Social Network Analysis

Social Networking

A **social networking** service (also **social networking site**, or **SNS** or **social media**) is an online platform which people use to build **social networks** or **social** relations with other people who share similar personal or career interests, activities, backgrounds or real-life connections.

------ Wikipedia

Largest social networking services

Service	Active users (in millions)
Facebook	2,234
YouTube	1,900
WhatsApp	1,500
Facebook Messenger	1,300
WeChat	1,058
Instagram	1,000
QQ	803
QZone	548
TikTok	500
Sina Weibo	431
Twitter	335
Reddit	330
LinkedIn	303
Baidu Tieba	300
Skype	300
Snapchat	291
Viber	260
Pinterest	250
LINE	203
Telegram	200
Talkoon	110

Social Network Analysis

- **Social network analysis (SNA)** is the process of investigating social structures through the use of networks and graph theory.

  ------ Wikipedia

- **SNA** is the mapping and measuring of relationships and flows between people, groups, organizations, computers, URLs, and other connected information/knowledge entities.

Introduction

- Much information gained by analysing social networks data.
- Example - “friends” relation found on Facebook.
- Important question - how to identify “communities?
- Some techniques are similar to clustering algorithms
- Communities almost never partition network, communities usually overlap.

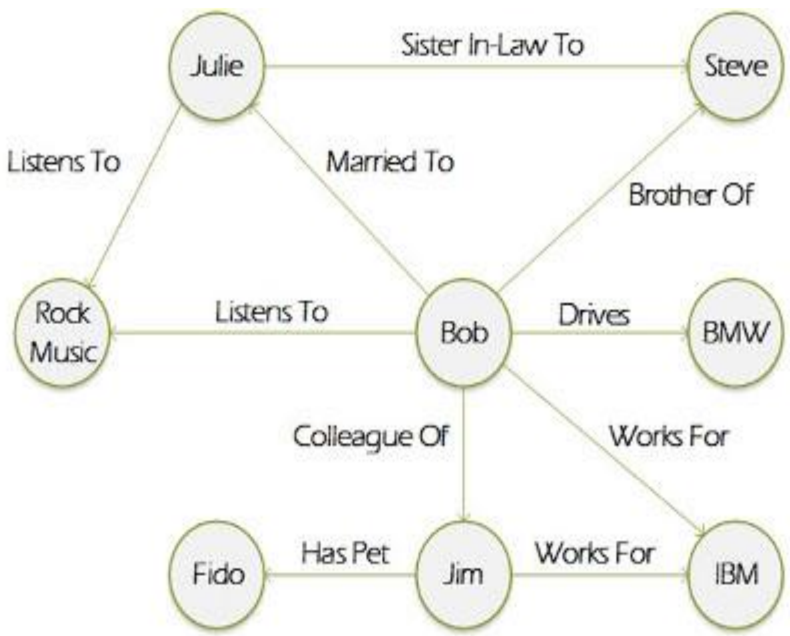
Introduction Cont…

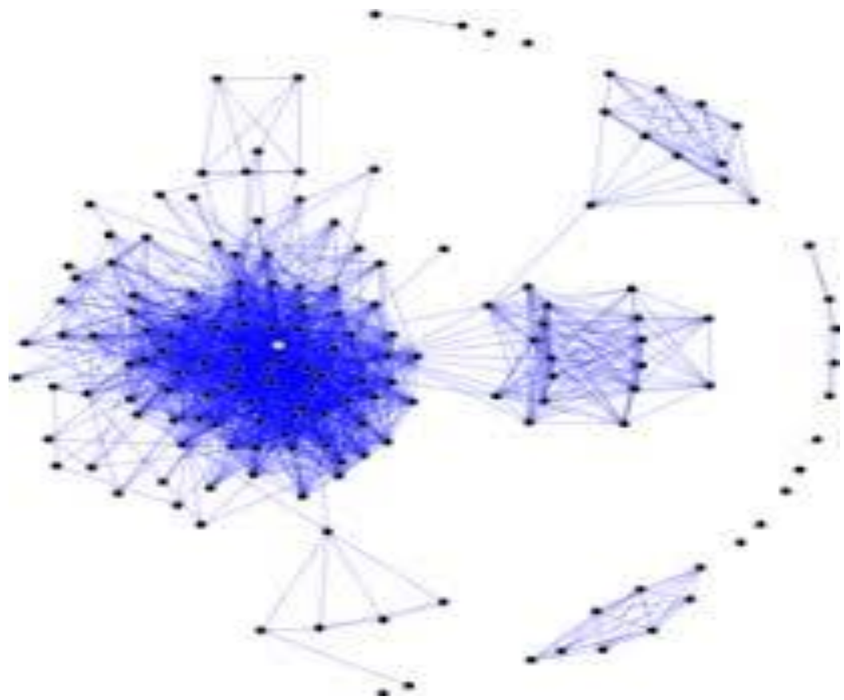
We will discuss

- a way to discover similarities among nodes of a graph
- way to measure the connectedness of a community (triangle counting)

Social Networks as Graphs

- A social graph is a diagram that illustrates interconnections among people, groups and organizations in a social network.
- The term is also used to describe an individual's social network.



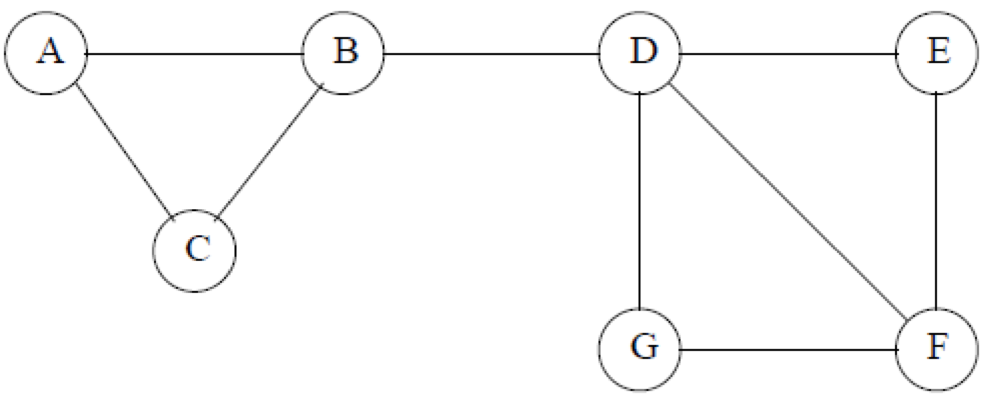


What is a Social Network?

- **Social Network** is a website that brings people together to talk, share ideas and interests, or make new friends.
- E.g. Facebook, Twitter, Google+, or another website that is called a "social network
- Characteristics of Social Network
  - Collection of entities that participate in the network
  - At least one relationship between entities of network
  - There is assumption of non-randomness or locality - relationships tend to cluster. (if entity A is related to both B and C, then there is a higher probability than average that B and C are related)

Social Networks as Graphs

- Social networks are naturally modeled as graphs, called as a social graph.
- The entities are the nodes
- An edge connects two nodes if the nodes are related by the relationship that characterizes the network.
- Degree is represented by labeling the edges.
- Often, social graphs are undirected



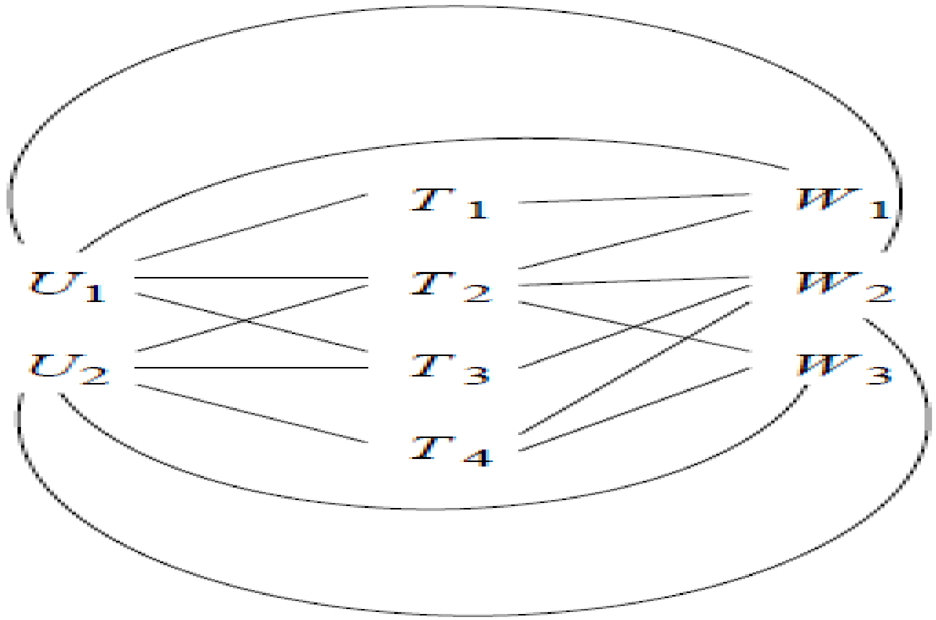
Example of a small social network

Varieties of Social Networks

- Telephone Networks
- Email Networks
- Collaboration Networks

Graphs With Several Node Types

- Seen many examples of social networks
- **Complex Example** - users at a site del.icio.us place tags on Web pages.
- Three different kinds of entities: users, tags, and pages.
- Natural way to represent such information is as a k-partite graph for some k > 1.



Clustering of Social-Network Graphs

- Communities in SN are connected by many edges.
- These correspond to groups of friends at school or groups of researchers interested in the same topic
- Graphs can be clustered to identify communities.
- Earlier techniques unsuitable for the problem of clustering social-network graphs.

Distance Measures for Social-Network Graphs

- Need to define a distance measure to apply standard clustering techniques.
- Edge labels might be usable as a distance measure
- Edges may be unlabeled like “Friend Relation” in Facebook
- Nodes are close if they have an edge between them and distant if not
- distance d(x, y) is 0 if there is an edge (x, y) and 1 if no edge.

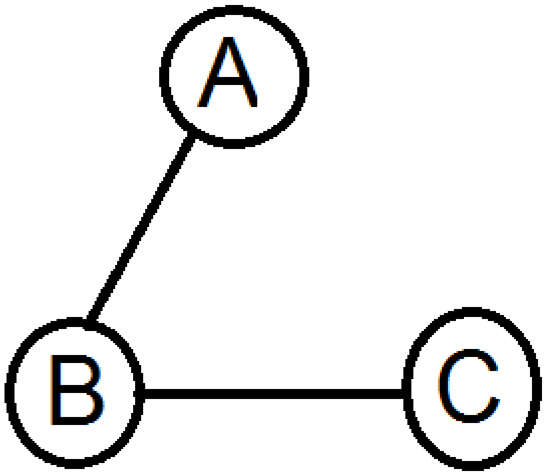
  or

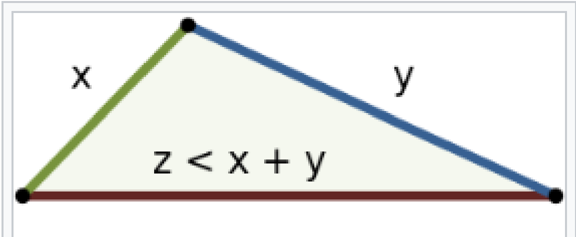
  distance d(x, y) = 1 if there is an edge (x, y) = ∞ if no edge
- Neither of these two-valued “distance measures” – is a true distance measure.
- The reason is that they violate the triangle inequality

Distance Measures for Social-Network Graphs

Triangle Inequality

In mathematics, the **triangle inequality** states that for any triangle, the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side.





- Distance from A to C exceeds the sum of the distances from A to B to C.
- Fix this problem, say, distance 1 for an edge and distance 1.5 for a missing edge

Applying Standard Clustering Methods Cont…

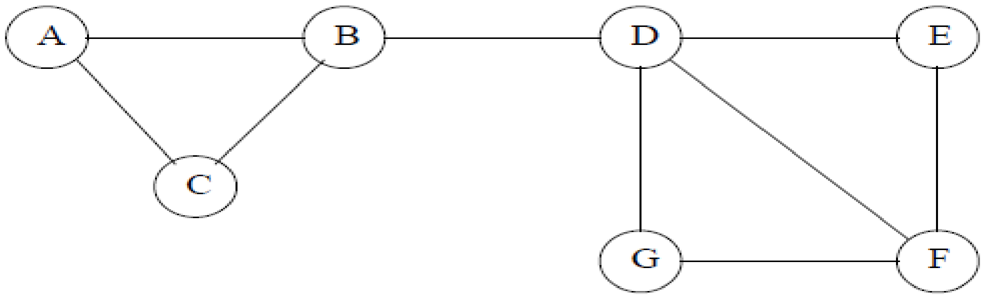
- Two general approaches to clustering:
  - hierarchical (agglomerative) and
  - point-assignment

Hierarchical Clustering

- Hierarchical clustering of a social-network graph starts by combining some two nodes that are connected by an edge.
- Successively, edges that are not between two nodes of the same cluster would be chosen randomly to combine the clusters to which their two nodes belong
- The choices would be random, because all distances represented by an edge are the same

Applying Standard Clustering Methods Cont…

Hierarchical Clustering



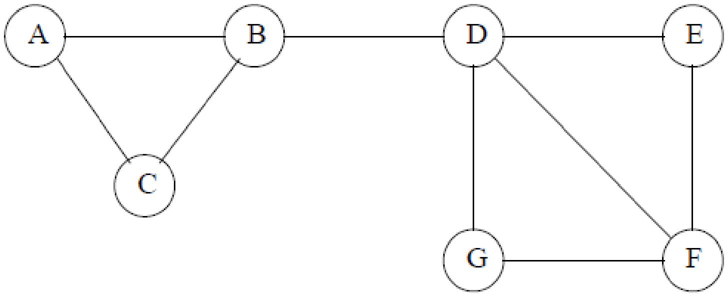
Example of a small social network

- At highest level it appears that there are two communities {A,B,C} and {D,E, F,G}.
- {D,E, F} and {D, F, G} subcommunities of {D,E, F,G} - ever be identified by a pure clustering algorithm
- **Problem** - likely to chose to combine B and D, even though they surely belong in different clusters.
- **Solution** -
  - Run hierarchical clustering several times and pick the run that gives most coherent clusters
  - Can use a more sophisticated method for measuring the distance between clusters

Applying Standard Clustering Methods Cont…

Point-assignment approach to clustering social networks:

- Suppose we try a **k-means** approach, pick k = 2.
- Pick two starting nodes at random.
- start with one randomly chosen node and then pick another as far away as possible e.g., E and G.
- suppose we get two starting nodes, B and F
  - assign A and C to the cluster of B
  - E and G to the cluster of F
  - But D is as close to B as it is to F,
- Deferred decision about D, until all are assigned
- Shortest average distance to all the nodes of the cluster, then D should be assigned **to cluster of F**
- **In large graphs, we shall surely make mistakes**.

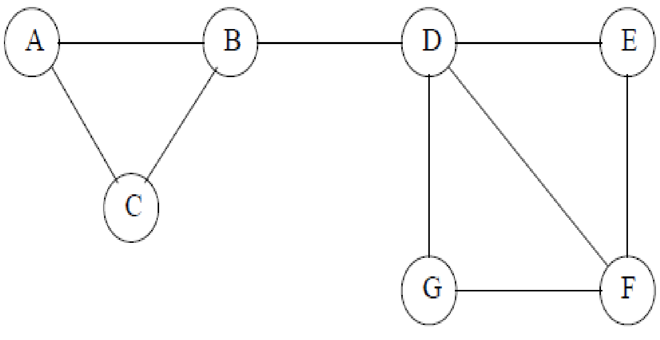


Example of a small social network

Betweenness

- Problems with standard clustering methods
- Several specialized clustering techniques have been developed to find communities in social networks.
- Simplest, based on finding the edges that are least likely to be inside a community.
- Define the **betweenness** of an edge (a, b) to be the number of pairs of nodes x and y such that the edge (a, b) lies on the shortest path between x and y.
- To be more precise, since there can be **several shortest paths between x and y**, edge (a, b) is credited with the **fraction of those shortest** paths that include the edge (a, b).

- **Example**:
  - ✓ (B,D) has the highest betweenness
  - ✓ edge is on every shortest path between any of A, B, and C to any of D, E, F, and G **(3 x 4 = 12)**
  - ✓ betweenness edge (D, F) = 4 **(on four SP from A, B, C, and D to F)**



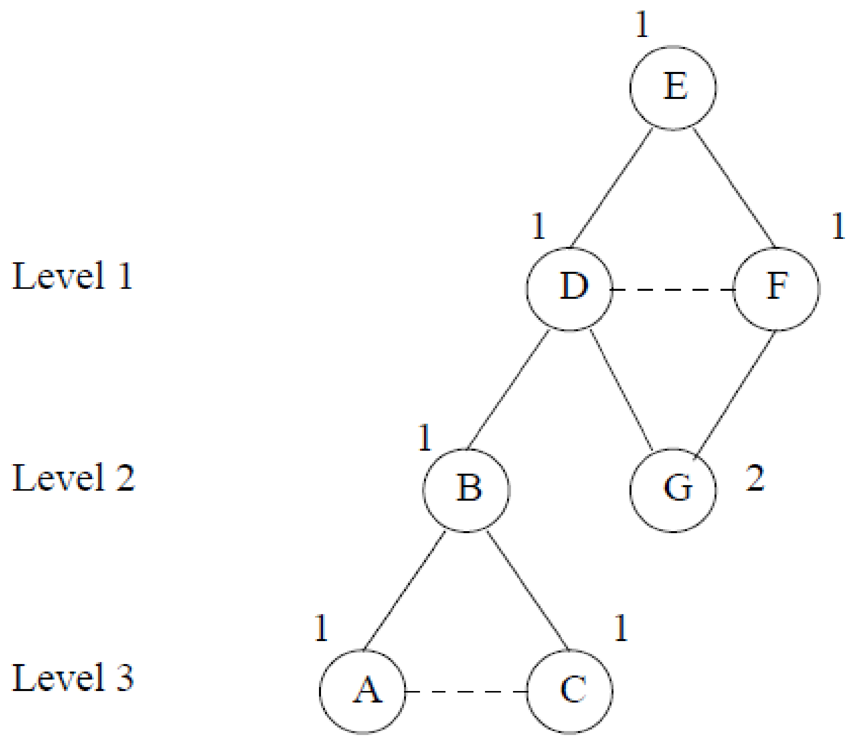
Example of a small social network

The Girvan-Newman Algorithm

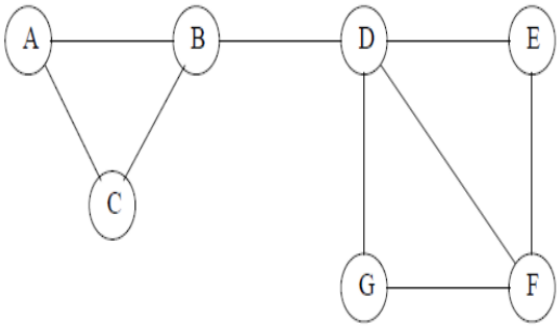
- In order to exploit betweenness of edges, need to calculate number of Shortest Paths (SPs) going through each edge.
- Can use **Girvan-Newman (GN) Algorithm.**
- GN Algorithm visits each node X once and computes the number of shortest paths from X.
- Makes use of **BFS** starting at the node X.
- Level of each node in BFS is length of SP from X to that node
- Edges between levels are called DAG edges.
- Each DAG edge will be part of at least one SP.

The Girvan-Newman Algorithm Cont…

Step – 1 - breadth-first presentation of the graph starting at **E**



Step 1 of the Girvan-Newman Algorithm



Example of a small social network

Step – 2 - label each node by the number of SPs that reach it from root.

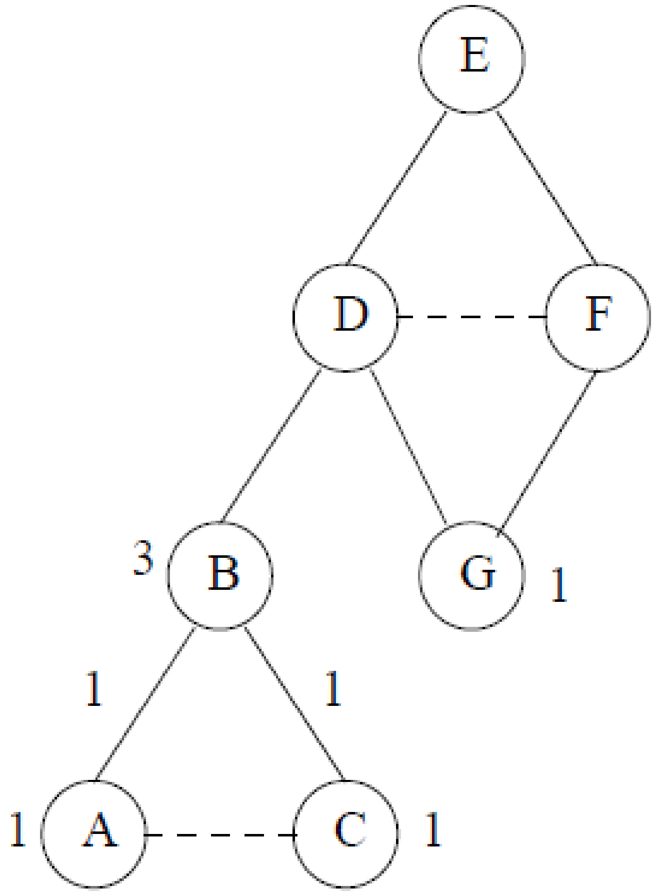
The Girvan-Newman Algorithm Cont…

Step – 3

- Calculate for each edge e sum over all nodes Y of the fraction of SPs from root X to Y that go through e.
- Calculation involves computing this sum for both nodes and edges, from the bottom.
- Each Nodes and Edges are given Credit.
- The rules for the calculation of Credit are as follows:
  1. Each leaf in the DAG gets a credit of 1.
  2. Each node that is not a leaf gets a credit equal to 1 plus the sum of credits of the DAG edges from that node to the level below.
  3. A DAG edge e entering node Z from the level above is given a share of the credit of Z proportional to the fraction of SPs from the root to Z that go through e.

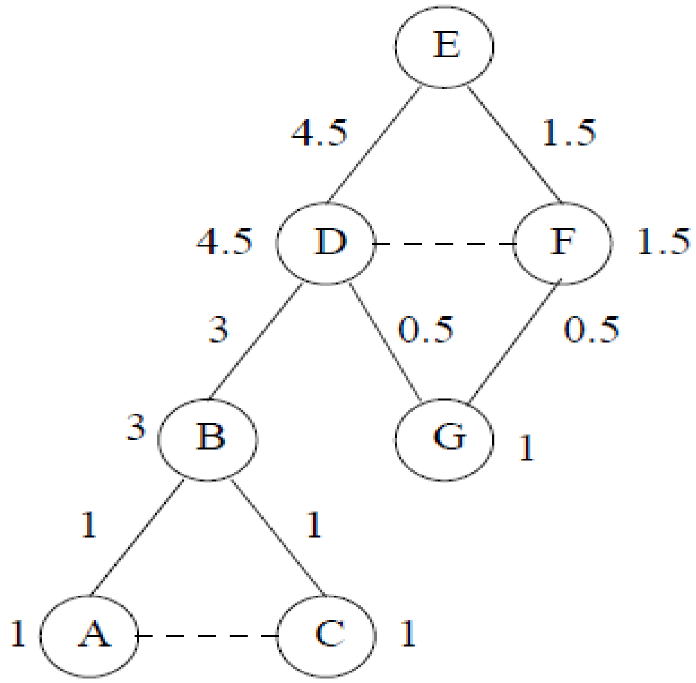
The Girvan-Newman Algorithm Cont…

Step – 3



The Girvan-Newman Algorithm Cont…

Step – 3

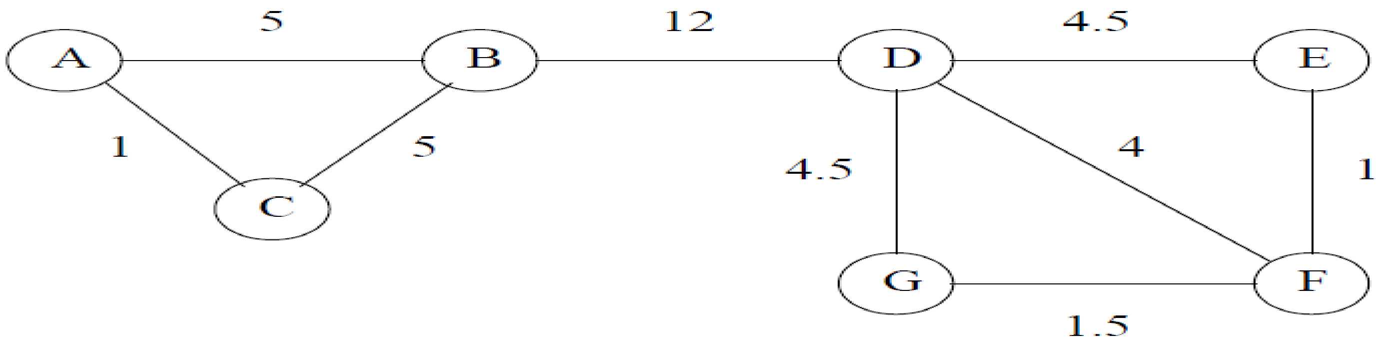


- To complete the betweenness calculation,
  - repeat this calculation for every node as the root
  - sum the contributions.
  - Finally, we must divide by 2 to get the true betweenness

Using Betweenness to Find Communities

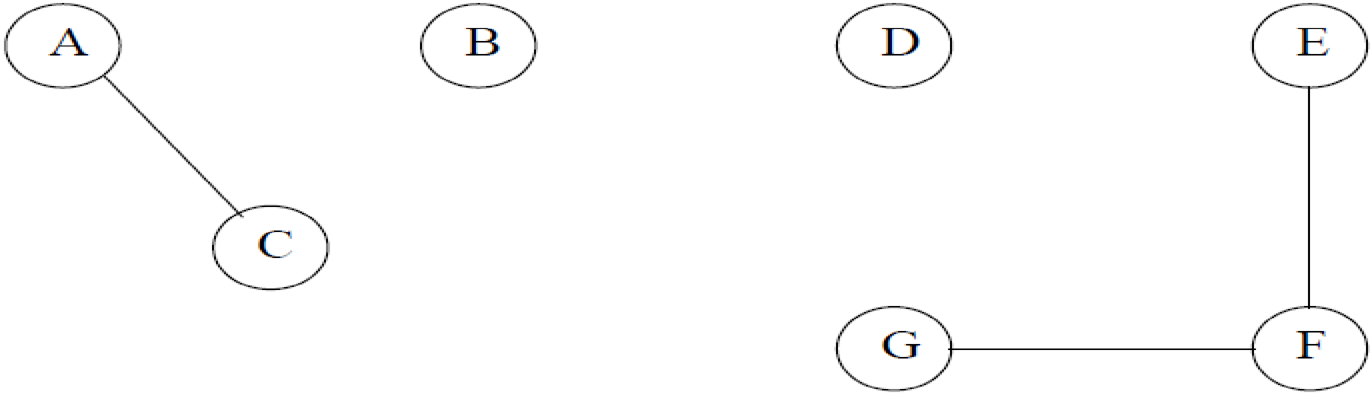
- Betweenness scores for the edges of a graph behave something like a distance measure
- Not exactly a distance measure, because it is not defined for pairs of nodes that are unconnected by an edge, and might not satisfy triangle inequality
- Idea is expressed as a process of edge removal
- Remove edges with the highest betweenness

Using Betweenness to Find Communities Cont…



Betweenness scores for the graph

Remove edges with betweenness four or more



All the edges with betweenness 4 or more have been removed

- **B is a "traitor" to the community {A,B,C}**
- **D can be seen as a "traitor" to the group {D,E, F,G}**

Speeding Up the Betweenness Calculation

- If we apply the above method to a graph of n nodes and e edges, it takes O(ne) running time
- BFS from a single node takes O(e) time, there are n of the computations
- If the graph is large – and even a million nodes will high running time
- Can pick a subset of nodes at random and use these as the roots of breadth-first searches
- Can get an approximation to the betweenness of each edge

Direct Discovery of Communities

Limitations of Betweenness to Find Communities

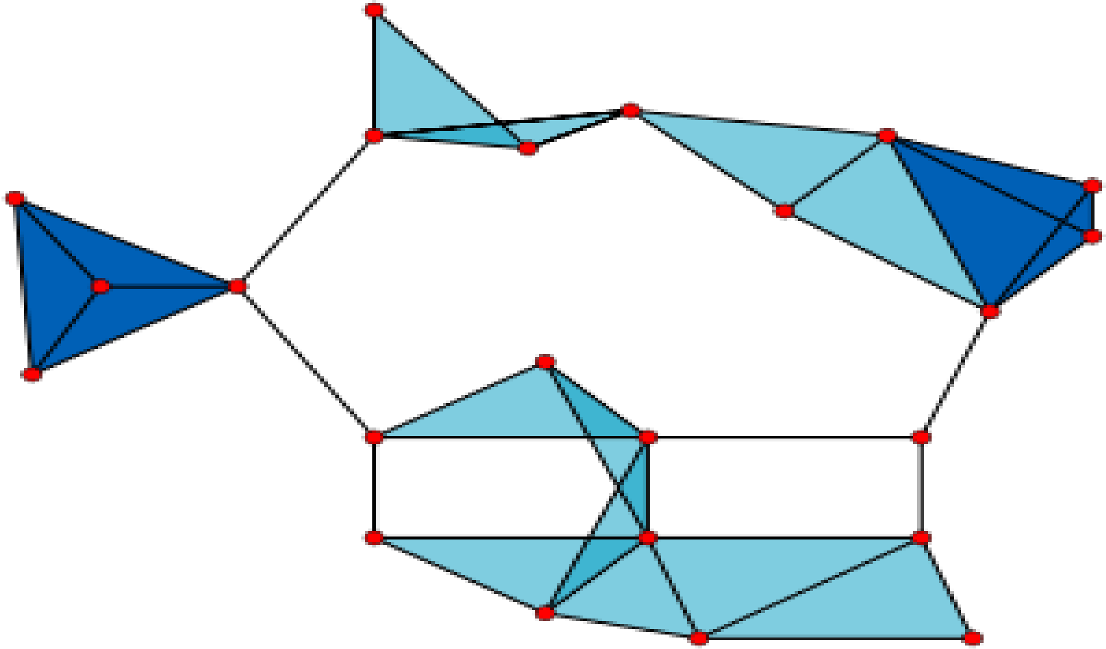
Searched for communities by partitioning all individuals in SN.

- **Limitations :** It is not possible to place an individual in two different communities
- Need a technique for discovering communities directly by looking for subsets of the nodes that have a relatively large number of edges among them.

Direct Discovery of Communities Cont…

Clique (graph theory)

In the mathematical area of graph theory, a **clique** is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete.



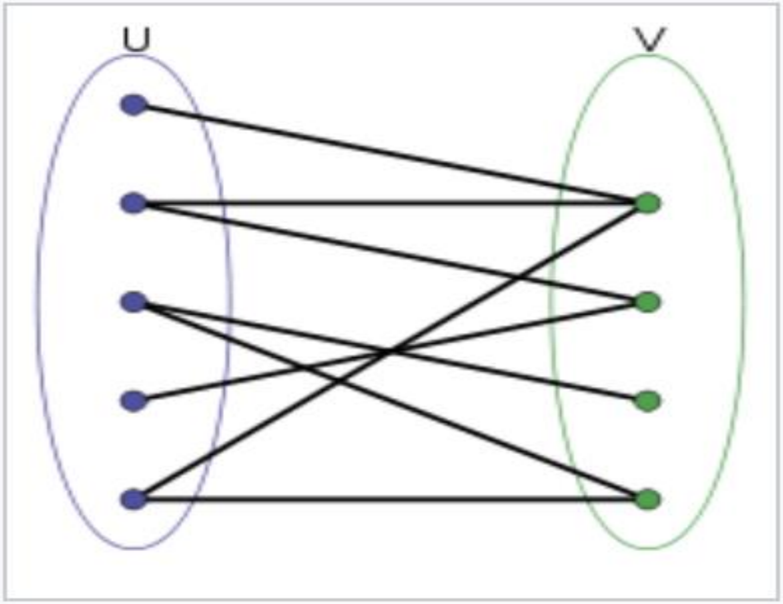
Direct Discovery of Communities Cont…

Finding Cliques

- Find sets of nodes with many edges by finding a large clique
- However, that task is not easy.
- Finding maximal cliques NP-complete, but hardest of the NP
- Even approximating the maximal clique is hard.
- Cliques may be relatively small – result in identifying small size community.

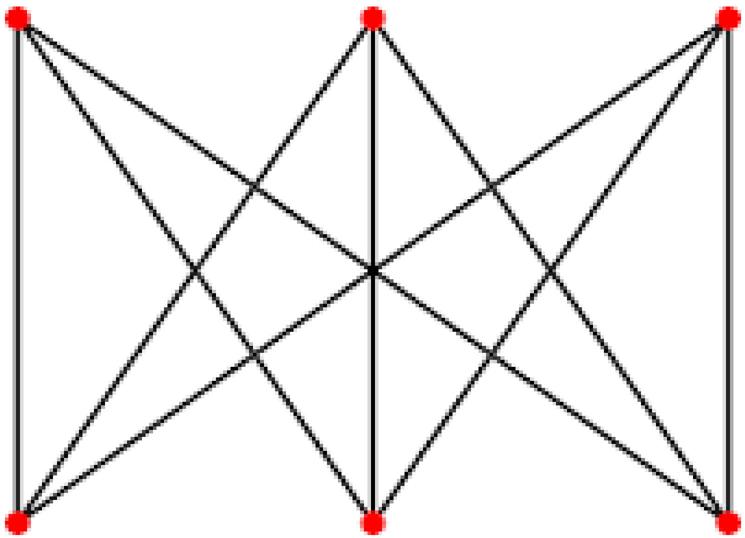
Bipartite graphs (graph theory)

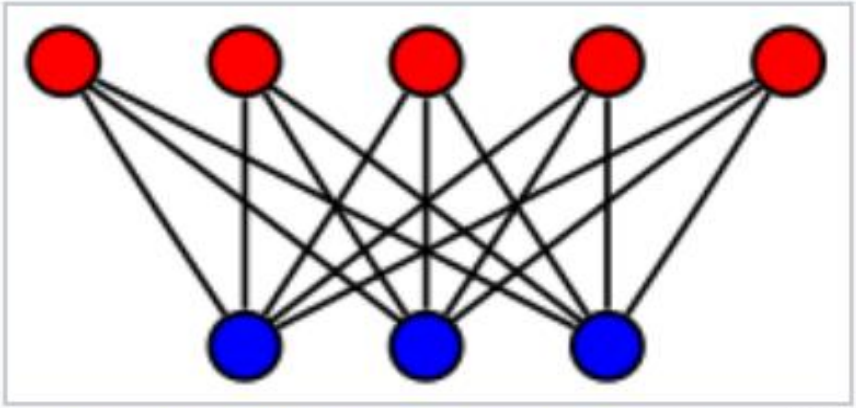
- In the mathematical field of graph theory, **a bipartite graph (or bigraph)** is a graph whose vertices can be divided into two disjoint and independent sets U and V such that every edge connects a vertex in U to one in V.
- Vertex sets U and V are called the parts of the graph.



Compete Bipartite graphs (graph theory)

In the mathematical field of graph theory, a **complete bipartite graph or biclique** is a special kind of bipartite graph where every vertex of the first set is connected to every vertex of the second set.





A complete bipartite graph with m = 5 and n = 3

Compete Bipartite graphs

- A complete bipartite graph consists of s nodes on one side and t nodes on the other side, with all st possible edges between the nodes of one side and the other present.
- We denote this graph by Ks,t.
- Complete bipartite graphs as subgraphs of general bipartite graphs and cliques as subgraphs of general graphs.
- No guarantee that a graph with many edges necessarily has a large clique.
- But possible to guarantee that a bipartite graph with many edges has a large complete bipartite subgraph.
- Can regard a complete bipartite subgraph as the nucleus of a community.

Direct Discovery of Communities Cont…

- We can also use complete bipartite subgraphs for community finding in ordinary graphs.
- Divide the nodes into two equal groups at random.
- If a community exists, then we would expect about half its nodes to fall into each group, and half its edges would go between groups.
- Chance of identifying a large complete bipartite subgraph in the community.
- To this nucleus we can add nodes from either of the two groups, if they have edges to many of the nodes already identified as belonging to the community.

Direct Discovery of Communities Cont…

Finding Complete Bipartite Subgraphs

- Given a large bipartite graph G , find instances of $K_{s,t}$
- Can view as finding frequent itemsets
- Let "items" be the nodes on left side of G
- Assume that the instance of $K_{s,t}$ looking for has t nodes on left side, and assume $t \le s$.
- "Baskets" correspond to nodes on right side of G
- Members of basket for node v are nodes to which v is connected.
- Let support threshold be s, number of nodes that instance of $K_{s,t}$ has on right side.

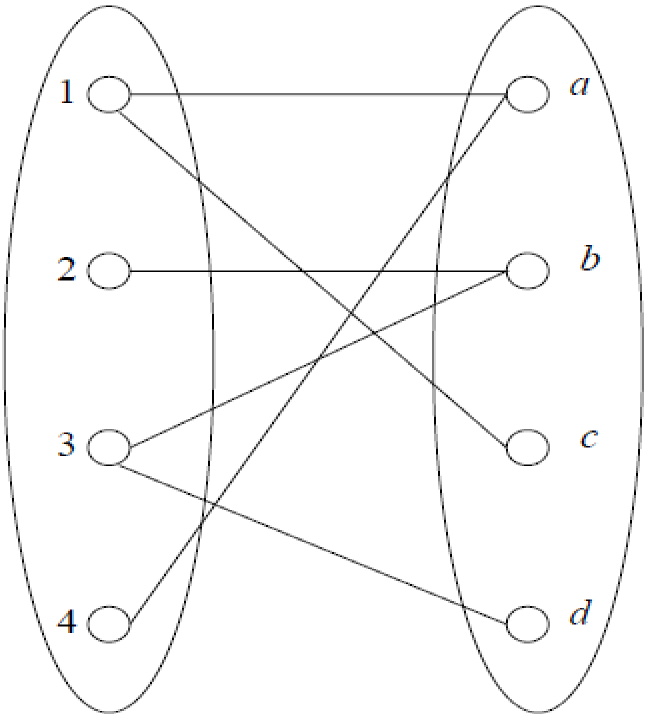
Direct Discovery of Communities Cont…

Finding Complete Bipartite Subgraphs

- Can state the problem of finding instances of $K_{s,t}$ as that of finding frequent item-sets F of size t.
- If a set of t nodes on left side is frequent, then they all occur together in at least s baskets.
- Baskets are the nodes on right side
- Each basket corresponds to a node that is connected to all t of nodes in F.
- Thus, frequent itemset of size t and s of baskets in which all those items appear form an instance of Ks,t.

Direct Discovery of Communities Cont…

Finding Complete Bipartite Subgraphs Example



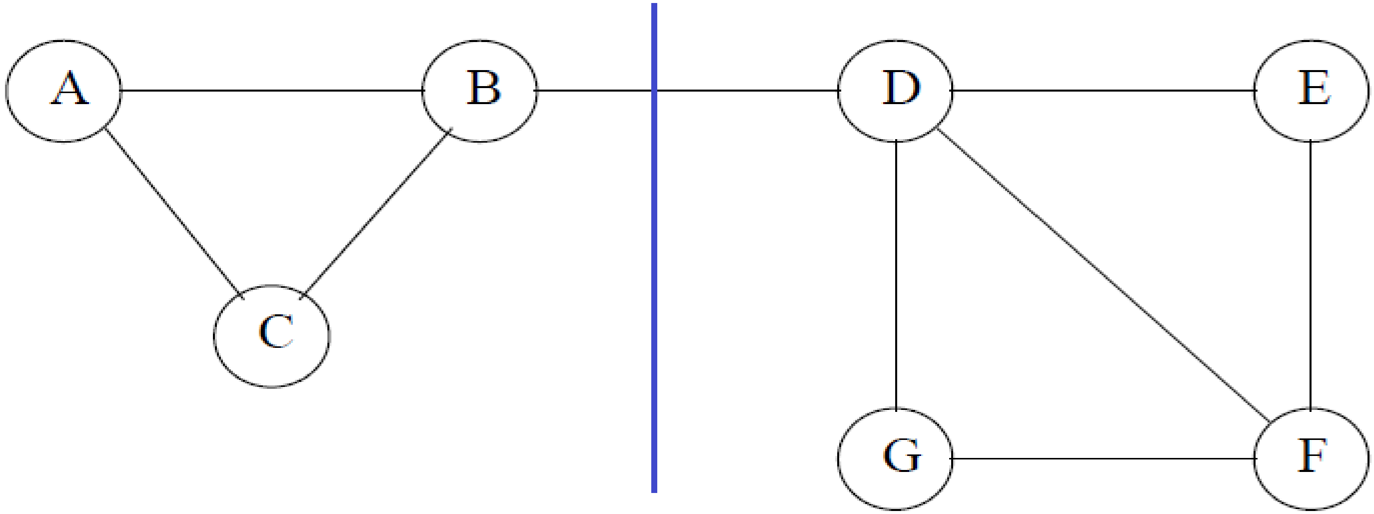
The bipartite graph

- Left side is nodes {1, 2, 3, 4} - **items** and right side is {a, b, c, d} – **baskets**
- basket *a* consists of "items" 1 and 4;
- a = {1, 4}, b = {2, 3}, c = {1} and d = {3}.
- If s = 2 and t = 1, must find item-sets of size 1 that appear in at least two baskets.
- {1} is one such itemset, and {3} is another.

In this tiny example there are no item-sets for larger, more interesting values of s and t, such as s = t = 2.

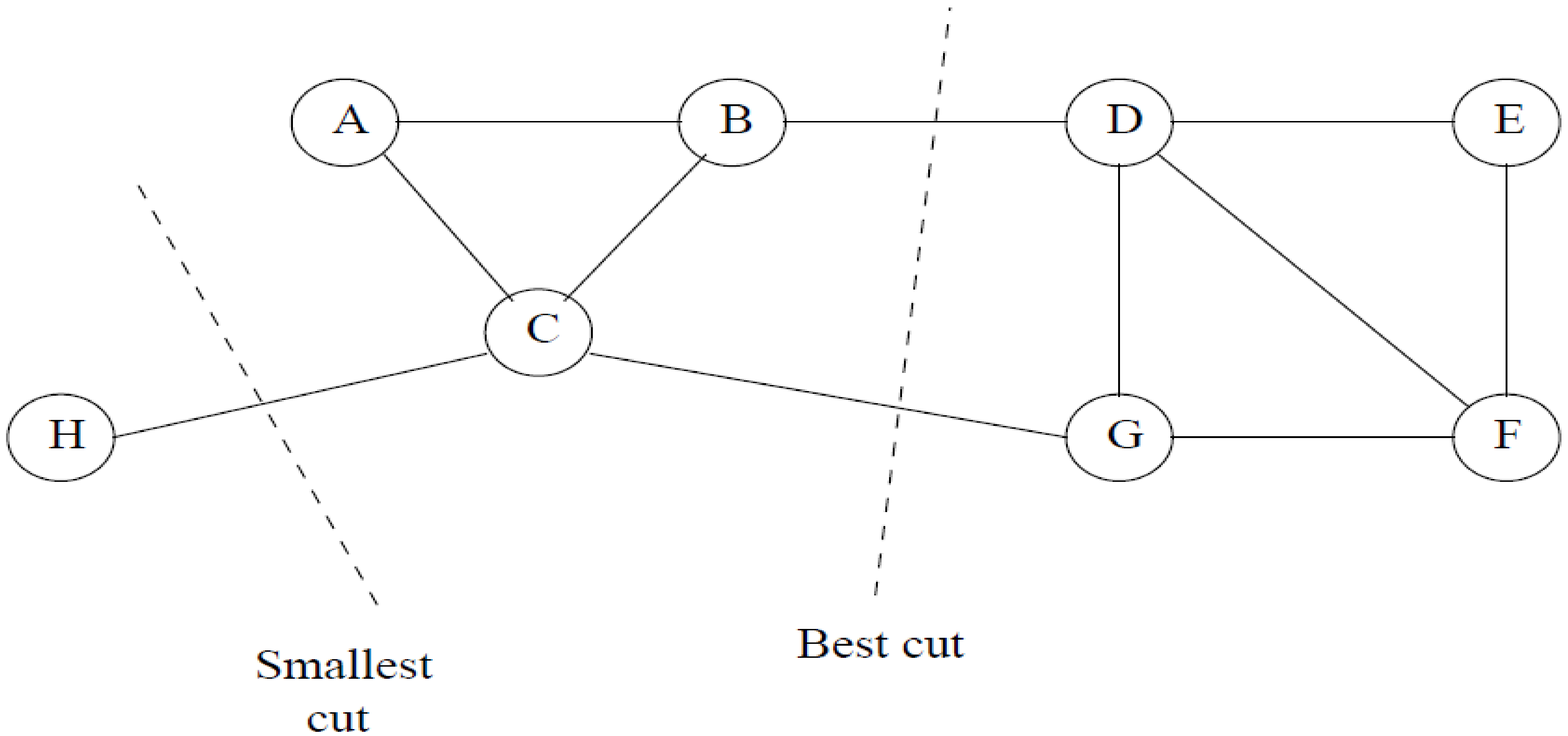
Partitioning of Graphs

- To partitioning a graph, we will use some important tools from matrix theory
- **Objective** – minimize number of edges that connect different components.
- Goal of minimizing the "cut" size needs to be understood carefully
- What makes Good Partition?



Partitioning of Graphs Cont…

- ❑ Divide nodes into two sets so that cut, or set of edges that connect nodes in different sets is minimized.
- ❑ Two sets are approximately equal in size



The smallest cut might not be the best cut

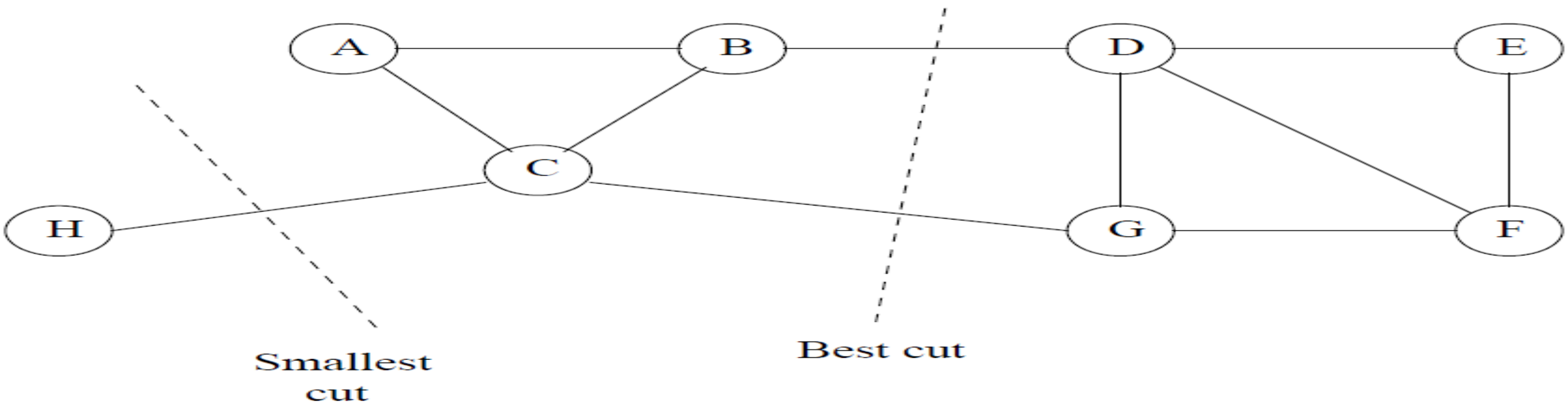
Partitioning of Graphs Cont…

Normalized Cuts

- Good" cut must balance the size of the cut against sizes of the sets
- One choice - "normalized cut."
- Volume of a set S (Vol (S)) - to be number of edges with at least one end in S
- Suppose nodes of graph are partitioned into two disjoint sets S and T
- Let Cut (S, T) be the number of edges that connect a node in S to a node in T .
- Then normalized cut value for S and T is

$$\frac{Cut(S,T)}{Vol(S)} + \frac{Cut(S,T)}{Vol(T)}$$

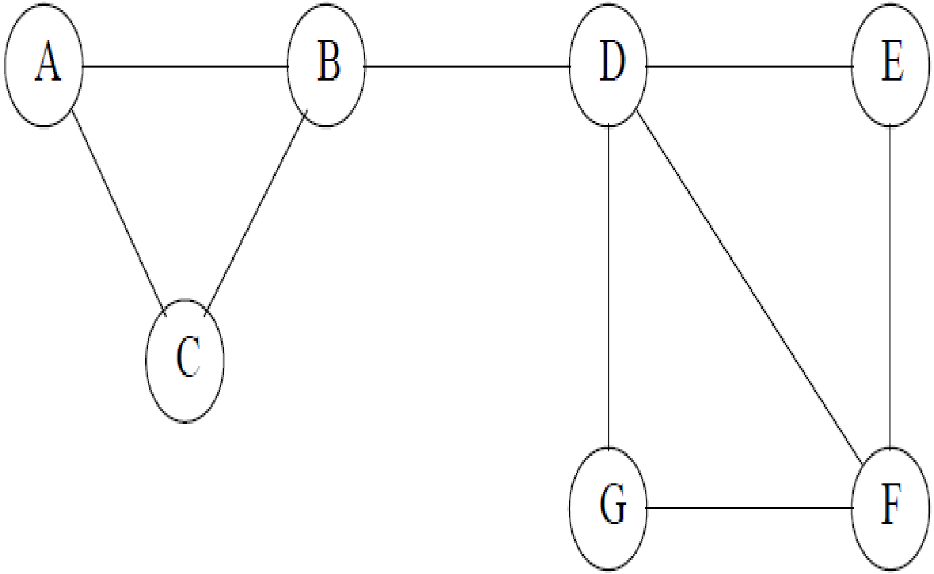
Normalized Cuts - Example



- If we choose S = {H} and T = {A,B,C,D,E, F,G}, then Cut (S, T) = 1.
- Vol(S) = 1 and Vol(T) = 11
- **normalized cut = 1/1 + 1/11 = 1.09**
- **Consider other cut**, then S = {A,B,C,H} and T = {D,E, F,G}
- Cut (S, T) = 2
- Vol (S) = 6, and Vol(T) = 7.
- **Normalized cut = 2/6 + 2/7= 0.62.**

Some Matrices That Describe GraphsCuts

- Matrix algebra can help us find good graph partitions



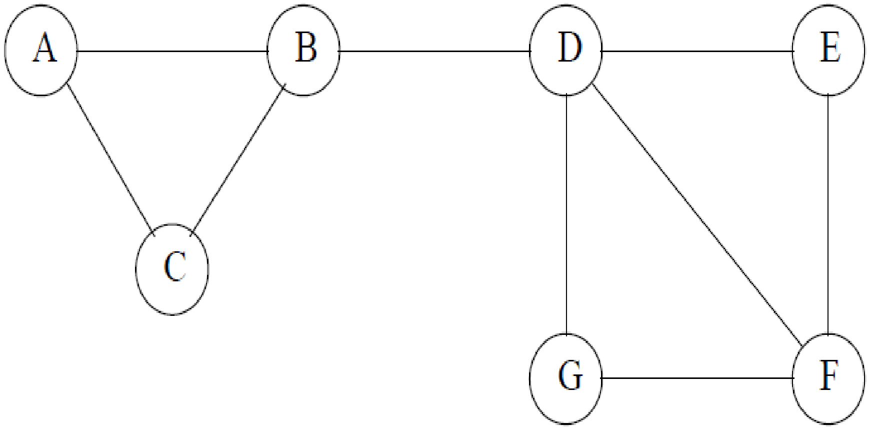
$$
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0
\end{bmatrix}
$$

The adjacency matrix

Partitioning of Graphs Cont…

Some Matrices That Describe GraphsCuts

- The second matrix we need is degree matrix for a graph.
- This graph has nonzero entries only on the diagonal.
- The entry for row and column *i* is the degree of the *ith* node.



$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

The degree matrix

Partitioning of Graphs Cont…

Some Matrices That Describe GraphsCuts

- Suppose our graph has adjacency matrix A and degree matrix D.
- Matrix, called the Laplacian matrix, is L = D – A

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

The adjacency matrix | The degree matrix | The Laplacian matrix

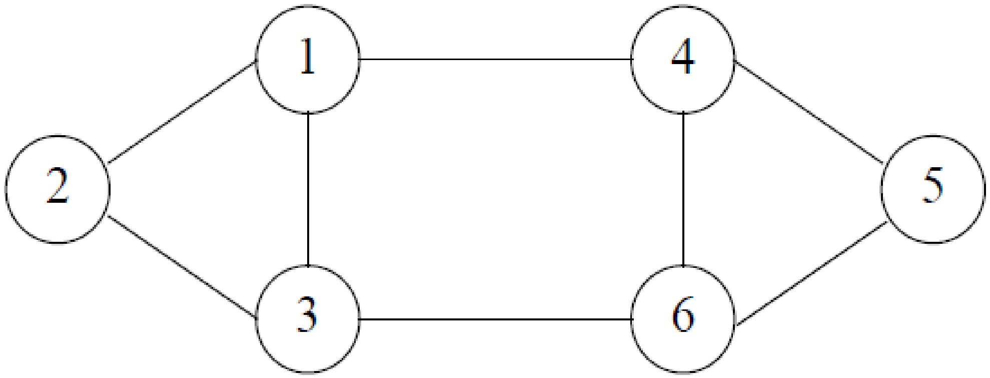
Partitioning of Graphs Cont…

Some Matrices That Describe GraphsCuts

- Best way to partition a graph from the eigenvalues and eigenvectors of its Laplacian matrix.

- Can obtain a partition by taking one set to be the nodes i whose corresponding vector component xi is positive and other whose components are negative.

Some Matrices That Describe GraphsCuts



$$\begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 \\ -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

Laplacian matrix

Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	−5	−1	−1	−1
	1	2	4	−2	1	0
	1	1	1	3	−1	1
	1	−1	−5	−1	1	1
	1	−2	4	−2	−1	0
	1	−1	1	3	1	−1

Eigenvalues and eigenvectors for the matrix

- Second eigenvector has 3 +ve and 3 –ve components.
- One group should be **{1, 2, 3}**, and other group should be **{4, 5, 6}**.

Partitioning of Graphs Cont…

Alternative Partitioning Methods

- We could set the threshold at some point other than zero.
- Threshold was not zero, but −1.5.
- Then nodes 4 and 6, with components −1 in the second eigenvector, would join 1, 2, and 3, and only node 5 in the other.
- **Other approach:** Use the method described above to split the graph into two, and then use it repeatedly on the components to split them.
- Can use several of the eigenvectors, not just the **second.**

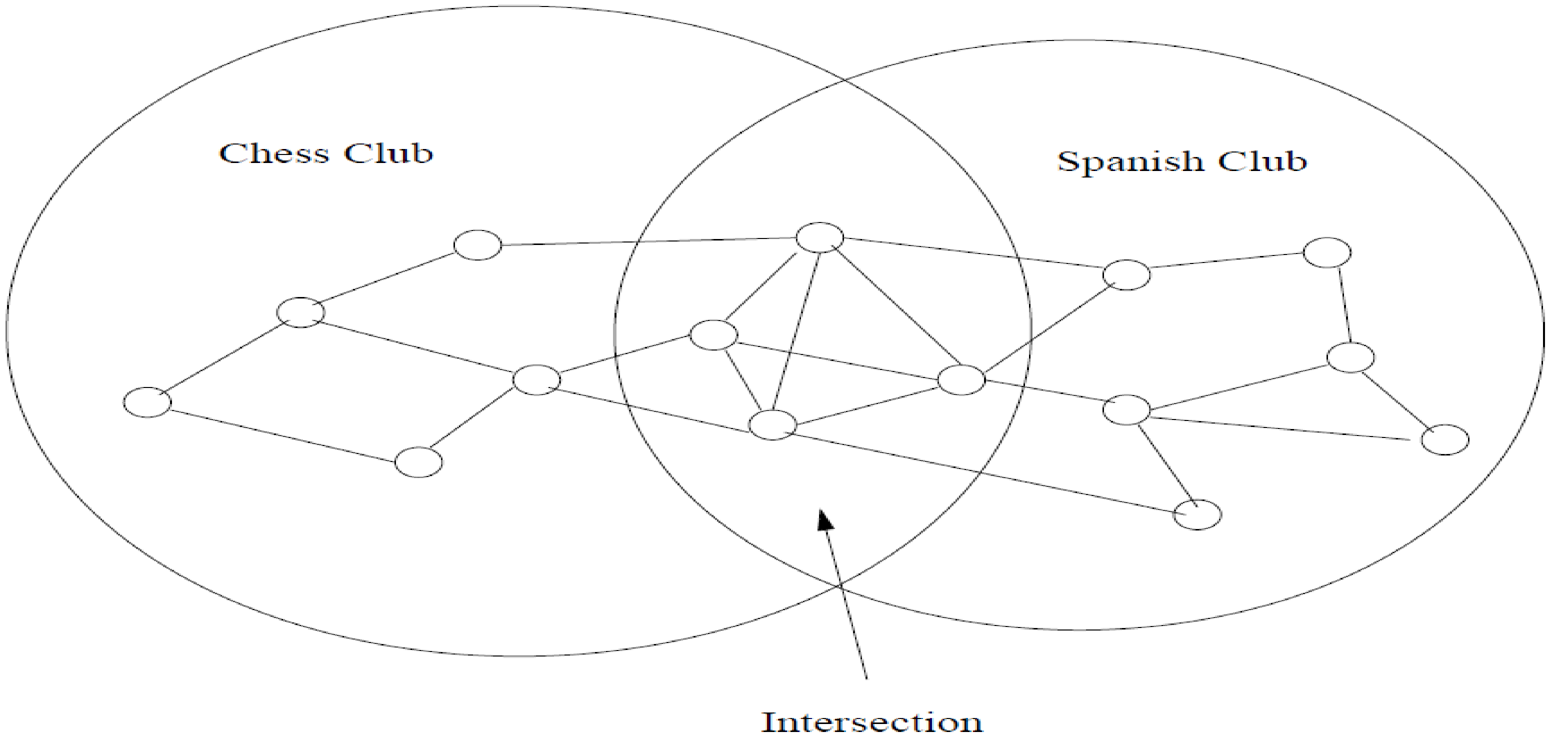
Finding Overlapping Communities

- Concentrated on clustering a social graph to find communities.
- In practice communities are rarely disjoint.
- We explain a method for taking a social graph and fitting a model to it that best explains how it could have been generated by a mechanism.
- This assumes that probability that two individuals are connected by an edge (are “friends”) increases as they become **members of more communities in common.**
- An important tool in this analysis is “maximum-likelihood estimation” (MLE).

Finding Overlapping Communities Cont…

Nature of Communities

- Nodes are people and there is an edge between two nodes if the people are “friends.”



The overlap of two communities is denser than the nonoverlapping parts of these communities

Maximum-Likelihood Estimation

- In statistics, Maximum Likelihood Estimation (MLE) is a method of estimating the parameters of a statistical model, given observations.
- The method obtains the parameter estimates by finding the parameter values that maximize the likelihood function.
- The estimates are called maximum likelihood estimates, which is also abbreviated as **MLE**.

………**Wikipedia**

Finding Overlapping Communities Cont…

Maximum-Likelihood Estimation (MLE)

- We will learn a useful modeling tool called Maximum Likelihood Estimation (**MLE**)
- Assumption about the generative process (the model) that creates instances of some artifact, for example, “friends graphs”.
- Model has parameters that determine probability of generating any particular instance of the artifact.
- This probability is called the likelihood of those parameter values.
- We assume that value of parameters that gives largest value of likelihood is correct model for **observed artifact**.

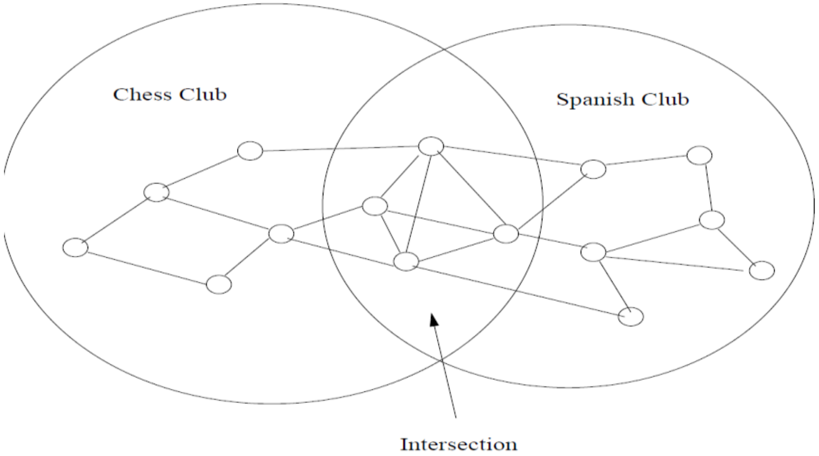
Finding Overlapping Communities Cont…

Maximum-Likelihood Estimation

- An example should make the MLE principle clear.
- For instance, we might wish to generate random graphs.
- Suppose that ***each edge is present*** with probability ***p*** and not present with probability ***1−p.***
- The only parameter we can adjust is ***p***.
- Each value of p there is a small but nonzero probability that the graph generated will be exactly the one we see.
- Following the MLE principle, we shall declare that true value of p is the one for which the probability of generating the observed graph is the highest.

Finding Overlapping Communities Cont…

- There are 15 nodes and 23 edges.
- 105 pairs of 15 nodes
- probability (likelihood) of generating exactly the graph is given by the function $p^{23}(1-p)^{82}$



- But function does have a maximum, which we can determine by taking its derivative and setting that to 0. That is:

$$23p^{22}(1-p)^{82} - 82p^{23}(1-p)^{81} = 0$$

- We can group terms to rewrite the above as

$$p^{22}(1-p)^{81}\, 23(1-p) - 82p^{)} = 0$$

- Only way the right side can be 0 is if p is 0 or 1, or the last factor $23(1-p) - 82p$ is 0.
- Likelihood of generating graph is maximized when $23 - 23p - 82p = 0$ or $p = 23/105$.

The Affiliation-Graph Model

- We shall now introduce a reasonable mechanism, called the affiliation-graph model, to generate social graphs from communities.
- Once we see how parameters of model influence likelihood of seeing a given graph, we can address how one would solve for values of the parameters that give maximum likelihood.
- The mechanism, called **community-affiliation graphs**.

Finding Overlapping Communities Cont…

The Affiliation-Graph Model

1. There is a given number of communities, and there is a given number of individuals (nodes of the graph).
2. Each community can have any set of individuals as members. That is, the memberships in the communities are parameters of the model.
3. Each community C has a probability pC associated with it, probability that two members of community C are connected by an edge because they are both members of C. These probabilities are also parameters of the model.
4. If a pair of nodes is in two or more communities, then there is an edge between them if any of communities of which both are members, justifies that edge according to rule (3).

Finding Overlapping Communities Cont…

The Affiliation-Graph Model

- We must compute the likelihood that a given graph with the proper number of nodes is generated by this mechanism.
- The key observation is how the edge probabilities are computed, **given an assignment of individuals to communities** and values of the pC's.
- Consider an edge (u, v) between nodes u and v.

Finding Overlapping Communities Cont…

The Affiliation-Graph Model

- Suppose u and v are members of communities C and D, but not any other communities.
- Then the probability that there is no edge between u and v is the product of the probabilities that there is no edge due to community C and no edge due to community D.
- That is, with probability $(1 - p_C)(1 - p_D)$ there is no edge (u, v) in the graph, and of course probability that there is such an edge is 1 minus that.

Finding Overlapping Communities Cont…

The Affiliation-Graph Model

- More generally, if u and v are members of a nonempty set of communities M and not any others, then p^{uv}, probability of an edge between u and v is given by:

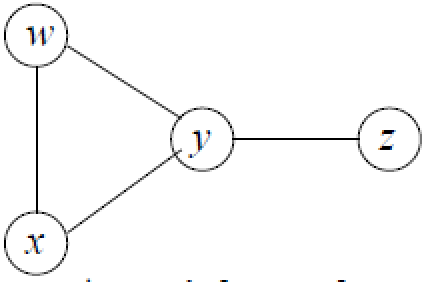
$$p_{uv} = 1 - \prod_{C \text{ in } M} (1 - p_C)$$

- As an important special case, if u and v are not in any communities together, then we take p_{uv} to be ϵ, some very tiny number.
- If we know which nodes are in which communities, then we can compute likelihood of given graph.
- Let M_{uv} be set of communities to which both u and v are assigned.
- Then likelihood of E being exactly the set of edges in observed graph is

$$\prod_{(u,v) \text{ in } E} p_{uv} \prod_{(u,v) \text{ not in } E} (1 - p_{uv})$$

Finding Overlapping Communities Cont…

The Affiliation-Graph Model Example



A social graph

- Suppose thereare two communities C and D, w probabilities pC and pD.
- C = {w, x, y} and D = {w, y, z}.
- Consider the pair of nodes w and x, M_{wx} = {C};

  $p_{wx} = 1 − (1 − pC) = pC.$
- Similarly, x and y are only together in C, y and z are only together in D, and likewise w and z are only together in D.
- Thus, we find p_{xy} = pC and $p_{yz} = p_{wz}$ = pD.

Finding Overlapping Communities Cont…

- Now the pair w and y are together in both communities, so

  $p_{wy} = 1-(1-pC)(1-pD) = pC + pD - pCpD.$

- Finally, x and z are not together in either community, so $p_{xz} = \epsilon$.
- Now, we can compute the likelihood of the graph,

  $p_{wx}p_{wy}p_{x}yp_{yz}(1 - p_{wz})(1 - p_{xz})$

- Substituting the expressions we developed above

  $(pC)^2pD(pC + pD - pCpD)(1 - pD)\underline{(1 - \epsilon)}$

  Note that ϵ is very small, so last factor is 1 and can be dropped.

- We must find values of pC and pD that maximize above expression
- So $pD \leq 1$, so $pC + pD - pCpD$ must grow positively with pC.

Finding Overlapping Communities Cont…

- Likelihood is maximized when pC = 1
- The expression becomes pD(1 − pD), and it is easy to see that this expression has its maximum at pD = 0.5
- That is, given C = {w, x, y} and D = {w, y, z}, the maximum likelihood for the graph occurs when members of C are certain to have an edge between them and there is a 50% chance that joint membership in D will cause an edge between the members.
- This reflects only part of the solution.
- We also need to find an assignment of members to communities such that the maximum likelihood solution for that assignment is the largest solution for any assignment.
- Once we fix on an assignment, we can find the probabilities, pC, even for very large graphs with large numbers of communities.
- The general method for doing so is called ***"gradient descent"***.

Counting Triangles

- One of the most useful properties of social-network graphs is the count of triangles and other simple subgraphs.
- We will discuss methods for estimating or getting an exact count of triangles in a very large graph.
- We begin with a motivation for such counts and then give some methods for counting efficiently.

Counting Triangles Cont…

Why Count Triangles?

- Consider Graph with n nodes and m edges
- Can calculate this number of triangles without difficulty.
- There are $\binom{n}{3}$ sets of three nodes, or approximately $\frac{n^3}{6}$
- Probability of an edge between any two given nodes being added is $m/\binom{n}{2}$ or approximately $2m/n^2$.
- The probability that any set of three nodes has edges between each pair, is approximately $(2m/n^2)^3 = 8m^3/n^6$.
- Expected number of triangles is $(8m^3/n^6)(n^3/6) = \frac{4}{3}(m/n)^3$

Counting Triangles Cont…

Why Count Triangles?

- If a graph is a social network with **n participants** and **m pairs of "friends,"** then number of triangles to be much greater than value for a random graph.
- The reason is that if A and B are friends, and A is also a friend of C, there should be a much greater chance than average that B and C are also friends.
- Counting the number of triangles helps us to measure **extent to which a graph looks like a social network**.
- The age of a community is related to the density of triangles.
- New community number of triangles is relatively small.

Counting Triangles Cont…

An Algorithm for Finding Triangles

- Consider a graph of n nodes and m ≥ n edges and nodes are integers 1, 2, . . . , n.
- Call a node a heavy hitter if its degree is at least $\sqrt{m}$.
- A heavy-hitter triangle is a triangle whose all three nodes are heavy hitters.
- Note that the number of heavy-hitter nodes is no more than 2 $\sqrt{m}$
- Since each edge contributes to degree of only two nodes, there would then have to be more than m edges.

Counting Triangles Cont…

An Algorithm for Finding Triangles

Assuming graph is represented by its edges, pre-process graph as follows:

1. Compute the degree of each node. The total time required is O(m)
2. Create an index on edges, with the pair of nodes at its ends as the key. A hash table suffices. It can be constructed in O(m) time.
3. Create another index of edges, this one with key equal to a single node.

We shall order the nodes as follows

- First, order nodes by degree.
- If v and u have the same degree, recall that both v and u are integers, so order them numerically.
- That is, we say $v \prec u$ if and only if either
    1. The degree of v is less than the degree of u, or
    2. The degrees of u and v are the same, and $v < u$.

Counting Triangles Cont…

An Algorithm for Finding Triangles

Heavy-Hitter Triangles:

- There are only O($\sqrt{m}$) heavy-hitter nodes, can consider all sets of three of these nodes.
- There are O($m^{3/2}$) possible heavy hitter triangles, and using the index on edges we can check if all three edges exist in O(1) time.
- Therefore, O($m^{3/2}$) time is needed to find all the heavy-hitter triangles.

Counting Triangles Cont…

An Algorithm for Finding Triangles: Other Triangles:

- Consider each edge (v_1, v_2). If both v_1 and v_2 are heavy hitters, ignore this edge.
- Suppose, however, that v_1 is not a heavy hitter and moreover $v_1 \prec v_2$. Let $u_1, u_2, \ldots, u_k$ be the nodes adjacent to v_1. Note that k <$\sqrt{m}$.
- We can find these nodes, using index on nodes, in O(k) time, which is surely O($\sqrt{m}$) time.
- For each u_i we can use first index to check whether edge (u_i, v_2) exists in O(1) time. We can also determine the degree of u_i in O(1) time, because we have counted all the nodes' degrees.
- We count triangle {v_1, v_2, u_i} if and only if the edge (u_i, v_2) exists, and $v_1 \prec u_i$.
- In that way, a triangle is counted only once – when v_1 is the node of the triangle that precedes both other nodes of the triangle according to the $\prec$ ordering.
- Thus, the time to process all the nodes adjacent to v_1 is O($\sqrt{m}$). Since there are m edges, the total time spent counting other triangles is O($m^{3/2}$).

Thank You !!!