Social Network Analysis

Session 1 - Why Networks?

Social network is a structure in which nodes are a set of social vertices that are connected together by different types of relationships. Because of the complexity of the nodes and the relationships between nodes, social networks are always represented by weighted, labeled and directed graphs. Social network analysis (SNA) is a set of techniques for determining and measuring the magnitude of the pressure. Social network analysis is focused also on visualization techniques for exploring the networks structure. It has gained a significant following in many fields of applications. It has been used to examine how the problems have been solved, how organizations interact with others, to understand the role of an individual in an organization. Networks are general language for describing complex systems.

You have all sorts of networks that I will be illustrating further below, but to mention a few it can be internet, Twitter, Facebook, Instagram, transportation, flights, citations, neural networks, financial networks, recommendation systems, friendship, professional networks.

Representing networks

Networks are illustrated with graphs and are collection of points joined by lines. These points are called vertex/nodes, and the lines are called edges.

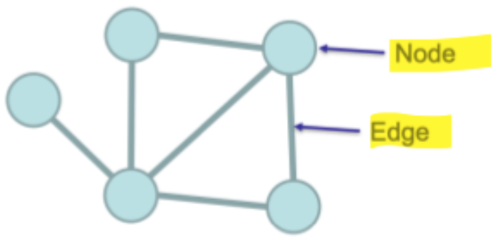


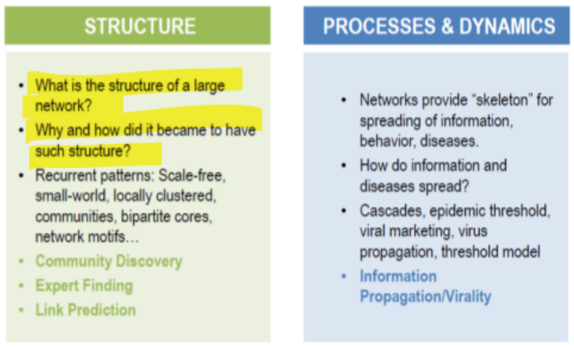
Figure 1: Illustration of networks

Mining social network data

Empirical: Study network data to find organizational principles
Mathematical models: probabilistic, graph theory
Algorithms: methods for analyzing graphs

Models Goals:

- Patterns and statistical properties of network data
- Design principles and models
- Understand why networks are organized the way they are (predict behavior of networked systems)


STRUCTURE

- What is the structure of a large network?
- Why and how did it became to have such structure?
- Recurrent patterns: Scale-free, small-world, locally clustered, communities, bipartite cores, network motifs…
- Community Discovery
- Expert Finding
- Link Prediction

PROCESSES & DYNAMICS

- Networks provide "skeleton" for spreading of information, behavior, diseases.
- How do information and diseases spread?
- Cascades, epidemic threshold, viral marketing, virus propagation, threshold model
- Information Propagation/Virality

Figure 2: What do we study?

How do you define a network?

It is important to proper represent the network for us to be able to use the network successfully. That means that there is sometimes a unique way to represent the data.

There are many tools, but we are using Igraph in R. Although you can use Igraph in python as well.

We have many different types of graphs, and here are some of them:

- **Simple graphs**: nodes of same type, edges are symmetric, binary
- **Directed graphs**: nodes of same type, edges are asymmetric
- **Weighted graphs**: nodes of same type, edges are numeric
- **Bi-partite graphs and projections**: nodes of different types

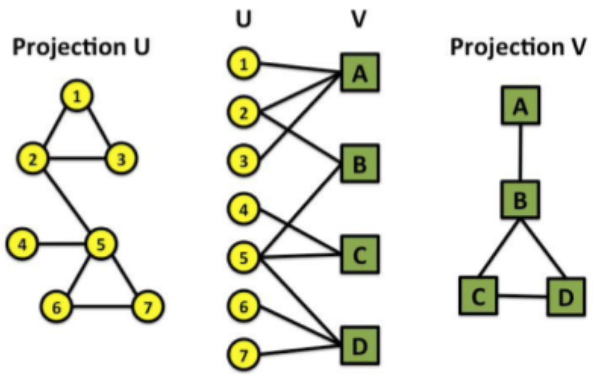
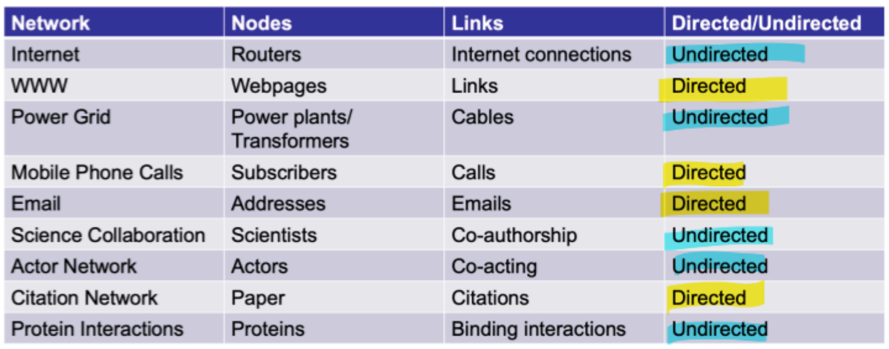


Figure 3: Projections



Network	Nodes	Links	Directed/Undirected
Internet	Routers	Internet connections	Undirected
WWW	Webpages	Links	Directed
Power Grid	Power plants/ Transformers	Cables	Undirected
Mobile Phone Calls	Subscribers	Calls	Directed
Email	Addresses	Emails	Directed
Science Collaboration	Scientists	Co-authorship	Undirected
Actor Network	Actors	Co-acting	Undirected
Citation Network	Paper	Citations	Directed
Protein Interactions	Proteins	Binding interactions	Undirected

Figure 4: Examples of networks

Session 2 - Network Properties

Representing graphs: Adjacency matrix

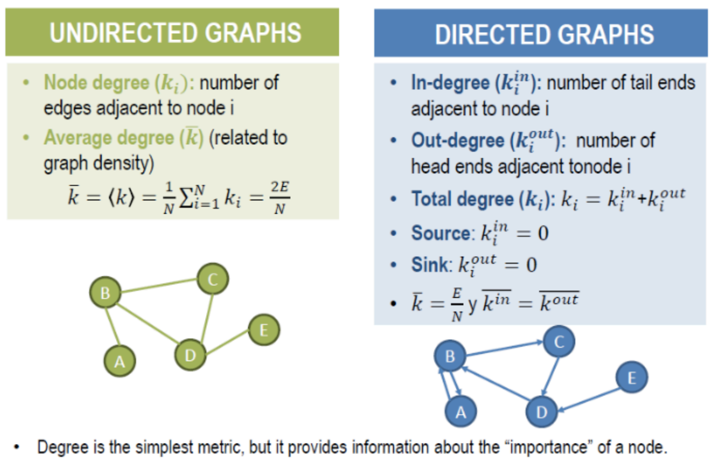
Aij = 1 if there is a link from node i to j
Aij = 0 otherwise

$$A = \begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 0 & 0 \\ B & 1 & 0 & 1 & 1 & 0 \\ C & 0 & 1 & 0 & 1 & 0 \\ D & 0 & 1 & 1 & 0 & 1 \\ E & 0 & 0 & 0 & 1 & 0 \end{array}$$

Figure 5: Adjacency matrix

Directed and undirected graphs

As briefly mentioned above there is something called directed graphs. Naturally the opposite of this is the undirected graphs. What does this mean? As the name kind off reveals, undirected graphs do not have a direction in its edges, making their adjacency matrix symmetrical and has a reciprocal relationship. i.e. friends on Facebook and collaborations. The opposite is directed graphs, and these are asymmetrical in their adjacency matrix. i.e. phone call, follow on Twitter/Instagram.



UNDIRECTED GRAPHS

- Node degree (k_i): number of edges adjacent to node i
- Average degree ($\bar{k}$) (related to graph density)

$$\bar{k} = \langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2E}{N}$$

DIRECTED GRAPHS

- In-degree (k_i^{in}): number of tail ends adjacent to node i
- Out-degree (k_i^{out}): number of head ends adjacent tonode i
- Total degree (k_i): $k_i = k_i^{in}$+k_i^{out}
- Source: $k_i^{in} = 0$
- Sink: $k_i^{out} = 0$
- $\bar{k} = \frac{E}{N}$ y $\overline{k^{in}} = \overline{k^{out}}$

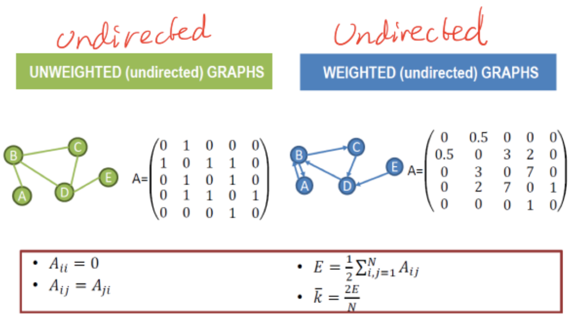
- Degree is the simplest metric, but it provides information about the "importance" of a node.

Figure 6: Node degree

Node degree will be mentioned later in this document, so don't worry about this illustration! Example from class: newspaper is like a source. Undirected has no arrows.

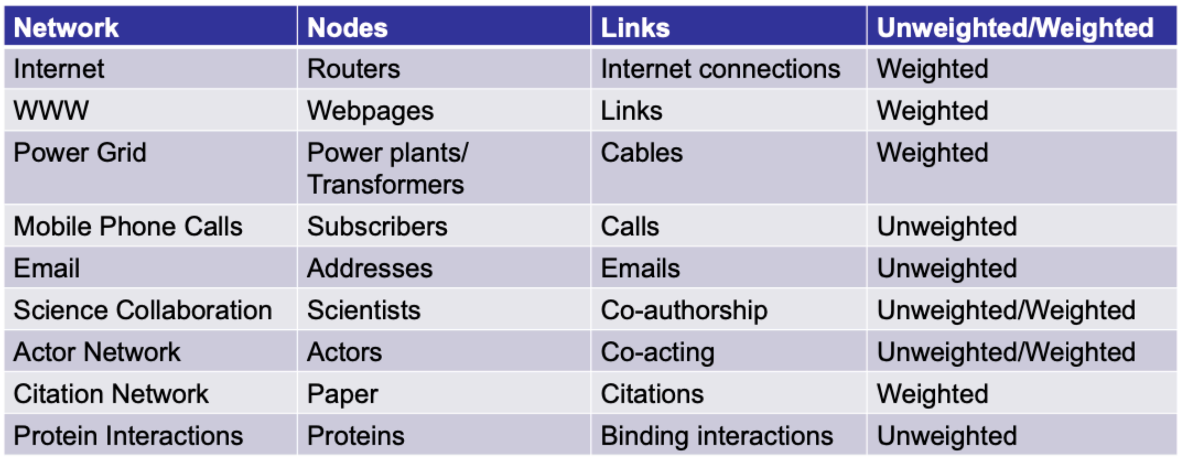
Unweighted and weighted graphs

Don’t look at the blue graph, it is wrong. It should have had lines wider than others and no arrows. Why? Let's say 3 people A, B, and C form a network, if A talks to B more than to C its link with B is wider, d that is what the weighted edges/links are. Can it be said in other words? Sure, let’s just say that it defines the most important connections.



Undirected
UNWEIGHTED (undirected) GRAPHS
Undirected
WEIGHTED (undirected) GRAPHS
A B C D E
$A=\begin{pmatrix}0&1&0&0&0\\1&0&1&1&0\\0&1&0&1&0\\0&1&1&0&1\\0&0&0&1&0\end{pmatrix}$
$A=\begin{pmatrix}0&0.5&0&0&0\\0.5&0&3&2&0\\0&3&0&7&0\\0&2&7&0&1\\0&0&0&1&0\end{pmatrix}$
$A_{ii}=0$
$A_{ij}=A_{ji}$
$E=\frac{1}{2}\sum_{i,j=1}^{N}A_{ij}$
$\bar{k}=\frac{2E}{N}$

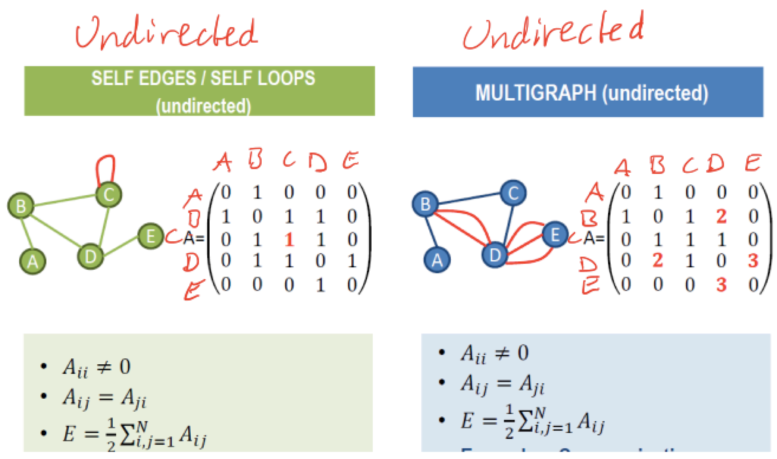
Figure 7: Illustrated by undirected graphs (blue drawing is incorrect)



Network	Nodes	Links	Unweighted/Weighted
Internet	Routers	Internet connections	Weighted
WWW	Webpages	Links	Weighted
Power Grid	Power plants/ Transformers	Cables	Weighted
Mobile Phone Calls	Subscribers	Calls	Unweighted
Email	Addresses	Emails	Unweighted
Science Collaboration	Scientists	Co-authorship	Unweighted/Weighted
Actor Network	Actors	Co-acting	Unweighted/Weighted
Citation Network	Paper	Citations	Weighted
Protein Interactions	Proteins	Binding interactions	Unweighted

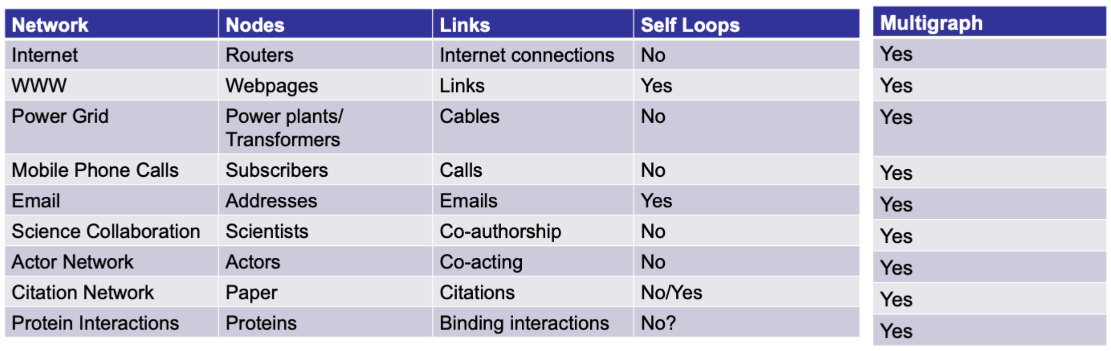
Figure 8: Is it weighted or unweighted?

Loops and Multigraphs



Undirected
SELF EDGES / SELF LOOPS (undirected)
Undirected
MULTIGRAPH (undirected)
A B C D E
$A=\begin{pmatrix}0&1&0&0&0\\1&0&1&1&0\\0&1&1&1&0\\0&1&1&0&1\\0&0&0&1&0\end{pmatrix}$
$A=\begin{pmatrix}0&1&0&0&0\\1&0&1&2&0\\0&1&1&1&0\\0&2&1&0&3\\0&0&0&3&0\end{pmatrix}$
$A_{ii}\neq 0$
$A_{ij}=A_{ji}$
$E=\frac{1}{2}\sum_{i,j=1}^{N}A_{ij}$
$A_{ii}\neq 0$
$A_{ij}=A_{ji}$
$E=\frac{1}{2}\sum_{i,j=1}^{N}A_{ij}$

Figure 8: Self loops and multigraph



Network	Nodes	Links	Self Loops	Multigraph
Internet	Routers	Internet connections	No	Yes
WWW	Webpages	Links	Yes	Yes
Power Grid	Power plants/ Transformers	Cables	No	Yes
Mobile Phone Calls	Subscribers	Calls	No	Yes
Email	Addresses	Emails	Yes	Yes
Science Collaboration	Scientists	Co-authorship	No	Yes
Actor Network	Actors	Co-acting	No	Yes
Citation Network	Paper	Citations	No/Yes	Yes
Protein Interactions	Proteins	Binding interactions	No?	Yes

Figure 10: Is it self-loop or multigraph?

Connectivity

Connected (undirected) graph: any two vertices can be joined by a path. A disconnected graph is made up by two or more connected components. When we talk about connectivity, we get two new special functions in a network called bridge edge and articulation node. **Bridge edge** are basically a bridge that if erased, the graph becomes disconnected with two separate components. An **articulation node** if erased disconnects the graph as well.

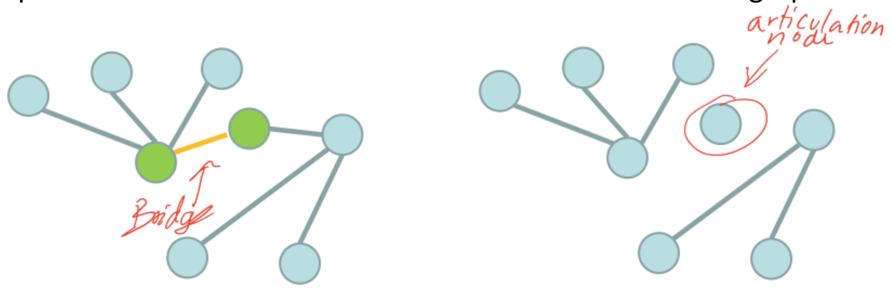
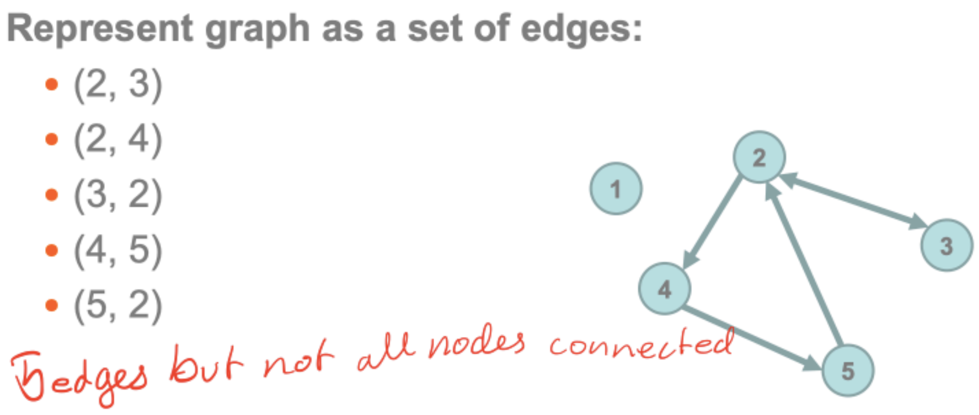


Figure 11: Bridge edge and articulation node

Connectivity for directed graphs (Important for exam):

- **Strongly connected directed graph**: Has a path from each node to every node and vice-versa. I.e. Path from A to B, and B to A.
- **Weakly connected directed graph**: Is connected if we disregard the edge direction. I.e. Path from A to B, B to A, but only B to C, A to C.
- **Strongly connected components (SCCs):** can be identified, but not every node is part of a nontrivial strongly connected component. Not every node can go back.

If a graph is strongly connected that means that all nodes have a path to each node. In other words, they are all reachable from each other. A strongly connected component is a part of the graph where all these have a path to each other, but where the whole graph is not. Lets say the network with nodes 1, 2, 3, 4, 5 and 6 is directed. They are connected like this: 1,2 2,3 3,4 4,1 and 2,5 5,6. In this example 1,2,3 and 4 are connected in a circle and forms a strongly connected component. 2,5 and 6 is also connected, but they are not strongly connected like the rest of the network.



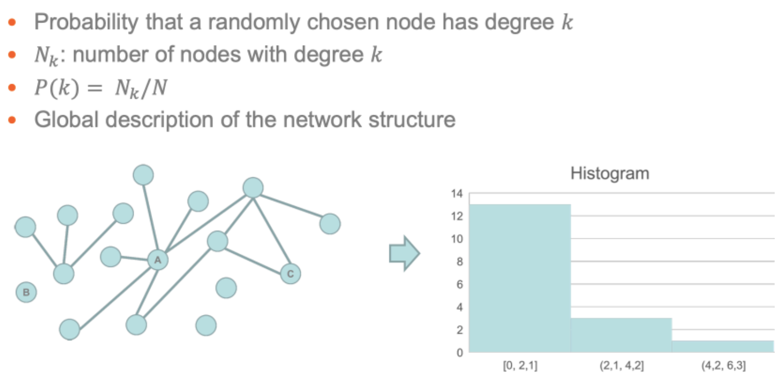
Represent graph as a set of edges:

- (2, 3)
- (2, 4)
- (3, 2)
- (4, 5)
- (5, 2)

Edges but not all nodes connected

Figure 12: Graph as a set of edges

(In-out) Degree Distribution *P(k)*



- Probability that a randomly chosen node has degree k
- N_k: number of nodes with degree k
- $P(k) = N_k/N$
- Global description of the network structure

Figure 13: Degree distribution

Diameter and Average Path Length

Distance: A-B-C. A is two connections to C

Diameter: longest shortest-path distance. Why this confusing name? Well, it can't do loops or one thing, and the diameter is basically the longest path of the graph, but not containing loops or going back and forth between nodes.

Average path length: the average path length of the network.

Dijkstra's Algorithm – not gone through in class

1. All nodes are initially **unvisited**. From the unvisited set of nodes, **the one that has the minimum shortest path length is selected**. We denote this node as smallest in the algorithm.
2. For this node, **we check all its neighbors that are still unvisited**. For each unvisited neighbor, we check if its current distance can be improved by considering the shortest path that goes through smallest. This can be performed by comparing its current shortest path length (distance(neighbor)) to the path length that goes through smallest (distance(smallest)+w(smallest, neighbor)).
3. **If the current shortest path can be improved, the path and its length are updated**. The paths are saved based on predecessors in the path sequence. Since, we only need the predecessor to reconstruct a path recursively.

4. **A node is marked as visited after all its neighbors are processed and is no longer changed.**

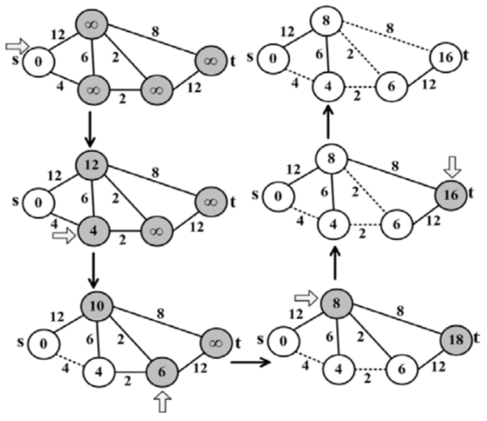


Figure 14: Dijkstra's Algorithm example

How does this really work? Wasn't that VERY confusing? I think so, so thanks Mohamed for explaining this in the course discussion board!

So, the goal of the algorithm here is to find the shortest path from the starting node (in this example a) to a target node (b).

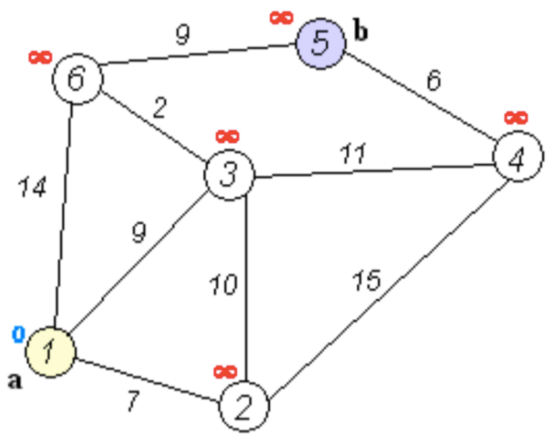


Figure 15: Illustration for the example explanation

The algorithm will work as follows:
It uses 2 lists, one to follow the nodes that the algorithm already visited and one that it hasn't. So, in every stage detailed below, it will update those 2 lists: Visited = [ ], and Unvisited [1,2,3,4,5]. And it updates this table as well:

Vertex	Shortest Distance from a (node 1)	Previous Node
1		
2		
3		
4		
5		
6		

Table 1: First setup of empty table

It starts from node 1: the distance from node 1 to node 1 is 0. And the distances from all other nodes from node 1 are unknown so for the algorithm, it is set at a very high value which is the infinity sign we see in the initial graph.

Vertex	Shortest Distance from a (node 1)	Previous Node
1	0	
2	∞	
3	∞	
4	∞	
5	∞	
6	∞	

Table 2: Shortest distance from node 1 to node 1

From there, the algorithm can begin visiting the unvisited nodes with the smallest known distance from the start vertex.

Examine the unvisited neighbors of node 1: the unvisited neighbors are here nodes 2,3 and 6 (the ones sharing edges with node 1). So, the distance from node 1 is calculated. The distance from node 1 to node 3 is 0+9=9. For node 2 it is 0+1=1 and for node 6 it is 0+14= 14.

If the calculated distance of a node is less than the known distance, we update the shortest distance in the table. And also, we can update the Previous Node for the distances we updated. Here we visited nodes 2, 3, and 6 via node 1:

Vertex	Shortest Distance from a (node 1)	Previous Node
1	0	
2	1	1
3	9	1
4	∞	
5	∞	
6	14	1

Table 3: Shortest distance from N1 to N2, N3 and N6

And we can update the initial list: Visited [N1], Unvisited: [N2, N3, N4, N5, N6]. Node 1 will not be revisited.

Now the algorithm will repeat the process: visiting the nodes with the smallest known distance from the start vertex. So, from the table we can see that this is Node 2.

Examine the unvisited neighbors of node 2: we look at its unvisited neighbors, which are nodes 3 and 4. And we calculate the distance of each neighbor from the starting node and the distance from node 4 to node 1 is 15+1= 16. And from node 1 to node 3 passing by node 2 is 10+1= 11.

Now, if the calculated distance of a node is less than the known distance, we update the shortest distance. The shortest known distance from node 1 to node 3 in the table is 9. Here we got a distance of 11 so we do not update this one. But we will update the distance from node 1 to node 4 to 16. And we also update the previous node:

Vertex	Shortest Distance from a (node 1)	Previous Node
1	0	
2	1	1
3	9	2
4	16	
5	∞	
6	14	2

Table 3: Shortest distance

We can update the initial list: Visited [N1, N2], Unvisited: [N3, N4, N5, N6]. Node 1 and node 2 will not be revisited.

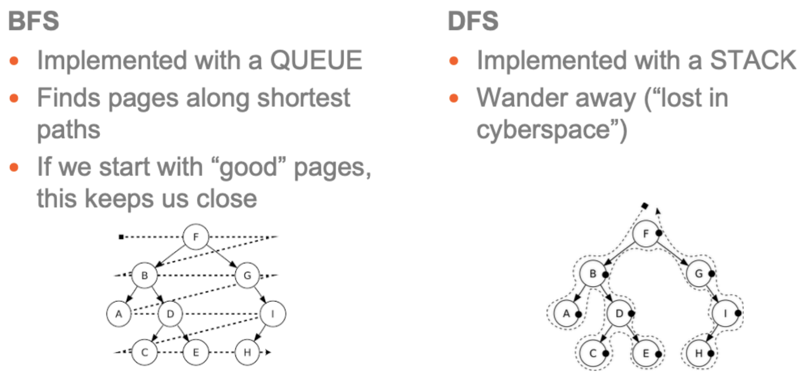
This process is then repeated over and over until the algorithm reaches the target node (b) in Node 5 and finds the shortest path to be: Node 1 -> Node 3 -> Node 6 -> Node 5 and the distance would be 20.

Breadth-First Search and Depth-First Search

BFS is typically used when you would like to find the shortest path from a certain source node to the specified destination node. Whilst DFS is typically used to exhaust all possibilities, then we check which path is the best/count all the possible ways. Essentially BFS won't get trapped exploring a useless path forever (if there is a solution amongst many solutions, it find the minimal one) whilst DFS might find useless paths.

It will not always work to find the shortest path in any kind of graph because the shortest path using BFS is calculated edgewise. So, for an unequally weighted graph, going from the same source to sink for two different paths may not necessarily provide the shortest path. This is because the BFS will find all paths that are an edge away. Thereafter followed by paths that are 2 edges away etc.

Example: Imagine source, A and sink Z with two different paths.
Path 1: A->B->C->D->Z with all edges weighted to 1.
Path 2: A->G->Z with (A,G) weighted 10 and (G.Z) weighted 1. BFS will incorrectly identify path 2 as the shortest path because pf the number of edges and not considering the weight.



BFS
- Implemented with a QUEUE
- Finds pages along shortest paths
- If we start with "good" pages, this keeps us close

DFS
- Implemented with a STACK
- Wander away ("lost in cyberspace")

Figure 16: BFG and DFS

Session 3 - Centrality

Centrality defines how important a node is within a network. It depends on the definition of the network.

Different metrics:

- Degree Centrality
- Eigenvector Centrality
- Betweenness Centrality
- Closeness Centrality
- Katz Centrality
- PageRank
- Group Centrality
- +++

Degree Centrality – Do you have many connections?

We had a look at this in the previous session, but this time we dive deeper into the topic. Central nodes are those that have mot links with other nodes.

Degree tells you how many is going in and out to that specific vertex in question, that's why you have the mode ="in", "out", or "all/total".

Undirected graph: In an undirected graph, the degree centrality of a node is simply the node degree (number of edges) of the node, denoted by d(i), normalized with the maximum degree, n-1. The value of this measure ranges between 0 and 1 as n-1 is the maximum value of d(i).

Directed graph: In this case, we need to distinguish in-links and out-links. The degree centrality is defined based on only the out-degree. It is important to know if we are measuring the in-degree or out-degree.

Eigenvector Centrality – Do you have many connections to important people?

Eigenvector centrality generalizes degree centrality by incorporating the importance of the neighbors. In other words, ec tries to measure how important the other nodes are for the centrality.

Closeness Centrality – Do you have many interactions?

A node is central if it can easily interact with all other nodes, that is the distance to all other nodes are short. The value here also ranges between 0 and 1, and is only meaningful for a connected graph. The difference for directed and undirected is only that the directed graph needs to consider the direction of links to edges, while undirected does not need to take this into consideration.

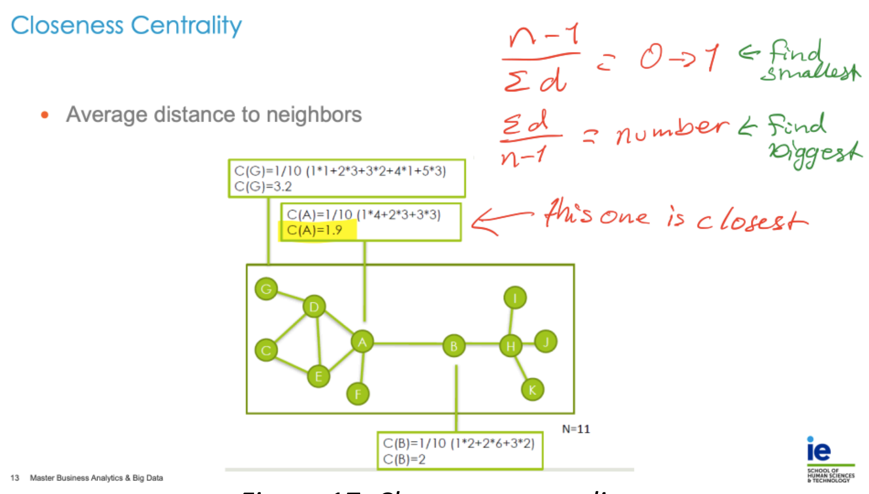

Closeness Centrality
Average distance to neighbors
C(G)=1/10 (1*1+2*3+3*2+4*1+5*3)
C(G)=3.2
C(A)=1/10 (1*4+2*3+3*3)
C(A)=1.9
C(B)=1/10 (1*2+2*6+3*2)
C(B)=2
N=11
$\frac{n-1}{\Sigma d} = 0 \to 1$ ← find smallest
$\frac{\Sigma d}{n-1}$ = number ← find biggest
← this one is closest
13 Master Business Analytics & Big Data
ie SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

Figure 17: Closeness centrality

Betweenness Centrality – Do you help to connect different parts of the network?

When it comes to betweenness it is again to find an important vertex, and the betweenness is the number of shortest paths that goes through a vertex. In other words, central nodes that have “control” of other pairs of nodes.

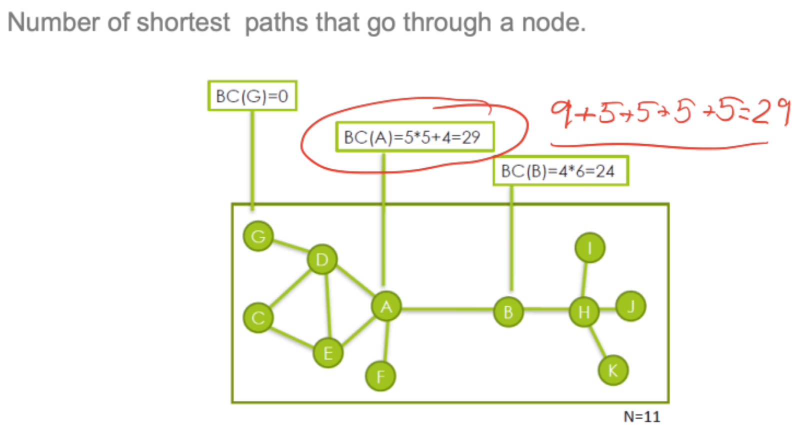


Figure 18: Betweenness centrality

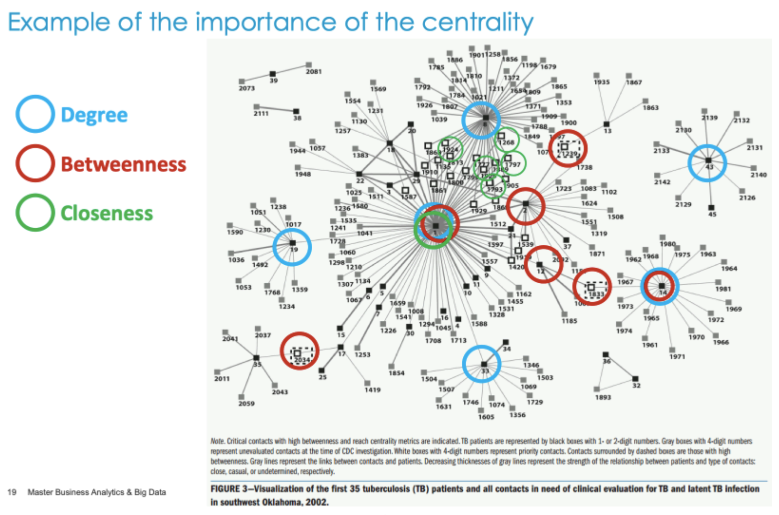

Example of the importance of the centrality
Degree
Betweenness
Closeness
19 Master Business Analytics & Big Data
FIGURE 3—Visualization of the first 35 tuberculosis (TB) patients and all contacts in need of clinical evaluation for TB and latent TB infection in southwest Oklahoma, 2002.

Figure 19: Example of centrality importance

PageRank Centrality – Do you have many interactions with important people?

Google offered the algorithm to Yahoo for one million dollars in 1997, but Yahoo refused it. In 2002, Yahoo tried to buy PageRank for 3.000 million dollars, but Google refused it.

“A page is important if it is pointed to by other important pages”

Session 4 – Transitivity and Reciprocity

Often we need to observe the linking behavior. Linking behavior determines how links (edges) are formed in a social graph. There is two well-known ways of analyzing linking behavior. What can you say generally about linking and job search? It can be smart to contact someone a bit further away than your close friends to get a job.

Transitivity

"when a friend of mine has a friend that is also my friend"

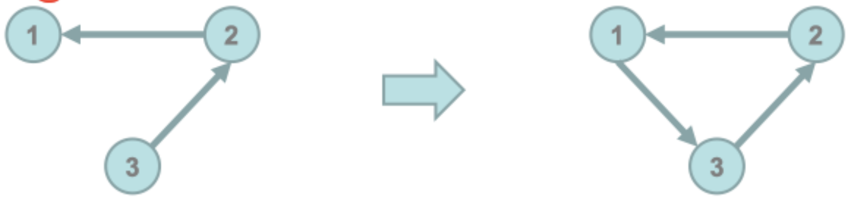


Figure ..: Friends of friends circle

Clustering coefficient: A path of length two uvw is closed if u and w are connected.

$$C = \frac{\text{number of closed paths of length } 2}{\text{number of paths of length } 2}$$

Equivalently:

$$C = \frac{\text{number of triangles x } 3}{\text{number of connected triples of vertices}}$$

Figure ..: Cluster coefficient formula

A complete graph is transitive, as for example this one:

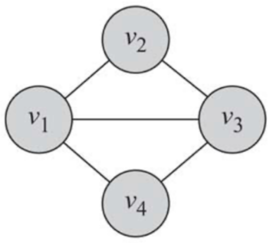


Figure ..: Complete transivity

The clustering coefficient analyzes transitivity in an undirected graph. **The local clustering coefficient**, *cl(v),* gives for node *v* the proportion of neighbors of *v* that are also connected to each other.

$$C_i = \frac{\text{number of pairs of neighbors of } i \text{ that are connected}}{\text{number of pairs of neighbours of } i}$$

Global clustering:

$$C_{WS} = \frac{1}{N}\sum_{i=1}^{N} C_i$$

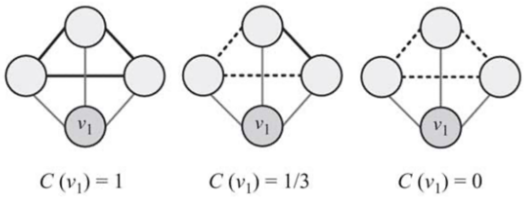


Figure ..: Clustering coefficient with example

Differences

- Given a network, C and C_{WS} **can differ substantially**.
- C_{WS} has been used very often for **historical reasons** (C_{WS} was proposed first).
- C can be dominated by the contribution of vertices of **high degree** (which have many adjancent nodes).
- C_{WS} is can be dominated by the contribution of vertices of **low degree** (which are many in the majority of networks).
- C_{WS} needs taking further decision on C_i when $k_i < 2$ (C is more elegant from a mathematical point of view).

Reciprocity

Reciprocity is a simplified version of transitivity, because it considers closed loops of length 2, which can only happen in directed graphs. Formally, if node v is connected to node u, u by connecting to v exhibits reciprocity. Any directed graph can have a maximum of |E|/2 pairs. Thus, this value can be used as a normalization factor.

Reciprocity can be computed using the adjacency matrix A:

$$R = \sum_{i,j\ i<j} A_{i,j}A_{j,i}$$

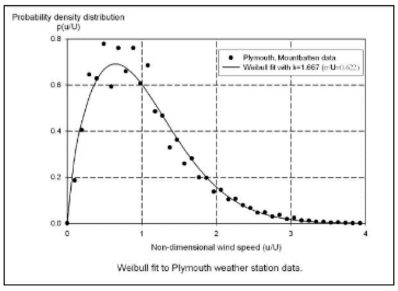
$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

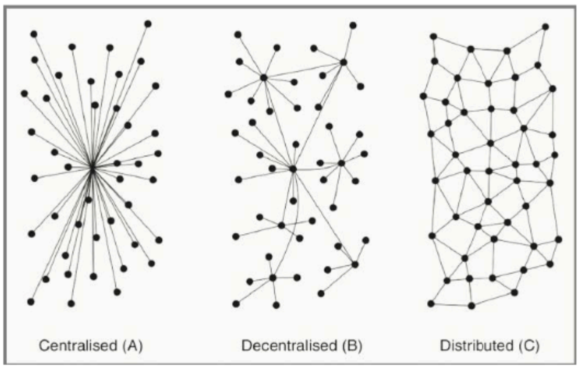
Figure ..: Formula and example of reciprocity

Session 5 – Real World Networks

Two different approaches:

- Identify one or more microscopic mechanisms driving the formation of the network and using them to define a dynamic model.
- Identify a set of characteristic static properties of real systems and then building networks that have the same properties but are otherwise maximally random.

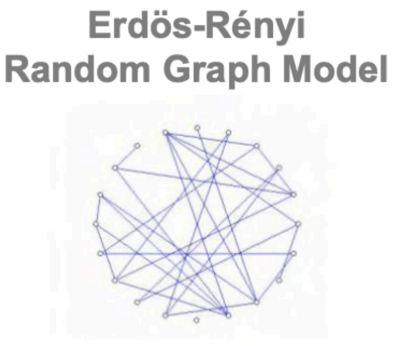


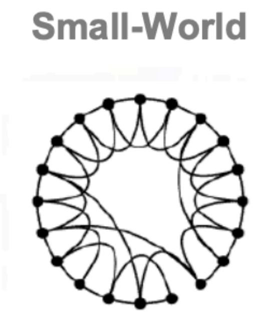


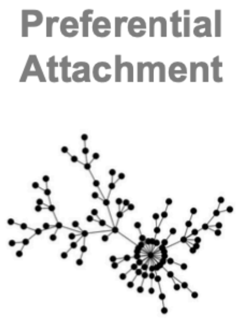
Why? It is a simple representation of complex network.

- We try to reproduce some properties of the network (degree distribution, average path, clustering coefficient, size of the giant component, etc.).
- Static versus dynamic models.
- It is not so simple given the complexity of the data. However, some models are good approximations of certain properties of the network and can be used as a "baseline".
  - We can derive mathematical properties.
  - We can simulate networks that are "similar" to the one of interest.
  - We can predict properties and outcomes (find trends and structural patterns, predict the behavior of the network to certain processes, etc.).

Models







Erdös-Rényi: Random Graph Model

Basic G(n,p) Erdös-Rényi random graph model: probability – that two nodes are connected.

- parameter n is the number of vertices.
- parameter p is s.t. $0 \leq p \leq 1$.

- Generate and edge (i ; j) independently at random with probability p.

n and p do not uniquely determine the graph!
The graph is a result of a random process.
We can have many different realizations given the same n and p.

Degree distribution of $\boldsymbol{Gnp}$ is binomial
Mean: $\boldsymbol{np}$
It can be approximated by a Gaussian distribution for N large.
Absence of hubs(nodes that have extremely high connectivity or degree).
Emergence of giant component.
Average shortest path. Difficult to derive exact shortest path.

- How does the shortest path scale with the number of nodes? $O(\log n)$ ↘Small diameter.

Low clustering coefficient.

- Not realistic in general. “Friends of my friends are my friends”.

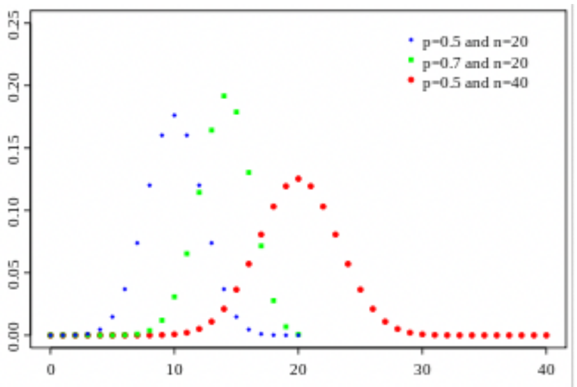


Figure..:

Are real networks like random graphs? It is not easy to have high clustering coefficient unless probability is high for random graphs. 2 million node graphs use 10 %, but how do they choose 10 %? Products in Amazon – choose only music instruments etc. or it will be created in a random way.

The Small-World Experiment - Six Degrees of Separation

[Milgram, Stanley. "The small world problem." Psychology today2.1 (1967): 60-67]. They picked 300 people in Omaha, Nebraska and Wichita, Kansas and ask them to get a letter to a stockbroker in Boston by passing it through friends. Of all 300 letters, 64 chains were completed: (64 letters reached the target). It took 6.2 steps on the average. Random networks are not like real life. We need high clustering and small distances between most of the nodes.

Could a network with high clustering be at the same time a small world? Yes! You don’t need more than a few random links

The Watts Strogatz Model provides insight on the interplay between clustering and the small-world, captures the structure of many realistic networks, accounts for the high clustering of real networks and does not lead to the correct degree distribution.

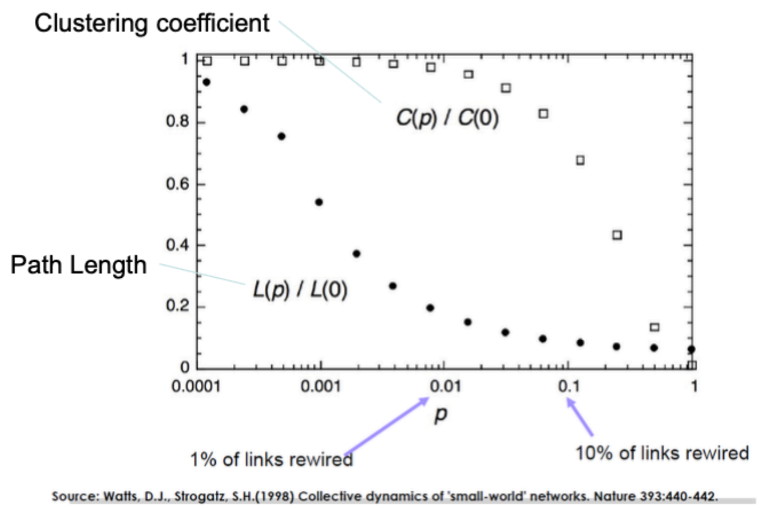


Figure ..:

Preferential Attachment

Goal: To create a model with power-law degree distribution.
Barabási and Albert and Price proposed a generative mechanism (preferential attachment) to explain the appearance of power-law distributions. It is based on Herbet Simon's result from power-laws arise from "Rich get richer" (cumulative advantage).

- From an initial graph with few connections, add a new node, create m out-links.
- The probability of linking a node is proportional to its degree.