Economic Networks

Theory and Empirics

Giorgio Fagiolo
Laboratory of Economics and Management (LEM)
Sant'Anna School of Advanced Studies, Pisa, Italy

http://www.lem.sssup.it/fagiolo/
giorgio.fagiolo@sssup.it

Lecture 3

This Lecture

- What is a network? Examples of networks
- Why networks are important for economists?
- Networks and graphs
- Measures and metrics on networks
- Distributions of metrics and measures in large networks
- Models of network formation
- Null statistical network models
- Economic applications

Why Network Statistics?

- Visualization may be useful but often is not enough
  - ✓ Large (many nodes) and dense networks (many links)
  - ✓ Are two networks similar or different?

Why Network Statistics?

- Visualization may be useful but often is not enough
  - ✓ Large (many nodes) and dense networks (many links)
  - ✓ Are two networks similar or different?
- Goal: Characterize networks by means of a set of statistics that capture graph-theoretic properties (topology)
  - ✓ Network-wide indicators: one value attached to the network
  - ✓ Node-specific indicators: one value attached to any single node
  - ✓ Link-specific indicators: one value attached to any single link

Why Network Statistics?

- Visualization may be useful but often is not enough
  - ✓ Large (many nodes) and dense networks (many links)
  - ✓ Are two networks similar or different?
- Goal: Characterize networks by means of a set of statistics that capture graph-theoretic properties (topology)
  - ✓ Network-wide indicators: one value attached to the network
  - ✓ Node-specific indicators: one value attached to any single node
  - ✓ Link-specific indicators: one value attached to any single link
- Main problems:
  - ✓ How can one tell whether a value of a statistic computed on a given network is large or small?
  - ✓ Comparing statistics across different networks or time snapshots

Density

- Network density: fraction of existing links (L) over all possible links

$$d = \frac{2L}{N(N-1)} = \frac{2\sum_{i>j} a_{ij}}{N(N-1)}$$

Undirected

$$d = \frac{L}{N(N-1)} = \frac{\sum_{i,j} a_{ij}}{N(N-1)}$$

Directed

Density

- Network density: fraction of existing links (L) over all possible links

$$d = \frac{2L}{N(N-1)} = \frac{2\sum_{i>j} a_{ij}}{N(N-1)}$$

Undirected

$$d = \frac{L}{N(N-1)} = \frac{\sum_{i,j} a_{ij}}{N(N-1)}$$

Directed

- Densities range from 0 (empty graph) to 1 (complete graph)

Density

- Network density: fraction of existing links (L) over all possible links

$$d = \frac{2L}{N(N-1)} = \frac{2\sum_{i>j} a_{ij}}{N(N-1)}$$

Undirected

$$d = \frac{L}{N(N-1)} = \frac{\sum_{i,j} a_{ij}}{N(N-1)}$$

Directed

- Densities range from 0 (empty graph) to 1 (complete graph)

- Bilateral density in a BDN: fraction of reciprocated links (L)

$$r = \frac{tr(A^2)}{N(N-1)} = \frac{\sum_{i,j} a_{ij}a_{ji}}{N(N-1)}$$

Components: Number and Size Distribution

Components: Number and Size Distribution

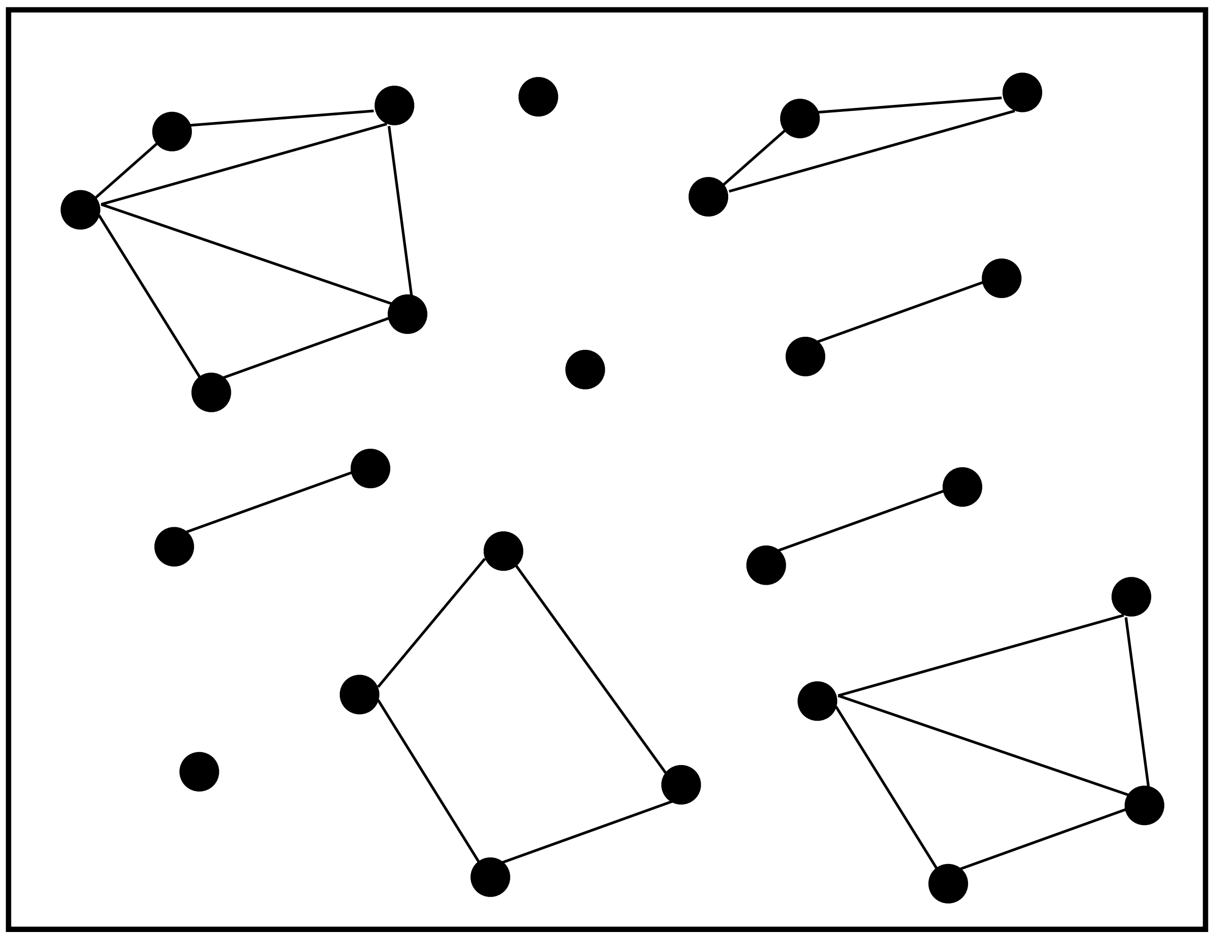
- Number of connected components in the graph

Components: Number and Size Distribution

- Number of connected components in the graph
- Size distribution of connected components

Components: Number and Size Distribution

- Number of connected components in the graph
- Size distribution of connected components

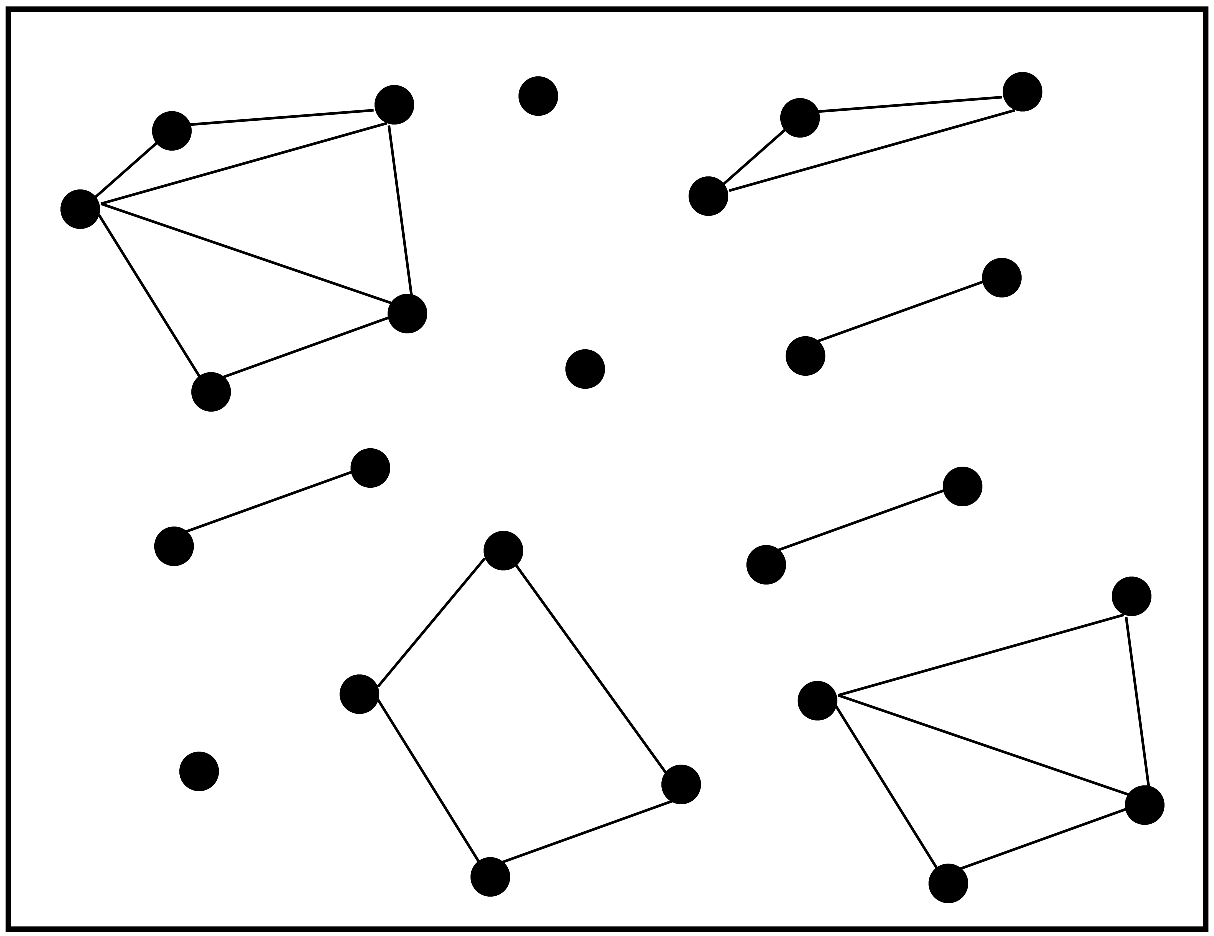


10 components

s(1)=3
s(2)=3
s(3)=1
s(4)=2
s(5)=1

Components: Number and Size Distribution

- Number of connected components in the graph
- Size distribution of connected components



10 components

s(1)=3
s(2)=3
s(3)=1
s(4)=2
s(5)=1

- Extensions to digraphs: weakly and strongly connected components

Diameter, Distances, and Length

- Path length matrix: A symmetric NxN matrix L whose generic element l(i,j) is the path length (length of geodesic path) between i and j

Diameter, Distances, and Length

- Path length matrix: A symmetric NxN matrix L whose generic element l(i,j) is the path length (length of geodesic path) between i and j

- Diameter of a graph (D): length of the longest geodesic path between any pair of nodes, i.e. max among all l(i,j)

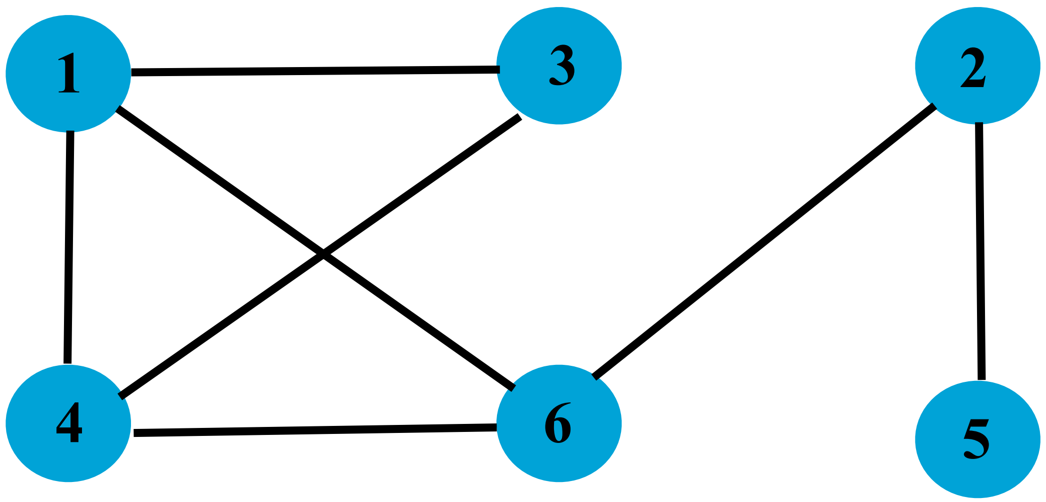
Diameter, Distances, and Length

- Path length matrix: A symmetric NxN matrix L whose generic element l(i,j) is the path length (length of geodesic path) between i and j

- Diameter of a graph (D): length of the longest geodesic path between any pair of nodes, i.e. max among all l(i,j)

A=

	1	2	3	4	5	6
1	0	0	1	1	0	1
2	0	0	0	0	1	1
3	1	0	0	1	0	0
4	1	0	1	0	0	1
5	0	1	0	0	0	0
6	1	1	0	1	0	0



L=

	1	2	3	4	5	6
1	0	2	1	1	3	1
2	2	0	3	2	1	1
3	1	3	0	1	4	2
4	1	2	1	0	3	1
5	3	1	4	3	0	2
6	1	1	2	1	2	0

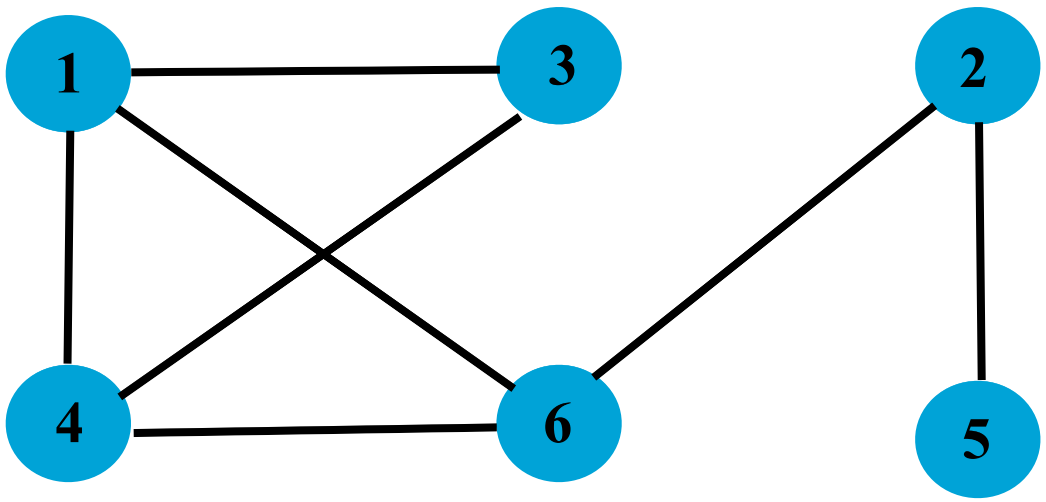
D=4

Diameter, Distances, and Length

- Path length matrix: A symmetric NxN matrix L whose generic element l(i,j) is the path length (length of geodesic path) between i and j

- Diameter of a graph (D): length of the longest geodesic path between any pair of nodes, i.e. max among all l(i,j)

$$A = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 1 & 1 \\ 3 & 1 & 0 & 0 & 1 & 0 & 0 \\ 4 & 1 & 0 & 1 & 0 & 0 & 1 \\ 5 & 0 & 1 & 0 & 0 & 0 & 0 \\ 6 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}$$



$$L = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 2 & 1 & 1 & 3 & 1 \\ 2 & 2 & 0 & 3 & 2 & 1 & 1 \\ 3 & 1 & 3 & 0 & 1 & 4 & 2 \\ 4 & 1 & 2 & 1 & 0 & 3 & 1 \\ 5 & 3 & 1 & 4 & 3 & 0 & 2 \\ 6 & 1 & 1 & 2 & 1 & 2 & 0 \end{array}$$

D=4

- Extensions of path length to WUN/WDN do exist

Average Path Length

- Given a path length matrix L, we can compute the average node path length simply as:

$$ANPL_i = \frac{\sum_{j \in J_i} l_{ij}}{|J_i|} \qquad J_i = \{j = 1, \ldots, N, j \neq i : l_{ij} < \infty\}$$

Average Path Length

- Given a path length matrix L, we can compute the average node path length simply as:

$$ANPL_i = \frac{\sum_{j \in J_i} l_{ij}}{|J_i|} \qquad J_i = \{j = 1, \ldots, N, j \neq i : l_{ij} < \infty\}$$

- The average path length of the graph is the average of all $ANPL_i$

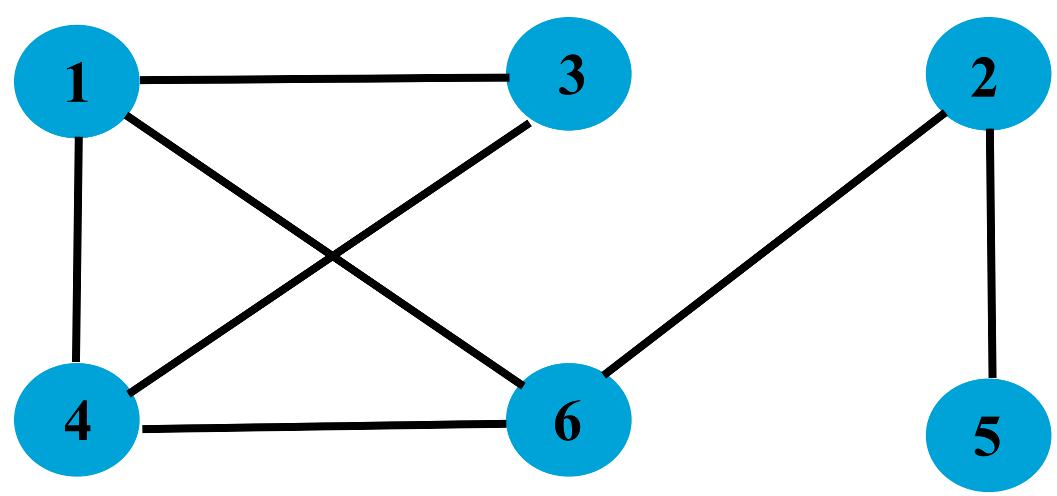
Average Path Length

- Given a path length matrix L, we can compute the average node path length simply as:

$$ANPL_i = \frac{\sum_{j \in J_i} l_{ij}}{|J_i|} \qquad J_i = \{j = 1, \ldots, N, j \neq i : l_{ij} < \infty\}$$

- The average path length of the graph is the average of all ANPL_i

$$A= \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 1 & 1 \\ 3 & 1 & 0 & 0 & 1 & 0 & 0 \\ 4 & 1 & 0 & 1 & 0 & 0 & 1 \\ 5 & 0 & 1 & 0 & 0 & 0 & 0 \\ 6 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}$$



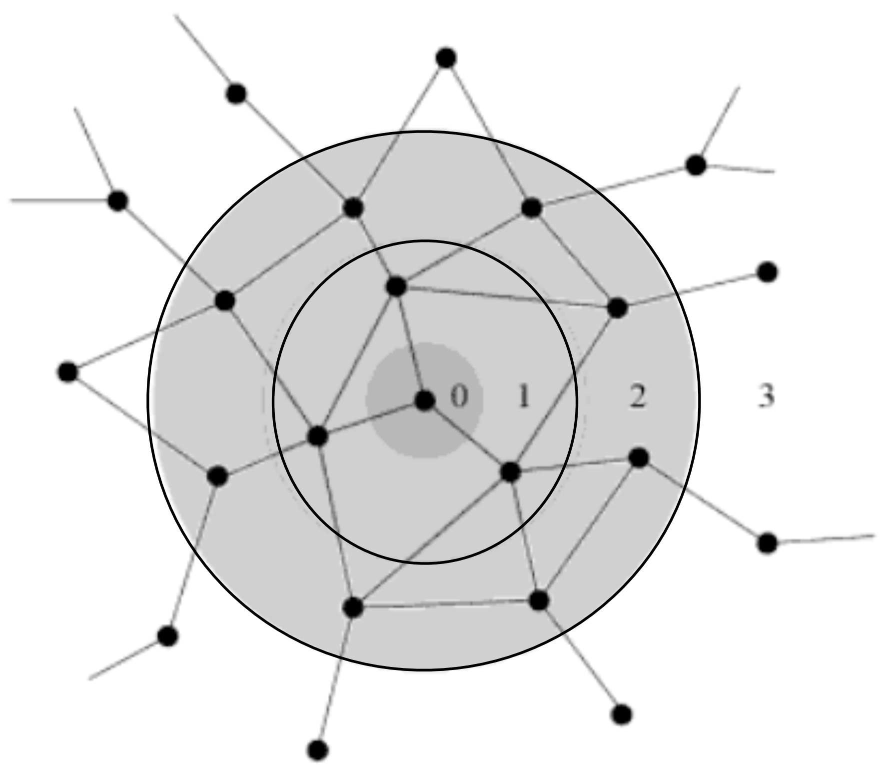
$$L= \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 2 & 1 & 1 & 3 & 1 \\ 2 & 2 & 0 & 3 & 2 & 1 & 1 \\ 3 & 1 & 3 & 0 & 1 & 4 & 2 \\ 4 & 1 & 2 & 1 & 0 & 3 & 1 \\ 5 & 3 & 1 & 4 & 3 & 0 & 2 \\ 6 & 1 & 1 & 2 & 1 & 2 & 0 \end{array}$$

ANPL(1)=8/5
ANPL(2)=9/5
ANPL(3)=11/5
ANPL(4)=8/5
ANPL(5)=13/5
ANPL(6)=7/5

APL=56/(6*5)=28/15≅1.8667

Shortest Paths and Breadth-First Search (1)

- How can we find shortest paths and components?
  - ✓ Naive implementation of a simple algorithm (breadth-first search, BFS)
  - ✓ More sophisticated implementations and algorithms are possible
- BFS: finds shortest distance from a given starting node s to every other node in the same component as s
  - ✓ We know s has d=0 from itself
  - ✓ Find all neighbors of s: they have distance 1 from s
  - ✓ Find all neighbors of neighbors of s excluding those we have already visited: they are distance=2 from s
  - ✓ ... Go on with the cycle by growing on each the set of visited node by one step



Shortest Paths and Breadth-First Search (2)

Shortest Paths and Breadth-First Search (2)

<u>Implementation</u>

Shortest Paths and Breadth-First Search (2)

<u>Implementation</u>

- Create an Nx1 array **x** to store the distance of each vertex from *s*

Shortest Paths and Breadth-First Search (2)

<u>Implementation</u>

- Create an Nx1 array **x** to store the distance of each vertex from *s*
- Set distance of *s* from itself to 0 (all other distances can be set to NaN)

Shortest Paths and Breadth-First Search (2)

Implementation

- Create an Nx1 array **x** to store the distance of each vertex from *s*
- Set distance of *s* from itself to 0 (all other distances can be set to NaN)
- Create a distance variable **d=0** keeping track of how far we are from *s*

Shortest Paths and Breadth-First Search (2)

Implementation

- Create an Nx1 array **x** to store the distance of each vertex from *s*
- Set distance of *s* from itself to 0 (all other distances can be set to NaN)
- Create a distance variable **d=0** keeping track of how far we are from *s*
    1. Find all vertices at distance *d* from *s* in **x**

Shortest Paths and Breadth-First Search (2)

Implementation

- Create an Nx1 array **x** to store the distance of each vertex from *s*
- Set distance of *s* from itself to 0 (all other distances can be set to NaN)
- Create a distance variable **d=0** keeping track of how far we are from *s*
    1. Find all vertices at distance *d* from *s* in **x**
    2. Find all neighbors of those nodes and check each one to see if its distance from *s* is NaN

Shortest Paths and Breadth-First Search (2)

Implementation

- Create an Nx1 array **x** to store the distance of each vertex from *s*
- Set distance of *s* from itself to 0 (all other distances can be set to NaN)
- Create a distance variable **d=0** keeping track of how far we are from *s*
    1. Find all vertices at distance *d* from *s* in **x**
    2. Find all neighbors of those nodes and check each one to see if its distance from *s* is NaN
    3. Exit if the number of neighbors with NaN distances is zero. Otherwise, if that number is >0 then set the distance of each of those neighbors to *d+1*

Shortest Paths and Breadth-First Search (2)

<u>Implementation</u>

- Create an Nx1 array **x** to store the distance of each vertex from *s*
- Set distance of *s* from itself to 0 (all other distances can be set to NaN)
- Create a distance variable **d=0** keeping track of how far we are from *s*
    1. Find all vertices at distance *d* from *s* in **x**
    2. Find all neighbors of those nodes and check each one to see if its distance from *s* is NaN
    3. Exit if the number of neighbors with NaN distances is zero. Otherwise, if that number is >0 then set the distance of each of those neighbors to *d+1*
    4. Set *d=d+1*

Shortest Paths and Breadth-First Search (2)

Implementation

- Create an Nx1 array **x** to store the distance of each vertex from *s*
- Set distance of *s* from itself to 0 (all other distances can be set to NaN)
- Create a distance variable **d=0** keeping track of how far we are from *s*
  1. Find all vertices at distance *d* from *s* in **x**
  2. Find all neighbors of those nodes and check each one to see if its distance from *s* is NaN
  3. Exit if the number of neighbors with NaN distances is zero. Otherwise, if that number is >0 then set the distance of each of those neighbors to *d+1*
  4. Set *d=d+1*
  5. Repeat from step 1

Shortest Paths and Breadth-First Search (2)

<u>Implementation</u>

- Create an Nx1 array **x** to store the distance of each vertex from *s*
- Set distance of *s* from itself to 0 (all other distances can be set to NaN)
- Create a distance variable **d=0** keeping track of how far we are from *s*
  1. Find all vertices at distance *d* from *s* in **x**
  2. Find all neighbors of those nodes and check each one to see if its distance from *s* is NaN
  3. Exit if the number of neighbors with NaN distances is zero. Otherwise, if that number is >0 then set the distance of each of those neighbors to *d+1*
  4. Set *d=d+1*
  5. Repeat from step 1

- When the BFS algorithm stops, we get an array with distances to every node in the component of the network that contains s (every node in other components have a NaN distance)

Shortest Paths and Breadth-First Search (2)

Implementation

- Create an Nx1 array **x** to store the distance of each vertex from *s*
- Set distance of *s* from itself to 0 (all other distances can be set to NaN)
- Create a distance variable **d=0** keeping track of how far we are from *s*
  1. Find all vertices at distance *d* from *s* in **x**
  2. Find all neighbors of those nodes and check each one to see if its distance from *s* is NaN
  3. Exit if the number of neighbors with NaN distances is zero. Otherwise, if that number is >0 then set the distance of each of those neighbors to *d+1*
  4. Set *d=d+1*
  5. Repeat from step 1

- When the BFS algorithm stops, we get an array with distances to every node in the component of the network that contains s (every node in other components have a NaN distance)
- Therefore, BFS also finds the component to which s belongs, and can be employed to find all components of the graph

Node Degrees (BUN, BDN)

- Binary undirected: Node degree=number of links of a node

$$k_i = \sum_{j=1}^{N} a_{ij} = A_{(i)} \mathbf{1_N} = A_{(i)}^T \mathbf{1_N}$$

Node Degrees (BUN, BDN)

- Binary undirected: Node degree=number of links of a node

$$k_i = \sum_{j=1}^{N} a_{ij} = A_{(i)}\mathbf{1_N} = A^T_{(i)}\mathbf{1_N}$$

- Binary directed:

  ✓ Node in-degree=number of incoming links of a node

  ✓ Node out-degree=number of outcoming links of a node

  ✓ Node total degree=Node in-degree+Node out-degree

$$k_i^{in} = \sum_{j=1}^{N} a_{ji} = A^T_{(i)}\mathbf{1_N} \qquad k_i^{out} = \sum_{j=1}^{N} a_{ij} = A_{(i)}\mathbf{1_N}$$

$$k_i^{tot} = \sum_{j=1}^{N} (a_{ij} + a_{ji}) = (A_{(i)} + A^T_{(i)})\mathbf{1_N}$$

Node Strength (WUN, WDN)

- Weighted undirected: Node strength=sum of link weights of a node

$$s_i = \sum_{j=1}^{N} w_{ij} = W_{(i)} \mathbf{1_N} = W_{(i)}^T \mathbf{1_N}$$

Node Strength (WUN, WDN)

- Weighted undirected: Node strength=sum of link weights of a node

$$s_i = \sum_{j=1}^{N} w_{ij} = W_{(i)}\mathbf{1_N} = W_{(i)}^T\mathbf{1_N}$$

- Weighted directed:

  ✓ Node in-strength=sum of incoming link weights of a node

  ✓ Node out-strength=sum of outcoming link weights of a node

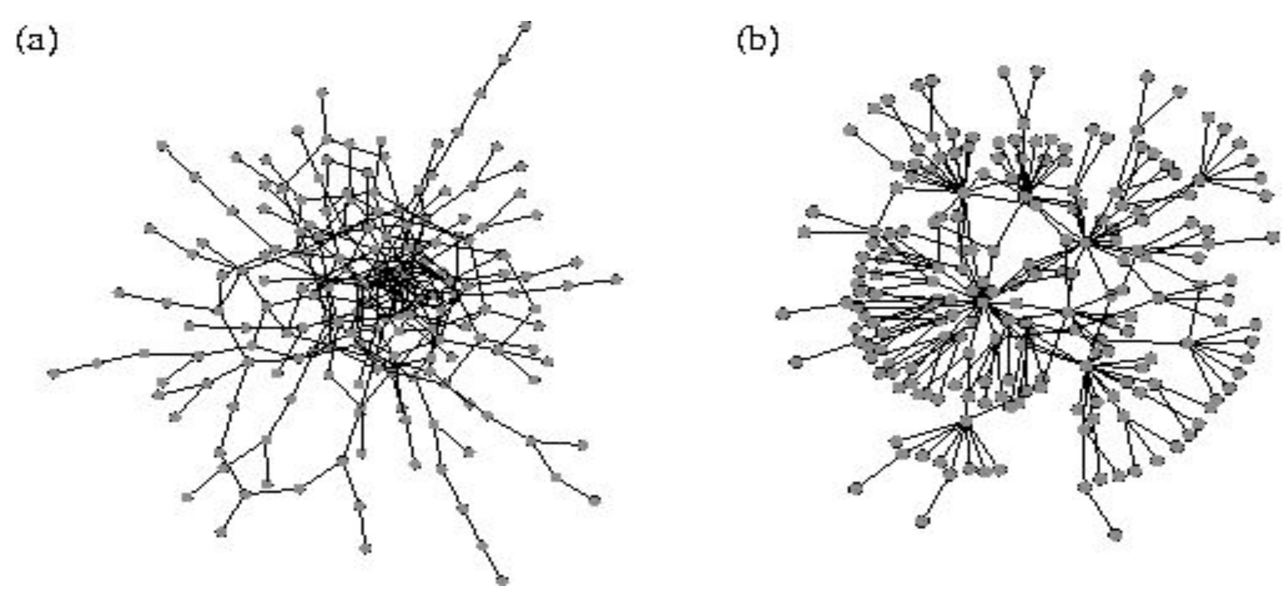
  ✓ Node total strength=Node in-strength+Node out-strength

$$s_i^{in} = \sum_{j=1}^{N} w_{ji} = W_{(i)}^T\mathbf{1_N} \qquad s_i^{out} = \sum_{j=1}^{N} w_{ij} = W_{(i)}\mathbf{1_N}$$

$$s_i^{tot} = \sum_{j=1}^{N} (w_{ij} + w_{ji}) = (W_{(i)} + W_{(i)}^T)\mathbf{1_N}$$

Homophily and Assortative Mixing (1)

- Nodes have tendency to link to other nodes with similar characteristics
  - ✓ Node-specific characteristics other than network-related (e.g. people form links in a social network if they are similar according to age, nationality, language, income, education level, etc.)
  - ✓ Node-specific network characteristics, e.g. degree or strength: Do high-degree (or strength) nodes tend to be linked to nodes that in turn have a high degree or strength (assortativity) or they end up linked to low-degree or low-strength ones (disassortativity)?
  - ✓ How can we measure assortativity or disassortativity in a network?



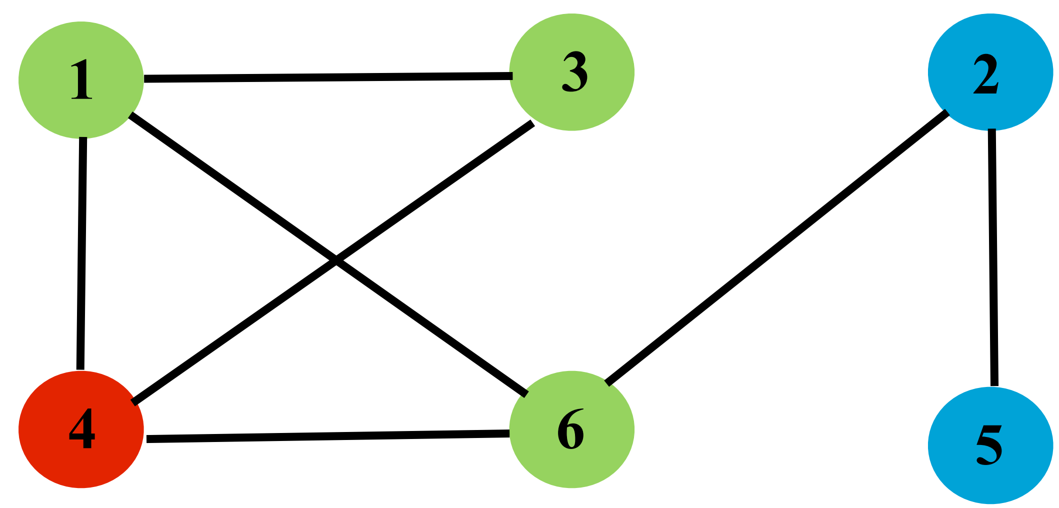
Assortative network

Disassortative network

Average Nearest-Neighbor Degree (ANND)

- Node ANND in BUNs: Average degree of a node's neighbors

$$A=\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 1 & 1 \\ 3 & 1 & 0 & 0 & 1 & 0 & 0 \\ 4 & 1 & 0 & 1 & 0 & 0 & 1 \\ 5 & 0 & 1 & 0 & 0 & 0 & 0 \\ 6 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}$$



- Node 4 has k(i)=3 and its 3 neighbors are (1,3,6)
- k(1)=3, k(3)=2, k(6)=3
- Thus ANND(4)=(3+2+3)/3=8/3

Average Nearest-Neighbor Degree/Strength

- Node ANND in BUNs: Average degree of a node's neighbors

$$ANND_i = \frac{\sum_j a_{ij} k_j}{k_i} = \frac{\sum_j \sum_h a_{ij} a_{jh}}{k_i} = \frac{A_{(i)} A \mathbf{1_N}}{A_{(i)} \mathbf{1_N}}$$

Average Nearest-Neighbor Degree/Strength

- Node ANND in BUNs: Average degree of a node's neighbors

$$ANND_i = \frac{\sum_j a_{ij} k_j}{k_i} = \frac{\sum_j \sum_h a_{ij} a_{jh}}{k_i} = \frac{A_{(i)} A \mathbf{1_N}}{A_{(i)} \mathbf{1_N}}$$

- Node ANNS in WUNs: Average strength of a node's neighbors

$$ANNS_i = \frac{\sum_j a_{ij} s_j}{k_i} = \frac{\sum_j \sum_h a_{ij} w_{jh}}{k_i} = \frac{A_{(i)} W \mathbf{1_N}}{A_{(i)} \mathbf{1_N}}$$

ANND/ANNS in Directed Networks

- Total ANND or ANNS (d stands for degree)

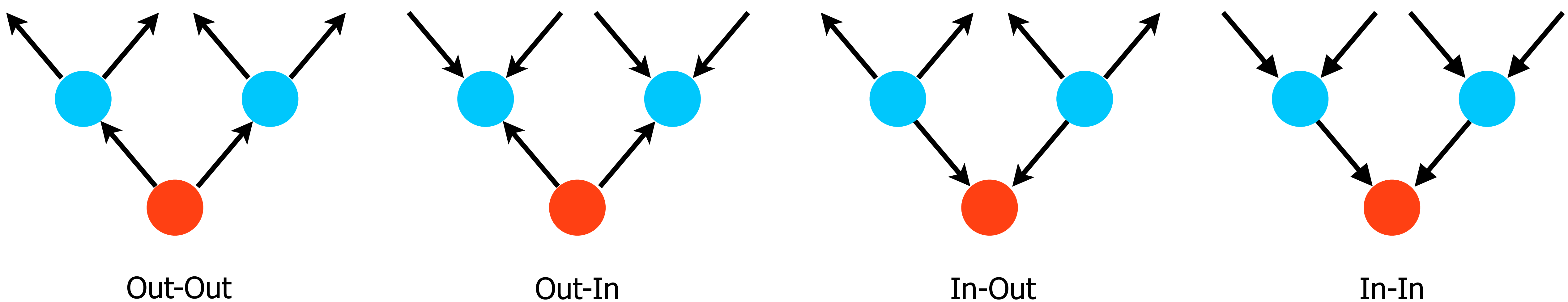
$$
\begin{aligned}
annd_i^{tot} = (d_i^{tot})^{-1} \sum_j (a_{ji} d_j^{tot} + a_{ij} d_j^{tot}) = \\
= (d_i^{tot})^{-1} \sum_j (a_{ji} + a_{ij}) d_j^{tot} = \\
= (d_i^{tot})^{-1} \sum_j \sum_h (a_{ji} + a_{ij})(a_{jh} + a_{hj}) = \\
= \frac{(A^T + A)_{(i)} (A^T + A) \mathbf{1}}{(A^T + A)_{(i)} \mathbf{1}}
\end{aligned}
$$

ANND/ANNS in Directed Networks

- Total ANND or ANNS (d stands for degree)

$$
\begin{aligned}
annd_i^{tot} &= (d_i^{tot})^{-1} \sum_j (a_{ji} d_j^{tot} + a_{ij} d_j^{tot}) = \\
&= (d_i^{tot})^{-1} \sum_j (a_{ji} + a_{ij}) d_j^{tot} = \\
&= (d_i^{tot})^{-1} \sum_j \sum_h (a_{ji} + a_{ij})(a_{jh} + a_{hj}) = \\
&= \frac{(A^T + A)_{(i)} (A^T + A)\mathbf{1}}{(A^T + A)_{(i)} \mathbf{1}}
\end{aligned}
$$

- Two additional dimensions to account for:
  - ✓ Nearest neighbors may be either in-neighbors or out-neighbors
  - ✓ Neighbors of nearest neighbors may be either in-neighbors or out-neighbors



Homophily and Assortative Mixing (2)

- How can we measure assortativity or disassortativity in a network?
  - ✓ Computing node-level **correlation coefficient between ND and ANND or NS and ANNS**: are well connected nodes linked with nodes whose neighbors are themselves well connected?

$$m = \frac{cov\{k_i, ANND_i\}}{\sigma(k_i) \cdot \sigma(ANND_i)} \in [-1, +1]$$

  - ✓ Computing link-level **degree-degree** or **strength-strength correlation coefficient**. Let x_i be a node-level statistics, i.e. degree or strength:

$$r = \frac{cov\{x_i, x_j\}}{\sigma^2(x)} = \frac{\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x)}{\sum_i \sum_j a_{ij}(x_i - \mu_x)^2}$$

Homophily and Assortative Mixing (2)

- The link-level degree-degree or strength-strength correlation coefficient can be further simplified:

Homophily and Assortative Mixing (2)

- The link-level degree-degree or strength-strength correlation coefficient can be further simplified:

$$\boxed{r = \frac{cov\{x_i, x_j\}}{\sigma^2(x)} = \frac{\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x)}{\sum_i \sum_j a_{ij}(x_i - \mu_x)^2}}$$

:

Homophily and Assortative Mixing (2)

- The link-level degree-degree or strength-strength correlation coefficient can be further simplified:

$$\boxed{r = \frac{cov\{x_i, x_j\}}{\sigma^2(x)} = \frac{\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x)}{\sum_i \sum_j a_{ij}(x_i - \mu_x)^2}} \quad :$$

$$\boxed{\mu_x = \frac{\sum_i \sum_j a_{ij} x_i}{\sum_i \sum_j a_{ij}} = \frac{\sum_i x_i \sum_j a_{ij}}{2L} = \frac{\sum_i k_i x_i}{2L}} \quad + \quad \boxed{\mu_x^2 = \frac{\sum_i \sum_j k_i k_j x_i x_j}{(2L)^2}}$$

Homophily and Assortative Mixing (2)

- The link-level degree-degree or strength-strength correlation coefficient can be further simplified:

$$\boxed{r = \frac{cov\{x_i, x_j\}}{\sigma^2(x)} = \frac{\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x)}{\sum_i \sum_j a_{ij}(x_i - \mu_x)^2}}$$

:

$$\boxed{\mu_x = \frac{\sum_i \sum_j a_{ij} x_i}{\sum_i \sum_j a_{ij}} = \frac{\sum_i x_i \sum_j a_{ij}}{2L} = \frac{\sum_i k_i x_i}{2L}}$$

+

$$\boxed{\mu_x^2 = \frac{\sum_i \sum_j k_i k_j x_i x_j}{(2L)^2}}$$

+

+

$$\boxed{\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x) = \sum_i \sum_j a_{ij} x_i x_j - 2L\mu_x^2}$$

Homophily and Assortative Mixing (2)

- The link-level degree-degree or strength-strength correlation coefficient can be further simplified:

$$\boxed{r = \frac{cov\{x_i, x_j\}}{\sigma^2(x)} = \frac{\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x)}{\sum_i \sum_j a_{ij}(x_i - \mu_x)^2}} \quad :$$

$$\boxed{\mu_x = \frac{\sum_i \sum_j a_{ij} x_i}{\sum_i \sum_j a_{ij}} = \frac{\sum_i x_i \sum_j a_{ij}}{2L} = \frac{\sum_i k_i x_i}{2L}} \quad + \quad \boxed{\mu_x^2 = \frac{\sum_i \sum_j k_i k_j x_i x_j}{(2L)^2}} \quad +$$

$$+ \quad \boxed{\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x) = \sum_i \sum_j a_{ij} x_i x_j - 2L\mu_x^2}$$

$$\Rightarrow \quad \boxed{\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x) = \sum_i \sum_j (a_{ij} - \frac{k_i k_j}{2L}) x_i x_j} \quad !!$$

Homophily and Assortative Mixing (2)

- The link-level degree-degree or strength-strength correlation coefficient can be further simplified:

$$r = \frac{cov\{x_i, x_j\}}{\sigma^2(x)} = \frac{\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x)}{\sum_i \sum_j a_{ij}(x_i - \mu_x)^2}$$:

$$\mu_x = \frac{\sum_i \sum_j a_{ij} x_i}{\sum_i \sum_j a_{ij}} = \frac{\sum_i x_i \sum_j a_{ij}}{2L} = \frac{\sum_i k_i x_i}{2L}$$ +

$$\mu_x^2 = \frac{\sum_i \sum_j k_i k_j x_i x_j}{(2L)^2}$$ +

+ $$\sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x) = \sum_i \sum_j a_{ij} x_i x_j - 2L\mu_x^2$$

$$\Rightarrow \sum_i \sum_j a_{ij}(x_i - \mu_x)(x_j - \mu_x) = \sum_i \sum_j (a_{ij} - \frac{k_i k_j}{2L}) x_i x_j$$!!

$$\Rightarrow \sum_i \sum_j a_{ij}(x_i - \mu_x)^2 = \sum_i \sum_j (a_{ij} - \frac{k_i k_j}{2L}) x_i^2$$

This Lecture: What we have done so far...

This Lecture: What we have done so far...

- Introduced a number of network measures and metrics

This Lecture: What we have done so far...

- Introduced a number of network measures and metrics
- Network-wide, node-specific, link-specific

This Lecture: What we have done so far...

- Introduced a number of network measures and metrics
- Network-wide, node-specific, link-specific

    1. Connectivity (density, components, distances, degrees, strength)

This Lecture: What we have done so far...

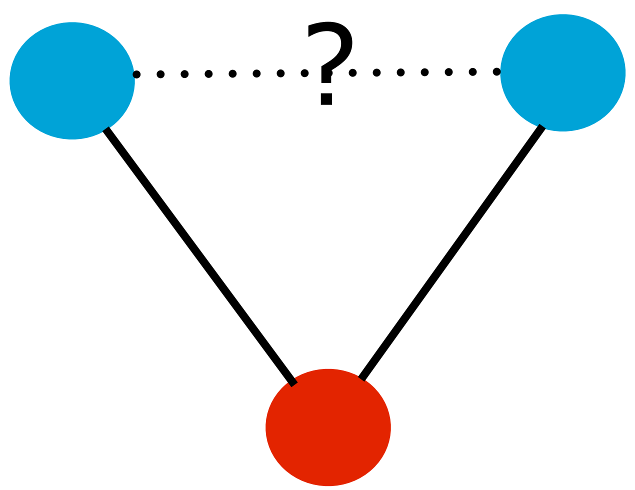
- Introduced a number of network measures and metrics
- Network-wide, node-specific, link-specific
    1. Connectivity (density, components, distances, degrees, strength)
    2. Homophily (ANND/ND and ANNS/NS correlation, ND-ND and NS-NS correlation)

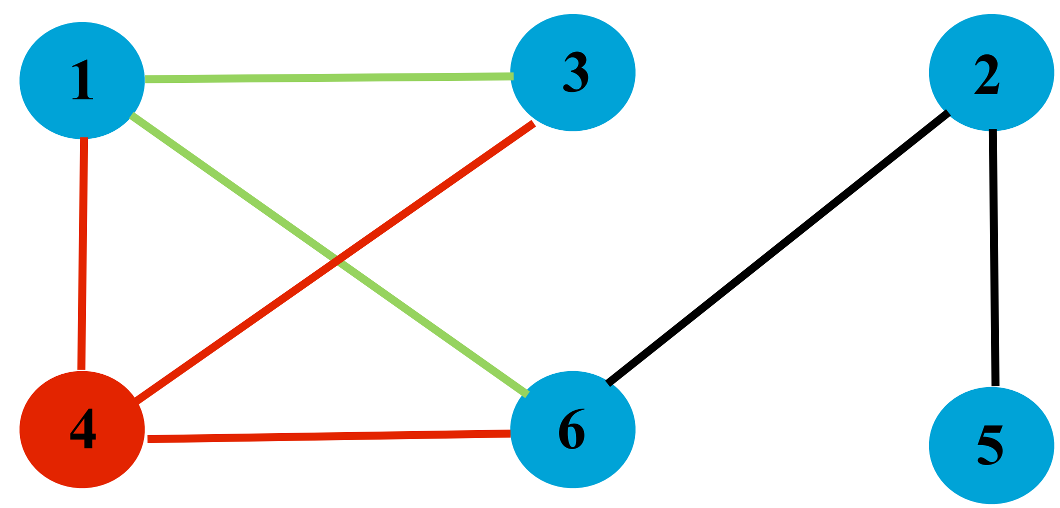
Node Clustering (1)

- What is the likelihood that any two neighbors of a node are themselves neighbors? Computing this likelihood is about counting triangles...

Node Clustering (1)

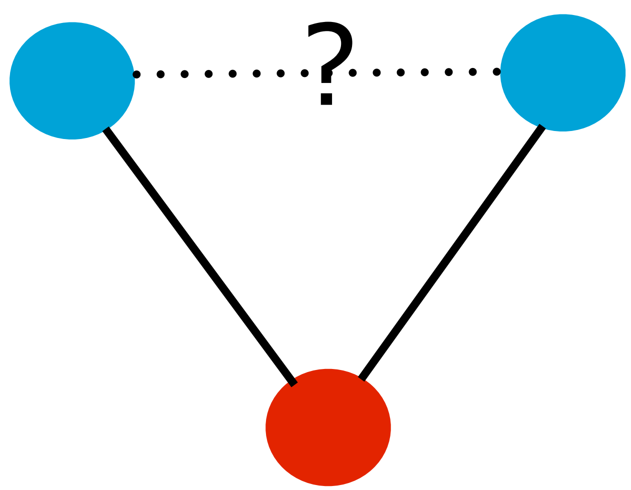
- What is the likelihood that any two neighbors of a node are themselves neighbors? Computing this likelihood is about counting triangles...

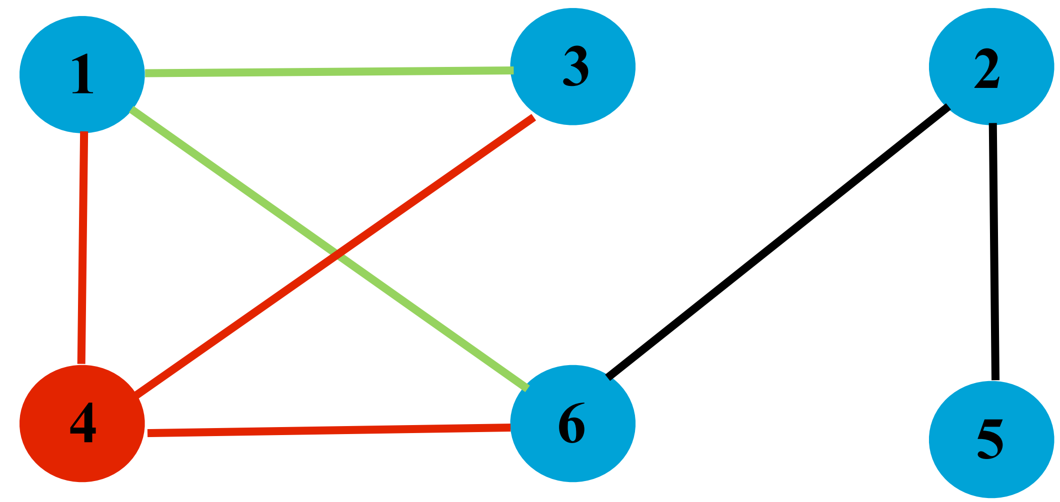




Node Clustering (1)

- What is the likelihood that any two neighbors of a node are themselves neighbors? Computing this likelihood is about counting triangles...

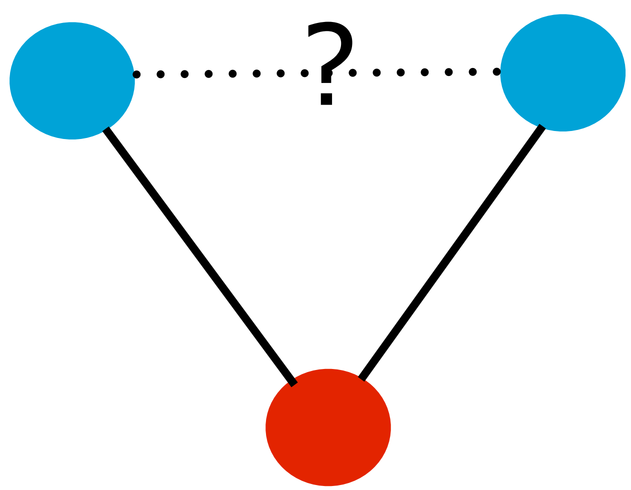


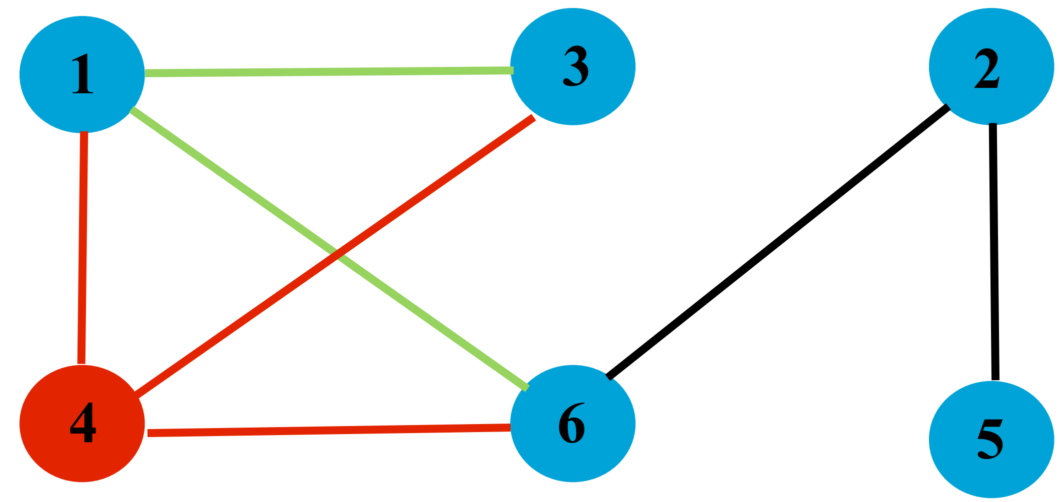


- Node 4 has k(i)=3: how many pairs of distinct neighbors can one count? It’s 3*2/2=3. In general, k(i)(k(i)-1)/2 pairs can be formed out of k(i) neighbors.

Node Clustering (1)

- What is the likelihood that any two neighbors of a node are themselves neighbors? Computing this likelihood is about counting triangles...

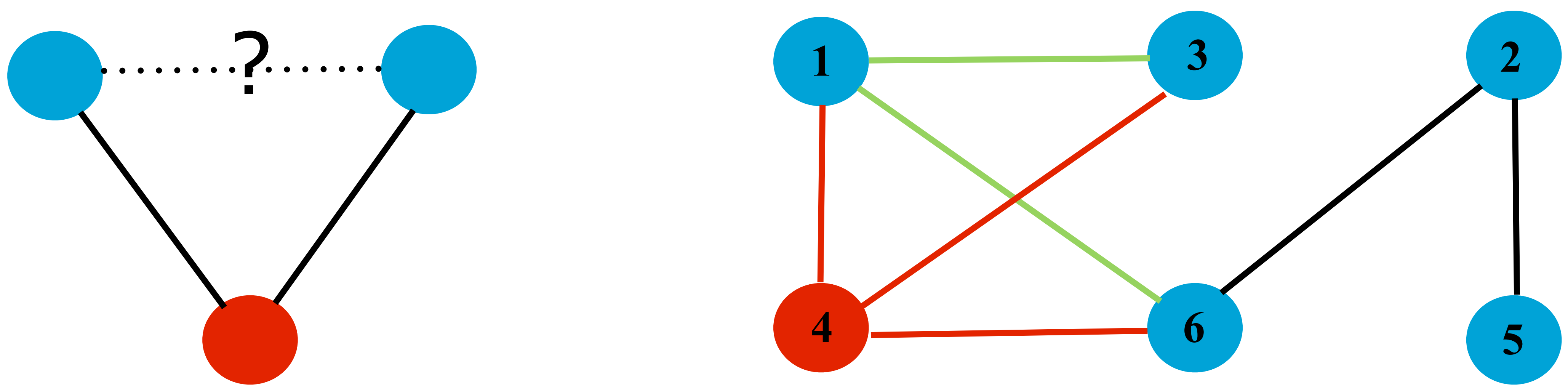




- Node 4 has k(i)=3: how many pairs of distinct neighbors can one count? It's 3*2/2=3. In general, k(i)(k(i)-1)/2 pairs can be formed out of k(i) neighbors.
- How many pairs of neighbors are themselves neighbors, i.e. how many triangles are present in the neighborhood of node 4? 2 out of 3, because (1,3) and (1,6) are neighbors, but (3,6) are not.

Node Clustering (1)

- What is the likelihood that any two neighbors of a node are themselves neighbors? Computing this likelihood is about counting triangles...



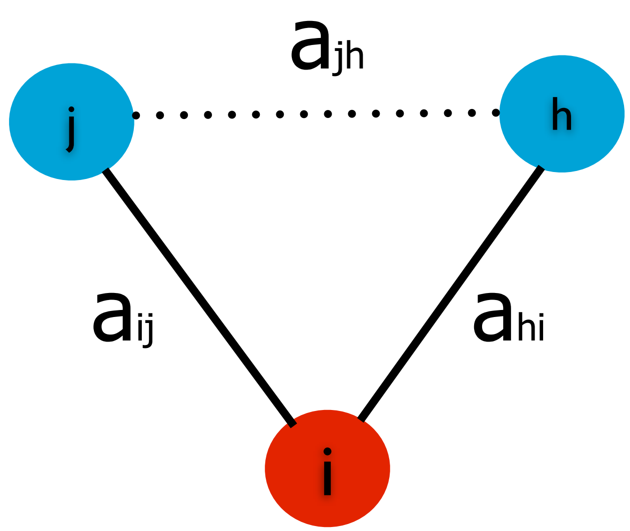
- Node 4 has k(i)=3: how many pairs of distinct neighbors can one count? It's 3*2/2=3. In general, k(i)(k(i)-1)/2 pairs can be formed out of k(i) neighbors.
- How many pairs of neighbors are themselves neighbors, i.e. how many triangles are present in the neighborhood of node 4? 2 out of 3, because (1,3) and (1,6) are neighbors, but (3,6) are not.
- Therefore the **clustering coefficient** of node 4 is 2/3

Node Clustering (2)

- How can one compute starting from A if a certain triangle is closed?

Node Clustering (2)

- How can one compute starting from A if a certain triangle is closed?

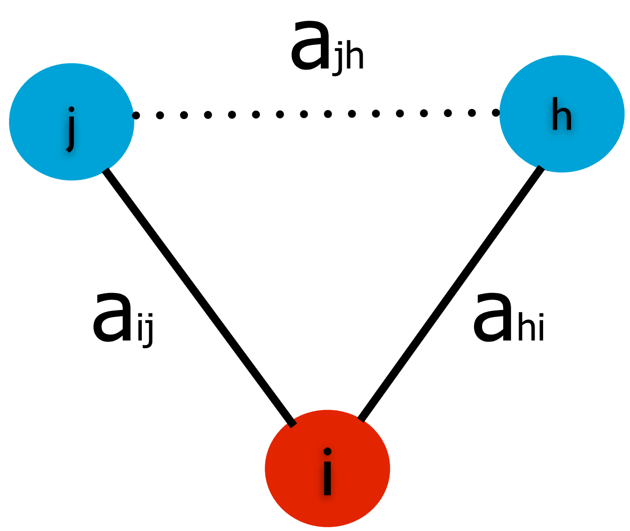


If: $a_{ij} \cdot a_{jh} \cdot a_{hi} = 1$ then triangle (i,j,h) is closed

But triangles are cycles of order 3... thus the number of triangles in i's neighborhood is equal to the number of 3-cycles starting and ending in i!

Node Clustering (2)

- How can one compute starting from A if a certain triangle is closed?



If: $a_{ij} \cdot a_{jh} \cdot a_{hi} = 1$ then triangle (i,j,h) is closed

But triangles are cycles of order 3... thus the number of triangles in i's neighborhood is equal to the number of 3-cycles starting and ending in i!

- The number of 3-cycles starting and ending in node i can be recovered looking at the entry z_{ii} of the matrix $Z=A^3$ and dividing that number by 2 (a cycle i>j>h is different from i>h>j but is the same triangle). Thus:

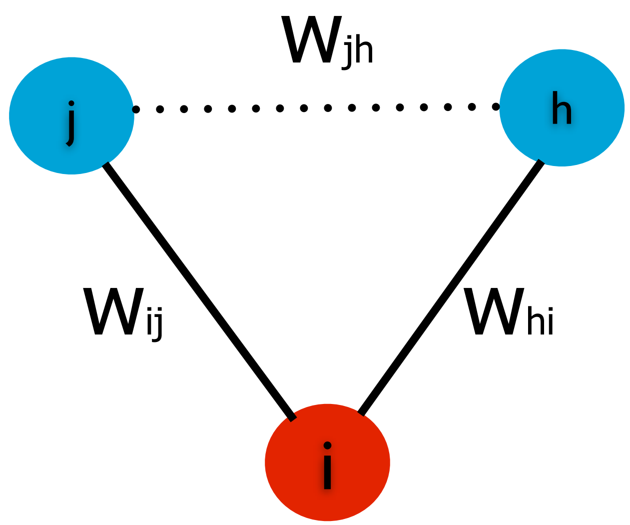
$$C_i = \frac{\frac{1}{2}\sum_j \sum_h a_{ij} a_{ih} a_{jh}}{\frac{1}{2} k_i (k_i - 1)} = \frac{(A^3)_{ii}}{k_i(k_i - 1)}$$

Clustering in WUNs

- How can one compute clustering coefficients in weighted undirected nets?

Clustering in WUNs

- How can one compute clustering coefficients in weighted undirected nets?

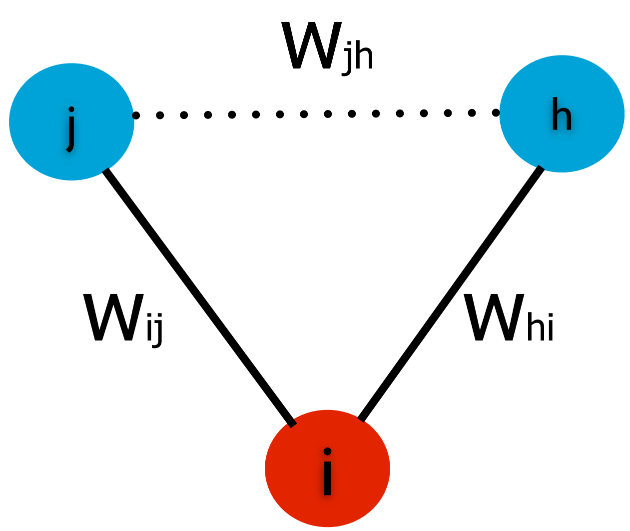


If: $a_{ij} \cdot a_{jh} \cdot a_{hi} = 1$ then triangle (i,j,h) is closed

Triangles must be weighted by their total intensity of interactions, as measured by some function of (w_{ij}, w_{jh}, w_{hi})

Clustering in WUNs

- How can one compute clustering coefficients in weighted undirected nets?



If: $a_{ij} \cdot a_{jh} \cdot a_{hi} = 1$ then triangle (i,j,h) is closed

Triangles must be weighted by their total intensity of interactions, as measured by some function of (w_{ij}, w_{jh}, w_{hi})

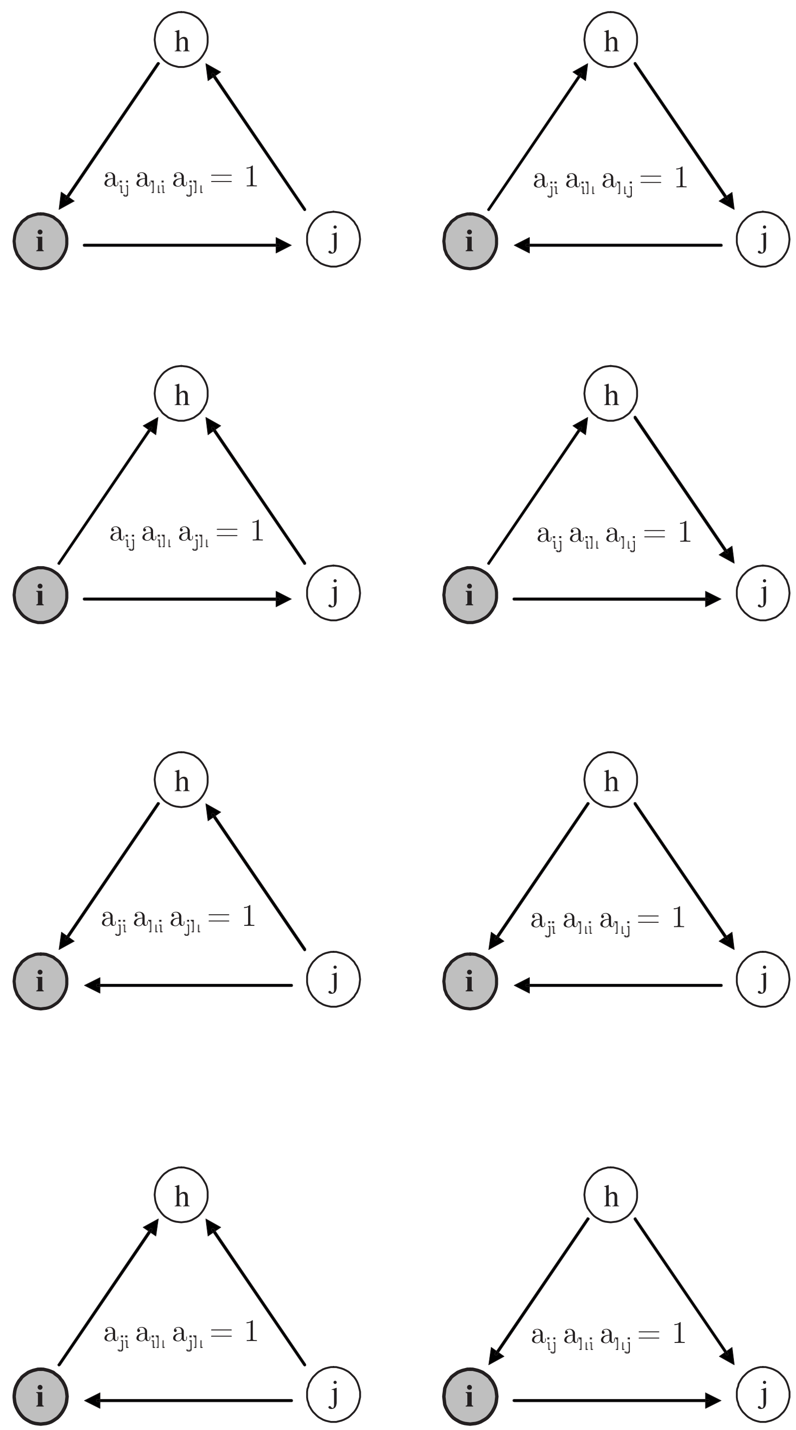
- There are many ways to weight a triangle, here's one of the most used:

$$C_i(W) = \frac{\frac{1}{2}\sum_i \sum_j w_{ij}^{1/3} w_{ih}^{1/3} w_{jh}^{1/3}}{\frac{1}{2}k_i(k_i - 1)} = \frac{(W^{[1/3]})^3_{ii}}{k_i(k_i - 1)}$$

- Where $(W^{[1/3]})^3_{ii}$ is the (i,i) entry of the matrix obtained first by raising all entries of W to 1/3 and then by taking the 3-rd power

Clustering in Directed Networks (1)

- Link directionality implies that there can be 8 different types of triangles and 4 classes that can be formed with node i as the reference node



Cycles

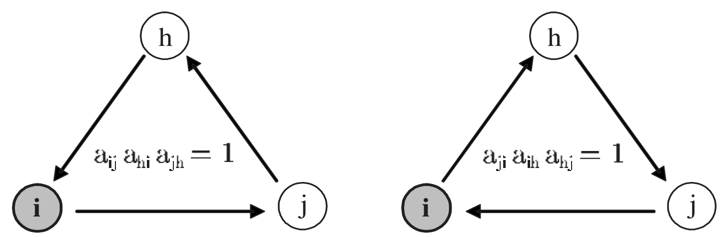
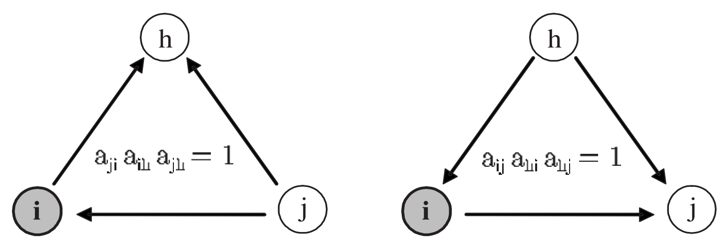
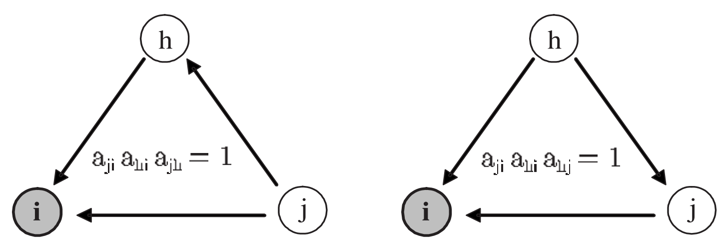
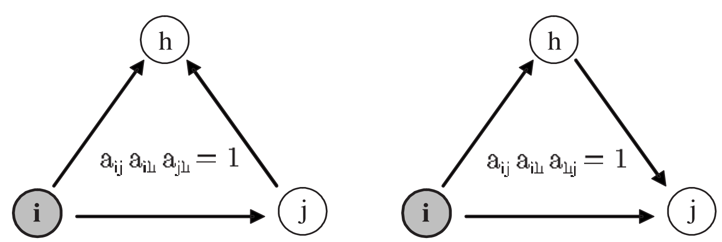
Out

In

Middleman

Clustering in Directed Networks (2)

- CC in BDN and WDN (see Fagiolo, PRE, 2007)

Patterns	Graphs	t_i^*	T_i^*	CCs for BDNs	CCs for WDNs
Cycle		$(A)^3_{ii}$	$d_i^{in}d_i^{out} - d_i^{\leftrightarrow}$	$C_i^{cyc} = \frac{(A)^3_{ii}}{d_i^{in}d_i^{out} - d_i^{\leftrightarrow}}$	$\tilde{C}_i^{cyc} = \frac{(\hat{W})^3_{ii}}{d_i^{in}d_i^{out} - d_i^{\leftrightarrow}}$
Middleman		$(AA^TA)_{ii}$	$d_i^{in}d_i^{out} - d_i^{\leftrightarrow}$	$C_i^{mid} = \frac{(AA^TA)_{ii}}{d_i^{in}d_i^{out} - d_i^{\leftrightarrow}}$	$\tilde{C}_i^{mid} = \frac{(\hat{W}\hat{W}^T\hat{W})_{ii}}{d_i^{in}d_i^{out} - d_i^{\leftrightarrow}}$
In		$(A^TA^2)_{ii}$	$d_i^{in}(d_i^{in} - 1)$	$C_i^{in} = \frac{(A^TA^2)_{ii}}{d_i^{in}(d_i^{in} - 1)}$	$\tilde{C}_i^{in} = \frac{(\hat{W}^T\hat{W}^2)_{ii}}{d_i^{in}(d_i^{in} - 1)}$
Out		$(A^2A^T)_{ii}$	$d_i^{out}(d_i^{out} - 1)$	$C_i^{out} = \frac{(A^2A^T)_{ii}}{d_i^{out}(d_i^{out} - 1)}$	$\tilde{C}_i^{out} = \frac{(\hat{W}^2\hat{W}^T)_{ii}}{d_i^{out}(d_i^{out} - 1)}$
All (D)	All 8 graphs above	$\frac{(A+A^T)^3_{ii}}{2}$	$d_i^{tot}(d_i^{tot} - 1) - 2d_i^{\leftrightarrow}$	$C_i^D = \frac{(A+A^T)^3_{ii}}{2T_i^D}$	$\tilde{C}_i^D = \frac{(\hat{W}+\hat{W}^T)^3_{ii}}{2T_i^D}$

Node Centrality

- Which are the most **central** nodes in a network?
  - ✓ Depends on the definition of “centrality”... many difference measures
  - ✓ Local node-centrality measures: take into account only the neighborhood of a node to measure its centrality in the network
  - ✓ Global node-centrality measures: account for the position of the node in the whole network

Node Centrality

- Which are the most **central** nodes in a network?
  - ✓ Depends on the definition of "centrality"... many difference measures
  - ✓ Local node-centrality measures: take into account only the neighborhood of a node to measure its centrality in the network
  - ✓ Global node-centrality measures: account for the position of the node in the whole network

- A simple and obvious local node-centrality measure
  - ✓ (Total) node degree (divided by N-1): a node is more (locally) central if it is more connected (**degree centrality**)

$$\Gamma_i^D = \frac{k_i}{N-1}$$

  - ✓ **Network centralization**: How much centralized is the whole network?

$$\Gamma^D = \frac{\sum_i (max_j\{k_j\} - k_i)}{(N-1)(N-2)}$$

Node Centrality

- Which are the most **central** nodes in a network?
  - ✓ Depends on the definition of "centrality"... many difference measures
  - ✓ Local node-centrality measures: take into account only the neighborhood of a node to measure its centrality in the network
  - ✓ Global node-centrality measures: account for the position of the node in the whole network

- A simple and obvious local node-centrality measure
  - ✓ (Total) node degree (divided by N-1): a node is more (locally) central if it is more connected (**degree centrality**)

$$\Gamma_i^D = \frac{k_i}{N-1}$$

  - ✓ **Network centralization**: How much centralized is the whole network?

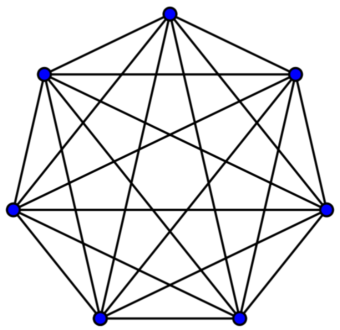
$$\Gamma^D = \frac{\sum_i (max_j\{k_j\} - k_i)}{(N-1)(N-2)}$$

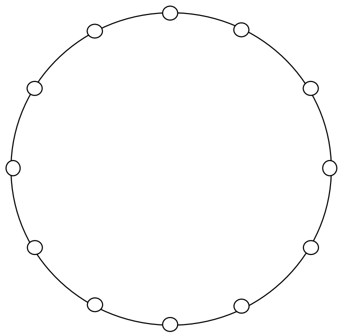
Value attained by the numerator in a star network with N nodes

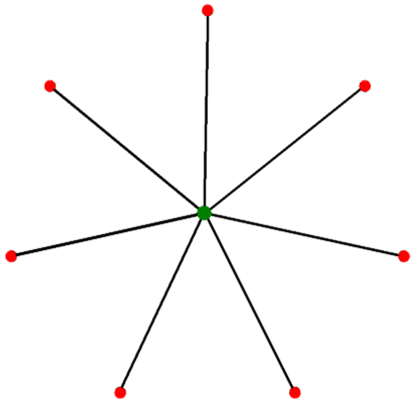
Network Centralization: Examples

- Regular networks (lattices, full networks)
  - ✓ All nodes have the same degree, thus $\Gamma^D = 0$
- Star Networks
  - ✓ There is 1 node (the center) with k=N-1 and N-1 nodes with k=1. Thus the numerator is equal to (N-1)(N-2) and $\Gamma^D = 1$ (check it)

$$\Gamma^D = \frac{\sum_i (max_j\{k_j\} - k_i)}{(N-1)(N-2)}$$



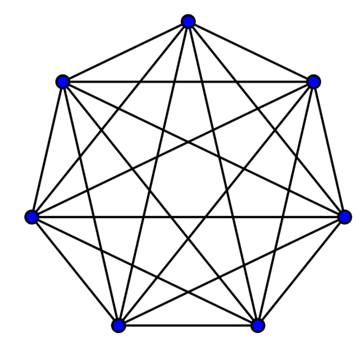


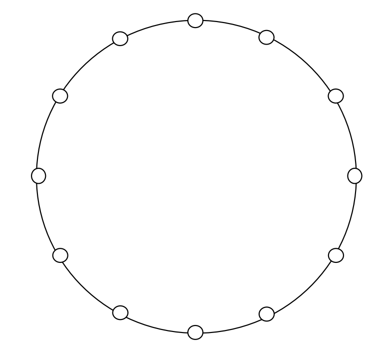


Network Centralization: Examples

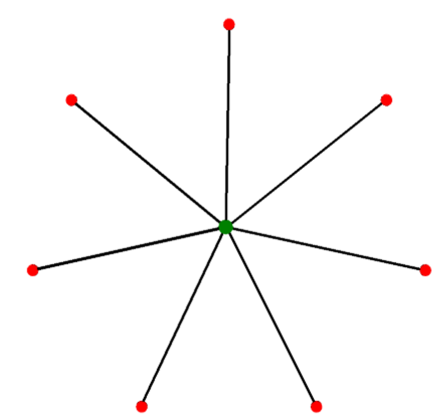
$$\Gamma^D = \frac{\sum_i (max_j\{k_j\} - k_i)}{(N-1)(N-2)}$$

- Regular networks (lattices, full networks)
  - ✓ All nodes have the same degree, thus $\Gamma^D = 0$



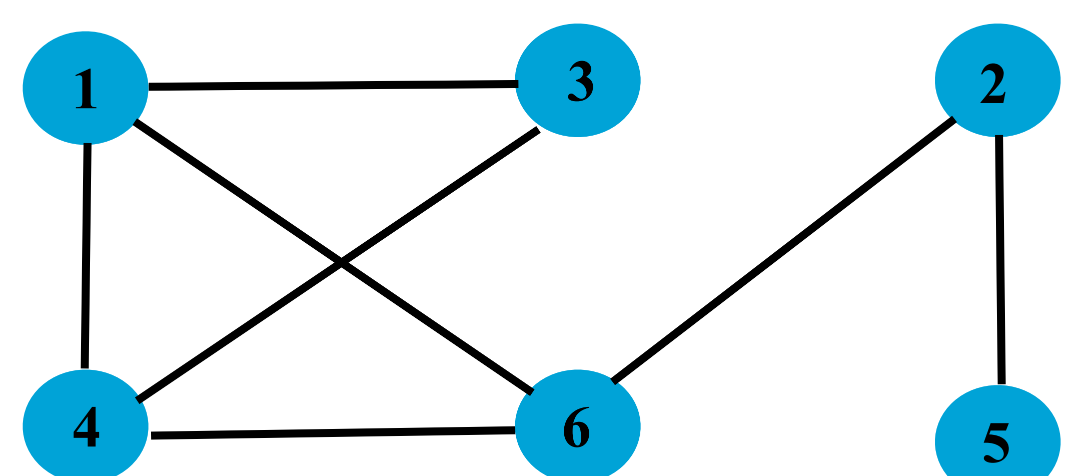


- Star Networks
  - ✓ There is 1 node (the center) with k=N-1 and N-1 nodes with k=1. Thus the numerator is equal to (N-1)(N-2) and $\Gamma^D = 1$ (check it)



- For the network below we have N=6 and degrees equal to {3,2,2,3,1,3}, thus max degree = 3 and:

$$\Gamma^D = \frac{(3-2)+(3-2)+(3-1)}{5 \cdot 4} = \frac{4}{20} = \frac{1}{5}$$



Node Closeness Centrality

- A node is more (globally) central the closer is on average to other nodes

$$CL_i = \frac{1}{\ell_i} = \frac{1}{\frac{1}{n}\sum_j d_{ij}} = \frac{n}{\sum_j d_{ij}}$$

ℓ_i = ANPL

d_{ij} = Distance between i and j

Node Closeness Centrality

- A node is more (globally) central the closer is on average to other nodes

$$CL_i = \frac{1}{\ell_i} = \frac{1}{\frac{1}{n}\sum_j d_{ij}} = \frac{n}{\sum_j d_{ij}}$$

ℓ_i = ANPL

d_{ij} = Distance between i and j

- Problems
  - ✓ Geodesic distances in networks tend to be very small, hence CL tends to span very small ranges, making it difficult to compare more and less central nodes
  - ✓ What if the network is not connected? Some distances become infinite and closeness becomes zero. A solution: defining CL as the inverse of the harmonic mean distance between nodes

$$CL'_i = \frac{1}{N-1}\sum_{j \neq i} \frac{1}{d_{ij}}$$

Node Betweenness Centrality (1)

- A node is more (globally) central the more it lies on (geodesic) paths connecting any other two nodes in the network
  - ✓ Assume that (i) something flows through the network (message); (ii) every pair of nodes exchange messages with equal probability per unit time; (iii) messages always take the shortest path between any two nodes (or choose one at random if there are several)
  - ✓ How many messages will be passed through a given node after a suitably long period of time? A number proportional to the number of geodesic paths the node lies on. This number is called betweenness centrality (BC) of a node.

Node Betweenness Centrality (1)

- A node is more (globally) central the more it lies on (geodesic) paths connecting any other two nodes in the network
  - ✓ Assume that (i) something flows through the network (message); (ii) every pair of nodes exchange messages with equal probability per unit time; (iii) messages always take the shortest path between any two nodes (or choose one at random if there are several)
  - ✓ How many messages will be passed through a given node after a suitably long period of time? A number proportional to the number of geodesic paths the node lies on. This number is called betweenness centrality (BC) of a node.

- Nodes with higher BC:
  - ✓ have higher influence because control flow (and might get paid for it)
  - ✓ are crucial for the network: if they fail, most communication is disrupted

Node Betweenness Centrality (2)

- More formally

$$BC_i = \sum_{h,k} \frac{\nu_{hk}^i}{g_{hk}}$$

ν_{hk}^i = # of geodesics from h to k passing through i

g_{hk} = total # of geodesics from h to k

Node Betweenness Centrality (2)

- More formally

$$BC_i = \sum_{h,k} \frac{\nu^i_{hk}}{g_{hk}}$$

ν^i_{hk} = # of geodesics from h to k passing through i

g_{hk} = total # of geodesics from h to k

- Remarks:

Node Betweenness Centrality (2)

- More formally

$$BC_i = \sum_{h,k} \frac{\nu_{hk}^i}{g_{hk}}$$

ν_{hk}^i = # of geodesics from h to k passing through i

g_{hk} = total # of geodesics from h to k

- Remarks:
  - ✓ We count also geodesics from h to h, node i included; we count h to k separately from k to h

Node Betweenness Centrality (2)

- More formally

$$BC_i = \sum_{h,k} \frac{\nu_{hk}^i}{g_{hk}}$$

= # of geodesics from h to k passing through i

= total # of geodesics from h to k

- Remarks:
  - ✓ We count also geodesics from h to h, node i included; we count h to k separately from k to h
  - ✓ In undirected networks we count paths twice, but this is irrelevant as we are not interested in BC levels but in rankings (who is more central)

Node Betweenness Centrality (2)

- More formally

$$BC_i = \sum_{h,k} \frac{\nu^i_{hk}}{g_{hk}}$$

= # of geodesics from h to k passing through i

= total # of geodesics from h to k

- Remarks:
  - ✓ We count also geodesics from h to h, node i included; we count h to k separately from k to h
  - ✓ In undirected networks we count paths twice, but this is irrelevant as we are not interested in BC levels but in rankings (who is more central)
  - ✓ Definition still applies in BDN and can be extended in WUN/WDN by appropriately weighting paths

Node Betweenness Centrality (2)

- More formally

$$BC_i = \sum_{h,k} \frac{\nu_{hk}^i}{g_{hk}}$$

ν_{hk}^i = # of geodesics from h to k passing through i

g_{hk} = total # of geodesics from h to k

- Remarks:
  - ✓ We count also geodesics from h to h, node i included; we count h to k separately from k to h
  - ✓ In undirected networks we count paths twice, but this is irrelevant as we are not interested in BC levels but in rankings (who is more central)
  - ✓ Definition still applies in BDN and can be extended in WUN/WDN by appropriately weighting paths
  - ✓ The denominator g is needed to account for cases where there are more than one geodesic between h and k

Node Betweenness Centrality (2)

- More formally

$$BC_i = \sum_{h,k} \frac{\nu^i_{hk}}{g_{hk}}$$

ν^i_{hk} = # of geodesics from h to k passing through i

g_{hk} = total # of geodesics from h to k

- Remarks:
  - ✓ We count also geodesics from h to h, node i included; we count h to k separately from k to h
  - ✓ In undirected networks we count paths twice, but this is irrelevant as we are not interested in BC levels but in rankings (who is more central)
  - ✓ Definition still applies in BDN and can be extended in WUN/WDN by appropriately weighting paths
  - ✓ The denominator g is needed to account for cases where there are more than one geodesic between h and k
  - ✓ The maximum value of BC is attained by the center of a star network: it lies on all N^2 geodesics between node pairs, except the N-1 paths between peripheral nodes to themselves. Thus BC(center) $=N^2-(N-1)$. Prove it. Compute max when one only counts paths from any two nodes different from i

Node Betweenness Centrality (2)

- More formally

$$BC_i = \sum_{h,k} \frac{\nu_{hk}^i}{g_{hk}}$$

= # of geodesics from h to k passing through i

= total # of geodesics from h to k

- Remarks:
  - ✓ We count also geodesics from h to h, node i included; we count h to k separately from k to h
  - ✓ In undirected networks we count paths twice, but this is irrelevant as we are not interested in BC levels but in rankings (who is more central)
  - ✓ Definition still applies in BDN and can be extended in WUN/WDN by appropriately weighting paths
  - ✓ The denominator g is needed to account for cases where there are more than one geodesic between h and k
  - ✓ The maximum value of BC is attained by the center of a star network: it lies on all N^2 geodesics between node pairs, except the N-1 paths between peripheral nodes to themselves. Thus BC(center) $=N^2-(N-1)$. Prove it. Compute max when one only counts paths from any two nodes different from i
  - ✓ What is the the minimum of BC? If the network is connected, then there must be at minimum: N-1 geodesics from all j≠i to i; N-1 geodesics from i to all j≠i; and a geodesic from i to i. Therefore min(BC) =2(N-1)+1=2N-1. This happens to “leafs” in a network or peripheral nodes of a star (prove it).

Eigenvector Centrality

- **Main Idea**
  - ✓ Degree centrality awards to a given node one centrality point for every neighbor it has. More generally: giving each node a score proportional to the sum of the scores of its neighbors.
  - ✓ Eigenvector node centrality: node centrality x is proportional to the sum of centralities of its neighbors

$$x_i = \lambda \sum_j a_{ij} x_j \Rrightarrow \mathbf{x} = \lambda \mathbf{A} \mathbf{x}$$

Eigenvector Centrality

- **Main Idea**
  - ✓ Degree centrality awards to a given node one centrality point for every neighbor it has. More generally: giving each node a score proportional to the sum of the scores of its neighbors.
  - ✓ Eigenvector node centrality: node centrality x is proportional to the sum of centralities of its neighbors

$$x_i = \lambda \sum_j a_{ij} x_j \Rightarrow \mathbf{x} = \lambda \mathbf{A} \mathbf{x}$$

  - ✓ Hence **x** is an eigenvector of A. Usually we take the eigenvector associated to the largest eigenvalue of A to ensure positive **x**s
  - ✓ This is called **Bonacich centrality**: a node is more central either because it has many neighbors or because it has important neighbors (it is connected with nodes that count), or both

Eigenvector Centrality in Digraphs

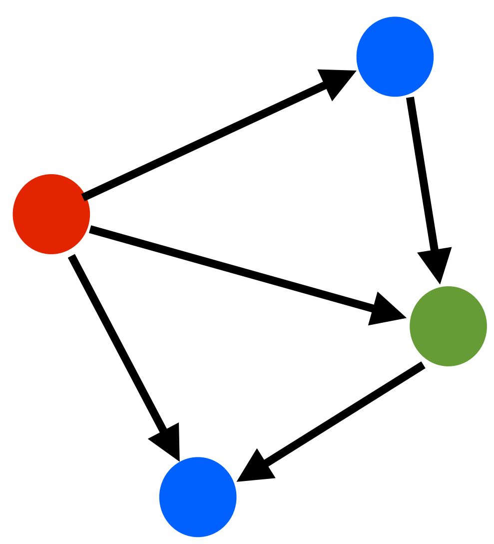
- **Problem #1**
  - ✓ If A is symmetric, there is only one eigenvector sequence. In digraphs A is asymmetric, so there may be two ways to define centrality, according to which type (inward or outward) of link contributes to centrality

$$x_i = \lambda^{out} \sum_j A_{ij} x_j \qquad\qquad x_i = \lambda^{in} \sum_j A_{ji} x_j$$

  - ✓ Example: in the WWW centrality depends on how many pages point to you, not from the fact that you build a page that point to many others... but in other networks it may not be so...

Eigenvector Centrality in Digraphs

- **Problem #2**
  - ✓ Any node who is in a chain of directed paths starting from a node who has zero centrality score, will end up having zero centrality score as well!



$$x_i = \lambda^{in} \sum_j A_{ji} x_j$$

Red node: no incoming links, so zero centrality

Green node: only one incoming link, but from a zero-centrality node, thus zero centrality as well

  - ✓ Only nodes that are in a strongly connected component (SCC) of two or more vertices, or the out-component of such a component, can have positive centrality scores
  - ✓ Acyclic networks? They have SCC of 1 node... only zero centrality nodes

A First Solution

- **Katz Centrality Index**
  - ✓ Idea: assigning to every node a small initial (positive) centrality bonus

$$x_i = \lambda \sum_j A_{ji} x_j + \beta$$

  - ✓ Setting the "bonus" equal to 1 for all and solving:

$$\mathbf{x} = \lambda \mathbf{A} \mathbf{x} + \mathbf{1} \qquad \mathbf{x} = (\mathbf{I} - \lambda \mathbf{A})^{-1} \mathbf{1}$$

- **Problems**
  - ✓ How to set λ?
  - ✓ Centrality scores are passed via incoming links... so a highly influential node with many outgoing links will give high centrality to all of them... but received centrality should be smaller the more links are pointed... if you are one among many you should receive less centrality.... Example: importance of a web page received from hubs in the WWW

Google Page-Rank Centrality

- **Diluting centrality scores from hubs**
  - ✓ Idea: rescaling induced centrality by out-degree

$$x_i = \lambda \sum_j A_{ji} \frac{x_j}{k_j^{out}} + \beta$$

  - ✓ Setting the "bonus" equal to 1 for all and solving:

$$\mathbf{x} = \lambda \mathbf{A}\mathbf{D}^{-1}\mathbf{x} + \beta \mathbf{1} \qquad \mathbf{x} = (\mathbf{I} - \lambda \mathbf{A}\mathbf{D}^{-1})^{-1}\mathbf{1}$$

- **How does Google search engine work?**
  - ✓ Searching using text queries and other methods in pre-assembled lists of web pages
  - ✓ Ranking pages according to a number of criteria, including Page-Rank centrality: Google is not efficient in searching/finding but in ranking
  - ✓ Setting λ=0.85... Why?

Hub and Authority Centrality (I)

- **Problem**
  - ✓ So far: Centrality scores can be received only through incoming edges.
  - ✓ Problem: in many cases (e.g., citation networks) nodes can be central also if they point to many "selected" nodes
    - Citation networks: review articles can be central because they cite many other "influential" papers; influential papers can become important because they are pointed by reviews

- **Hubs and Authorities**
  - ✓ Authorities: nodes that contain useful information, they are pointed by many nodes, in particular by many hubs
  - ✓ Hubs: nodes that tell us where authorities are located, they point to many authorities
  - ✓ A node can then have two measures of centrality and can be central because it's a hub or an authority, or both!

Hub and Authority Centrality (II)

- **How to Compute Hub and Authority Centrality Scores?**
  - ✓ Assigning to each node an authority score (x) and a hub score (y)
  - ✓ Authority score (x) depends on how many links a node receives from nodes that have a (high) hub score
  - ✓ Hub score (y) depends on how many nodes with a (high) autorithy score a node points to

$$x_i = \alpha \sum_j A_{ji} y_j \qquad\qquad y_i = \beta \sum_j A_{ij} x_j$$

  - ✓ Letting $\lambda=\alpha\beta$ and solving

$$\mathbf{x} = (\mathbf{I} - \lambda \mathbf{A}^{\mathbf{T}} \mathbf{A})^{-1} \mathbf{1} \qquad\qquad \mathbf{y} = (\mathbf{I} - \lambda \mathbf{A} \mathbf{A}^{\mathbf{T}})^{-1} \mathbf{1}$$

  - ✓ Authority (hub) scores are the eigenvectors of A^TA (AA^T) associated to the same (largest) eigenvalue, which can be shown to exist

This Lecture: What we have done so far...

This Lecture: What we have done so far...

- Introduced a number of network measures and metrics

This Lecture: What we have done so far...

- Introduced a number of network measures and metrics
- Network-wide, node-specific, link-specific

This Lecture: What we have done so far...

- Introduced a number of network measures and metrics
- Network-wide, node-specific, link-specific
    1. Connectivity (density, components, distances, degrees, strength)

This Lecture: What we have done so far...

- Introduced a number of network measures and metrics
- Network-wide, node-specific, link-specific

    1. Connectivity (density, components, distances, degrees, strength)
    2. Homophily (ANND/ND and ANNS/NS correlation, ND-ND and NS-NS correlation)

This Lecture: What we have done so far...

- Introduced a number of network measures and metrics
- Network-wide, node-specific, link-specific

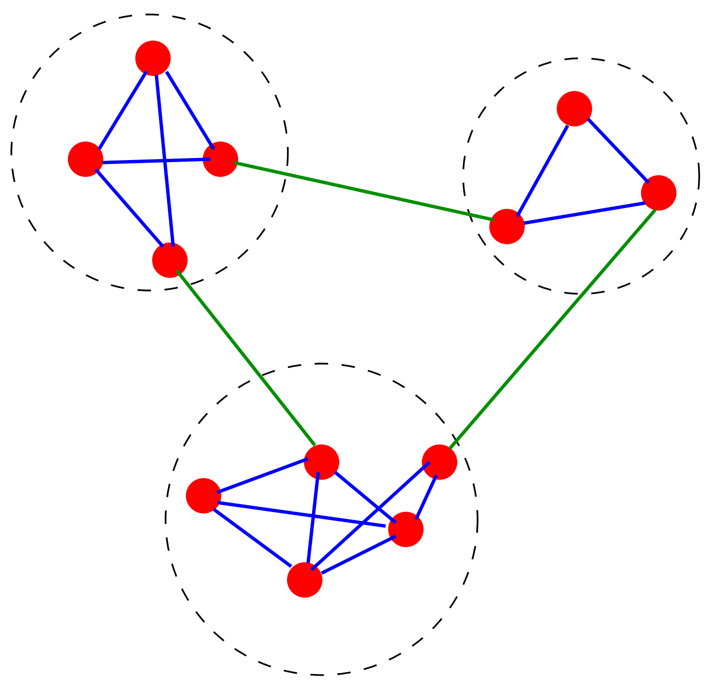
1. Connectivity (density, components, distances, degrees, strength)
2. Homophily (ANND/ND and ANNS/NS correlation, ND-ND and NS-NS correlation)
3. Clustering (undirected/directed; binary/weighted)

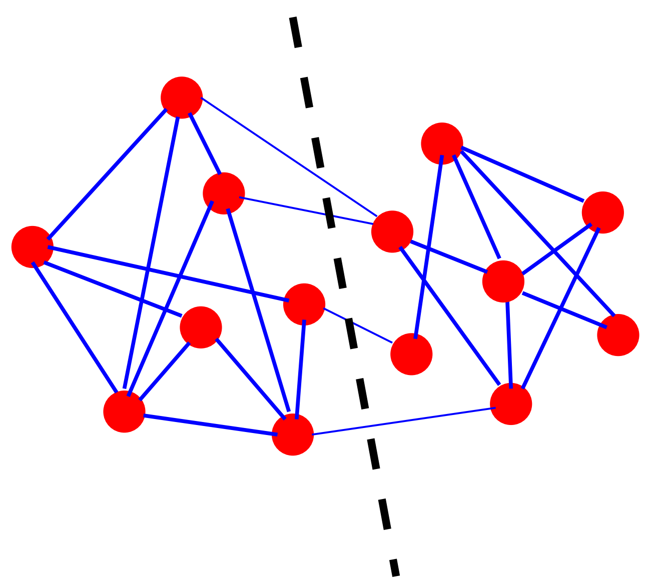
This Lecture: What we have done so far...

- Introduced a number of network measures and metrics
- Network-wide, node-specific, link-specific
    1. Connectivity (density, components, distances, degrees, strength)
    2. Homophily (ANND/ND and ANNS/NS correlation, ND-ND and NS-NS correlation)
    3. Clustering (undirected/directed; binary/weighted)
    4. Centrality

Community Structure (I)

- Detecting groups of tightly interconnected vertices (Fortunato, 2009)
  - ✓ In many networks the distribution of links is globally and locally inhomogeneous
  - ✓ High concentration of links among special groups of vertices, and low concentration of links between these groups
- Community structure (CS) detection
  - ✓ Identifying clusters or modules that due to high inter-connectivity among them may share common properties and/or play similar roles
  - ✓ Definition of CS is not clear: therefore a huge set of CS detection methods are available
  - ✓ CS: Non-overlapping vs. overlapping
- Here: Non-overlapping CS detection via maximization of modularity function
  - ✓ Assigning each partition of the N nodes a “quality” indicator





Source: Fortunato (2009)

Community Structure: Examples

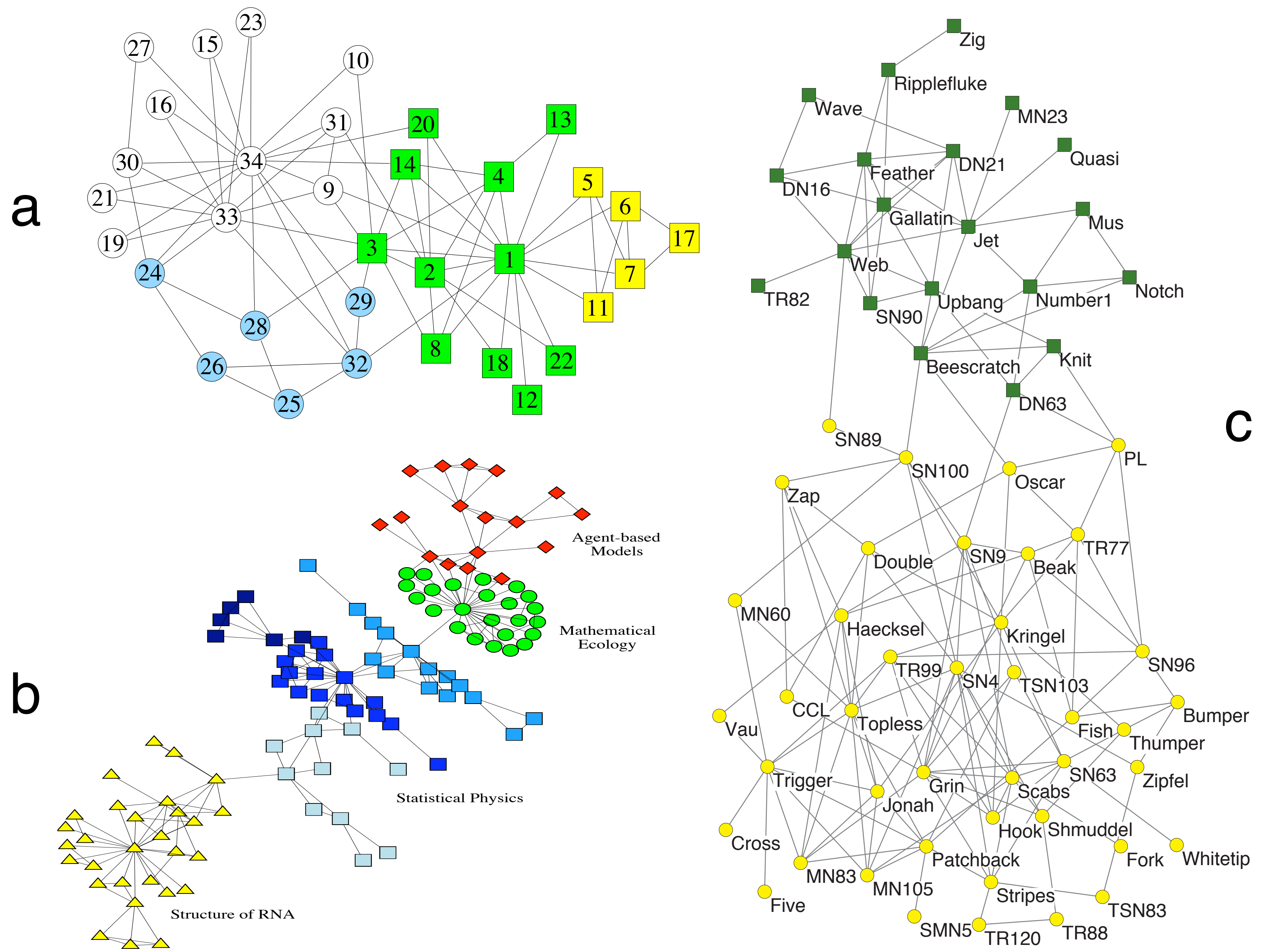


FIG. 2 Community structure in social networks. a) Zachary's karate club, a standard benchmark in community detection. The colors correspond to the best partition found by optimizing the modularity of Newman and Girvan (Section VI.A). Reprinted figure with permission from (Donetti and Muñoz, 2004). ©2004 by IOP Publishing and SISSA. b) Collaboration network between scientists working at the Santa Fe Institute. The colors indicate high level communities obtained by the algorithm of Girvan and Newman (Section V.A) and correspond quite closely to research divisions of the institute. Further subdivisions correspond to smaller research groups, revolving around project leaders. Reprinted figure with permission from (Girvan and Newman, 2002). ©2002 by the National Academy of Science of the USA. c) Lusseau's network of bottlenose dolphins. The colors label the communities identified through the optimization of a modified version of the modularity of Newman and Girvan, proposed by Arenas et al. (Arenas *et al.*, 2008b) (Section XII.A). The partition matches the biological classification of the dolphins proposed by Lusseau. Reprinted figure with permission from (Arenas *et al.*, 2008b). ©2008 by IOP Publishing.

Source: Fortunato (2009)

Community Structure (II)

- Consider one of all possible partitions of the N nodes of the network. Let this partition be **C**={C_1,...,C_K}. To evaluate how good this partition is we can compute the function:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij})\, \delta(C_i, C_j)$$

where: i,j=1,...,N, A_{ij} are the entries of the adjacency matrix; P_{ij} represents the expected number of edges between i and j; m is the total number of links; and δ yields one if i and j are in the same community, zero otherwise

Community Structure (II)

- Consider one of all possible partitions of the N nodes of the network. Let this partition be **C**={C_1,...,C_K}. To evaluate how good this partition is we can compute the function:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij}) \, \delta(C_i, C_j)$$

where: i,j=1,...,N, A_{ij} are the entries of the adjacency matrix; P_{ij} represents the expected number of edges between i and j; m is the total number of links; and δ yields one if i and j are in the same community, zero otherwise

- Suppose that the probability of connection between i and j is proportional to the product of k_i and k_j. Thus the expected number of links between i and j is equal to k_i*k_j/2m (prove it). This is the configuration model that we will study in Lecture 6. Then the modularity function becomes:

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j)$$

Community Structure (III)

- Grouping all contributions that come from the same community together, the modularity function can be rewritten as

$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right]$$

where now n_c=k, c spans all clusters in **C**, l_c is total number of links joining nodes of cluster c, and d_c is the sum of degrees of nodes in cluster c

Community Structure (III)

- Grouping all contributions that come from the same community together, the modularity function can be rewritten as

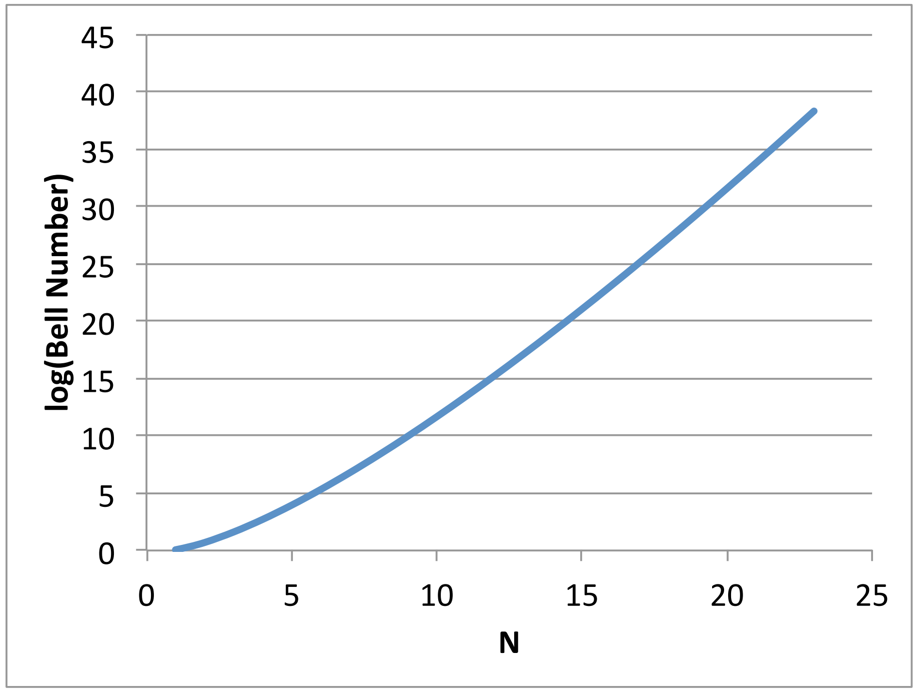
$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right]$$

where now n_c=k, c spans all clusters in **C**, l_c is total number of links joining nodes of cluster c, and d_c is the sum of degrees of nodes in cluster c

- Modularity maximization: since high values of Q indicate good partitions (as compared to the null model), then finding the max of Q over the space of all partitions would yield the best one
- Unfortunately maximizing Q is impossible: it is an NP-complete problem. No fast solution is known and there is no known efficient way to locate a solution
- That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows.

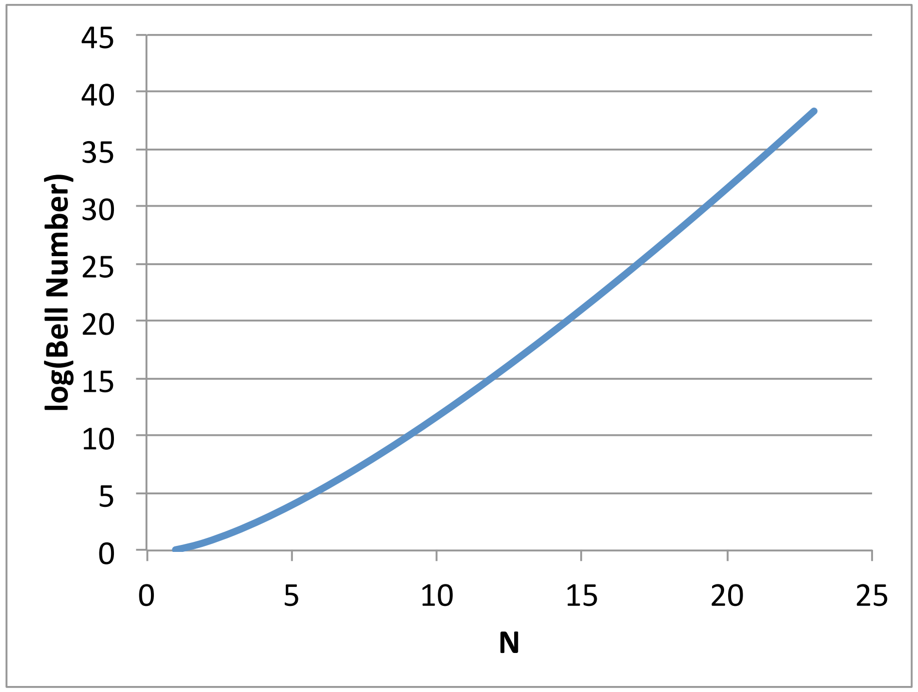
Community Structure (IV)

- What is the number of all partitions of a set of N units? They are known as Bell's numbers and grow very quickly as N increases



Community Structure (IV)

- What is the number of all partitions of a set of N units? They are known as Bell's numbers and grow very quickly as N increases



- Therefore modularity maximization needs clever optimization algorithms to deliver solutions (greedy techniques, simulated annealing, genetic algorithms)
- Extensions of modularity to the case of weighted directed networks are possible

Next Lecture

- What is a network? Examples of networks
- Why networks are important for economists?
- Networks and graphs
- Measures and metrics on networks
- Distributions of metrics and measures in large networks
- Models of network formation
- Null statistical network models
- Economic applications