School of Information Sciences
University of Pittsburgh

TELCOM2125: Network Science and Analysis

Konstantinos Pelechrinis

Spring 2015

Figures are taken from:
M.E.J. Newman, “Networks: An Introduction”

Part 2: Measures and Metrics

Degree centrality

- **Centrality is a widely studied concept**
  - Which are the most important/central nodes in a network?

- **Simplest centrality metric is the degree centrality**
  - It is simply the degree of node
  - For a directed network we have in- and out-degree centralities
    - ✓ Each one appropriate for different circumstances

- **Simple, yet illuminating measure**
  - Social network → node of high degree might be though as one with access to more information sources, with more prestige etc.
  - Citation network → a paper with more citations (in-degree centrality) might be roughly thought as influential

Eigenvector centrality

- **Degree centrality assumes all neighbors are equal → Only the number of neighbors matter**
  - However, in many circumstances the importance of a vertex increases if it is connected to important nodes → eigenvector centrality

- **Assume initially everyone has score x_i=1 and we update its centrality using the sum of the scores/centralities of his neighbor's**

$$x_i' = \sum_j A_{ij} x_j \Rightarrow \vec{x}' = A\vec{x}$$

Eigenvector centrality

- **After t updates we will have:** $\vec{x}(t) = A^t \vec{x}(o)$
- **Expressing vector x(0) as a linear combination of the eigenvalues of A (why can we do this?) we have:** $\vec{x}(0) = \sum_i c_i \vec{v}_i$
- **Hence:** $\vec{x}(t) = A^t \sum_i c_i \vec{v}_i = \sum_i c_i \kappa_i^t \vec{v}_i = \kappa_1^t \sum_i c_i [\frac{\kappa_i}{\kappa_1}]^t \vec{v}_i$ **where κ_1 is the largest eigenvalue**
  - Since all, but for i=1, the terms in the sum decay exponentially as t grows, we have for $t \to \infty \Rightarrow \vec{x}(t) = c_1 \kappa_1^t \vec{v}_1$
  - Therefore, vector x is basically the leading eigenvector of A:

$$A\vec{x} = \kappa_1 \vec{x}$$

Eigenvector centrality

- **As wanted the centrality x_i is proportional to the sum of the centralities of i's neighbors**

$$x_i = \frac{1}{\kappa_i} \sum_j A_{ij} x_j$$

- **So the eigenvector centrality of a node can be large for two reasons**
  - Has many neighbors and/or
  - Has important neighbors

- **Note: All eigenvector centrality values are non-negative**

Eigenvector centrality for directed networks

- **The adjacency matrix is general asymmetric → two leading eigenvectors**
  - Which to use?

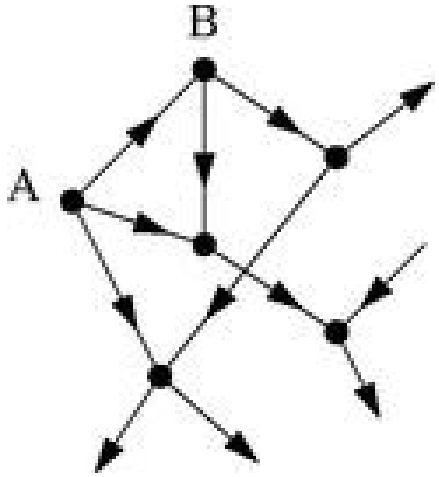
- **In most cases we use the right eigenvector**
  - The centrality in most of the cases is conferred by other vertices pointing towards you

$$x_i = \frac{1}{\kappa_i} \sum_j A_{ij} x_j \Rightarrow A\vec{x} = \kappa_i \vec{x}$$

  - Where x is the right leading eigenvector

Eigenvector centrality for directed networks

- **What is the eigenvector centrality of A ?**
  - Of B ?



Only vertices that are in a strongly connected component of two or more vertices, or the out-component of such a component can have non-zero eigenvector centrality.

In acyclic graphs eigenvector centrality is completely useless

Katz centrality

- **In order to overcome problems as the above in directed networks we can tweak the eigenvector centrality and give each vertex a small amount of centrality "for free"**

$$x_i = \alpha \sum_j A_{ij} x_j + \beta \Rightarrow \vec{x} = \alpha A \vec{x} + \beta \vec{1}$$

  - The first term is the normal eigenvector centrality and the second is the "free" part

- **By setting β=1 we have:** $\vec{x} = (I - \alpha A)^{-1} \vec{1}$
  - Katz centrality

Choice of α

- **α governs the balance between the eigenvector centrality contribution and the constant term**
  - For small α the Katz centrality is dominated by the constant factor and hence all vertices have a centrality of 1 (β=1)
  - For large values of α Katz centrality diverges

- **As we increase α from zero the centralities increase and there comes a point at which they diverge. This happens at the point where $(I-\alpha A)^{-1}$ diverges → det(I-αA)=0 → $\det(A-\alpha^{-1}I) = 0$**
  - The roots α^{-1} are equal to the eigenvalues of A
  - As α increases the determinant first crosses zero when $\alpha=1/\kappa_1$, where κ_1 is the largest eigenvalue of A
    - ✓ Hence α should be less than $1/\kappa_1$

Extension of Katz centrality

- **A possible extension of the above definition is to assign different constant centrality to different vertices**

$$x_i' = \alpha \sum_j A_{ij} x_j + \beta_i \Rightarrow \vec{x} = (I - \alpha A)^{-1} \vec{\beta}$$

- **β_i is some intrinsic, non-network contribution to the centrality for each vertex**
  - E.g., in a social network the importance of an individual might depend on non-network factors as well, such as age or income etc.

PageRank

- **One possible problem with Katz centrality is that a vertex v with high centrality and high out-degree will cause a large number of vertices to have high centrality as well**
  - In reality though each one of the vertices that v points to should get a fraction of v's centrality

$$x_i = \alpha \sum_j A_{ij} \frac{x_j}{k_j^{out}} + \beta$$

- **A problem can arise when a node j has no vertices that points to**
  - A_{ij} as well as j's out-degree will be 0 and their ratio is not defined
  - However, when a node has no out edges should contribute zero to the centralities of any other vertex and hence we can artificially set this contribution to 0 by setting its out-degree to 1

PageRank

- **Hence the PageRank can be expressed as**

$$\vec{x} = \alpha A D^{-1}\vec{x} + \beta\vec{1} \Rightarrow \vec{x} = (I - \alpha A D^{-1})^{-1}\vec{1} = D(D - \alpha A)^{-1}\vec{1}$$

  - Where D is a diagonal matrix: $D_{ii} = \max(k_{out}^{i}, 1)$

- **Again the choice of α is important**
  - Should be less than the inverse of the leading eigenvalue of the matrix AD^{-1}
  - Google search engine sets α=0.85
    - ✓ There is no rigorous theory behind this choice

PageRank

- **Again we can generalize PageRank**

$$x_i = \alpha \sum_j A_{ij} \frac{x_j}{k_j^{out}} + \beta_i \Rightarrow \vec{x} = D(D - \alpha A)^{-1} \vec{\beta}$$

  - β_i can be possibly based on the relevance of the page/vertex with the query

- **With no additive constant term we get:**

$$x_i = \alpha \sum_j A_{ij} \frac{x_j}{k_j}$$

  - For an undirected network this is simply the degree centrality
  - For directed networks it does not reduce to a known centrality measure, but it suffers from the same problem of the original eigenvector centrality

Centrality metrics summary

	with constant term	without constant term
divide by out-degree	$\mathbf{x} = \mathbf{D}(\mathbf{D} - \alpha\mathbf{A})^{-1} \cdot \mathbf{1}$ PageRank	$\mathbf{x} = \mathbf{A}\mathbf{D}^{-1}\mathbf{x}$ degree centrality
no division	$\mathbf{x} = (\mathbf{I} - \alpha\mathbf{A})^{-1} \cdot \mathbf{1}$ Katz centrality	$\mathbf{x} = \kappa_1^{-1}\mathbf{A}\mathbf{x}$ eigenvector centrality

Connection with random walk

- **Many of the eigenvector-based centrality metrics can be explained through random walks on graphs**

- **We will focus on PageRank and derive the same formula through a random walk model for the Web surfing process**
  - In this process a “random” surfer starts from a web pages and follows randomly the outgoing links
  - The steady state probability distribution of this random walk is the PageRank scores for every webpage
  - Problems will arise when a webpage does not have an outgoing link (dangling node) or when we are trapped in an infinite loop (“spider traps”)

Connection with random walk

- **We will be constructing several related matrices:** $H, \hat{H}, G$
  - Every matrix will be NxN where N is the number of webpages
- **Eventually the PageRank will be an eigenvector of matrix G**
- **Constructing matrix H**
  - The (i,j) element of H is $1/O_i$ if there is a hyperlink from webpage i to webpage j and 0 otherwise
  - This matrix essentially captures the structure of the hyperlinks

Matrix H

- **If x is the vector denoting the importance of the N webpages we can iterative compute x in a similar way we did for the eigenvector centrality:**

$$\vec{x}^T[k] = \vec{x}^T[k-1] \cdot H$$

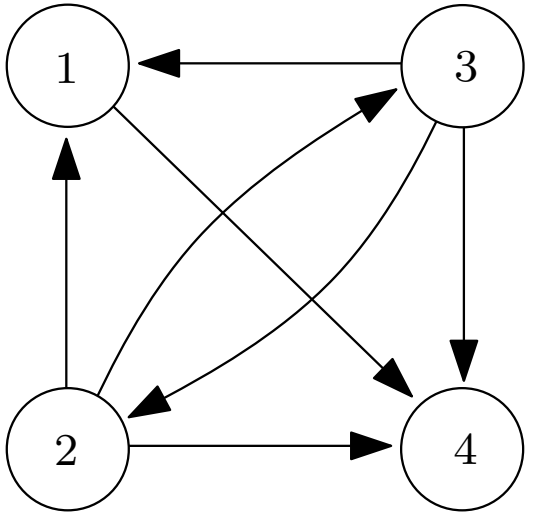
  - We also typically normalize the vector **x** so that its entries sum up to 1

- **Do these iterations converge irrespective of the initial value we pick for vector x ?**
  - Not as is!

Constructing $\hat{H}$

- **There will be a problem when you have dangling nodes**

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



- **To see this we write out the system of linear equations $x^T=x^TH$**

$$\begin{cases} \frac{1}{3}(x_2 + x_3) = x_1 \\ \frac{1}{3}x_3 = x_2 \\ \frac{1}{3}x_2 = x_3 \\ x_1 + \frac{1}{3}(x_2 + x_3) = x_4 \end{cases}$$

This system has solution $\mathbf{x}=[0\ 0\ 0\ 0]^T$

Constructing $\hat{H}$

- **One solution is to replace every row of 0s, with a row of 1/N**
  - Intuitively this is saying that even a webpage does not point to any other webpage, we will force it to spread its importance score event among the webpages

- **Mathematically this amounts to adding the matrix** $\frac{1}{N}(\vec{w} \cdot \vec{1}^T)$
  - **w** is a column indicator vector, with the i-th entry being 1 if webpage i is a dangling node – otherwise it is 0
  - **1** is a vector of 1s

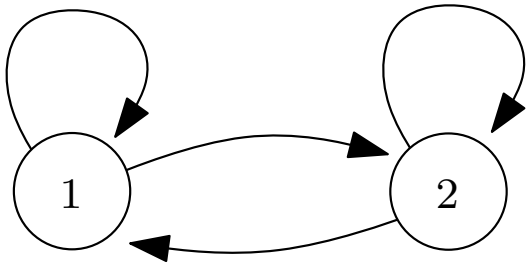
$$\hat{H} = H + \frac{1}{N}(\vec{w} \cdot \vec{1}^T)$$

Constructing $\hat{H}$

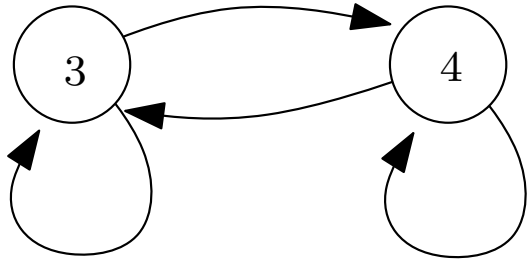
- **The matrix $\hat{H}$ has non-negative entries and each row adds up to 1**
  - We can think each row as a probability vector with the (i,j) entry indicating the probability of going to page j is you are currently at page i
  - This is a random walk on graphs with a teleport operation
  - Markov chain

- **This random walk model does not capture the web browsing behavior exactly but it does strike an effective balance between simplicity and usefulness**

Constructing G



- **Another problem that exists is the "spider traps" that can appear when our random surfer gets trapped in parts of the network**
- **For example, for the network on the right depending on the initial position of our surfer and the initialization vector x[0] we will get different results**



- **To solve this problem we can add some *randomization* to the iterative procedure of the surfer**
  - With probability (1-α) you teleport to a different node (e.g., uniformly at random), and with probability α you follow the outgoing links of the node you are currently at

Constructing G

- **Mathematically it means that we add another rank 1 matrix, weighted by (1-a):** $\frac{1}{N}\vec{1}\cdot\vec{1}^{T}$
- **Hence, the final matrix G is:**

$$G = a\hat{H} + (1-a)\frac{1}{N}(\vec{1}\cdot\vec{1}^{T})$$

- **This matrix is aperiodic, irreducible and stochastic and it can be shown that independently of the initialization x[0] the iterative procedure: $\mathbf{x}^{T}[k]=\mathbf{x}^{T}[k-1]G$ will always converge to the unique PageRank vector $\mathbf{x}^{*}$**
  - Vector $\mathbf{x}^{*}$ is the leading eigenvector of G (the leading eigenvalue of the *stochastic* matrix G is 1): $\mathbf{x}^{*T}=\mathbf{x}^{*T}G$

Computing PageRank

- **The above process provides as with a straightforward algorithm to compute the PageRank vector of a network**

- **After constructing the (Google) matrix G, we follow the iterative procedure: $\mathbf{x}^T[k]=\mathbf{x}^T[k-1]G$ until convergence**
  - You can check for convergence by examining by how much the two consecutive vectors **x** change

Generalized PageRank

- Instead of assuming uniformly at random teleportation we can have any probability distribution vector v over the nodes in the graph

- Hence G is: $G = aH + (a\vec{w} + (1-a)\cdot\vec{1})\cdot\vec{v}^T$

- You need to convince yourself that the dominant left eigenvector of matrix G is also the solution to this equation: $(I - aH)^T\vec{x} = \vec{v}$ which is the equation we saw before!

Hubs and authorities

- **For directed networks there is another twist to the centrality measures**
  - Authorities are nodes that contain useful information on a topic of interest
  - Hubs are nodes that tell us where the best authorities can be found

- **Hyperlink-Induced Topic Search or HITS defines two types of centralities for a vertex i**
  - Authority centrality x_i
  - Hub centrality y_i

HITS

- **The authority centrality of vertex i is proportional to the sum of the hub centralities of the vertices that point to i, while its hub centrality is proportional to the sum of the authority centralities of the vertices it points to**

$$x_i = \alpha \sum_j A_{ij} y_j \quad \text{and} \quad y_i = \beta \sum_j A_{ji} x_j$$

$$\vec{x} = \alpha A \vec{y} \quad \text{and} \quad \vec{y} = \beta A^T \vec{x}$$

$$A A^T \vec{x} = \lambda \vec{x} \quad \text{and} \quad A^T A \vec{y} = \lambda \vec{y}, \quad \lambda = (\alpha\beta)^{-1}$$

- **The authorities and hub centralities are the eigenvectors of AA^T and A^TA respectively with the same (leading) eigenvalue**

HITS

- **The above means that AA^T and A^TA should have the same leading eigenvalue λ_1**

- **It can be proven that in fact all the eigenvalues are the same for the two matrices**

$$AA^T\vec{x} = \lambda\vec{x} \Rightarrow A^TA(A^T\vec{x}) = \lambda(A^T\vec{x})$$

  - Hence A^Tx is an eigenvector of A^TA with the same eigenvalue λ
  - Furthermore we have $\vec{y} = A^T\vec{x}$, which means that we only need to compute the authority centrality

Notes on HITS

- **AA^T is the co-citation matrix → authority centrality is *roughly* the eigenvector centrality of the co-citation matrix**
  - Why roughly?

- **Similarly A^TA is the bibliographic coupling matrix → hub centrality is *roughly* the eigenvector centrality of the bibliographic coupling network**

- **HITS provides an elegant solution to the problem raised from eigenvector centrality in directed networks**

Closeness centrality

- **Measures the distance from the vertex to the rest of the network**

- **If d_{ij} is the length of the shortest path between i and j, then the mean distance from i to any other node is:**

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

  - Vertices that are separated by other nodes by short geodesic distances have a low l_i value

- **Not considering the summation term for j=i we get:**

$$\ell_i = \frac{1}{n-1}\sum_{j(\neq i)} d_{ij}$$

Closeness centrality

- **In contrast to other centrality measures, l_i gets smaller values for central nodes**
  - Hence closeness centrality C_i is given by:

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

- **A problem with closeness centrality is that its value's dynamic range is relative close**
  - Small changes in the network can cause large fluctuations and ranking changes
  - Other centrality measures we have examined do not typically suffer from similar problems
    - ✓ Especially the *leaders* are clearly separated

Closeness centrality

- **A second problem with the definition arises when we have networks with more than one connected components**
  - Nodes belonging to different components will have infinite distance, leading l_i to be infinite as well

- **One possible solution is to consider in the summation only the vertices within the same component**
  - What is the problem with this approach?

- **A more elegant solution is to redefine closeness in terms of the harmonic mean distance between vertices**

$$C_i' = \frac{1}{n-1} \sum_{j(\neq i)} \frac{1}{d_{ij}}$$

Betweenness centrality

- **Measures the extent to which a vertex lies on paths between other vertices**
  - Vertices with high betweenness may have considerable influence within a network by virtue of their control over information passing between others
  - Removal of vertices with high betweenness will disrupt the most communications between other vertices

- **Formally, assume a network with at most one shortest path between any pair of vertices**
  - Then the betweenness centrality of a vertex i is defined to be the number of those paths that pass through i

Betweenness centrality

- **Mathematically let n^i_{st} be 1 if vertex i lies on the shortest path between s and t, and 0 if not (or s and t belong to different components). Then the betweenness centrality of i, x_i is:**

$$x_i = \sum_s \sum_t n^i_{st}$$

  - Since we count both paths from s to t and from t to s, the above equation can be directly applied to directed networks
  - We also consider paths when s=t (it does not change the ranking of nodes)
  - Paths from s to t are considered to pass from s and t
    - ✓ Again this choice does not make any change in the relevant betweenness of nodes belonging at the same component (which is what we care about)

Betweenness centrality

- **The above hold if there is at most 1 shortest path between two vertices**
  - This is not the case though in real networks → multiple shortest paths (not necessarily vertex/edge independent)

- **The standard extension is to consider a weight for every shortest path between the pair (s,t)**
  - Weight is equal to the inverse of the number of shortest paths between (s,t)
    - ✓ Betweenness of a vertex is then simply the sum of the weights of the shortest paths crossing this vertex

Betweenness centrality

- **Formally, we redefine n^i_{st} to be the number of shortest paths between s and t that pass through i, while g_{st} is the total number of shortest paths between s and t. Then:**

$$x_i = \sum_s \sum_t \frac{n^i_{st}}{g_{st}}$$

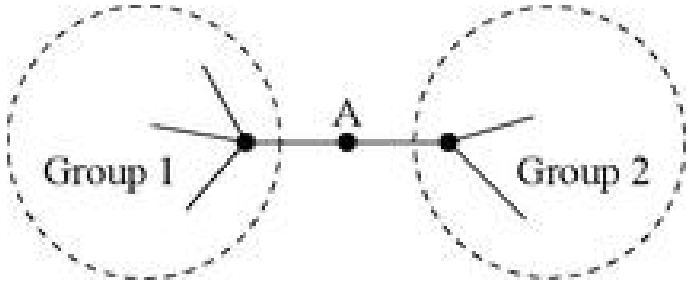
  - Terms where $g_{st}=0$, are not considered, since this means that s and t belong to different components

- **Similar definitions exist for directed networks, but betweenness is rarely used in this case**

Betweenness centrality

- **Betweenness centrality captures how much a vertex falls *between* others, in contrast with other centrality measures we have seen, which capture how well connected the node is**

- **A vertex can have low degree/eigenvector/closeness etc. centralities but high betweenness centrality**

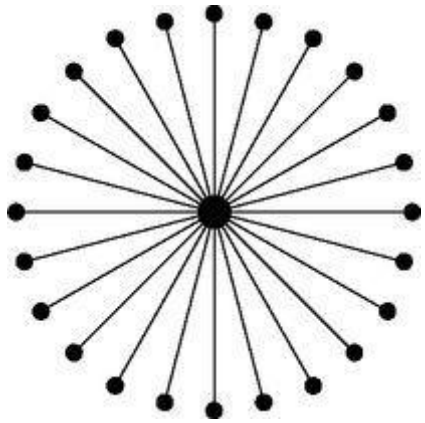


A is a broker vertex

Betweenness centrality

- **Betweenness exhibits a large dynamic range of values**

- **For instance consider a start topology**
  - Largest value is n^2-n+1 (why?)
  - Lowest value is 2n-1 (why?)
  - The ratio of largest to lowest is ≈(1/2)n



- **As network evolves betweenness values can shift but the ordering at the top of the list changes relatively infrequently**

Normalized betweenness

- Some software tools make use of a normalized version of betweenness centrality

- A natural choice is to divide it with the total number of (ordered) vertex pairs:

$$x_i = \frac{1}{n^2} \sum_s \sum_t \frac{n_{st}^i}{g_{st}}$$

- Other choice is to divide with the maximum possible value for the betweenness (rarely used):

$$x_i = \frac{1}{n^2 - n + 1} \sum_s \sum_t \frac{n_{st}^i}{g_{st}}$$

Flow betweeenness

- **In reality shortest paths are not always followed**
  - E.g., think of how many times you have heard some news about a friend of yours not from him/her directly but from a third person

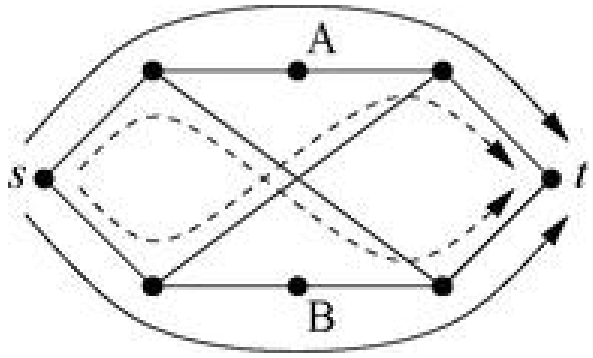
- **Flow betweenness makes some allowances for effects like this**
  - Relevance with maximum flow
  - Is defined from the same equation but now n^i_{st} is the amount of flow through vertex i when the max flow is transmitted from s to t

$$x_i = \sum_s \sum_t n^i_{st}$$

  - From Menger's theorem, n^i_{st} is the number of edge independent paths between s and t that pass through i

Flow betweenness

- **The edge independent paths are not necessarily unique**
  - Some vertices might be part of one maximal set of edge independent paths but not of others



- **Freeman *et al.* define the flow through a vertex, such as A, to be the maximum possible flow over all possible choices of paths**
  - Hence, in the topology presented the flow between s and t would contribute 1 to the centrality of A

Random-walk betweenness

- **Flow centrality might miss shortest paths at all**
  - Hence, no matter which definition we are using from the ones presented we might not consider important paths for different reasons

- **Random-walk betweenness aims at counting all paths**
  - Traffic between s and t is thought as following a random-walk with absorbing state (state t)
  - n^i_{st} is the number of times that the random walk from s to t passed through node i averaged over many repetitions of the walk

$$x_i = \sum_s \sum_t n^i_{st}$$

Random-walk betweenness

- **Random-walk betweenness captures the betweenness centrality of a vertex in a network where information wonders around in the network at random**

- **On the opposite extreme shortest path betweenness captures the centrality when information flows over a known path**

- **The truth is a real network should lay somewhere in between…**

S.P. Borgatti, “Centrality and Network Flow”, Social Networks **27**, 55-71 (2005)

Groups of vertices

- **Clique is a maximal subset of the vertices in an undirected network such that every member of the set is connected by an edge to any other vertex**
  - Cliques can overlap → they can share one or more vertices
- **A clique in an otherwise sparse network, indicates a highly cohesive subgroup**
- **The requirement of being connected to all other members is strict**
  - In real networks, people might be connected to most but not all of their acquaintances

k-plex

- **k-plex of size n is a maximal subset of n vertices within a network such that each vertex is connected (by an edge) to at least n-k of the others**
  - For k=1 we have the ordinary cliques
- **Similar to cliques k-plexes can be overlapping**
- **k-plexes is a useful concept on social network analysis but there is no solid rule what value k should take**
- **Another generalization of cliques (and k-plexes) is one where we could specify that each member should be connected to a *fraction* of the others (e.g., 50% or 75% etc.)**

k-core

- **Maximal subset of vertices such that each is connected (by an edge) to at least k others in the subset**
  - The total number of nodes in the core is not restricted
    - ✓ Hence, the set of k-cores is not the same with the set of k-plexes for a given value of k
  - A k-core of size n is also a (n-k)-plex

- **Two k-cores cannot overlap, since then they would just form another k-core of larger size (maximal subset)**

k-core

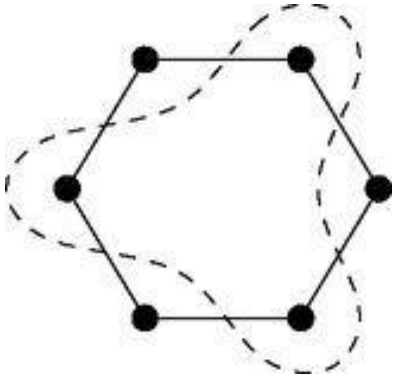
- **k-cores are of particular interest since there are straightforward algorithms to compute k-cores in a network:**
  - Start by removing from the network nodes of degree < k
    - ✓ Remove also edges of these nodes → update the degrees of nodes that remained after pruning
      - Continue this process iteratively until you cannot remove any other nodes
    - ✓ The remaining nodes form one or more k-cores

- **The resulting k-cores might not be connected, regardless if the network we started with was connected or not**

k-cliques and k-clans (k-clubs)

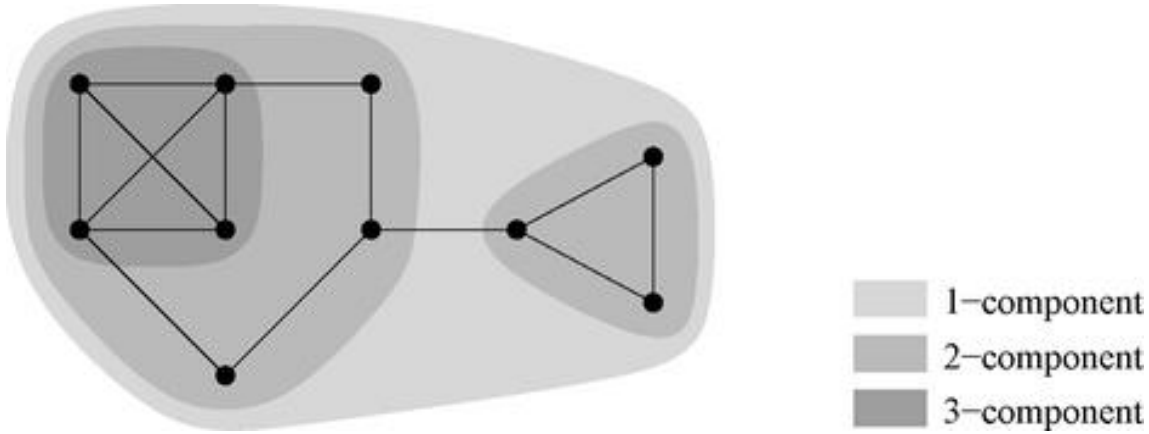
- **k-clique is a maximal subset of vertices such that each is no more than a distance k away from any of the others via the edges of the network**
  - For k=1 this just recovers ordinary cliques

- **The paths through which the members of a k-clique are connected, not need to be consisted of vertices within the group**
  - If we restrict to paths that run only within the subset then we obtain k-clans or k-clubs



k-components

- **A <u>k-component</u> is a maximal subset of vertices such that each is reachable from each of the others by at least k vertex-independent paths**



  - For k=2 → bicomponents and for k=3 → tricomponents

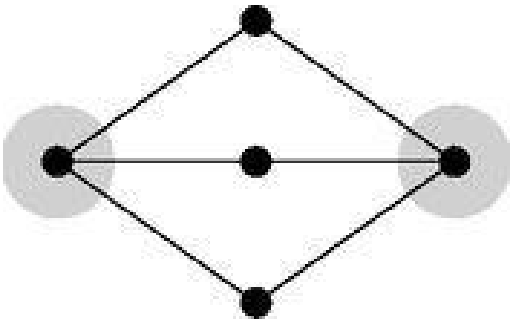
- **A k-component is a subset of a (k-1)-component**

k-components

- **k-components are tightly related with the notion of network robustness**

- **For k>2 the k-components need not be contiguous**

- **In social network sciences non-contiguous components are not desirable**
  - A more strict definition comes from the requirement that the paths should be entirely within the subset



Transitivity

- **Transitivity is important in social networks**
  - Not that much in other types of networks
- **A relation ° is transitive if** $\left.\begin{matrix} a \circ b \\ b \circ c \end{matrix}\right\} \Rightarrow a \circ c$
- **The most simple – but also important – relation in a social network is "connected by an edge"**
  - If vertex u is connected by an edge with v, and v is connected by an edge with w, then u is connected by an edge with w
- **Perfect transitivity only occurs in networks where each component is a fully connected subgraph or clique**

Transitivity

- **Partial transitivity can be also useful**
  - The fact that u knows v and v knows w, does not mean that u must know w
    - ✓ However, it increases the probability that u knows w

- **Clustering coefficient (cc) is used to capture this partial transitivity**

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

  - The above definition is for the network as a whole

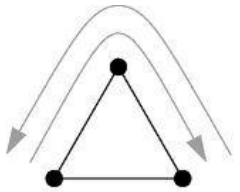
- **A high clustering coefficient is translated to a network with high transitivity**
  - People belong to *tight* groups

Transitivity

- Another way to compute the above expression is by counting the number of triangles that appear in the network

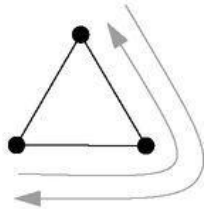
- Every triangle contains 6 paths of length two and hence:

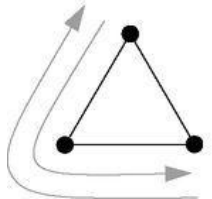
$$C = \frac{6 \times (\text{number of triangles})}{\text{number of paths of length two}}$$



- Alternatively, a connected triplet can be defined as a set of three nodes uvw with edges (u,v) and (v,w):

$$C = \frac{3 \times (\text{number of triangles})}{\text{number of connected triplets}}$$





Clustering coefficient in directed networks

- **In a directed network, traditionally cc is computed by assuming no direction on the edges**

- **There are extensions that consider edge direction**
  - E.g., if “u points/likes v” and “v points/likes w”, then “u points/likes w”
    - ✓ These extensions have not found a widespread application in the literature

Local clustering

- **As we said clustering coefficient can be defined for a single vertex as well:**

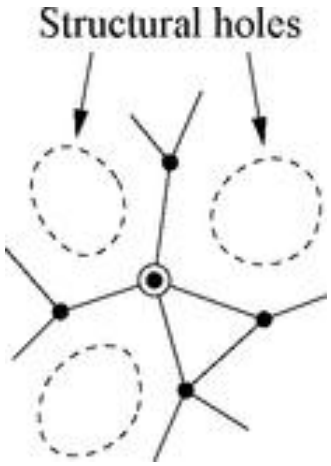
$$CC_i = \frac{\text{(number of pairs of neighbors of i that are connected)}}{\text{(number of pairs of neighbors of i)}}$$

  - If we find the average clustering coefficient from all vertices of a network, we practically have the clustering coefficient of the network

- **Clustering coefficient is closely related with structural holes**

Structural holes

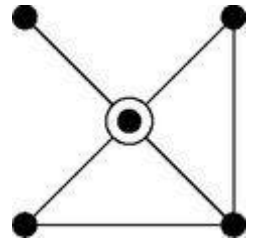
- **While it is common for the neighbors of a vertex to be connected among themselves, it happens sometimes that these expected connections between neighbors are missing**
  - These missing links are called structural holes



- **Structural holes can be a bad thing with regards to efficient spread of information**
  - Information has fewer routes to move around
  - They are good though for the vertex i
    - ✓ Hence, local clustering coefficient could be a measure of possible influence of the specific vertex (lower value is better)

Redundancy

- **The redundancy R_i of a vertex i is the mean number of connections from a neighbor of i to other neighbors of i**
  - R_i=1/4(0+1+1+2)=1
  - Redundancy for a node i takes values from 0 to k_i-1



- **R_i is closely related with CC_i**
  - When the average connections of a friend of i with the other friends of i is R_i, the total number of such edged is ½($k_i R_i$)
  - The total number of possible such edges is ½(k_i(ki-1))
  - Hence: $$CC_i = \frac{\frac{1}{2} k_i R_i}{\frac{1}{2} k_i (k_i - 1)} = \frac{R_i}{k_i - 1}$$

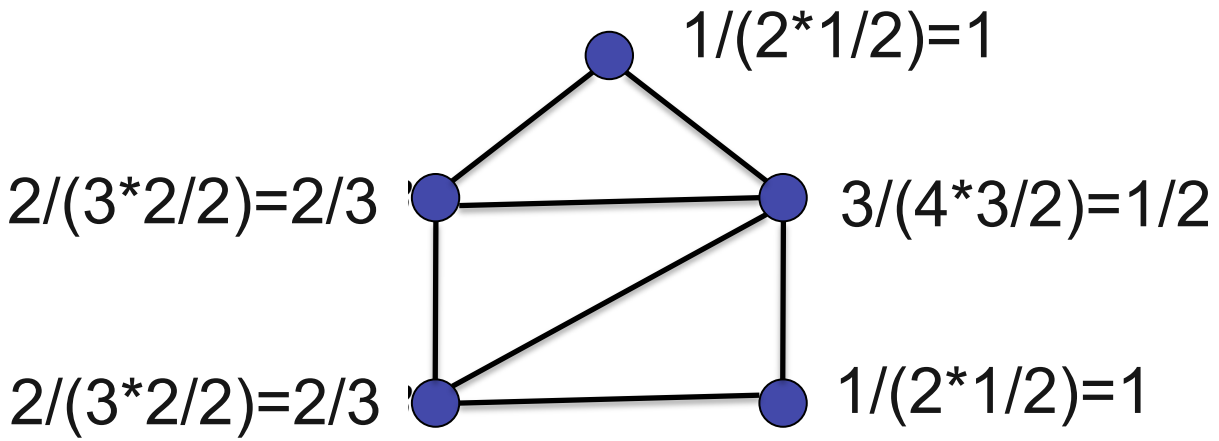
Local VS Global clustering coefficient

- **Watts and Strogatz have suggested computing the clustering coefficient of a network as the average over all the local clustering coefficients of the vertices:**

$$CC_{WS} = \frac{1}{n}\sum_{i=1}^{n} CC_i$$

- **NOTE: This definition gives different result than the previous one on global cc**
  - The above definition tends to give high cc values for networks dominated by vertices of low degree (why?) – which is regularly the case in real networks

What value for cc is high?



CC=(1+1/2+1+2/3+2/3)/5=0.7666…

What value is high?

- **The notion of high clustering coefficient is relevant**
- **We define the edge density p of a network as:**

$$p = \frac{|E|}{n(n-1)/2}$$

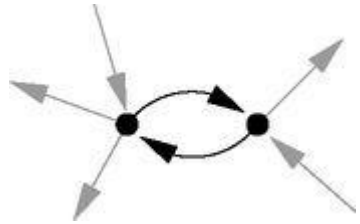
- **If we pick a pair of vertices at random, the probability that they are connected is p**
  - If we pick a pair of vertices with a common connection at random, then the probability that they are connected is C
    - ✓ Hence if C >> p, we say that the network has a high clustering coefficient
      - So is the cc high in the previous network?

Reciprocity

- **The cc is examining loops of length 3 – this is the smallest possible loop size in an undirected network**

- **However, in directed networks there can be loops of size 2**

- **<u>Reciprocity</u> measures the frequency of length 2 loops in a directed network**



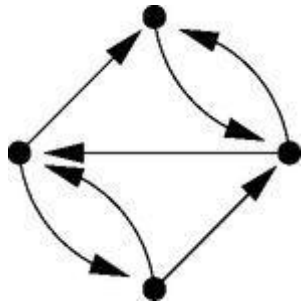
  - If there is a directed edge from i to j and there is also an edge from j to i then we say that the edge (i,j) – or the edge (j,i) – is reciprocated

Reciprocity

- **Formally, reciprocity is defined as the fraction of edges that are reciprocated**

- **$A_{ij}A_{ji}$ is one iff both edges (i,j) and (j,i) exist**
  - In other words, iff (i,j) is reciprocated
  - Hence, if m is the total number of directed edges:

$$r = \frac{1}{m}\sum_{ij} A_{ij}A_{ji} = \frac{1}{m}TrA^2$$



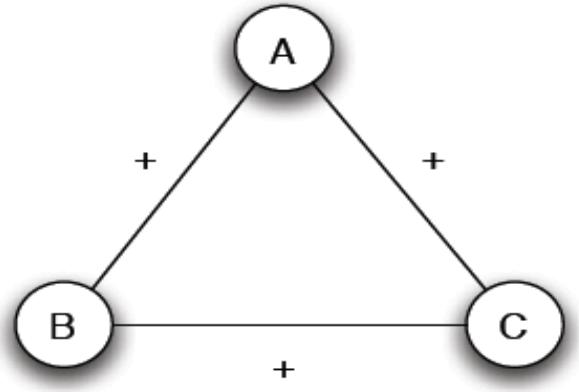
What is the reciprocity in this network?

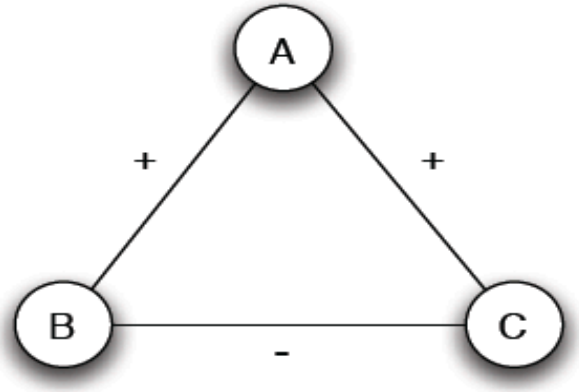
Signed edges

- **In social networks, edges can be annotated as positive or negative**
  - Positive → Friends
  - Negative → Enemies
  - Note that a negative edge represents people who interact, but in a negative way
    - ✓ It is not the same with the absence of an edge

- **A network with positive and negative labels on its edges is called signed network**
  - The corresponding edges are called signed edges

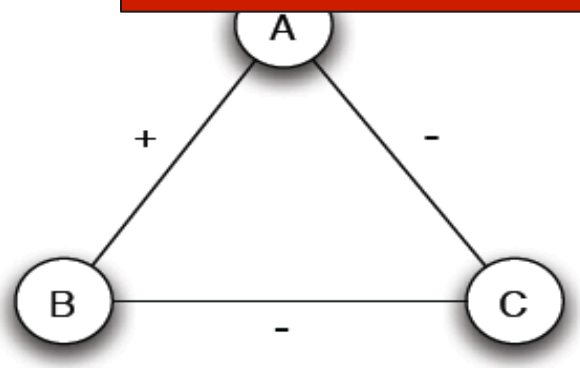
Structural balance

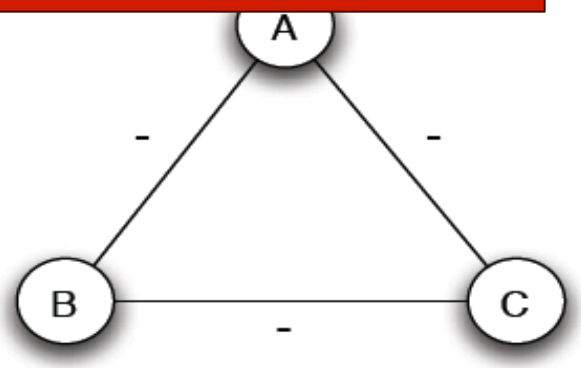




(a) *A, B, a* ... *n't get*

Triangles with 1 or 3 '+'s are *balanced.*
Triangles with 0 or 2 '+' are *unbalanced.*





(c) *A and B are friends with C as a mutual enemy: balanced.*

(d) *A, B, and C are mutual enemies: not balanced.*

Structural balance

- **Networks containing only loops with even number of minus signs are said to show structural balance**
  - This is also true for loops of length greater than three

- ***A balanced network can be divided into connected groups of vertices such that all connections between members of the same group are positive and all connections between members of different groups are negative***
  - Groups can consist even of a single node

- **A network that can be divided into groups as per the above theorem are called clusterable**
  - Hence, a balanced network is clusterable as well

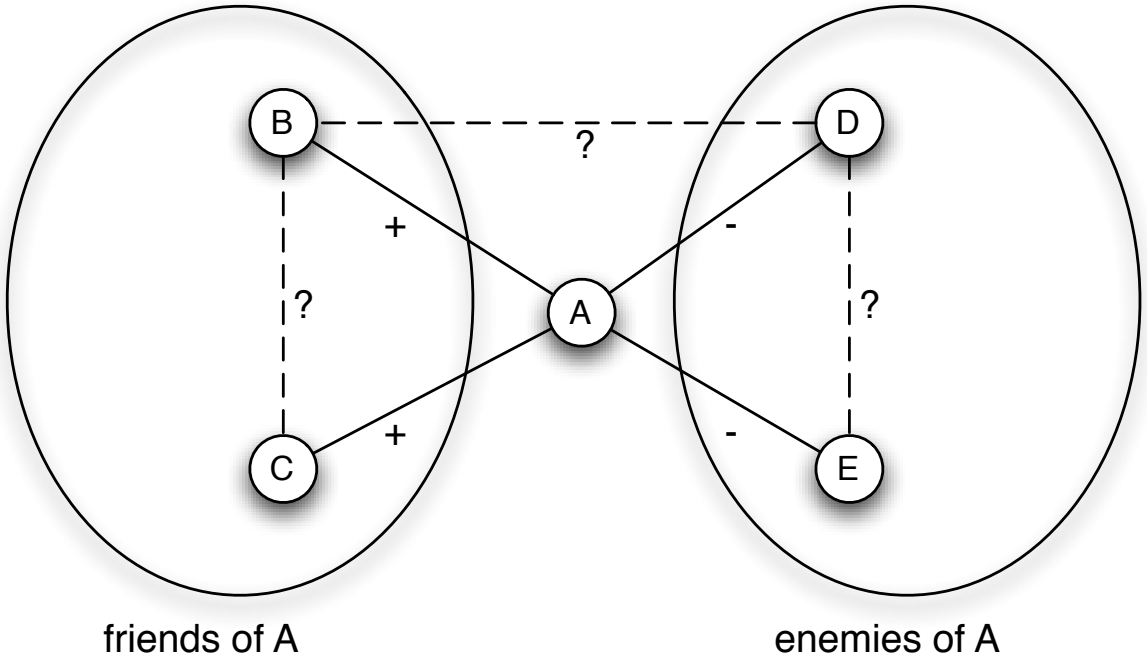
Proof of the structural balance theorem

- **We will prove it for complete graphs (2 groups X and Y)**
  - Extension to not connected graphs is trivial

- **Consider a random node A**
  - Every node is either a friend of A (+) or its enemy (-)
  - Hence, natural candidates for “clusters” are:
    - ✓Set X of A’s friends
    - ✓Set Y of A’s enemies

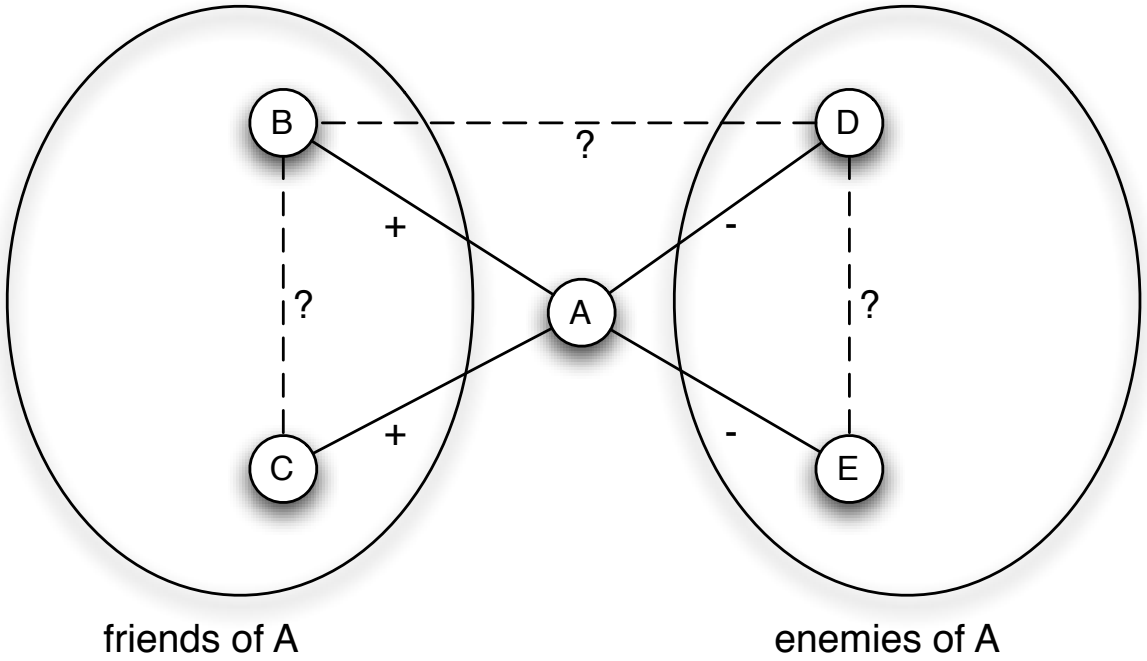
- **We must show:**
  - Every two nodes in X are friends
  - Every two nodes in Y are friends
  - Every node in X is an enemy with every node in Y

Proof of structural balance theorem



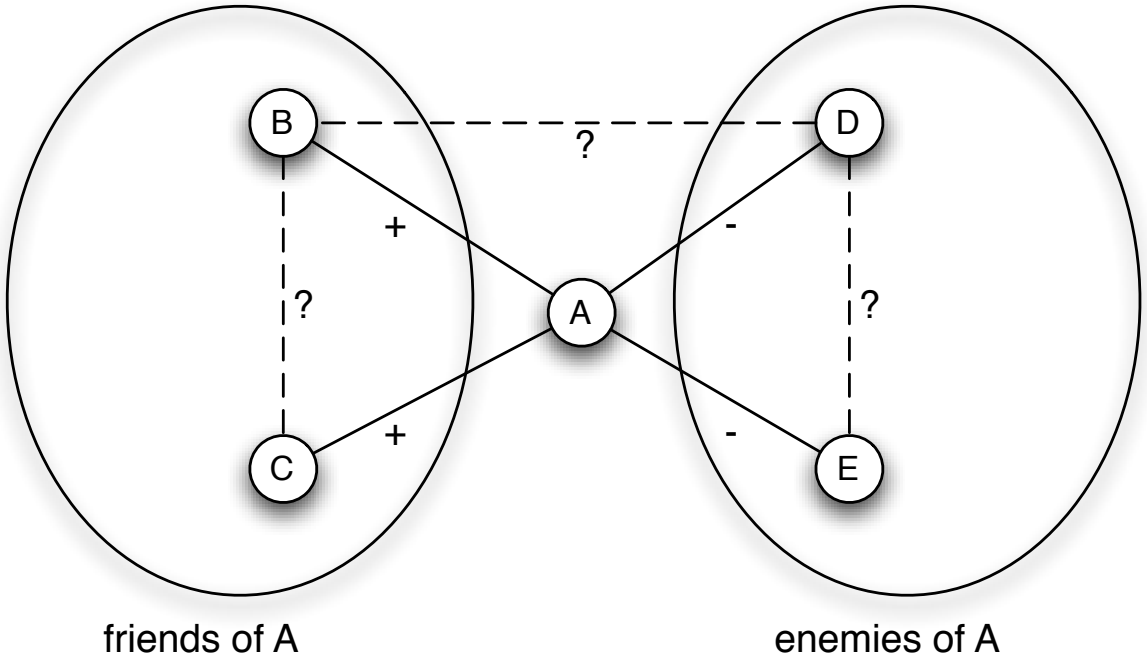
- **For A we know he is friend with any other person in X**
- **What about B and C?**
  - Since each edge (A,B) and (A,C) is +, the edge (B,C) needs to be (+) as well
    - ✓Otherwise the assumption for a balanced network would not hold

Proof of structural balance theorem



- **What about nodes in Y? Are they friends?**
- **D and E are enemies (-) with A**
  - Hence, (D,E) must be +, otherwise the balance assumption does not hold

Proof of structural balance theorem

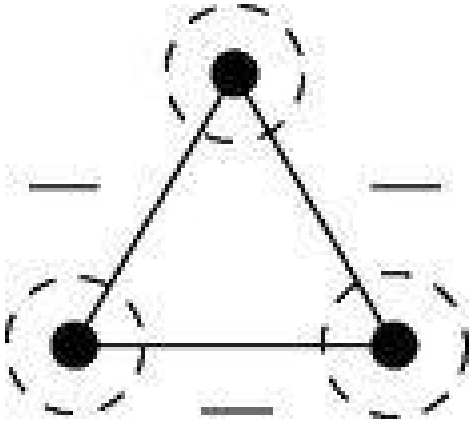


- **What about cross-set edges?**
  - E.g., (B,D)

Note

- **The structural balance theorem states that a balanced network is clusterable**
  - The opposite is NOT true

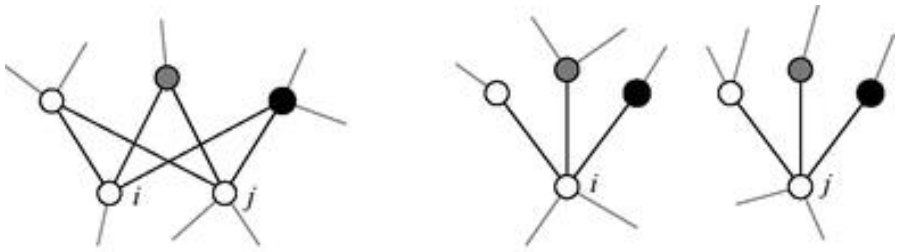
- **We can cluster the network in 3 clusters**
  - However, the network is not balanced



Similarity

- **A central concept in social network analysis is that of vertex similarity**

- **While similarity between vertices can be defined based on attributes of the vertices, we will focus on determining similarity using the information contained in the network structure**
  - Structural equivalence
    - ✓ Cosine similarity, Pearson coefficients
  - Regular equivalence



(a) Structural equivalence (b) Regular equivalence

Cosine similarity

- **A simple and straightforward metric for structural equivalence between nodes i and j, would be to measure the number of common neighbors:**

$$n_{ij} = \sum_k A_{ik} A_{kj}$$

- **However, what is a *high* value for n?**
  - We need to somehow normalize this quantity
  - Normalizing by the total number of possible edges in the network is not good (why?)

- **Cosine similarity inspired by geometry**
  - The inner product of two vectors x and y is given by:

$$\vec{x} \cdot \vec{y} = |\vec{x}||\vec{y}|\cos\theta$$

Cosine similarity

- **Rearranging we have:** $\cos\theta = \dfrac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|}$

- **Considering as the two vectors the i-th and j-th rows (or columns) of the adjacency matrix we get the cosine similarity of the two vertices:**

$$\sigma_{ij} = \cos\theta = \frac{\sum_k A_{ik}A_{kj}}{\sqrt{\sum_k A_{ik}^2}\sqrt{\sum_k A_{jk}^2}} = \frac{\sum_k A_{ik}A_{kj}}{\sqrt{k_i k_j}} = \frac{n_{ij}}{\sqrt{k_i k_j}}$$

- **Therefore, the cosine similarity normalizes the number of common neighbors by the geometric mean of the degrees of the nodes**

Pearson coefficients

- **The main idea is to compare the number of common neighbors between two vertices with what we should expect if connections were made at random**

- **If the degrees of vertices i and j are k_i and k_j respectively, then if connections were made at random the expected number of common neighbors is: $k_i k_j / n$**
  - A reasonable metric is then the difference between the actual number of common neighbors and the expected one

Pearson coefficients

$$\sum_k A_{ik}A_{jk} - \frac{k_i k_j}{n} = \sum_k A_{ik}A_{jk} - \frac{1}{n}\sum_k A_{ik}\sum_k A_{jk} =$$

$$= \sum_k A_{ik}A_{jk} - n < A_i >< A_j >= \sum_k [A_{ik}A_{jk} - < A_i >< A_j >] =$$

$$= \sum_k (A_{ik} - < A_i >)(A_{jk} - < A_j >)$$

- **The above quantity is positive if i and j have more neighbors as expected at random**
- **We further normalize the above quantity with the maximum value of the covariance between the i-th and j-th row of the adjacency matrix**
  - This is the variance of i-th or j-th row (they are both equal)

$$r_{ij} = \frac{\mathrm{cov}(A_i, A_j)}{\sigma_i \sigma_j} = \frac{\sum_k (A_{ik} - < A_i >)(A_{jk} - < A_j >)}{\sqrt{\sum_k (A_{ik} - < A_i >)^2}\sqrt{\sum_k (A_{jk} - < A_j >)^2}}$$

Other metrics of structural equivalence

- **We can normalize the number of common neighbors not by subtracting the expected number but by dividing**

$$\frac{n_{ij}}{k_i k_j / n} = n \frac{\sum_k A_{ik} A_{jk}}{\sum_k A_{ik} \sum_k A_{jk}}$$

- **Euclidian distance**
  - Dissimilarity measure

$$d_{ij} = \sum_k (A_{ik} - A_{jk})^2$$

  - We can normalize it by the maximum possible distance (all neighbors of i and j are different)

$$\frac{\sum_k (A_{ik} - A_{jk})^2}{k_i + k_j} = \frac{\sum_k (A_{ik} + A_{jk} - 2A_{ik}A_{jk})}{k_i + k_j} = 1 - 2\frac{n_{ij}}{k_i + k_j}$$

Regular equivalence

- **Two vertices are regular equivalent if they have neighbors who themselves are similar**
- **Metrics for regular equivalence are less developed**
- **i and j can be considered similar if they have neighbors k and l that are similar**

$$\sigma_{ij} = \alpha \sum_{kl} A_{ik} A_{jl} \sigma_{kl} \Rightarrow \sigma = \alpha A \sigma A$$

  - This expression does not necessarily give high values to "self-similarity", and hence, to nodes that have a lot of common neighbors

Regular equivalence

- **We can fix the above issue by adding a diagonal term:**

$$\sigma_{ij} = \alpha \sum_{kl} A_{ik} A_{jl} \sigma_{kl} + \delta_{ij} \Rightarrow \sigma = \alpha A \sigma A + I$$

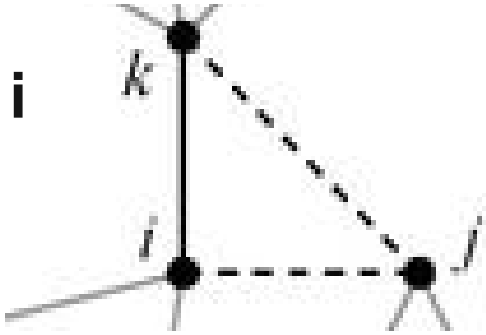
- **The above expression still has problems**
  - Assuming a null matrix as the initial value of the similarity matrix and iterating we get after k iterations:

$$\sigma^{(k)} = \sum_{m=0}^{k-1} a^{2(m-1)} A^{2m}$$

  - This measure of similarity is a weighted sum over the numbers of paths of even length between pairs of vertices
    - ✓ Why though only even length paths ?

Regular equivalence

- **New definition: Vertices i and j are similar, if i has a neighbor k that is itself similar to j:**



$$\sigma_{ij} = \alpha \sum_k A_{ik}\sigma_{kj} + \delta_{ij} \Rightarrow \sigma = aA\sigma + I$$

- **Iterating again starting with σ=0, we get:**

$$\sigma = \sum_{m=0}^{\infty} (aA)^m = (I - aA)^{-1}$$

  - The similarity of two nodes is basically the weighted sum of the number of the paths of different length that connected them
    - ✓When α<1, longer paths have smaller weight

Regular equivalence

- **The definition above inherently gives high similarity value to nodes with high degree (the summation includes more non-zero terms)**
  - Hence, in order to avoid this bias which is not always wanted we can divide with the degree of the node:

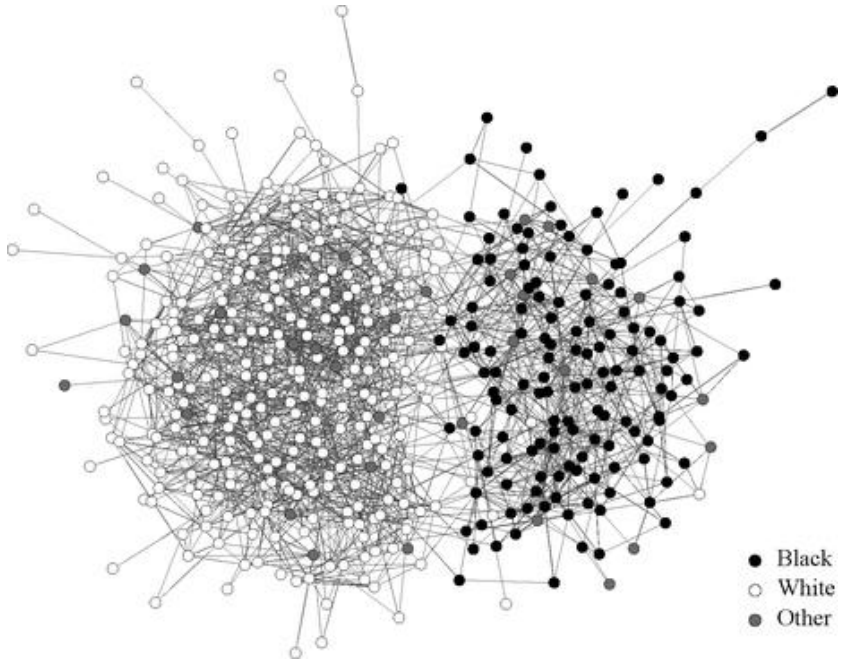
$$\sigma_{ij} = \frac{\alpha}{k_i} \sum_k A_{ik} \sigma_{kj} + \delta_{ij} \Rightarrow \sigma = (I - \alpha D^{-1} A)^{-1} = (D - \alpha A)^{-1} D$$

- **Another variations could force the additive term δ_{ij} have non-diagonal elements as well**
  - Nodes might have similarity based on other (non-network) information

Homophily and assortative mixing

- **People tend to associate with others whom they perceive as being similar to themselves in some ways**
  - Homophily or assortative mixing
  - Figure presents the friendships in a high school
    - ✓Nodes are annotated with regards to their race

- **Disassortative mixing is also possible**
  - E.g., sexual partnerships



Enumerative characteristics

- **Consider a network where each vertex is classified according to a *discrete* characteristic, that can take a finite number of values**
  - E.g., race, nationality, etc.

- **We find the number of edges that run between vertices of the same type and we subtract it from the fraction of such edges that we would expect to find if edges where placed at random (i.e., without regard for the vertex type)**
  - If the result is significantly positive → assortativity mixing is present
  - If the result is significantly negative → disassortativity mixing is present

Enumerative characteristics

- **Let c_i be the class of vertex i and let n_c possible classes**
  - The total number of edges connecting two vertices belonging to the same class is given by:

$$\sum_{edges(i,j)} \delta(c_i, c_j) = \frac{1}{2} \sum_{ij} A_{ij} \delta(c_i, c_j)$$

- **The expected number of edges between same type vertices is given by:** $\frac{1}{2} \sum_{ij} \frac{k_i k_j}{2m} \delta(c_i, c_j)$
- **Taking the difference of the above quantities we get:**

$$\frac{1}{2} \sum_{ij} A_{ij} \delta(c_i, c_j) - \frac{1}{2} \sum_{ij} \frac{k_i k_j}{2m} \delta(c_i, c_j) = \frac{1}{2} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(c_i, c_j)$$

Enumerative characteristics

- Conventionally we calculate the fraction of such edges, which is called modularity Q:

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})\delta(c_i, c_j)$$

- It is strictly less than 1 and takes positive values when there are more edges between vertices of the same type than we would expect

- The summation term above appears in many other situations and we define it to be the modularity matrix B:

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

Enumerative characteristics

- **The value of Q can be considerably lower than 1 even for perfectly mixed networks**
  - Depends on group size, degrees etc.
  - How can we decide whether a modularity value is large or not?

- **We normalize the modularity value Q, by the maximum value that it can get**
  - Perfect mixing is when all edges fall between vertices of the same type

$$Q_{\max} = \frac{1}{2m}(2m - \sum_{ij} \frac{k_i k_j}{2m}\delta(c_i, c_j))$$

- **Then, the assortativity coefficient is given by:**

$$\frac{Q}{Q_{\max}} = \frac{\sum_{ij}(A_{ij} - k_i k_j / 2m)\delta(c_i, c_j)}{2m - \sum_{ij}(k_i k_j / 2m)\delta(c_i, c_j)}$$

Enumerative characteristics

- The above equations might be computationally prohibitive to be used in large networks to compute modularity Q
- Let $e_{rs} = \frac{1}{2m}\sum_{ij} A_{ij}\delta(c_i, r)\delta(c_j, s)$ be the fraction of edges that join vertices of type r to vertices of type s
- Let $a_r = \frac{1}{2m}\sum_{ij} k_i\delta(c_i, r)$ be the fraction of ***ends*** of edges attached to vertices of type r
- We further have $\delta(c_i, c_j) = \sum_r \delta(c_i, r)\delta(c_j, r)$

Enumerative characteristics

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\sum_r \delta(c_i, r)\delta(c_j, r) =$$

$$= \sum_r \left[\frac{1}{2m}\sum_{ij} A_{ij}\delta(c_i, r)\delta(c_j, r) - \frac{1}{2m}\sum_i k_i \delta(c_i, r)\frac{1}{2m}\sum_j k_i \delta(c_j, r)\right] =$$

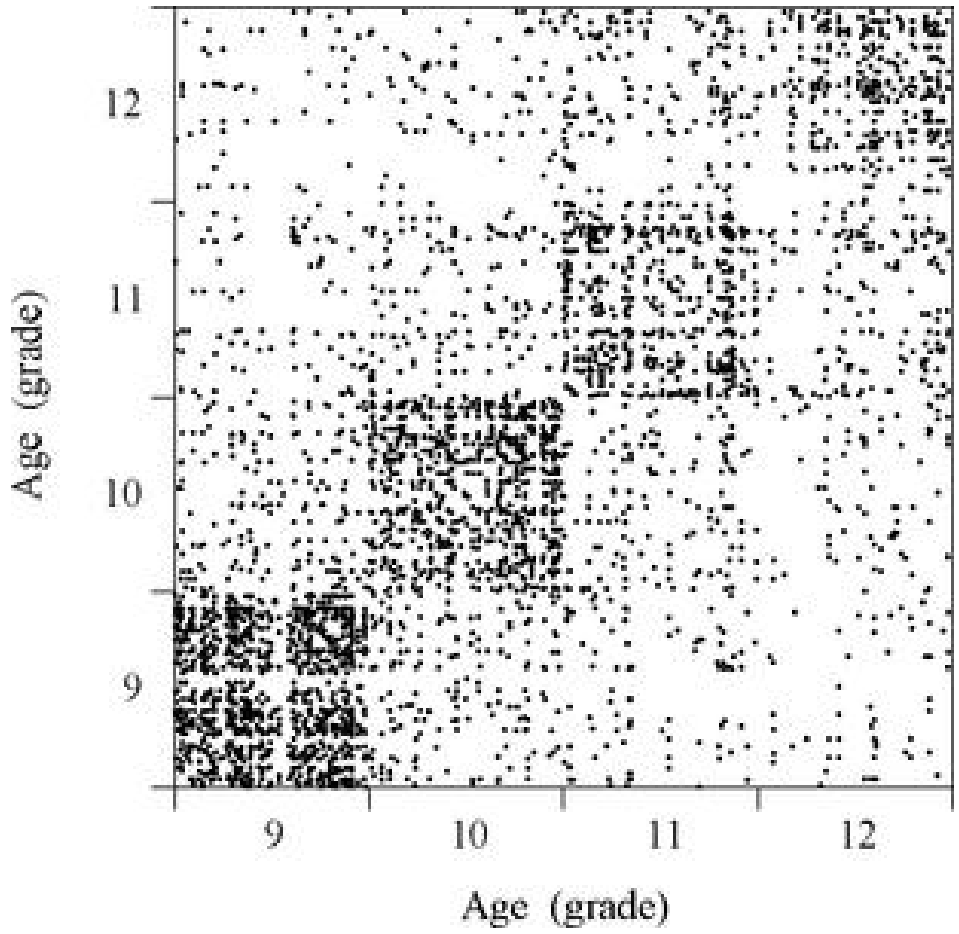
$$= \sum_r (e_{rr} - a_r^2)$$

Scalar characteristics

- **Scalar characteristics take values that come in particular order and can in theory take an infinite number of values**
  - E.g., age
  - In this case two people can be considered similar if they are born the same day or within a year or within 2 years etc.

- **When we consider scalar characteristics we basically have an approximate notion of similarity between connected vertices**
  - There is no approximate similarity when we talk for enumerative characteristics

Scalar characteristics



Scalar characteristics

- **Simple solution would be to quantize/bin scalar values**
  - Treat vertices that fall in the same bin as exactly same
  - Apply modularity metric for enumerative characteristics

- **The above misses much of the point of scalar characteristics**
  - Why two vertices that fall at the same bin are same ?
  - Why two vertices that fall in different bins are different ?

Scalar characteristics

- x_i is the value of the scalar characteristic of vertex i
- If X_i is the variable describing the characteristic at vertex i and X_j is the one for vertex j, then we want to calculate the covariance of these two variables over the edges of the graph
- First we need to compute the mean value of x_i at the end i of an edge:

$$\mu = \frac{\sum_{ij} A_{ij} x_i}{\sum_{ij} A_{ij}} = \frac{\sum_{i} k_i x_i}{\sum_{i} k_i} = \frac{1}{2m} \sum_{i} k_i x_i$$

Scalar characteristics

- **Hence, the covariance is:**

$$\mathrm{cov}(x_i, x_j) = \frac{\sum_{ij} A_{ij}(x_i - \mu)(x_j - \mu)}{\sum_{ij} A_{ij}} = \ldots = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})x_i x_j$$

- **If the covariance is positive, we have assortativity mixing, otherwise we have disassortativity**

- **As with the modularity Q we need to normalize the above value so as it gets the value of 1 for a perfectly mixed network**
  - All edges fall between vertices that have the same value x_i (highly unlikely in a real network)

Scalar characteristics

- **Setting $x_i=x_j$ for all the existing edges (i,j) we get the maximum possible value (which is the variance of the variable):**

$$\frac{1}{2m}\sum_{ij}(k_i\delta_{ij}-\frac{k_ik_j}{2m})x_ix_j$$

- **Then the assortativity coefficient r is defined as:**

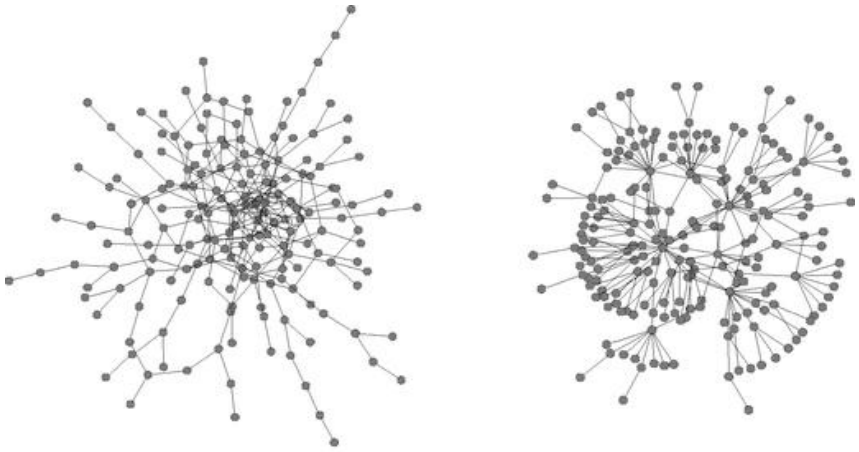
$$r=\frac{\sum_{ij}(A_{ij}-k_ik_j/2m)x_ix_j}{\sum_{ij}(k_i\delta_{ij}-k_ik_j/2m)x_ix_j}$$

  - r=1→Perfectly assortative network
  - r=-1→Perfecrtly disassortative network
  - r=0→no (linear) correlation

Assortative mixing by degree

- **A special case is when the characteristic of interest is the degree of the node**

- **In this case studying of assortative mixing will *show* if high/low-degree nodes connect with other high/low-degree nodes or not**
  - Perfectly mixed networks by degree → core/periphery
  - Disassortative networks by degree → uniform
    - ✓ Star like topology



Assortative mixing by degree

- **Assortative mixing by degree requires only structural information for the network**
  - No knowledge for any other attributes is required

$$\mathrm{cov}(k_i, k_j) = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})k_i k_j$$

$$r = \frac{\sum_{ij}(A_{ij} - k_i k_j / 2m)k_i k_j}{\sum_{ij}(k_i \delta_{ij} - k_i k_j / 2m)k_i k_j}$$

Assortative mixing by degree

- **Assortative mixing by degree requires only structural information for the network**
  - No knowledge for any other attributes is required

$$\mathrm{cov}(k_i, k_j) = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})k_i k_j$$

$$r = \frac{\sum_{ij}(A_{ij} - k_i k_j / 2m)k_i k_j}{\sum_{ij}(k_i \delta_{ij} - k_i k_j / 2m)k_i k_j}$$