

**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 1: Mathematics of Networks

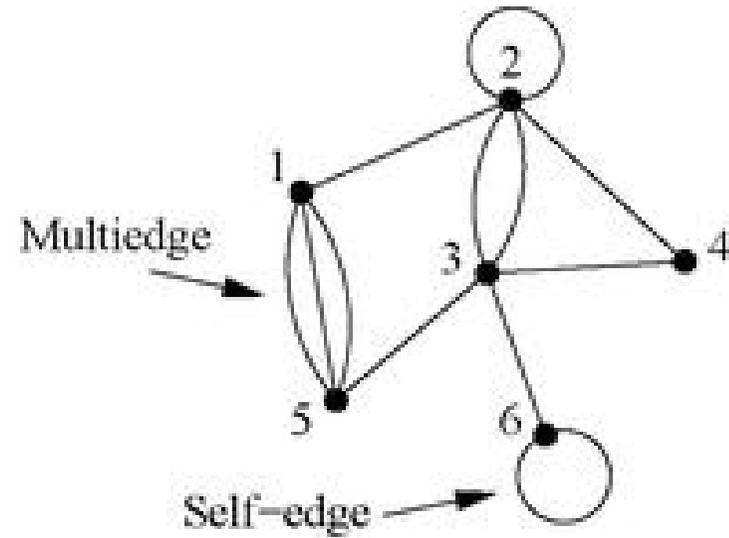
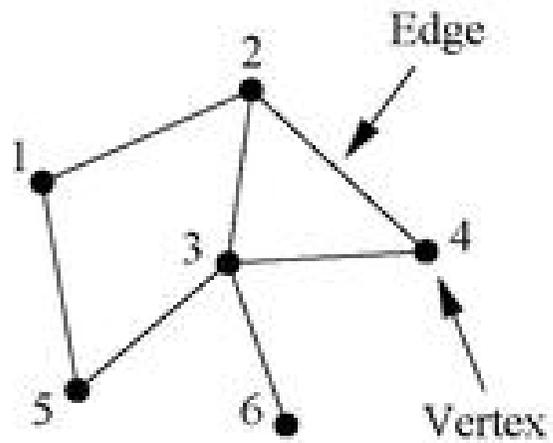
The representation of networks

- The network consists of *entities* connected with each other
- The structure of these connections are represented through graphs
- A graph is represented by two sets
 - A vertex set V of the entities participating in the network. In the rest of the slides typically, n will be the number of vertices
 - ✓ Also called node or actor set
 - An edge set E of the connections between vertices. In the rest of the slides typically, m will be the number of edges
 - ✓ Also called link or tie set

Graph terminology

- A graph whose vertices are connected by at most one link is called simple network or simple graph
 - Most of the graphs we will examine will be simple
- When two nodes connect with more than one edge, we refer to all those edges collectively as multiedge
 - The corresponding graph is called multigraph
- Depending on the type of connection a node might be connected to itself
 - Self-edges or self-loops

Example



The adjacency matrix

- If we label the nodes with IDs 1, 2, ... n we can denote each edge as a pair (i,j)
 - This is an edge list specification
 - ✓ Good for storing and processing networks in computers, but not for mathematical development
- The adjacency matrix **A** of a simple graph is a matrix with elements A_{ij} such that:

$$A_{ij} = \begin{cases} 1, & \text{if there is an edge between vertices } i \text{ and } j \\ 0, & \textit{otherwise} \end{cases}$$

Example

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

Observations

- For the (typical) case of no self-loops, the diagonal of the matrix is all 0
- For undirected networks, since the existence of edge (i,j) assumes the existence of the edge (j,i) , matrix A is symmetric
- If node i has a self edge then we set $A_{ii}=2$ (why?)
- Multiedges are represented by setting the corresponding entry equal to the number of distinct edges

Weighted networks

- **Some relations are not simple on/off (1/0) relations**
- **In a weighted network links can have weights**
 - The corresponding adjacency matrix entry is equal to the weight
 - Weights can represent the frequency of contacts between the actors, the capacity of a channel connecting two routers etc.
- **When weights are integer it might be convenient to think of the weight as multiedges**

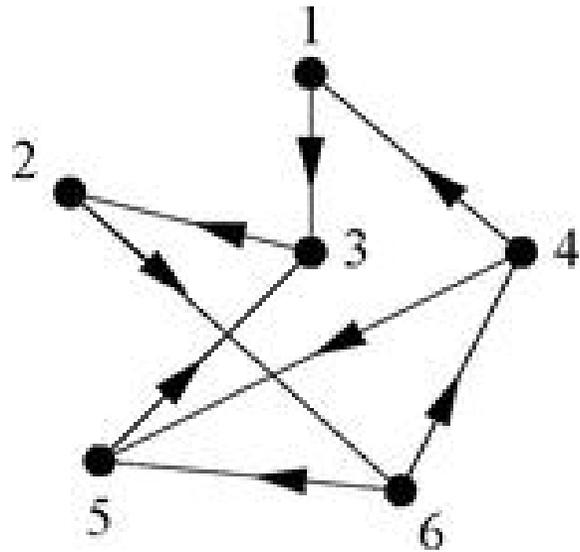
Directed networks

- In some phenomena the *direction* of the underlying relation between two nodes matters → relations are not reciprocal
 - E.g., twitter connections, world wide web links, paper citations etc.
 - These relations are captured through directed networks/graphs
 - The adjacency matrix of a directed graph (or a digraph) is given by:
- A is in general not symmetric

$$A_{ij} = \begin{cases} 1, & \text{if there is an edge from } j \text{ to } i \\ 0, & \text{otherwise} \end{cases}$$

Self edges are represented by setting $A_{ii}=1$

Example

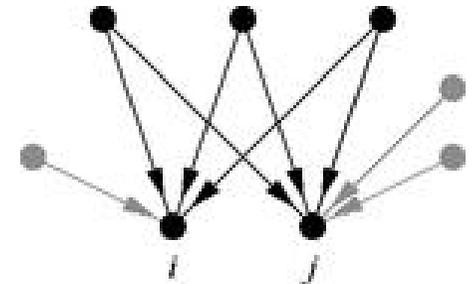


$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Co-citation coupling

- The co-citation of two vertices i and j in a directed network is the number of nodes that have outgoing edges pointing to both nodes i and j
 - So in order for node k to contribute to the co-citation of i and j , the following needs to be true: $A_{ik}A_{jk}=1$
 - Hence the co-citation of nodes i and j is:

$$C_{ij} = \sum_{k=1}^n A_{ik}A_{jk} = \sum_{k=1}^n A_{ik}A_{kj}^T$$



- The $n \times n$ adjacency matrix of the corresponding co-citation network is:

$$C = AA^T$$

Co-citation coupling

- **The non-diagonal elements can be larger than 1**
 - Either we consider it as a weighted undirected graph or we map every non-diagonal entry greater than 1 to 1

- **The diagonal elements of C can be non zero**

- Since either $A_{ik}=0$ or $A_{ik}=1$ (assuming a simple graph – no multiedges)

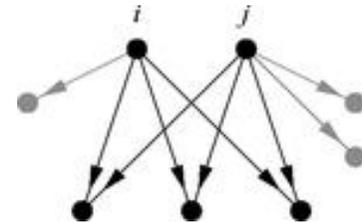
$$C_{ii} = \sum_{k=1}^n A_{ik}^2 = \sum_{k=1}^n A_{ik}$$

- C_{ii} provides the number of nodes that point to node i
 - ✓ In constructing the co-citation network we set by definition all the diagonal elements to zero!

Bibliographic coupling

- The bibliographic coupling of two vertices i and j in a directed network is the number of nodes that they both point to
 - So in order for node k to contribute to the bibliographic coupling of i and j , the following needs to be true: $A_{ki}A_{kj}=1$
 - Hence the bibliographic coupling of nodes i and j is:

$$B_{ij} = \sum_{k=1}^n A_{ki}A_{kj} = \sum_{k=1}^n A_{ik}^T A_{kj}$$



- The $n \times n$ adjacency matrix of the corresponding bibliographic coupling network is:

$$B = A^T A$$

Bibliographic coupling

- **The diagonal elements of B can be non zero**

- Since either $A_{ki}=0$ or $A_{ki}=1$ (assuming a simple graph – no multiedges)

$$B_{ii} = \sum_{k=1}^n A_{ki}^2 = \sum_{k=1}^n A_{ki}$$

- B_{ii} provides the number of nodes that node i points to
 - ✓ Again when considering the bibliographic coupling network, we set by definition all the diagonal elements to zero!

- **The off diagonal elements of B can again be greater than 0 and hence, we can define a weighted undirected network**

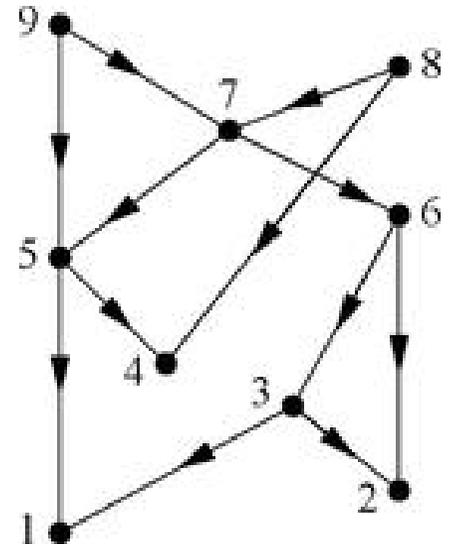
- We can also map any positive off diagonal element to 1

Notes on coupling

- **While both coupling methods are similar they can give completely different structures**
 - Co-citation is heavily based on incoming edges, while bibliographic coupling on outgoing edges
- **They can be used for vertex similarity**
 - E.g., the highest the co-citation of two papers in a paper citation network, the more possible is that these papers deal with similar topics
- **While these coupling methods help us convert a directed network to undirected, we lose some structural information during this transformation**

Acyclic directed networks

- A cycle in a directed network is a closed loop of edges with the arrows on each of the edges pointing the same way around the loop
 - Networks without cycles are called *acyclic*, while those with cycles are called *cyclic*
 - ✓ Self edges also count as cycles
- An acyclic network can be drawn with all edges pointing downward (not necessarily unique drawing)



Determining if a network is acyclic

- **A simple algorithm for determining whether a network is acyclic is:**
 - Find a vertex with no outgoing edges (step 1)
 - If no such vertex exists, the network is cyclic. Otherwise, if such a vertex does exist, remove it and all its incoming edges from the network (step 2)
 - If all vertices have been removed, the network is acyclic. Otherwise go back to step 1

Adjacency matrix of an acyclic network

- If we construct an ordering of the vertices of an acyclic graph as per the previous algorithm, there can be an edge from vertex j to vertex i only if $j > i$
 - Adjacency matrix is triangular (only the elements above the diagonal have non zero entries)
 - ✓ More precisely, since the acyclic graph does not have self loops, the diagonal elements are zero \rightarrow strictly triangular matrix
- All of the eigenvalues of an adjacency matrix are zero if and only if the network is acyclic

Bipartite networks

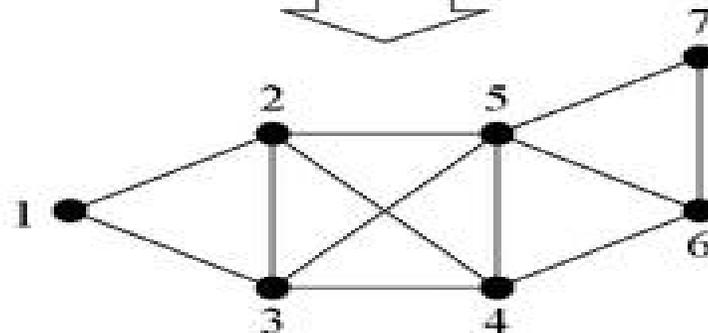
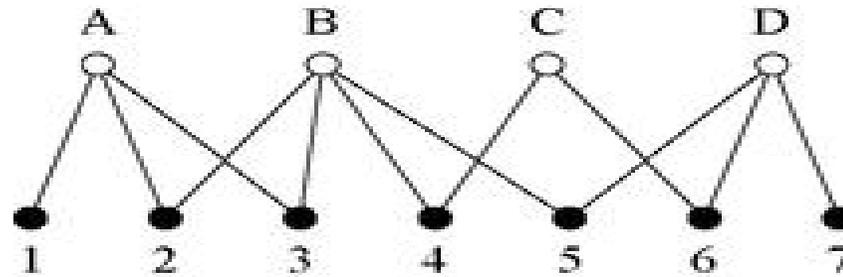
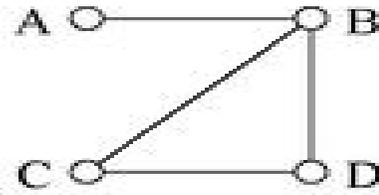
- In bipartite networks (aka *two-mode networks*) there are two kinds of vertices
 - One type represents that original vertices (e.g., people) and the other represents the groups – or affiliations or foci – that the first type of vertices belong to (e.g., company)
- The edges of a bipartite network run only between nodes of different type
- A bipartite network is defined by the incidence matrix B . If there are n number of actors and g number of groups:

$$B_{ij} = \begin{cases} 1, & \text{if vertex } j \text{ belongs to group } i \\ 0, & \textit{otherwise} \end{cases}$$

One-mode projections

- **We can use the bipartite network to infer connections between nodes of the same type**
 - Projection on the actors \rightarrow n -vertex network, where nodes are the actors of the original bipartite graph and an edge between two actors exists if they participate to at least one common group
 - Projection on the groups \rightarrow g -vertex network, where nodes are the groups of the original bipartite graph and an edge between two groups exists if they include at least one common actor
- **One mode projections discard a lot of the information present in the structure of the original bipartite graph**

Projections



Mathematics of actors one-mode projection

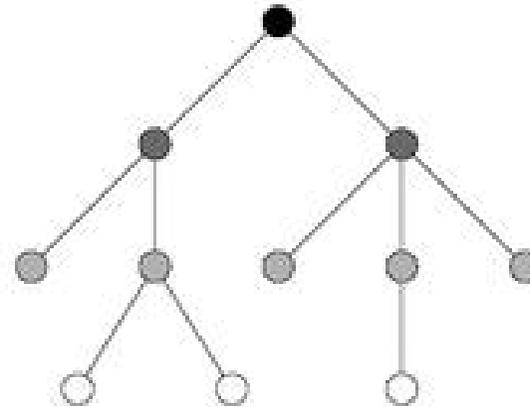
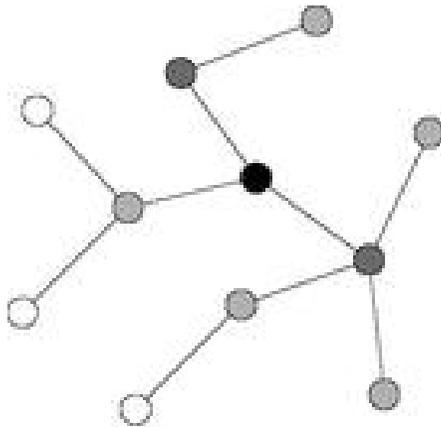
- **Two actors i and j belong to group k iff $B_{ki}B_{kj}=1$**
 - The total number of groups P_{ij} that i and j belong is given by:

$$P_{ij} = \sum_{k=1}^g B_{ki}B_{kj} = \sum_{k=1}^g B_{ik}^T B_{kj}$$

- **The $n \times n$ matrix $P=B^T B$ is similar to the adjacency matrix for the weighted one-mode projection onto the n vertices (actors)**
 - Why only *similar* and not the same?

Trees

- **Connected, undirected network with no closed loops**
 - They are usually drawn in a *rooted* manner
 - ✓ A *root node* at the top and a branching structure going down
 - ✓ Vertices at the bottom of the tree, connected only to one other vertex are called *leaves*



Trees

- **Since there are no closed loops there is exactly one path from every vertex to any other**
 - This property is important because it makes some calculations particularly simple
- **A tree with n vertices has exactly $n-1$ edges**
 - Reverse is also true!

Degree

- The degree k_i of a vertex i in a graph is the number of edges connected to it
 - For undirected graphs we have:

$$k_i = \sum_{j=1}^n A_{ij}$$

- And the number of edges of a graph is given by:

$$m = \frac{1}{2} \sum_{i=1}^n k_i = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n A_{ij}$$

- Mean degree c of a vertex in an undirected graph is:

$$c = \frac{1}{n} \sum_{i=1}^n k_i = \frac{2m}{n}$$

Connectance

- The maximum number of possible edges in a simple graph is

$$\binom{n}{2} = \frac{1}{2}n(n-1)$$

- Connectance or density ρ of a graph is the fraction of these edges that are actually present:

$$\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)} = \frac{c}{n-1}$$

- A network for which the density ρ tends to a (positive) constant as $n \rightarrow \infty$ is called dense
- A network for which the density ρ tends to zero as $n \rightarrow \infty$ is called sparse
 - ✓ This basically means that in a sparse network c tends to a constant when n increases arbitrarily

Degrees in directed networks

- In a directed network each vertex is associated with two degrees
 - In-degree is the number of incoming edges to a vertex
 - Out-degree is the number of outgoing edges from a vertex

$$k_i^{in} = \sum_{j=1}^n A_{ij} \quad k_i^{out} = \sum_{j=1}^n A_{ji}$$

- The total number of edges in a directed network is:

$$m = \sum_{i=1}^n k_i^{in} = \sum_{i=1}^n k_i^{out} = \sum_{ij} A_{ij}$$

- Thus, mean in (c_{in}) and out (c_{out}) degrees are equal

$$c_{in} = \frac{1}{n} \sum_{i=1}^n k_i^{in} = \frac{1}{n} \sum_{i=1}^n k_i^{out} = c_{out} = c = \frac{m}{n}$$

Paths

- **A sequence of vertices such that every consecutive pair of vertices in the sequence is connected by an edge in the network**
 - For directed graphs the edges traversed from the path needs to be traversed in the correct direction
 - Paths that do not intersect with themselves are called self-avoiding paths
- **Length of a path is the number of edges traversed along the path**
 - When a path traverses the same edge e two times, e is counted twice

Paths in undirected networks

- The product $A_{ik}A_{kj}$ is 1 iff there is a path between i and j through k (a path of length 2)

- Hence, if we want to find how many length 2 paths between i and j exist:

$$N_{ij}^{(2)} = \sum_{k=1}^n A_{ik}A_{kj} = [A^2]_{ij}$$

- This can easily be generalized to: $N_{ij}^{(r)} = [A^r]_{ij}$

- $[A^r]_{ii}$ gives the number of length r paths that originate and end at node i (cycles)

- Hence if we want to find the number L_r of length r cycles in a graph

$$L_r = \sum_{i=1}^n [A^r]_{ii} = \text{Tr}A^r$$

Paths in undirected networks

- **A is symmetric** → has n real, non-negative eigenvalues
 - $A = UKU^T$
 - ✓ U orthogonal matrix of eigenvectors
 - ✓ K diagonal matrix with eigenvalues

$$A^r = (UKU^T)^r = UK^rU^T$$

$$L_r = \text{Tr}A^r = \text{Tr}(UK^rU^T) = \text{Tr}(U^TUK^r) = \text{Tr}K^r = \sum_i \kappa_i^r$$

κ_i is the i-th eigenvalue of matrix A

Paths in directed networks

- The adjacency matrix of a directed network is in general non symmetric and hence not diagonalizable
- We will use the Schur decomposition
 - $A=QTQ^T$
 - ✓ Q is orthogonal
 - ✓ T is upper triangular
 - Its diagonal elements are its eigenvalues
 - *They are equal with those of A (why?)

$$L_r = \text{Tr}A^r = \text{Tr}(QT^r Q^T) = \text{Tr}(Q^T Q T^r) = \text{Tr}T^r = \sum_i K_i^r$$

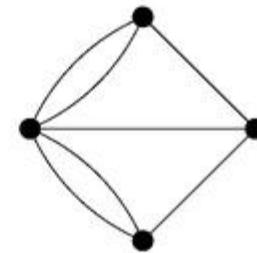
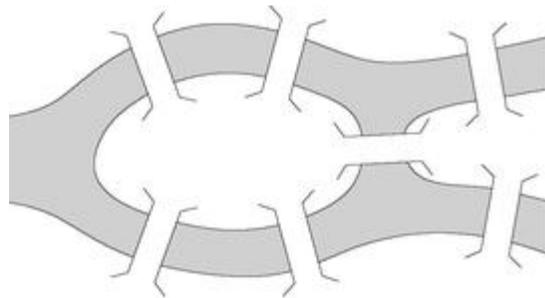
Geodesic paths

- A geodesic path (shortest path) is a path between two vertices such that no shorter path exists
 - The length of this path is called geodesic (or shortest) distance
 - If two nodes are not connected with any path their geodesic distance is infinite
- By definition shortest paths are self avoiding (why?)
- Diameter of a network is the length of the longest geodesic path between any pair of vertices in the network for which a path actually exists
 - Other definitions are extensively used in the literature as well, such as, the average value of all geodesic paths in the network etc.

Eulerian and Hamiltonian paths

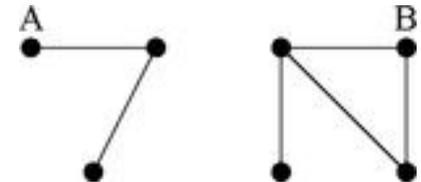
- An Eulerian path is a path that traverses each edge in a network exactly once
- A Hamiltonian path is a path that visits each vertex exactly once
 - Self avoiding

Konisberg Bridge Problem



Components

- A network for which there exists pairs of vertices that there is no path between them is called disconnected
 - If there exists a path between any possible pair of vertices in a network the latter is called connected



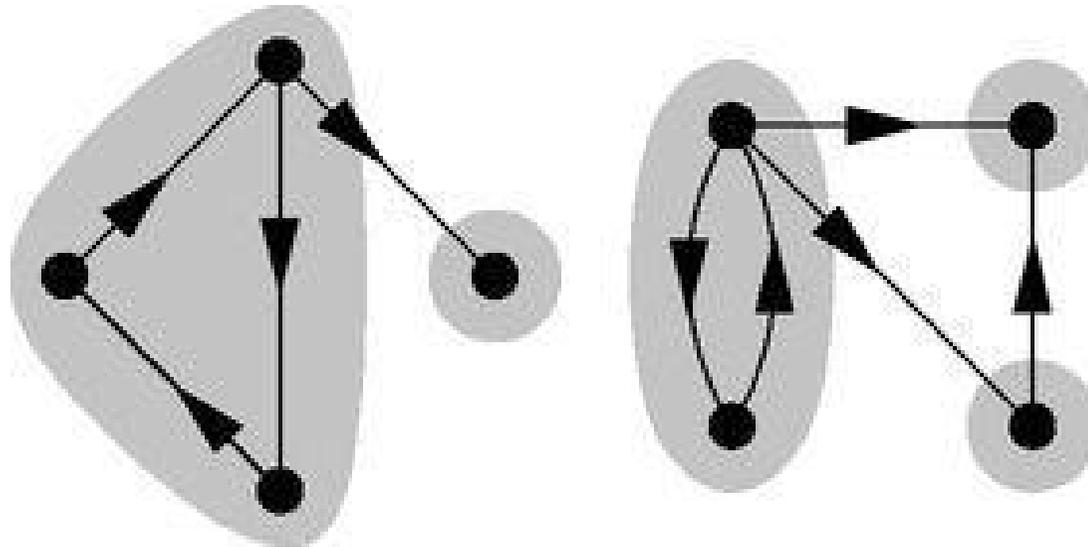
- Component is a *maximal* subset of vertices of a network such that there exists at least one path from every vertex of the subgroup to any other

How can the adjacency matrix of a network with more than one component be written?

Components in directed networks

- There are many different types of connected components that we can define for a directed network
- Assuming no direction on the edges, we can identify connected components as in an undirected graph
 - Weakly connected components
- Imposing a constraint on the direction of the edge we get the strongly connected components
 - Maximal subsets of vertices such that there is a directed path in both directions between every pair in the subset
 - ✓ Every vertex belonging to a strongly connected component with more than one vertex must belong to at least one cycle (why?)

Visually



Out-components

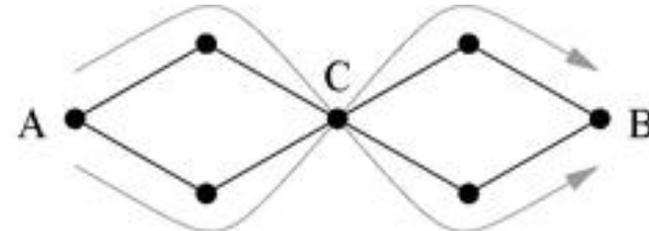
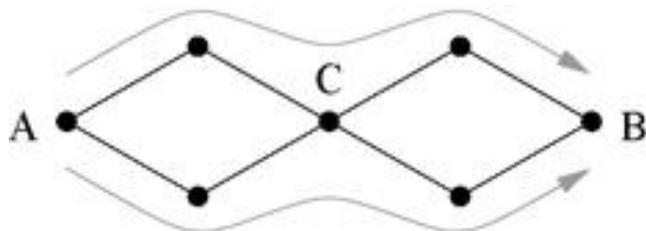
- **The set of vertices that are reachable via directed paths from node A (including A), form its out-component**
 - Out-component is a property of both the network and the starting vertex
 - ✓ Hence, a vertex can belong to more than one out-components
- **All members of the strongly connected component that A belongs to, are members of its out-component**
- **All vertices that are reachable from A are also reachable from any other member of its strongly connected component**
 - Out-components really “belong” not to individual vertices but to strongly connected components

In-components

- The in-component of a specific vertex A is the set of all vertices (including A) from which there is a directed path to A
- All the members of a strongly connected component have the same in-component
- A 's strongly connected component is the intersection of its out- and in-components

Independent paths and connectivity

- Two paths connecting a given pair of vertices are edge (vertex)-independent if they share no edges (vertices other than the starting and ending vertices)
 - Two vertex-independent paths are also edge-independent (the opposite is not true)
- The number of independent paths between a pair of vertices is called connectivity of the vertices
 - Edge- and vertex-connectivity



Cut set

- A vertex cut set, is a set of vertices whose removal will disconnect a specified pair of vertices
 - This set can be thought as the *bottleneck* for the connectivity of this specific pair of nodes
- An edge cut set, is a set of edges whose removal will disconnect a specified pair of vertices
- A minimum cut set is the smallest cut set that will disconnect a specified pair of vertices

Menger's theorem

- **The size of the minimum vertex cut set that disconnects a given pair of vertices in a network is equal to the vertex connectivity of the same vertices**
 - The same holds true for edges
- **The edge version of the theorem is important for the maximum flow problem**

Max-flow/min-cut theorem

- **In the general case we have weighted networks**
 - Minimum edge cut set is a cut set such that the sum of the weights on the edges has the minimum possible value
- **The maximum flow between a given pair of vertices in a network is equal to the sum of the weights on the edges of the minimum edge cut set that separates the same two vertices**
 - Intuitively, the “low” weight edges form bottlenecks that do not allow the flow between the two vertices to increase.

Diffusion

- Suppose we have some commodity on the vertices of a network (vertex i has amount ψ_i)
- The commodity moves along an edges from j to i with a rate $C(\psi_j - \psi_i)$
 - C is a constant (diffusion constant)
 - Hence, the rate at which ψ_i changes is:

$$\frac{d\psi_i}{dt} = C \sum_j A_{ij} (\psi_j - \psi_i) \Rightarrow$$

$$\frac{d\psi_i}{dt} = C \sum_j A_{ij} \psi_j - C \psi_i \sum_j A_{ij} = C \sum_j A_{ij} \psi_j - C \psi_i k_i = C \sum_j (A_{ij} - \delta_{ij} k_i) \psi_j$$

Diffusion

- Using matrix notation we have

$$\frac{d\vec{\psi}}{dt} = C(A - D)\vec{\psi}$$

- D is a diagonal matrix with the vertex degrees along the diagonal

- We define: $L = D - A$ and hence: $\frac{d\vec{\psi}}{dt} + CL\vec{\psi} = \vec{0}$

- The above is similar to the ordinary diffusion equation for gas, where the Laplacian operator has been replaced by matrix L

- ✓ L is called graph Laplacian $L_{ij} = \delta_{ij}k_i - A_{ij}$

Solving the diffusion equation

- We write vector ψ as a linear combination of the eigenvectors v_i of the Laplacian (why is this possible?):

$$\vec{\psi}(t) = \sum_{i=1}^n \alpha_i(t) \vec{v}_i$$

- We substitute at the diffusion equation and we get:

$$\frac{d \sum_{i=1}^n \alpha_i(t) \vec{v}_i}{dt} + CL \sum_{i=1}^n \alpha_i(t) \vec{v}_i = \vec{0} \Rightarrow \sum_{i=1}^n \left(\frac{d\alpha_i(t)}{dt} \vec{v}_i + C\alpha_i(t) L\vec{v}_i \right) = \vec{0} \Rightarrow \sum_{i=1}^n \left(\frac{d\alpha_i(t)}{dt} + C\lambda_i \alpha_i(t) \right) \vec{v}_i = \vec{0}$$

- Since the Laplacian is symmetric its eigenvectors are orthogonal \rightarrow multiplying with any eigenvector v_j we get:

$$\frac{d\alpha_i(t)}{dt} + C\lambda_i \alpha_i = 0, \forall i \Rightarrow \alpha_i(t) = \alpha_i(0) e^{-C\lambda_i t}$$

Eigenvalues of the Laplacian

- The eigenvalues of the Laplacian are real (L is symmetric) and also non-negative
- Let one end of an edge annotated with 1 and the other with 2. Then we can define the $m \times n$ edge incidence matrix

$$B_{ij} = \begin{cases} +1 & \text{if end 1 of edge } i \text{ is attached to vertex } j, \\ -1 & \text{if end 2 of edge } i \text{ is attached to vertex } j, \\ 0 & \text{otherwise.} \end{cases}$$

- $B_{ki}B_{kj}$ is non zero (actually -1) iff vertices i and j are connected through edge k
 - Since we consider a simple graph, taking the sum of all the above factors over k will be either -1 (iff vertices i and j are connected through an edge) or 0 (iff i and j are not neighbors)

Eigenvalues of the Laplacian

- If $i=j$, the previous sum is equal to the degree k_i of vertex i (why?)

- Hence, **$L=B^T B$**

- Consider \vec{v}_i being an eigenvector of L with eigenvalue λ_i

$$L\vec{v}_i = \lambda_i \vec{v}_i \implies \vec{v}_i^T L\vec{v}_i = \vec{v}_i^T \lambda_i \vec{v}_i \implies \vec{v}_i^T B^T B\vec{v}_i = \lambda_i \vec{v}_i^T \vec{v}_i \implies (B\vec{v}_i)^T \cdot (B\vec{v}_i) = \lambda_i$$

- Hence **$\lambda_i \geq 0$**

- It is easy to see that $(1,1,\dots,1)$ is an eigenvector of the L with eigenvalue 0

Components and algebraic connectivity

- **Suppose a network with c components**

- First component has n_1 vertices, second has n_2 and the c -th has n_c
- The Laplacian (just as the adjacent matrix) can be written in a block diagonal form

- There are c eigenvectors of L with eigenvalue zero \rightarrow vectors that have ones in all positions corresponding to vertices in single components and zero elsewhere. E.g., $\mathbf{v} = (\underbrace{1, 1, 1, \dots}_{n_1 \text{ ones}}, \underbrace{0, 0, 0, \dots}_{\text{zeros}})$

$$L = \begin{pmatrix} \square & 0 & \dots \\ 0 & \square & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

- **The number of zero eigenvalues is exactly equal to the number of connected components**

- The second eigenvalue λ_2 (algebraic connectivity or spectral gap) of the Laplacian is non-zero iff the network is connected

Random walks

- A random walk is a path across a network created by taking repeated random steps
 - Start at a specific vertex
 - Choose uniformly at random one of the edges of this vertex and move along the edge
 - Arrive at the vertex at the other end of the edge and repeat the above steps
- Let $p_i(t)$ be the probability that after t random steps the walk is at vertex i
 - If the walk is at vertex j at time $t-1$, the probability of taking a step along any particular one of the k_j edges of j is $1/k_j$

$$p_i(t) = \sum_j \frac{A_{ij}}{k_j} p_j(t-1) \Rightarrow \vec{p}(t) = AD^{-1}\vec{p}(t-1)$$

Random walks

- Since we are interested for the steady state distribution (i.e., $t \rightarrow \infty$), we have:

$$p_i(\infty) = \sum_j \frac{A_{ij}}{k_j} p_j(\infty) \Rightarrow \vec{p} = AD^{-1}\vec{p}$$

- The above equation can be written as:

$$(I - AD^{-1})\vec{p} = (D - A)D^{-1}\vec{p} = LD^{-1}\vec{p} = \vec{0}$$

- Hence, $D^{-1}\mathbf{p}$ is an eigenvector of L with zero eigenvalue

Random walk on a connected graph

- In this case we know that there is only one null eigenvalue, which corresponds to an eigenvector with all components equal:

$$D^{-1}\vec{p} = \alpha\vec{1} \Rightarrow \vec{p} = \alpha D\vec{1} \Rightarrow p_i = \alpha k_i$$

- In a connected network the probability that a random walk will be found in a vertex is proportional to its degree
- If we pick constant α to normalize p_i properly we have:

$$p_i = \frac{k_i}{\sum_j k_j} = \frac{k_i}{2m}$$

Random walk and first passage time

- The first passage time for a random walk from a vertex u to another vertex v is the number of steps before a walk starting at u first reaches v
- Absorbing random walk: there is one or more absorbing states, that is, vertices that the walk can arrive but not leave
- We can answer questions about the first passage time by considering the probability $p_v(t)$ that a walk is at vertex v (absorbing) after time t
 - The probability that a walk has first passage time exactly t is then $p_v(t) - p_v(t-1)$
 - Then the mean first passage time τ is:

$$\tau = \sum_{t=0}^{\infty} t [p_v(t) - p_v(t-1)]$$

Random walk and first passage time

- Here we have a “semi”-directed graph; the walk can get in v (A_{vi} is not necessarily 0) but cannot move out of it ($A_{iv} = 0$ for all i)

- For any $i \neq v$
$$p_i(t) = \sum_j \frac{A_{ij}}{k_j} p_j(t-1) = \sum_{j(\neq v)} \frac{A_{ij}}{k_j} p_j(t-1)$$

- In the sum there are no terms A_{vj} and hence we can write:

$$\vec{p}'(t) = A'D'^{-1}\vec{p}'(t-1) \Rightarrow \vec{p}'(t) = [A'D'^{-1}]^t \vec{p}'(0)$$

- Where vector p' does not include the entry for v and matrices A' and D' do not include the v -th row and column

Random walk and first passage time

- Since \mathbf{p}' does not include the entry of p_v we have:

$$p_v(t) = 1 - \sum_{i(\neq v)} p_i(t) = 1 - \vec{1}^T \vec{p}'(t)$$

- Substituting we get the mean passage time:

$$\begin{aligned} \tau &= \sum_{t=0}^{\infty} t[p_u(t) - p_u(t-1)] = \sum_{t=0}^{\infty} t[1 - \vec{1}^T \vec{p}'(t) - 1 + \vec{1}^T \vec{p}'(t-1)] = \sum_{t=0}^{\infty} t \vec{1}^T [\vec{p}'(t-1) - \vec{p}'(t)] = \\ &= \sum_{t=0}^{\infty} t \vec{1}^T [(A'D'^{-1})^{t-1} \vec{p}'(0) - (A'D'^{-1})^t \vec{p}'(0)] = \sum_{t=0}^{\infty} t \vec{1}^T \vec{p}'(0) [(A'D'^{-1})^{t-1} - (A'D'^{-1})^t] \quad \Rightarrow \\ &\tau = \vec{1}^T [I - A'D'^{-1}]^{-1} \vec{p}'(0) \end{aligned}$$

Random walk and first passage time

- If we further simplify:

$$[I - A'D'^{-1}]^{-1} = I - (A'D'^{-1})^{-1} = I - D'A'^{-1} = D'(D'^{-1} - A^{-1}) = D'(D' - A')^{-1} = D'L'^{-1}$$

- We get:

$$\tau = \vec{1}^T D'L'^{-1} \vec{p}'(0)$$

- Where L' is the v-th reduced Laplacian