- “Donat el caràcter i la finalitat exclusivament docent i eminentment il·lustrativa de les explicacions a classe d'aquesta presentació, l’autor s’acull a l’article 32 de la Llei de propietat intel·lectual vigent respecte de l'ús parcial d'obres alienes com ara imatges, gràfics o altre material contingudes en les diferents diapositives”

- “Dado el carácter y la finalidad exclusivamente docente y eminentemente ilustrativa de las explicaciones en clase de esta presentación, el autor se acoge al artículo 32 de la Ley de Propiedad Intelectual vigente respecto al uso parcial de obras ajenas como imágenes, gráficos u otro material contenidos en las diferentes diapositivas”.

UNIVERSITAT DE BARCELONA

Complex Networks

Albert Díaz Guilera

http://diaz-guilera.net

@anduviera

complexity lab barcelona

COMPLEXITAT

UNIVERSITAT DE BARCELONA

Characterizing Complex Networks

Albert Díaz Guilera

http://diaz-guilera.net

@anduviera

complexity lab barcelona

COMPLEXITAT

Levels of characterization

- Microscale: role of nodes in the network (centrality, degree, betweenness, …)
- Macroscale: distributions, statistical properties
- Mesoscale: motifs, modules, communities, …

Microscale

- Centrality
- Degree (local perspective)
- Other measures (global perspective)

Degree (microscopic scale)

- Number of links that a node has
- It corresponds to the local centrality in social network analysis
- It measures how important is a node with respect to its nearest neighbors

Degree centrality

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

The following are some elementary facts about the degree centrality. You are invited to prove these yourself.

1. $k_i = (A^2)_{ii}$,
2. $\sum_{i=1}^{n} k_i = 2m$, where m is the number of links,
3. $\sum_{i=1}^{n} k_i^{in} = \sum_{i=1}^{n} k_i^{out} = m$, where m is the number of links.

- Let us consider the network illustrated in Figure 14.1

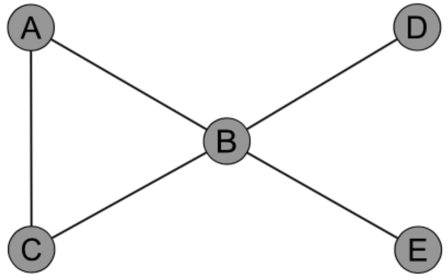


Figure 14.1: A simple labelled network

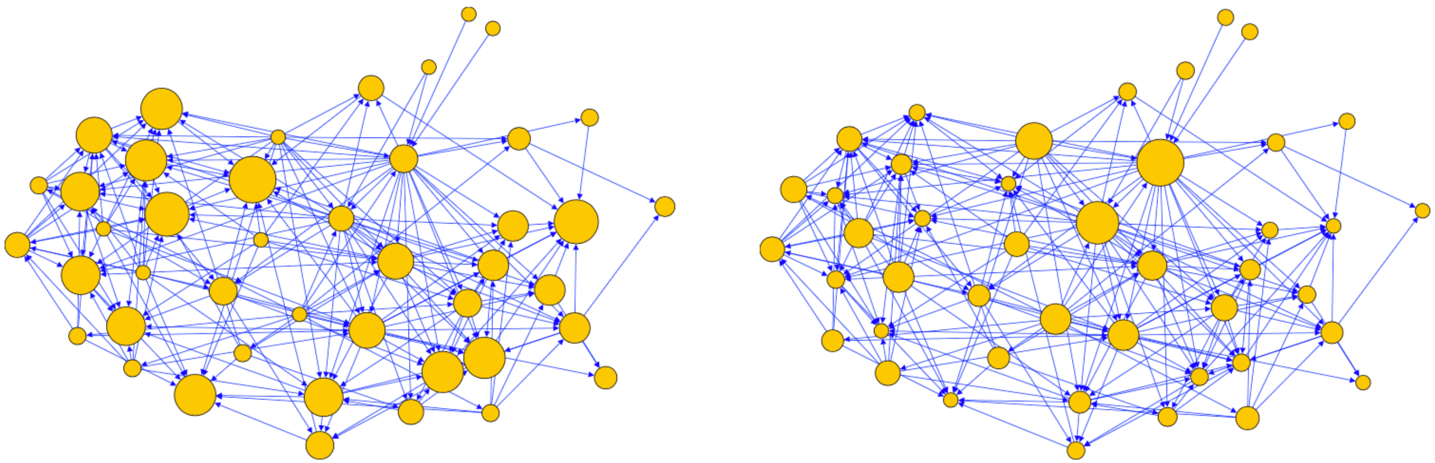
Since the adjacency matrix of the network is

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

the node degree vector is

$$\mathbf{k} = A\mathbf{e} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 2 \\ 1 \\ 1 \end{bmatrix}.$$

- Let us now consider a real-world network. It corresponds to the food web of St Martin island in the Caribbean, in which nodes represent species and food sources and the directed links indicate what eats what in the ecosystem. Here we represent the networks in Figure 14.3 by drawing the nodes as circles with radius proportional to the corresponding in-degree in (a) and out-degree in (b).

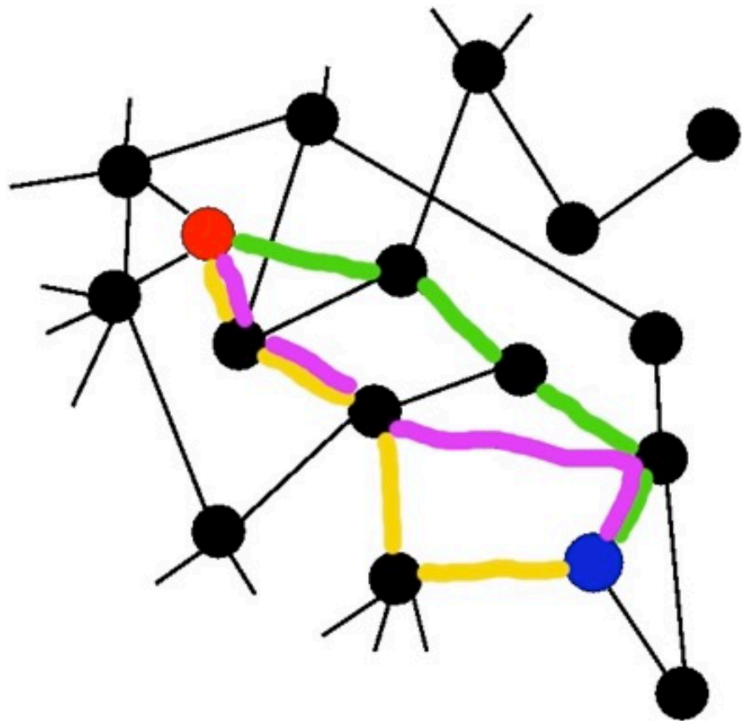


(a) (b)

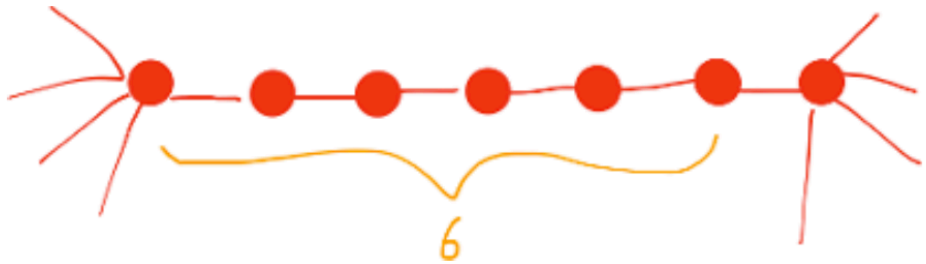
Figure 14.3: Food webs in St Martin with nodes drawns as circles of radii proportional to in-degree (a) and out-degree (b)

Distance between two nodes

- Number of links that make up the shortest-path between two nodes

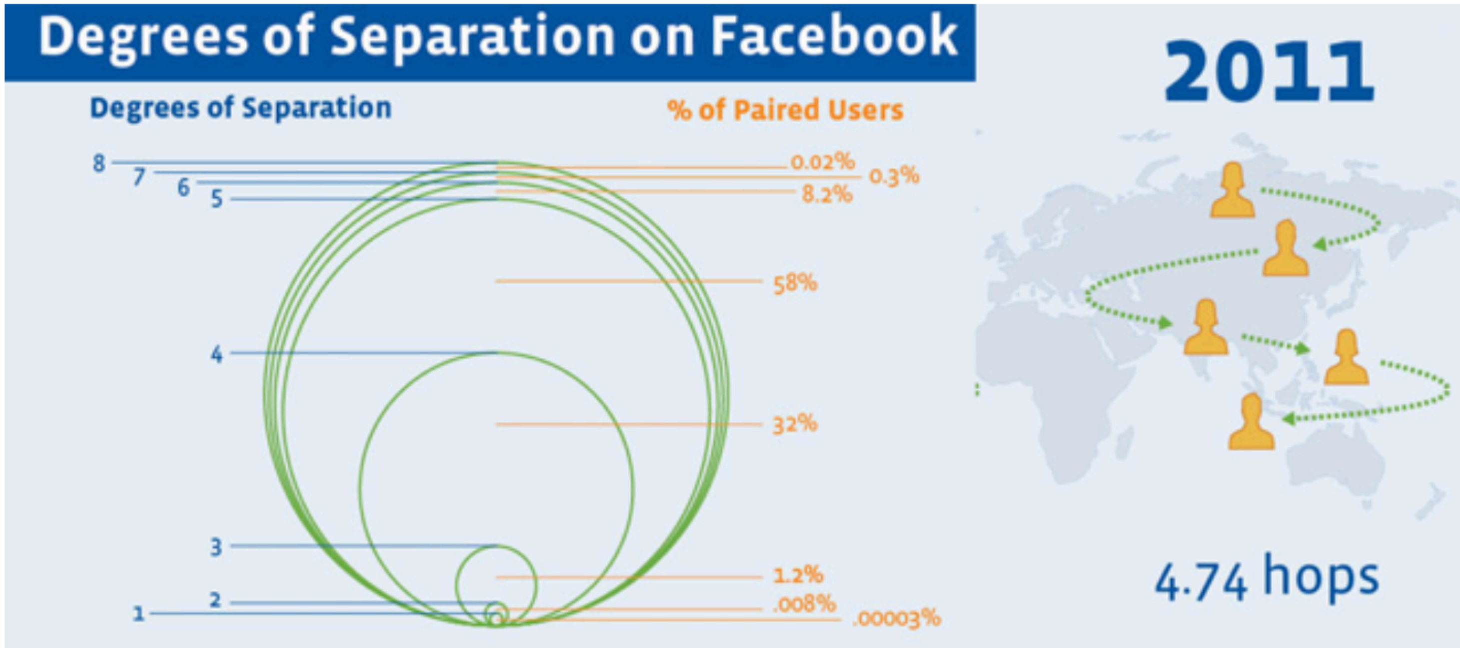


Small-World Phenomenon (Six Degrees of Separation)



Everybody is connected to everybody else by no more than six degrees of separation

by sociologist Stanley Milgram (1967)



Centrality

- Centrality: nodes that are “close” to many other nodes in the network.

Closeness centrality

- average distance from a given node to the other nodes

$$s(i) = \sum_{j \in V(G)} d(i,j).$$

$$CC(i) = \frac{n-1}{s(i)},$$

- Consider the network illustrated in Figure 14.4.

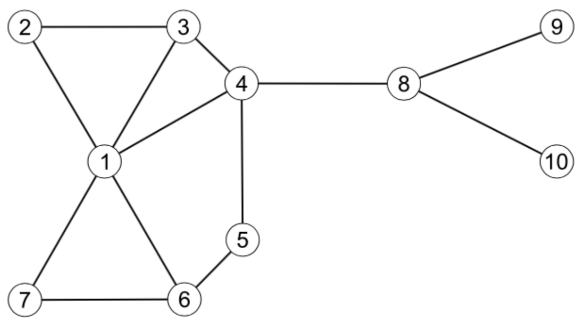


Figure 14.4: A network where closeness centrality does not match degree centrality

We start by constructing the distance matrix of this network, which is given by

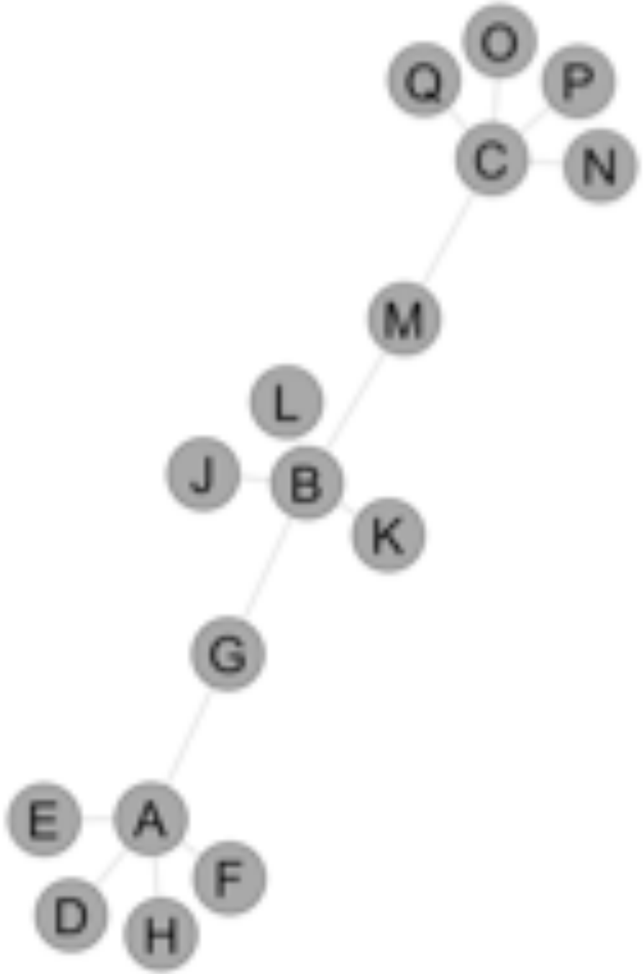
$$D = \begin{bmatrix} 0 & 1 & 1 & 1 & 2 & 1 & 1 & 2 & 3 & 3 \\ 1 & 0 & 1 & 2 & 3 & 2 & 2 & 3 & 4 & 4 \\ 1 & 1 & 0 & 1 & 2 & 2 & 2 & 2 & 3 & 3 \\ 1 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 2 & 2 \\ 2 & 3 & 2 & 1 & 0 & 1 & 2 & 2 & 3 & 3 \\ 1 & 2 & 2 & 2 & 1 & 0 & 1 & 3 & 4 & 4 \\ 1 & 2 & 2 & 2 & 2 & 1 & 0 & 3 & 4 & 4 \\ 2 & 3 & 2 & 1 & 2 & 3 & 3 & 0 & 1 & 1 \\ 3 & 4 & 3 & 2 & 3 & 4 & 4 & 1 & 0 & 2 \\ 3 & 4 & 3 & 2 & 3 & 4 & 4 & 1 & 2 & 0 \end{bmatrix}$$

The vector of distance-sum of each node is then

$$\mathbf{s} = D\mathbf{e} = (\mathbf{e}^T D)^T = [\ 15 \quad 22 \quad 17 \quad 14 \quad 19 \quad 20 \quad 21 \quad 18 \quad 26 \quad 26\ ]^T$$

And we use (11.4) to measure the closeness centrality of each node. For instance for node 1

$$CC(1) = \frac{9}{15} = 0.6.$$



	A,C	B	G,M	J,K,L	La resta
Local (grau)	5	5	2	1	1
Global (distància)	43	33	37	47	57

Distances and adjacency matrices

$$(A^n)_{ij}$$

NetworkX

Shortest Paths

Compute the shortest paths and path lengths between nodes in the graph.

These algorithms work with undirected and directed graphs.

`shortest_path`(G[, source, target, weight])	Compute shortest paths in the graph.
`all_shortest_paths`(G, source, target[, weight])	Compute all shortest paths in the graph.
`shortest_path_length`(G[, source, target, weight])	Compute shortest path lengths in the graph.
`average_shortest_path_length`(G[, weight])	Return the average shortest path length.
`has_path`(G, source, target)	Return True if G has a path from source to target, False otherwise.

Algorithm 3.12 Closeness Centrality

Input: $G(V, E)$ unweighted undirected network, `visit` function that operates on a node and marks it as visited

Output: Matrix of distances between nodes **D**, Closeness centrality **C**

```
for all x ∈ V do
    Initialize Queue Q                          ▷ To store pending nodes
    D(x)(x) ← 0
    visit(x)
    Enqueue x in Q
    while Q is not empty do
        a ← Dequeue(Q)
        for all Neighbors of a do
            if y Neighbor of a was not visited then
                visit(y)
                D(x)(y) ← D(x)(a) + 1
                Enqueue y in Q
            end if
        end for
    end while
end for
Compute C using Equation XXXXX
```

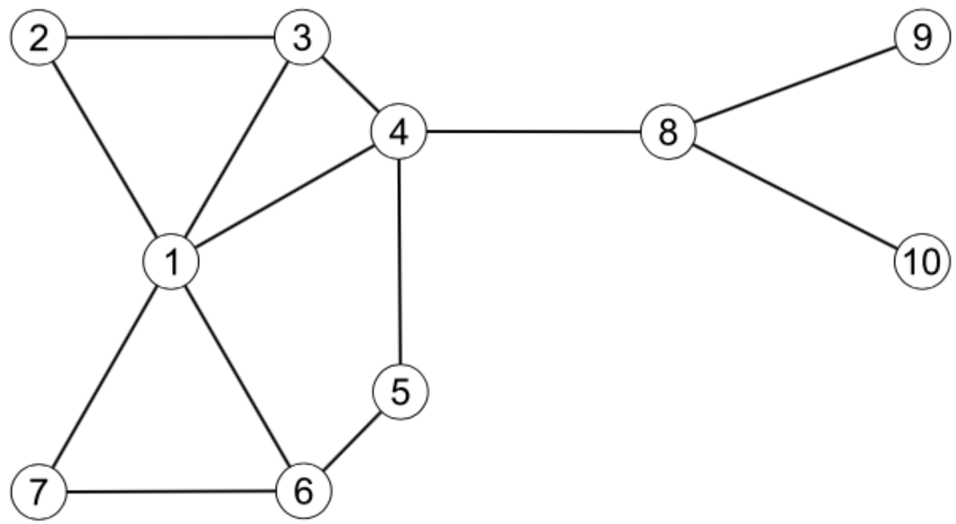
Betweenness centrality

- Measures the “intermediary” role in the network
- It is a set of matrices, one for ach node
- is a measure of the centrality, in terms of flow, of node k

$$BC(i) = \sum_i \sum_k \frac{\rho(j,i,k)}{\rho(j,k)}, \; i \neq j \neq k,$$

$$B^i_{kj}$$

where $\rho(j,k)$ is the number of shortest paths connecting the node j to the node k, and $\rho(j,i,k)$ is the number of these shortest paths that pass through node i in the network.



(j,k)	$\rho(j,1,k)$	$\rho(j,k)$	$\frac{\rho(j,i,k)}{\rho(j,k)}$
2,4	1	2	1/2
2,5	2	3	2/3
2,6	1	1	1
2,7	1	1	1
2,8	1	2	1/2
2,9	1	2	1/2
2,10	1	2	1/2
3,6	1	1	1
3,7	1	1	1
4,6	1	2	1/2
4,7	1	1	1
6,8	1	2	1/2
6,9	1	2	1/2
6,10	1	2	1/2
7,8	1	1	1
7,9	1	1	1
7,10	1	1	1
			12.667

Using a similar procedure we obtain the betweenness centrality for each node:

$$\mathbf{BC} = [\ 12.667 \quad 0.000 \quad 2.333 \quad 20.167 \quad 2.000 \quad 1.833 \quad 0.000 \quad 15.000 \quad 0.000 \quad 0.000\ ]^T,$$

which indicates that the node 4 is the most central one, i.e., it is the most important in allowing communication between other pairs of nodes.

- In Figure 14.6 we illustrate the urban street network of the central part of Cordoba, Spain. The most central nodes according to the betweenness correspond to those street intersections which surround the central part of the city and connect it with the periphery.

Figure 14.6: The street network of Cordoba with nodes of radii proportional to their betweenness centrality

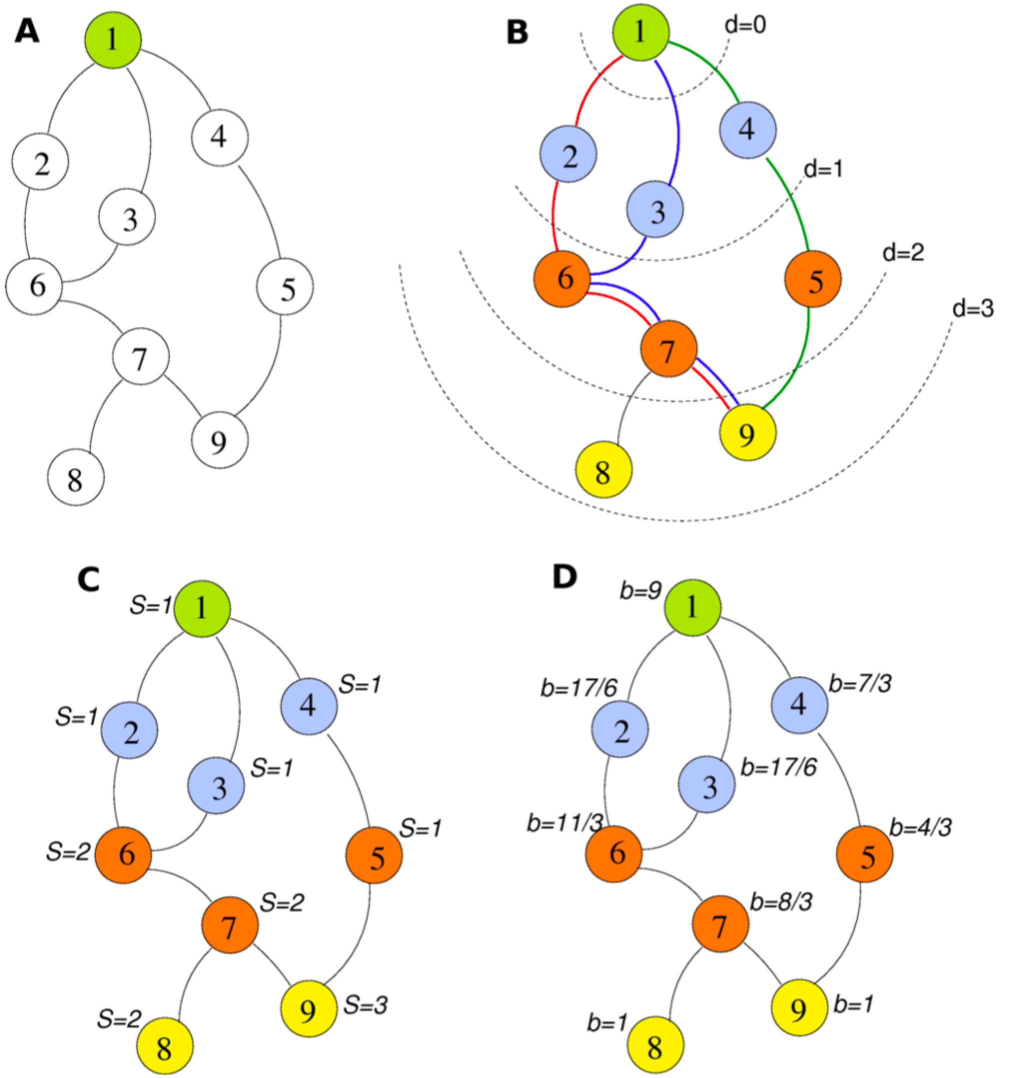


Figure 3.1 Representation of the computation of the betweenness centrality. In (A) we show a network composed of 9 nodes whereas node 1 is considered as the source. In (B) we show the distances of the rest of the nodes to the source highlighting the three possible shortest path from node 9 to node 1. In (C) we show the weights assigned to each of the nodes meaning the number of shortest paths from the node tp the source. Finally, in (D) we show the betweenness score of each node (when considering the source node 1).

Algorithm 3.13 Betweeness Centrality

```
Input: G(V,E) unweighted undirected network, visit function that oper-
   ates on a node and marks it as visited
Output: Matrix of distances between nodes D, Betweeness centrality B
  B ← 0
  for all x ∈ V do
      Initialize Queue Q                              ▷ To store pending nodes
      D(x)(x) ← 0
      visit(x)
      S(x) ← 1
      Enqueue x in Q
      while Q is not empty do
          a ← Dequeue(Q)
          for all Neighbors of a do
              if y Neighbor of a was not visited then
                  visit(y)
                  D(x)(y) ← D(x)(a) + 1
                  S(y) ← S(a)
                  Enqueue y in Q
              else
                  if D(y)(x)=D(x)(a)+1 then
                      S(y) ← S(y) + S(a)
                  end if
              end if
          end for
      end while
      for all y ∈ V do
          Enqueue y in Q
      end for
      d ← maximum distance from x
      while Q is not empty do
          for all y ∈ V do
              if D(y)(x) = d then
                  Sum to B(i) the value obtained by using Eq. (3.2)
                  i ← Dequeue(Q)
              end if
          end for
          d ← d − 1
      end while
  end for
```

NetworkX

Centrality

Degree

`degree_centrality`(G)	Compute the degree centrality for nodes.
`in_degree_centrality`(G)	Compute the in-degree centrality for nodes.
`out_degree_centrality`(G)	Compute the out-degree centrality for nodes.

Closeness

`closeness_centrality`(G[, u, distance, ...])	Compute closeness centrality for nodes.

Betweenness

`betweenness_centrality`(G[, k, normalized, ...])	Compute the shortest-path betweenness centrality for nodes.
`edge_betweenness_centrality`(G[, k, ...])	Compute betweenness centrality for edges.

Eigenvector centrality

- Generalization of degree
- Eigenvector centrality is a measure of the importance of a node in a network. It assigns relative scores to all nodes in the network based on the principle that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes.

$$x_i = \frac{1}{\lambda} \sum_{j \in M(i)} x_j = \frac{1}{\lambda} \sum_{j=1}^{N} a_{i,j} x_j$$

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

In general, there will be many different eigenvalues λ for which a non-zero eigenvector solution exists. However, the additional requirement that all the entries in the eigenvector be non-negative implies (by the Perron–Frobenius theorem) that only the greatest eigenvalue results in the desired centrality measure.[4] The v^{th}

Perron–Frobenius theorem

From Wikipedia, the free encyclopedia

In linear algebra, the **Perron–Frobenius theorem**, proved by Oskar Perron (1907) and Georg Frobenius (1912), asserts that a real square matrix with positive entries has a unique largest real eigenvalue and that the corresponding eigenvector can be chosen to have strictly positive components, and also asserts

(ii) Sometimes being connected to a few very important nodes make a node more central than being connected to many not so central ones. For instance, in Figure 15.3 node 4 is connected to only three other nodes, while 1 is connected to four. However, 4 is more central than 1 according to the eigenvector centrality because it is connected to two nodes with relatively high centrality while 1 is mainly connected to peripheral nodes. The vector of centralities is

$$\mathbf{q}_1 = \begin{bmatrix} 0.408 & 0.167 & 0.167 & \mathbf{0.500} & 0.408 & 0.167 & 0.167 & 0.167 & 0.408 & 0.167 & 0.167 & 0.167 \end{bmatrix}^T.$$

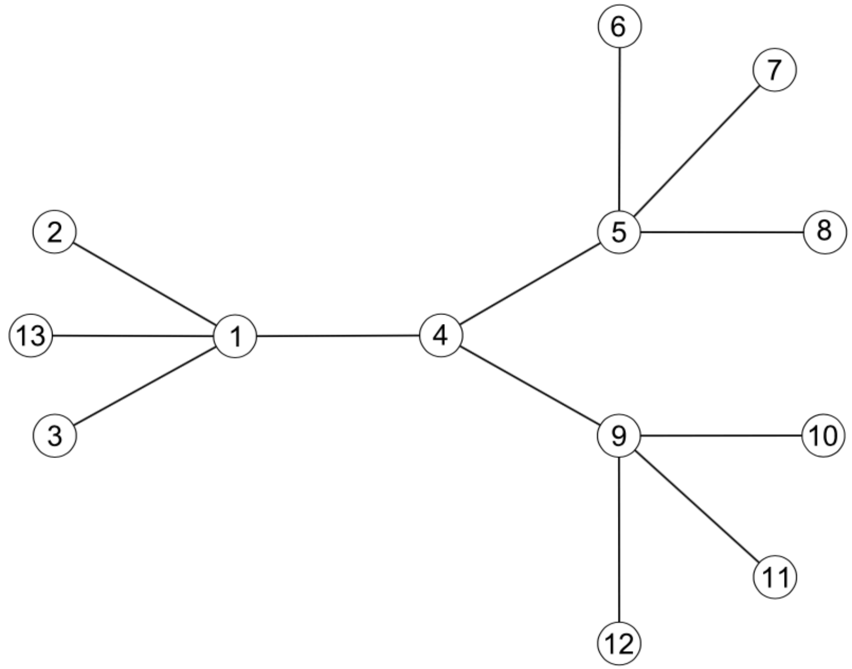


Figure 15.3: A network highlighting the difference between degree and eigenvector centrality

Page-rank

- Originally developed by the founders of Google (Page and Brin)
- Related to the probability that a random walker arrives to a given node
- Recursive relation (very fast convergence)

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

“teleportation” 0.85

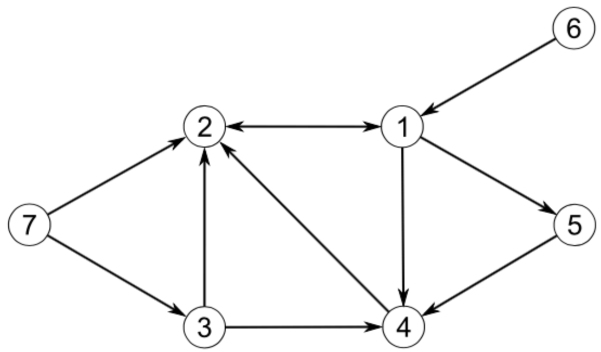


Figure 15.5: A directed network illustrating PageRank centrality

- The normalised PageRank of the network in Figure 15.5 is

$$\mathbf{PG} = \begin{bmatrix} 0.301 & \mathbf{0.308} & 0.030 & 0.211 & 0.107 & 0.021 & 0.021 \end{bmatrix}^T$$

when $\alpha = 0.85$. Notice that node 1 has higher PageRank than node 4 due to its in-link from node 4. In this example, the rankings vary little as we change α.

Katz centrality

The degree of the node i counts the number of walks of length one from i to every other node of the network. That is, $k_i = (A\mathbf{e})_i$. In 1953, Katz extended this idea to count not only the walks of length one, but those of any length starting at node i. Intuitively, we can reason that the closest neighbours have more influence over node i than more distant ones. Thus when combining walks of all lengths one can introduce an attenuation factor so that more weight is given to shorter walks than to longer ones. This is precisely what Katz did and the Katz index is given by

$$K_i = \left[(\alpha^0 A^0 + \alpha A + \alpha^2 A^2 + \cdots + \alpha^k A^k + \cdots)\mathbf{e}\right]_i = \left[\sum_{k=0}^{\infty}(\alpha^k A^k)\mathbf{e}\right]_i . \tag{15.1}$$

$$\alpha < 1$$

NetworkX

Eigenvector

`eigenvector_centrality`(G[, max_iter, tol, ...])	Compute the eigenvector centrality for the graph G.
`eigenvector_centrality_numpy`(G[, weight])	Compute the eigenvector centrality for the graph G.
`katz_centrality`(G[, alpha, beta, max_iter, ...])	Compute the Katz centrality for the nodes of the graph G.
`katz_centrality_numpy`(G[, alpha, beta, ...])	Compute the Katz centrality for the graph G.

Communicability

`communicability`(G)	Return communicability between all pairs of nodes in G.
`communicability_exp`(G)	Return communicability between all pairs of nodes in G.
`communicability_centrality`(G)	Return communicability centrality for each node in G.
`communicability_centrality_exp`(G)	Return the communicability centrality for each node of G
`communicability_betweenness_centrality`(G[, ...])	Return communicability betweenness for all pairs of nodes in G.
`estrada_index`(G)	Return the Estrada index of a the graph G.

PageRank

PageRank analysis of graph structure.

`pagerank`(G[, alpha, personalization, ...])	Return the PageRank of the nodes in the graph.

Clustering

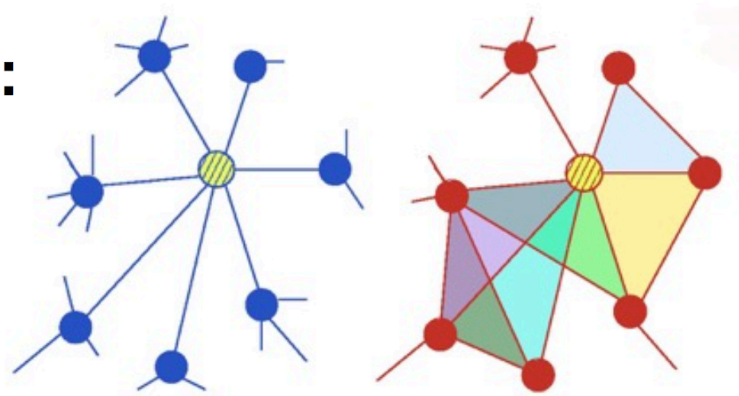
- Are my friends also friends among them?????

Clustering of a node:

$$C_i = \frac{\#\text{triangles connected to } i}{\#\text{possible triangles connected to } i} = \frac{2E_i}{k_i(k_i - 1)}$$

Clustering of the Network:

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



Algorithm 3.10 Average Clustering Coefficient

Input: $G(V, E)$ unweighted undirected network, $\mathbf{k}$ degree sequence
Output: Individual clustering coefficients $\mathbf{c}$, average clustering C

```
c ← 0
C ← 0
for all node i ∈ V do
    Γ_i ← neighbors of node i
    for all node j ∈ Γ_i do
        Γ_j ← neighbors of node j
        c(i) ← c(i)+ number of overlapping elements between Γ_i and Γ_j
    end for
    c(i) ← 2c(i) / (k(i)(k(i)−1))
    C ← C + c(i)
end for
C ← C / N
return c and C
```

NetworkX

Clustering

Algorithms to characterize the number of triangles in a graph.

`triangles`(G[, nodes])	Compute the number of triangles.
`transitivity`(G)	Compute graph transitivity, the fraction of all possible triangles present in G.
`clustering`(G[, nodes, weight])	Compute the clustering coefficient for nodes.
`average_clustering`(G[, nodes, weight, ...])	Compute the average clustering coefficient for the graph G.

Book: First Course Network Science

- https://cambridgeuniversitypress.github.io/FirstCourseNetworkScience/
- Tutorials (notebooks)
- Data sets

Table of Contents

- Chapter 0: Introduction
- Chapter 1: Basic Elements (Tutorial)
- Chapter 2: Small Worlds (Tutorial)

Additional Resources

- Dataset files