Building & Mining Knowledge Graphs

KEN4256

Lecture 9: Network Science and Graph Analytics

id: KEN4256_L9
version: 1.2024.0
created: March 15, 2021
last modified: March 26, 2024
published on: March 26, 2024

Network Science & Graph Analytics

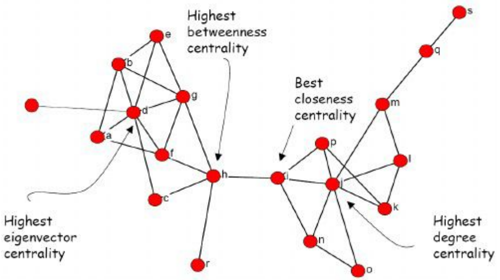
Network Science aims to understand and analyze the structure, behavior, and dynamics of complex networks and systems.

- A **network** refers to some real world phenomena of connected entities.
- A **graph** is the computational representation of that network.

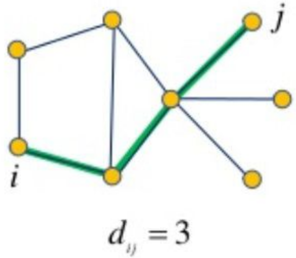
Graph analytics involves the application of algorithms over graphs to gain insights into network structure and dynamics.

Network Science & Graph Analytics

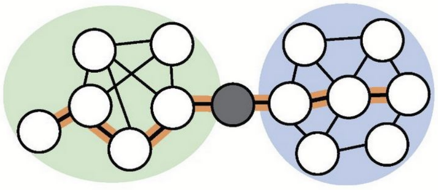
A key instrument of network science is the analysis of network topology: Analyzing the structure of networks, including the identification of important nodes (e.g., hubs or influencers in social networks), links, or substructures that play critical roles in the functioning of the network.



Node centrality



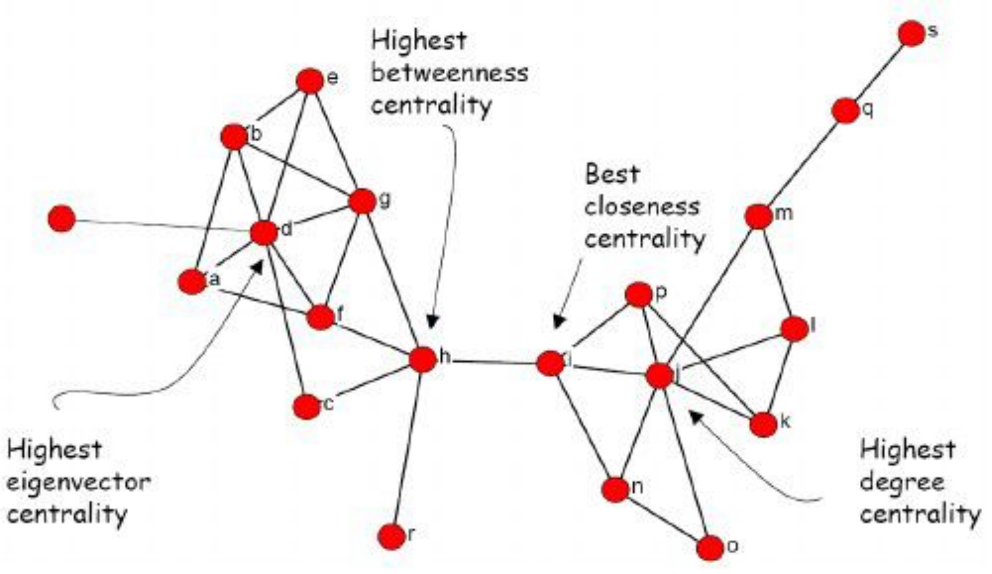
Path analysis



Topological clustering

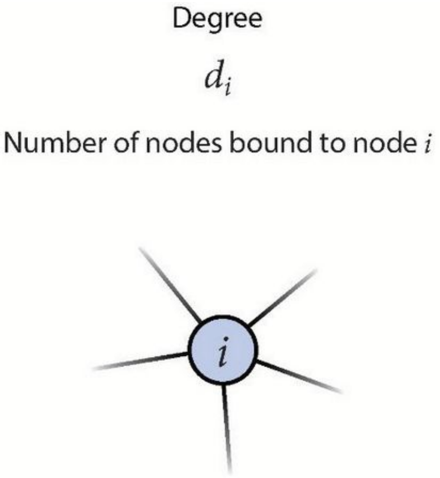
Centrality measures

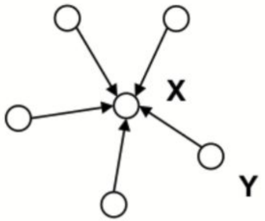
Importance of the node in a network based on the topological structure of the network



Node centralities

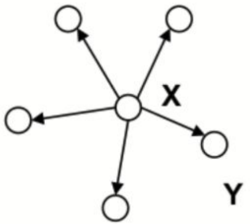
<u>Degree centrality</u>: measure the number of nodes adjacent to a node (degree)





indegree

# of incoming edges



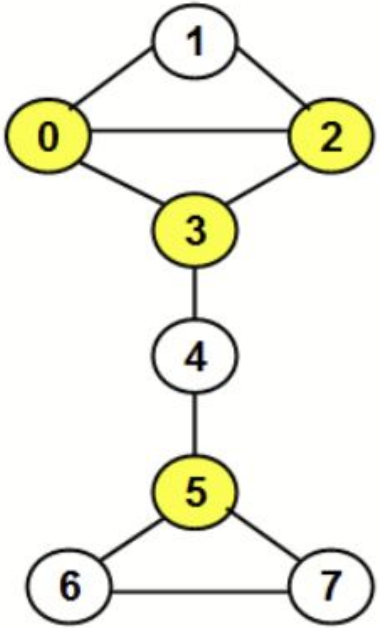
outdegree

# of outgoing edges

The adjacency matrix (and linear algebra) used as a basis for many network computations

$$\mathbf{A}(i,j) = \begin{cases} 1, & \text{if } v_i \sim v_j \\ 0, & \text{otherwise.} \end{cases}$$

$$\deg(v_i) = \sum_{j=1}^{n} \mathbf{A}(i,j) = \sum_{j=1}^{n} \mathbf{A}(j,i).$$



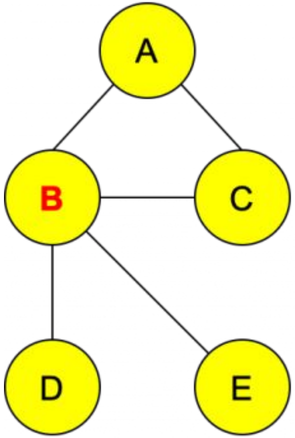
$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array}
\begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}
\times
\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}
=
\begin{bmatrix} 3 \\ 2 \\ 3 \\ 3 \\ 2 \\ 3 \\ 2 \\ 2 \end{bmatrix}
\begin{array}{c} \text{Vertex IDs} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array}$$

Adjacency Matrix — Column Vector — Degree Centrality

Closeness centrality

$$CC(i) = \frac{N-1}{\sum_j d(i,j)}$$

where
$i \neq j$,
d_{ij} is the length of the shortest path between nodes i and j in the network,
N is the number of nodes.



$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array}
\begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 2 & 2 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 2 \\ 2 & 1 & 2 & 0 & 2 \\ 2 & 1 & 2 & 2 & 0 \end{pmatrix} \end{array}$$

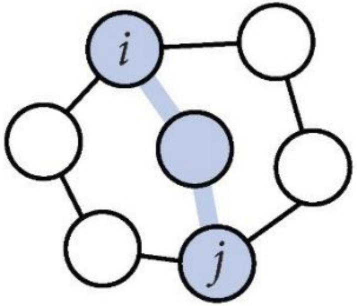
farness $\sum_{j=1}^{n} d(i,j)$	$CC(i) = \frac{N-1}{\sum_j d(i,j)}$
6	(5-1)/6= 0.67
4	1.00
6	0.67
7	0.57
7	0.57

N = 5 (# of nodes)

Shortest path distance

$$d_{ij} = \min\{|e_p| \subset E_{ij}\}$$

E_{ij}: all edge sets connecting nodes i and j

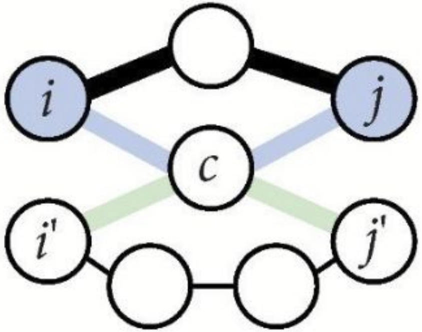


Betweenness centrality

$$b_c = \sum_i \sum_j I_{ij}/s_{ij}$$

s_{ij}: total number of shortest paths between i and j

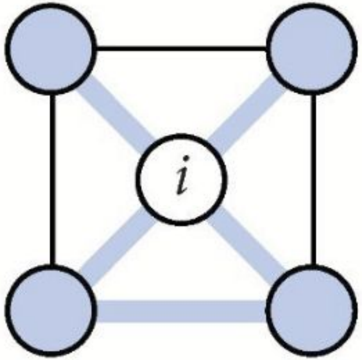
I_{ij}: 1 if c is within path; 0 otherwise



Clustering coefficient

$$c_i \Big/ \binom{n_i}{2}$$

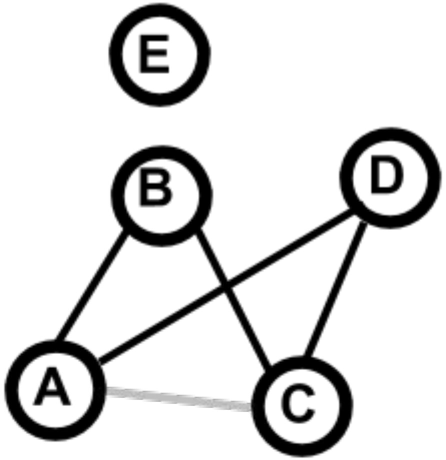
c_i: edges connecting all n_i nodes bound to i



Common Neighbors

$$s_{xy}^{\mathrm{CN}} = |\Gamma(x) \cap \Gamma(y)|,$$

$\Gamma(x) \leftarrow$ neighbours of x



CN(A,B) = 1
CN(A,C) = 2

Nodes with more common neighbors, are more likely similar to each other

Jaccard Similarity

$$s_{xy}^{\text{Jaccard}} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

same as common neighbors, adjusted for degree

Cosine Similarity

$$s_{xy}^{\text{Salton}} = \frac{|\Gamma(x) \cap \Gamma(y)|}{\sqrt{k_x \times k_y}}$$

K_i degree of x,y

Adamic/Adar Similarity

k_x ← degree of x

$$s_{xy}^{\mathrm{AA}} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log k_z}$$

weighting rarer neighbors more heavily

Eigenvector centrality

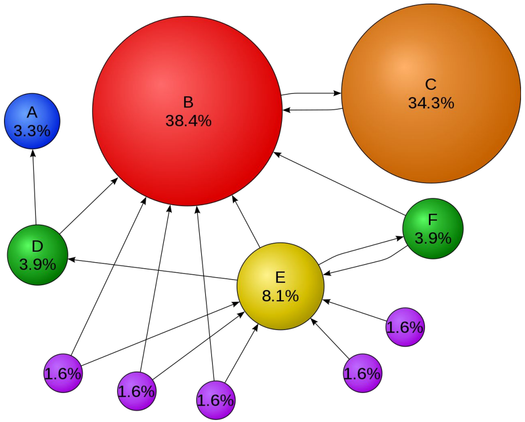
Eigenvector centrality is used to measure the level of influence of a node within a network

Relationships originating from high-scoring nodes contribute more to the score of a node than connections from low-scoring nodes.

The PageRank algorithm is a variant of Eigenvector Centrality with an additional jump probability.

$$x_v = \frac{1}{\lambda} \sum_{t \in M(v)} x_t = \frac{1}{\lambda} \sum_{t \in V} a_{v,t} x_t$$

PageRank



PageRank algorithm was developed by Larry Page and Sergey Brin, the founders of Google, while they were students at Stanford University.

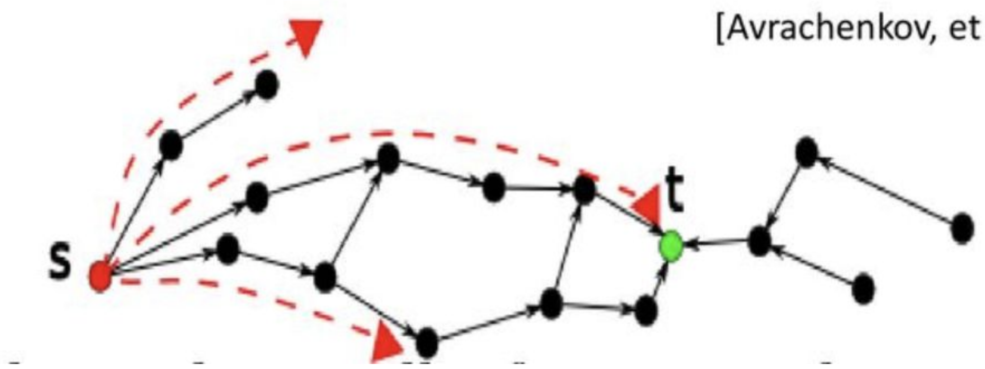
PageRank measures the importance of website pages based on the links between them. It is based on a normalized eigenvector centrality, combined with a random jump.

$$x_i = \sum_{j \to i} \frac{1}{N_j} x_j^{(k)}$$

Each node, x_i, has a PageRank as defined by the sum of pages *j* that link to *i* times one over the outlinks or "out-degree" of *j* times the "importance" or PageRank of *j* .

Rooted (Personalized) PageRank

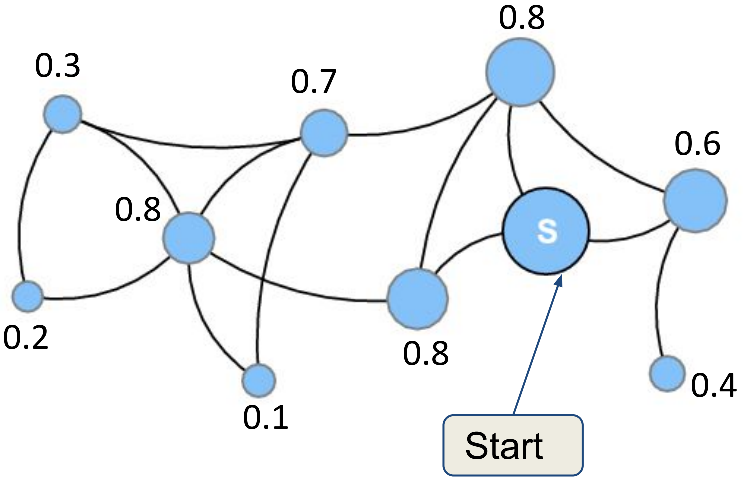
calculates popularity score for each node
with respect to root node s (starting node of “browsing”).
(given source s, target t and stopping probability.)



[Avrachenkov, et al 2007]

Random Walk with Restart

- Imagine a network, and starting at a specific node, you follow the edges randomly.
- But with some probability, you "jump" back to the node (restart!).



If you keep doing the random walk, you will obtain a ranking of other nodes with respect to the start node.

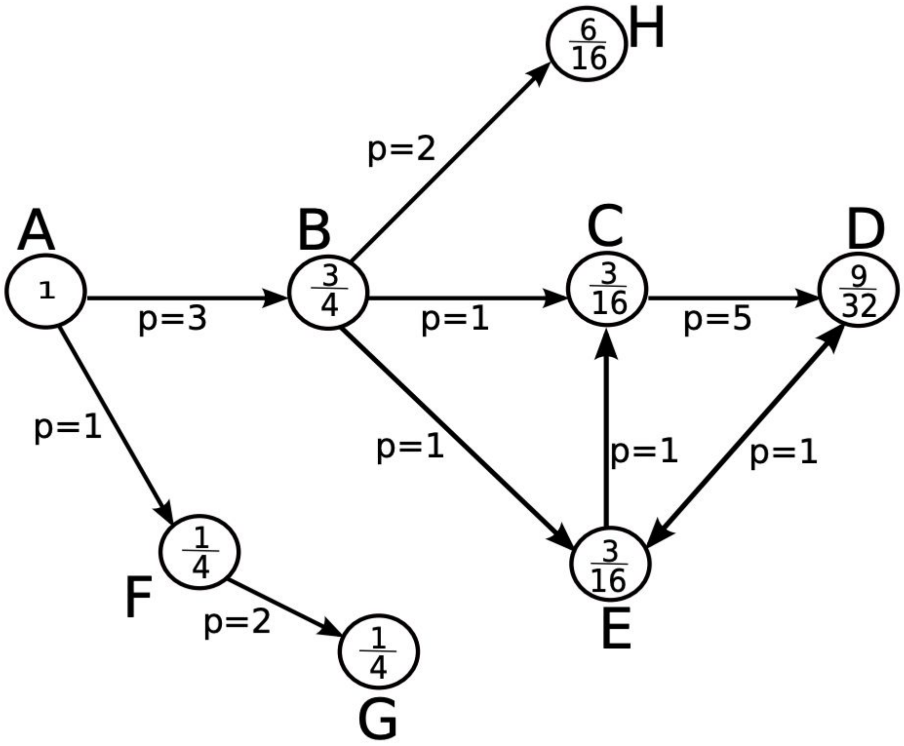
likelihood of forming a link from the start node.

PropFlow

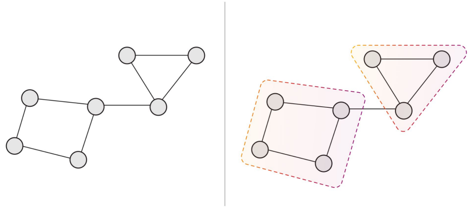
Similar to rooted PageRank

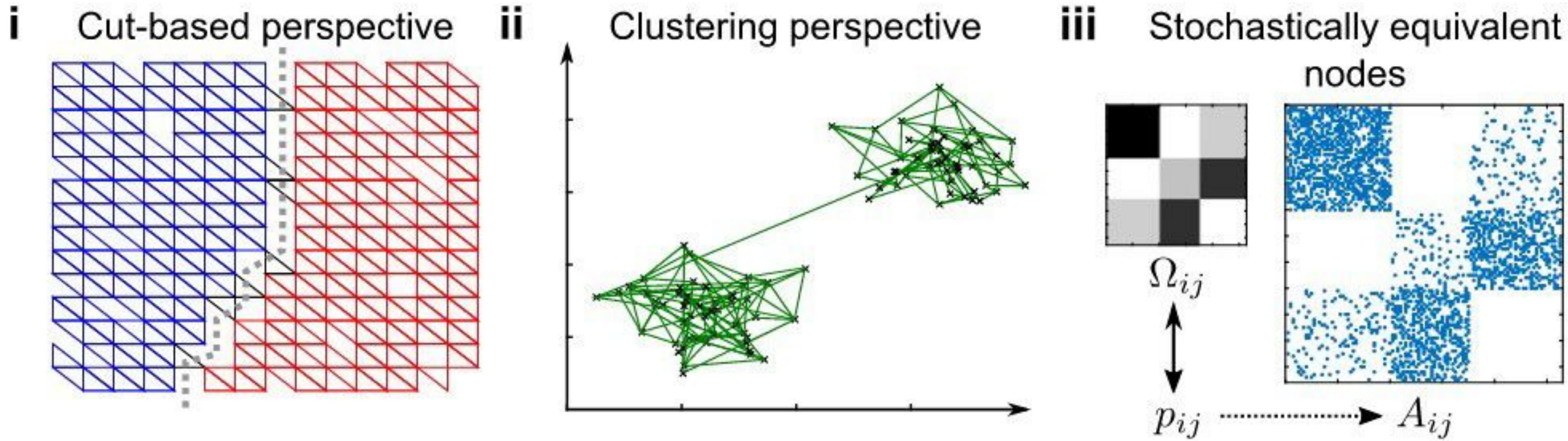
Makes restricted random walks of at most *h* steps
on edge-weighted graph
starting at *x* and ending at *y*

Probability of following a link is proportional to its edge-weight



Topological Clustering





Bipartite Graph

A **bipartite graph** is a graph whose nodes can be divided into two disjoint sets U and V such that every edge connects a node in U to one in V.

Two “types” of nodes. No edges between nodes of the same type

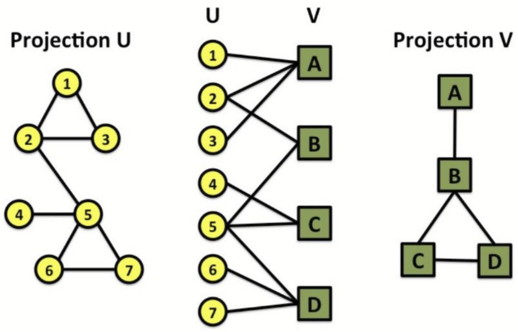
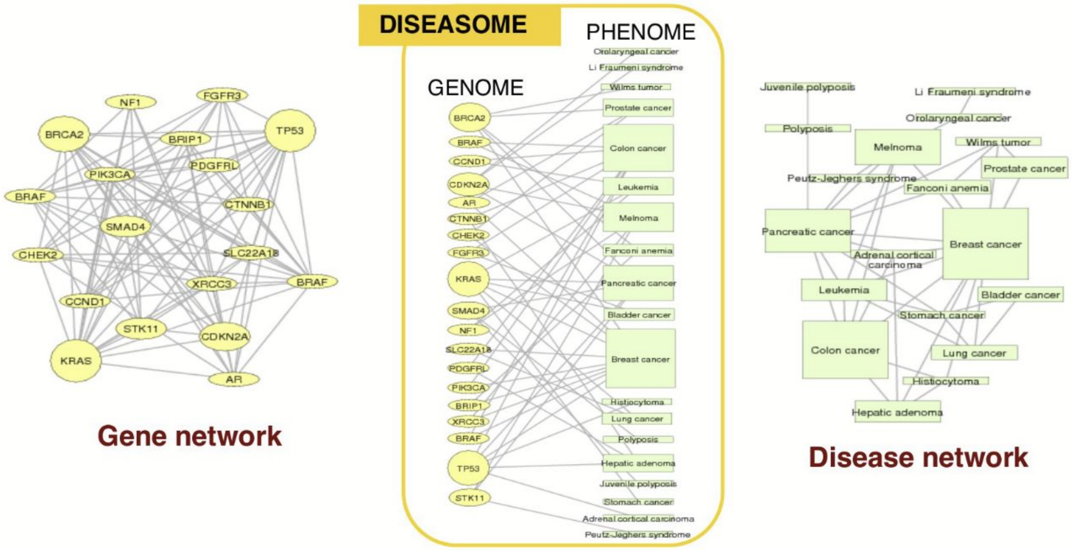
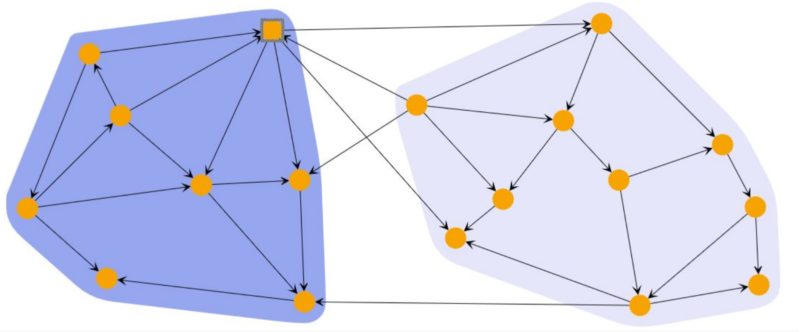


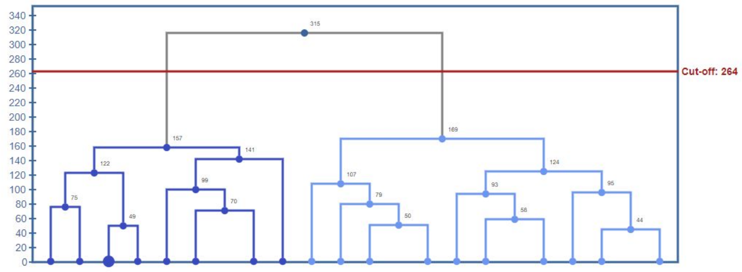
Image: Adapted from Leskovec, 2015



Goh, Cusick, Valle, Childs, Vidal & Barabási, PNAS (2007)

Hierarchical Clustering





Hierarchical clustering partitions the graph into a hierarchy of clusters.

* Agglomerative strategy applies a bottom-up approach: Each node put into a separate cluster and subsequent steps the algorithm merges pairs of clusters while moving up the hierarchy. The algorithm continues until all nodes belong to the same cluster.
* Divisive strategy applies a top down approach where all nodes are initially grouped into one cluster. At each step, the algorithm splits the largest cluster while moving down to the hierarchy.

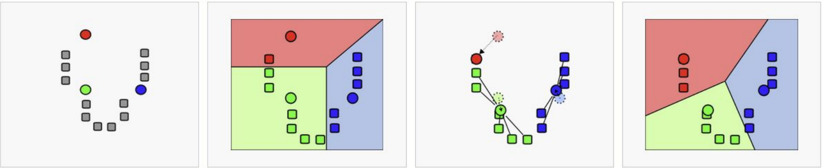
The dissimilarity between clusters is determined based on the given linkage criterion and an appropriate distance metric as euclidean distance, euclidean-squared distance, manhattan distance, or Chebyshev distance.

The result is a dendrogram which can be cut based on a given cut-off value.

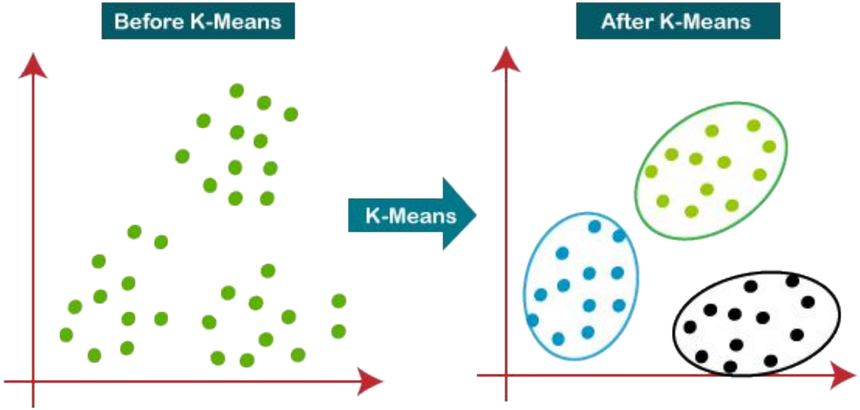
K-Means Clustering

K-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid).

Demonstration of the standard algorithm



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).
2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.
3. The centroid of each of the *k* clusters becomes the new mean.
4. Steps 2 and 3 are repeated until convergence has been reached.



Network Characterization

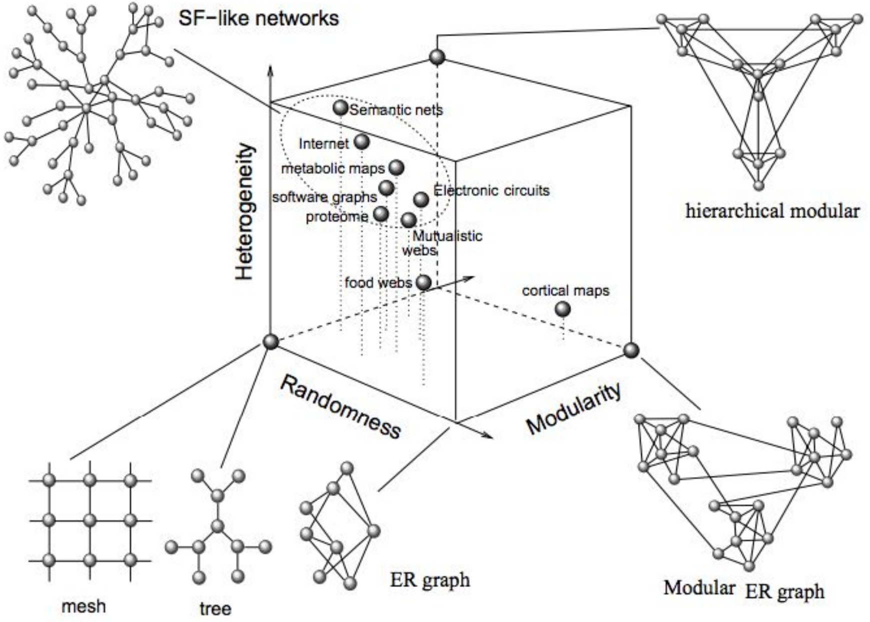
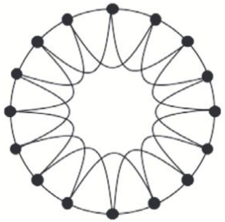
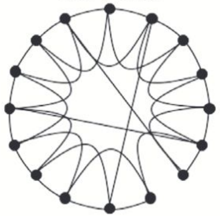
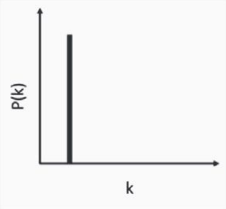
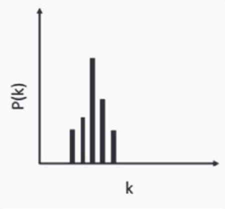
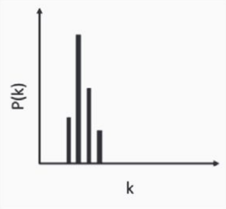
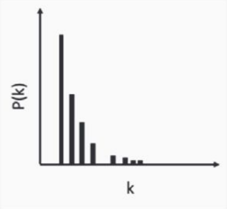
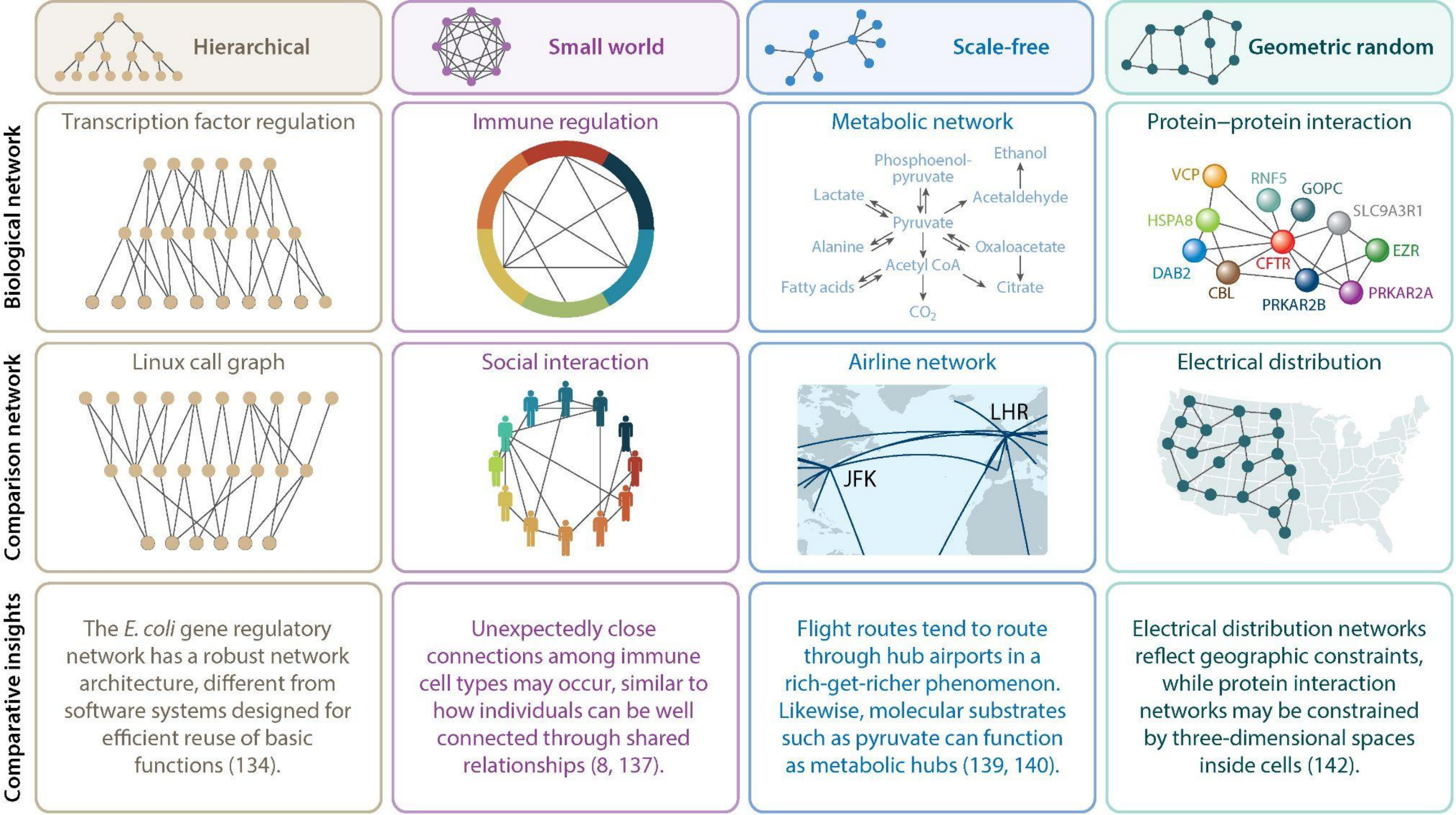


FIG. 3 A zoo of complex networks. In this qualitative space, three relevant characteristics are included: randomness, heterogeneity and modularity. The first introduces the amount of randomness involved in the process of network's building. The second measures how diverse is the link distribution and the third would measure how modular is the architecture. The position of different examples are only a visual guide. The domain of highly heterogeneous, random hierarchical networks appears much more occupied than others. Scale-free like networks belong to this domain.

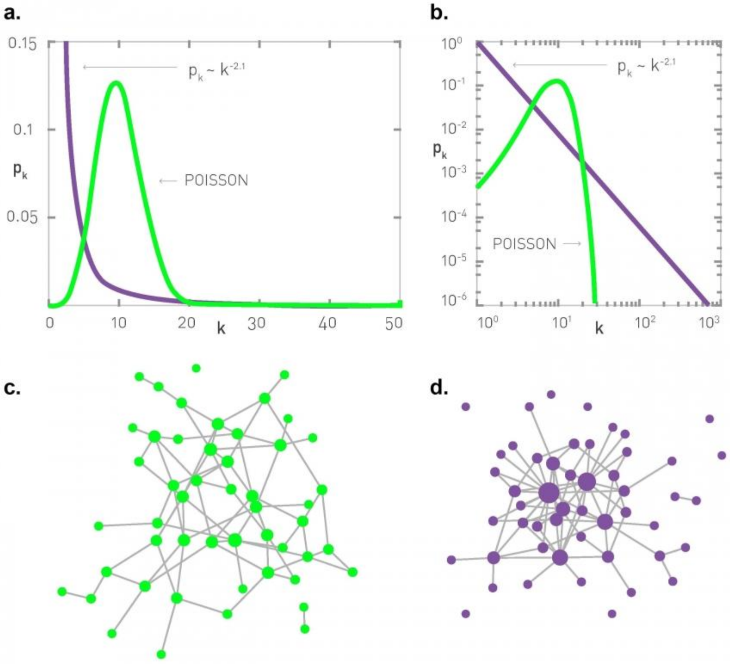
Information Theory of Complex Networks: on evolution and architectural constraints by Sole and Valverde (2004)

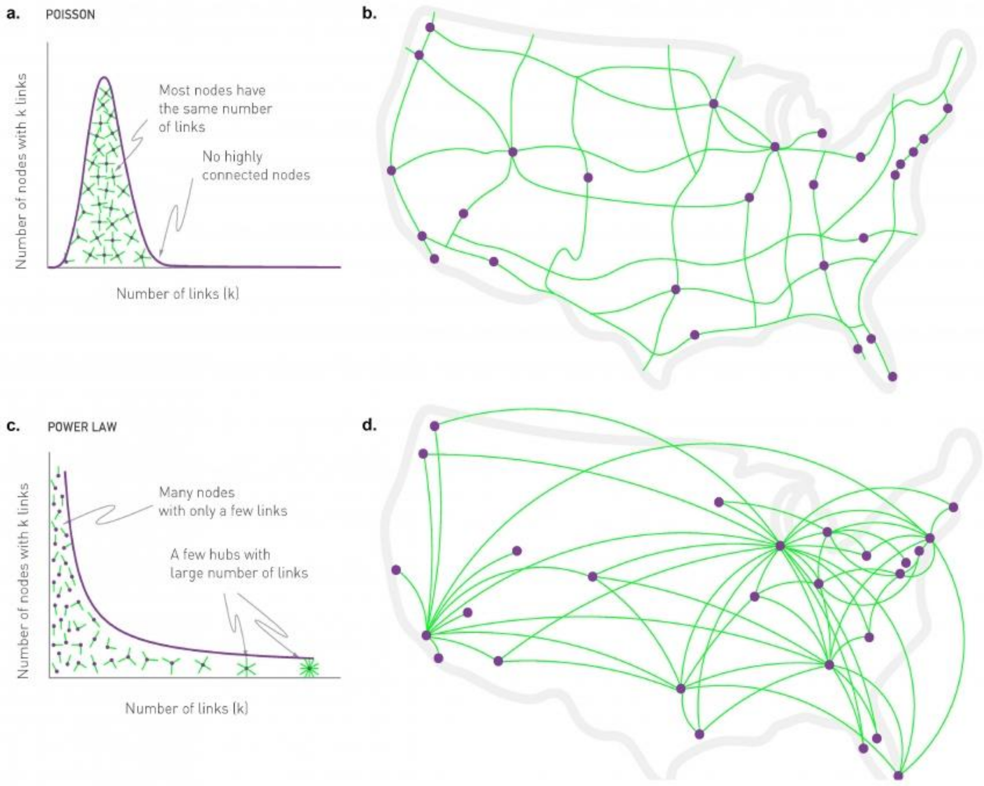
Network Structures

GRAPH TYPE	A. REGULAR GRAPH	B. RANDOM GRAPH	C. SMALL WORLD NETWORK	D. SCALE FREE NETWORK
				
DEGREE DISTRIBUTION	Fixed	Poisson	Poisson, sometimes skewed	Power Law
				
CLUSTERING COEFFICIENT	High	Low	High	Low
PATH LENGTH	Long	Short	Short	Short
RANDOMNESS	$p = 0$	$p = 1$	$0 < p < 1$	$0 < p < 1$
REAL WORLD EXAMPLES	Usually used to represent man-made networks and are not found in nature (Sole & Valverde, 2004).	The random graph does not capture real networks, but are used as a baseline for comparison (Barabasi, 2016).	Social networks, neural networks, power grids of the western United States, collaboration of graph of film actions (Watts & Strogatz, 1998).	World wide web (Barabasi et al., 2000), some airlines (Guimera & Amaral, 2004).



McGillivray P, et al. 2018.
Annu. Rev. Biomed. Data Sci. 1:153–80


a.
0.15
0.1
0.05
p_k
$p_k \sim k^{-2.1}$
POISSON
0
10
20
30
40
50
k
b.
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
p_k
$p_k \sim k^{-2.1}$
POISSON
10^0
10^1
10^2
10^3
k
c.
d.


a.
POISSON
Number of nodes with k links
Most nodes have
the same number
of links
No highly
connected nodes
Number of links (k)
b.
c.
POWER LAW
Number of nodes with k links
Many nodes
with only a few links
A few hubs with
large number of links
Number of links (k)
d.

Contact

Mailing list
Issue tracker
Source

Releases

Stable (notes)

3.0 — January 2023
download | doc | pdf

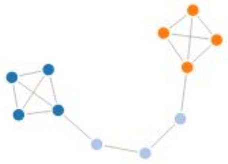
Latest (notes)

3.1 development
github | doc | pdf

Archive

NetworkX
Network Analysis in Python

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.



Software for complex networks

- Data structures for graphs, digraphs, and multigraphs
- Many standard graph algorithms
- Network structure and analysis measures
- Generators for classic graphs, random graphs, and synthetic networks
- Nodes can be "anything" (e.g., text, images, XML records)
- Edges can hold arbitrary data (e.g., weights, time-series)
- Open source 3-clause BSD license
- Well tested with over 90% code coverage
- Additional benefits from Python include fast prototyping, easy to teach, and multi-platform

https://networkx.org/