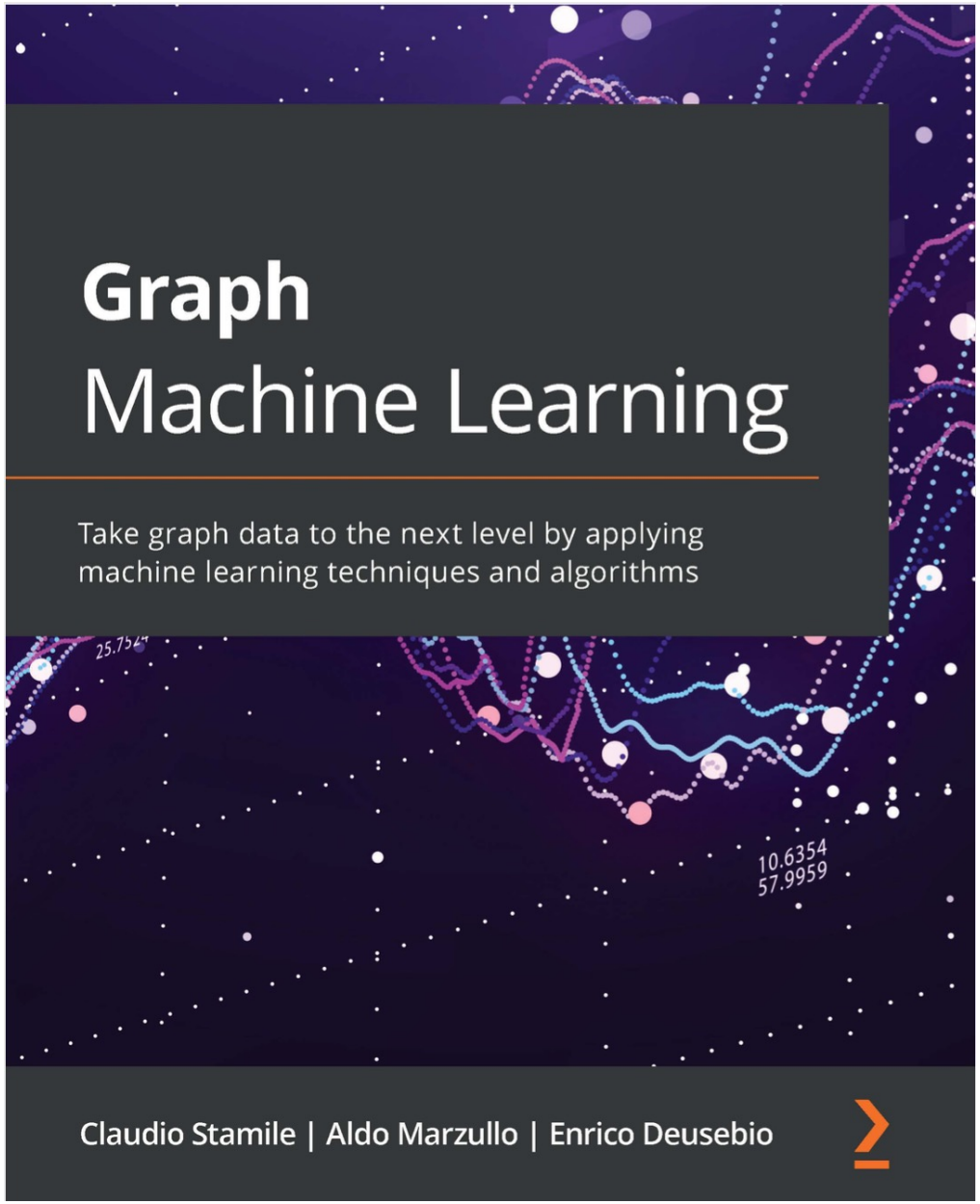
CGnal

business innovation through algorithms

Graph Machine Learning

CGnal S.p.A – Corso Venezia 43 - Milano

Novembre 2022| Milano



1. Introduction to Graphs and how to characterize them
2. Machine learning on Graphs
3. Graph Representation Learning
4. Graph Neural Network

What is a graph?

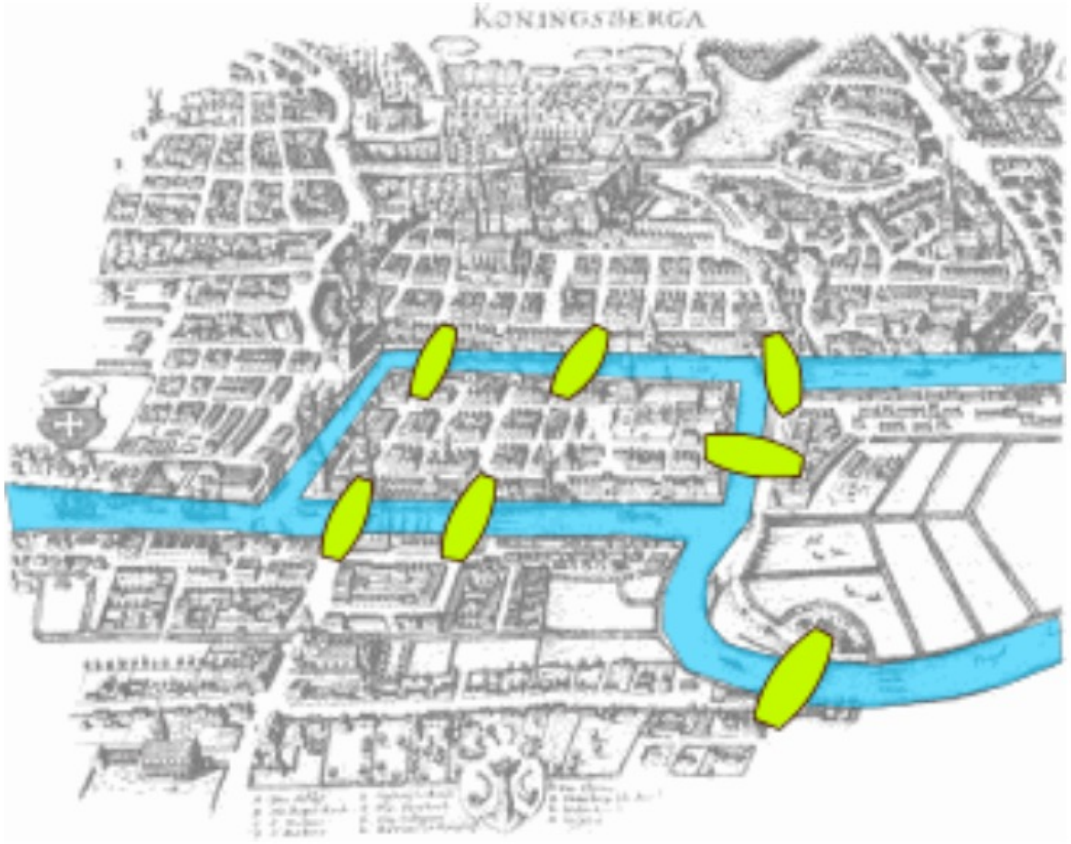
A **Simple Undirected Graph** (or simply graph) G is defined as a couple G=(V,E) where V={v1, ..., vn} is a set of Nodes (also called **vertices**) and E={{vk, vw}, …, {vi, vj}} is a set of two-sets (set of two elements) of **edges** (also called links) representing the connection between two nodes belonging to V.

Very **abstract** representation

↳ Very **general**

↳ Convenient to describe **many different** systems

Can one walk across the seven bridges and never cross the same bridge twice?



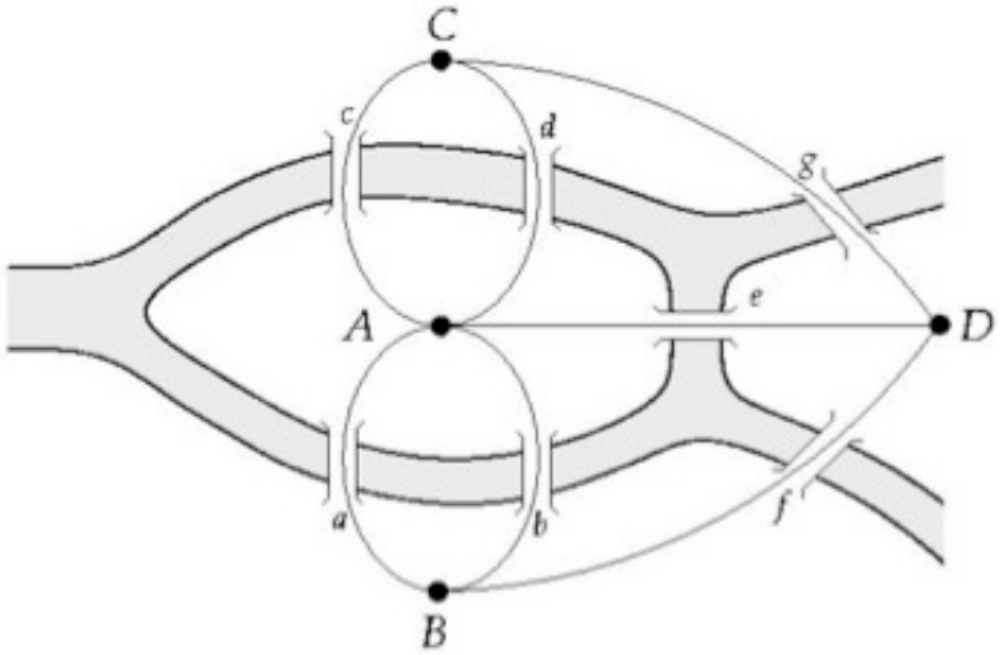
What is a graph?

A **Simple Undirected Graph** (or simply graph) G is defined as a couple G=(V,E) where V={v1, ..., vn} is a set of Nodes (also called **vertices**) and E={{vk, vw}, …, {vi, vj}} is a set of two-sets (set of two elements) of **edges** (also called links) representing the connection between two nodes belonging to V.

Very **abstract** representation

↳ Very **general**

↳ Convenient to describe **many different** systems



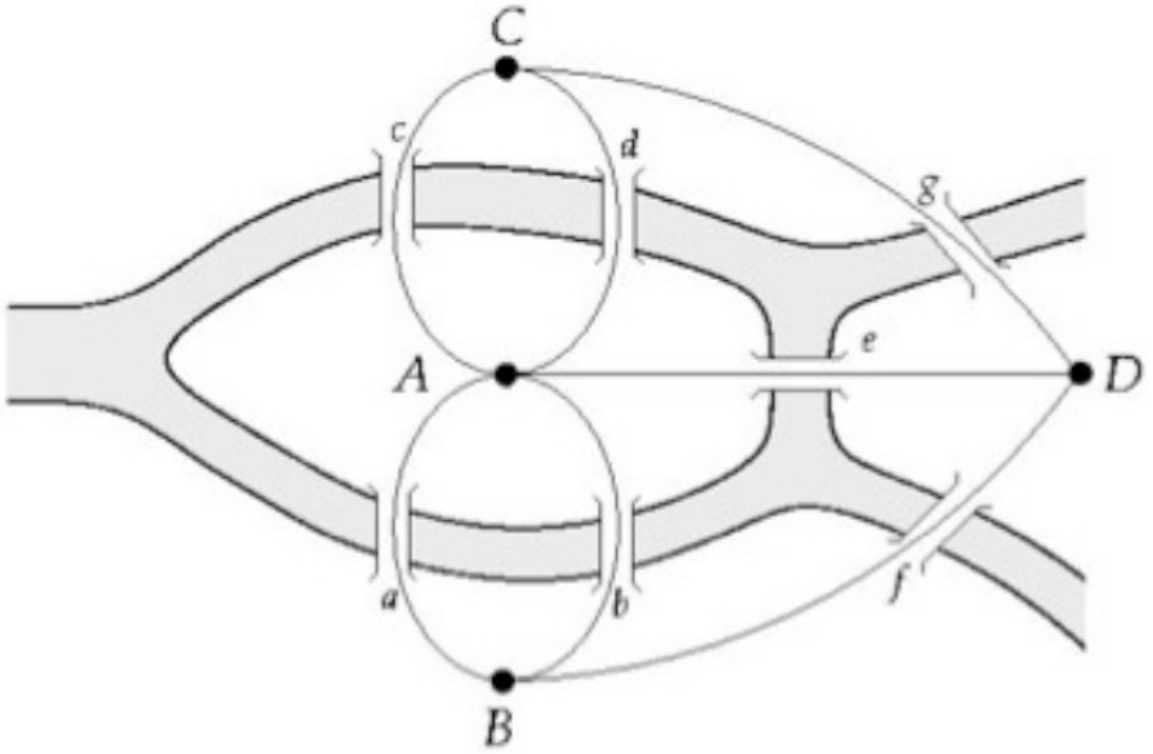
Can one walk across the seven bridges and never cross the same bridge twice?

1735: Leonhard Euler's theorem:
(a) If a graph has nodes of odd degree, there is no path.
(b) If a graph is connected and has no odd degree nodes, it has at least one path.

Graph are very interdiscipliary….

Science of complex networks can be found in many disciplices:

- **Graph theory**
- Sociology
- Communication science
- Biology / Chemistry
- Transportation / Logistics



Graph are very interdiscipliary….

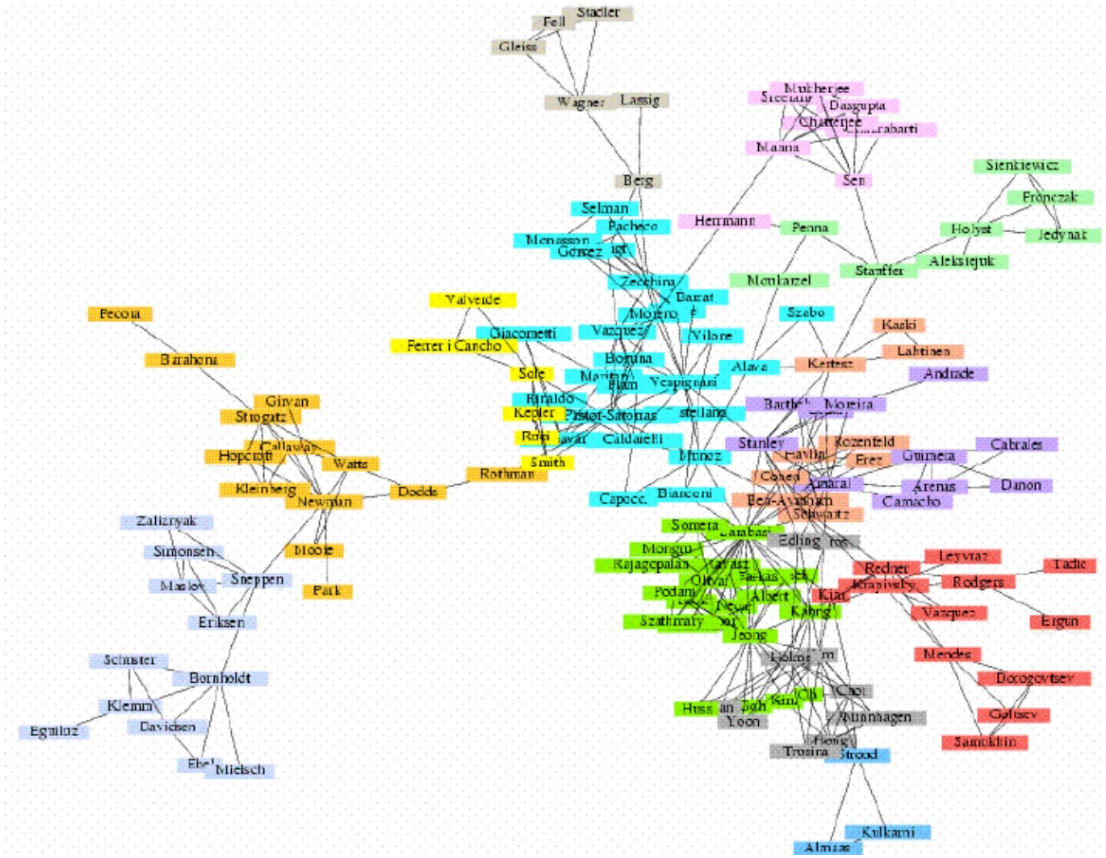
Science of complex networks can be found in many disciplices:

- Graph theory
- **Sociology**
- Communication science
- Biology / Chemistry
- Transportation / Logistics

Graph are very interdiscipliary….

Science of complex networks can be found in many disciplices:

- Graph theory
- Sociology
- **Communication science**
- Biology / Chemistry
- Transportation / Logistics



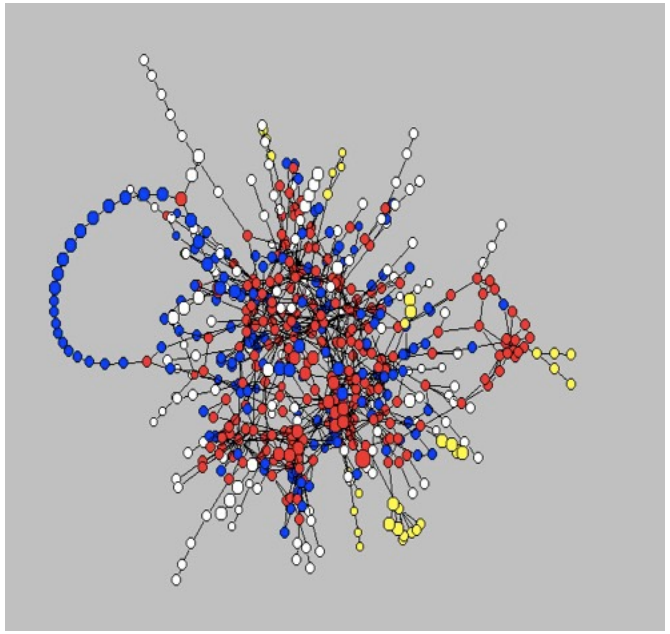
Citation or co-authorship networks

Graph are very interdiscipliary….

Science of complex networks can be found in many disciplices:

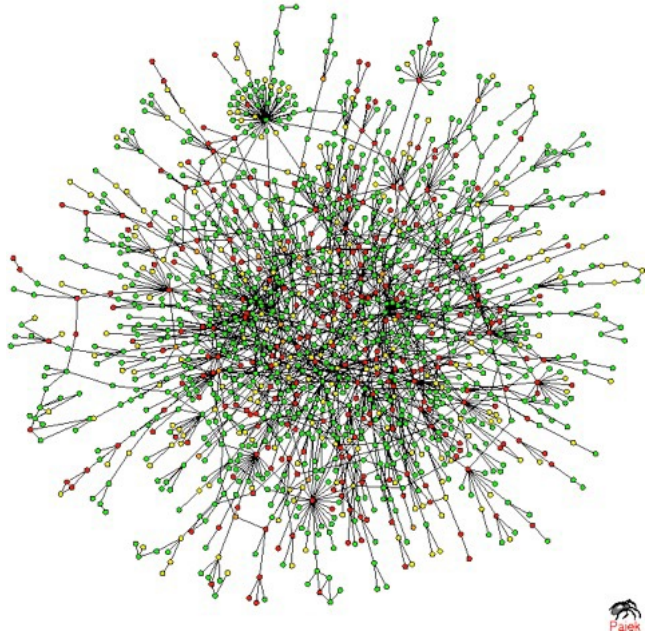
- Graph theory
- Sociology
- Communication science
- **Biology / Chemistry**
- Transportation / Logistics

Metabolic Network



Nodes: metabolites
Links:chemical reactions

Protein Interactions

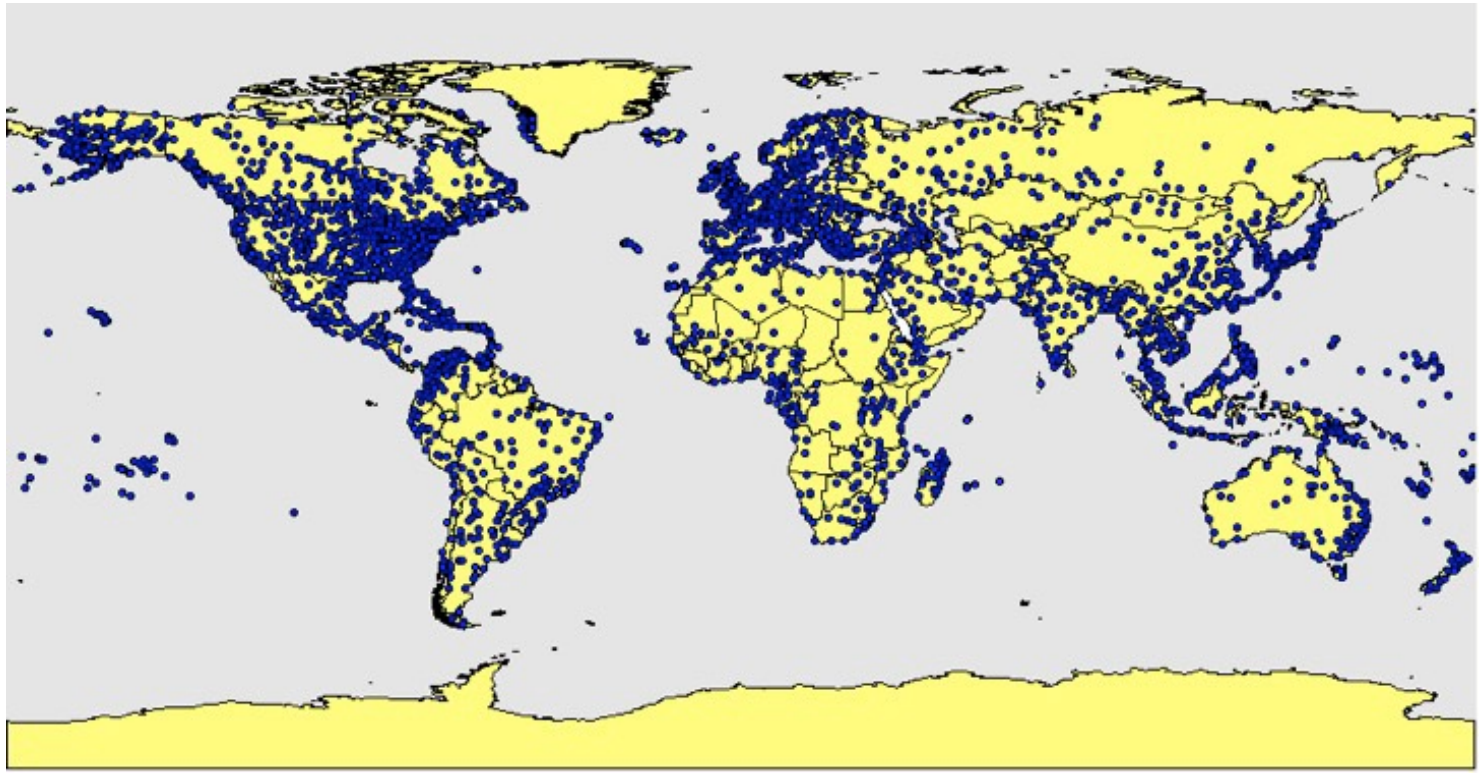


Nodes: proteins
Links: interactions

Graph are very interdiscipliary….

Science of complex networks can be found in many disciplices:

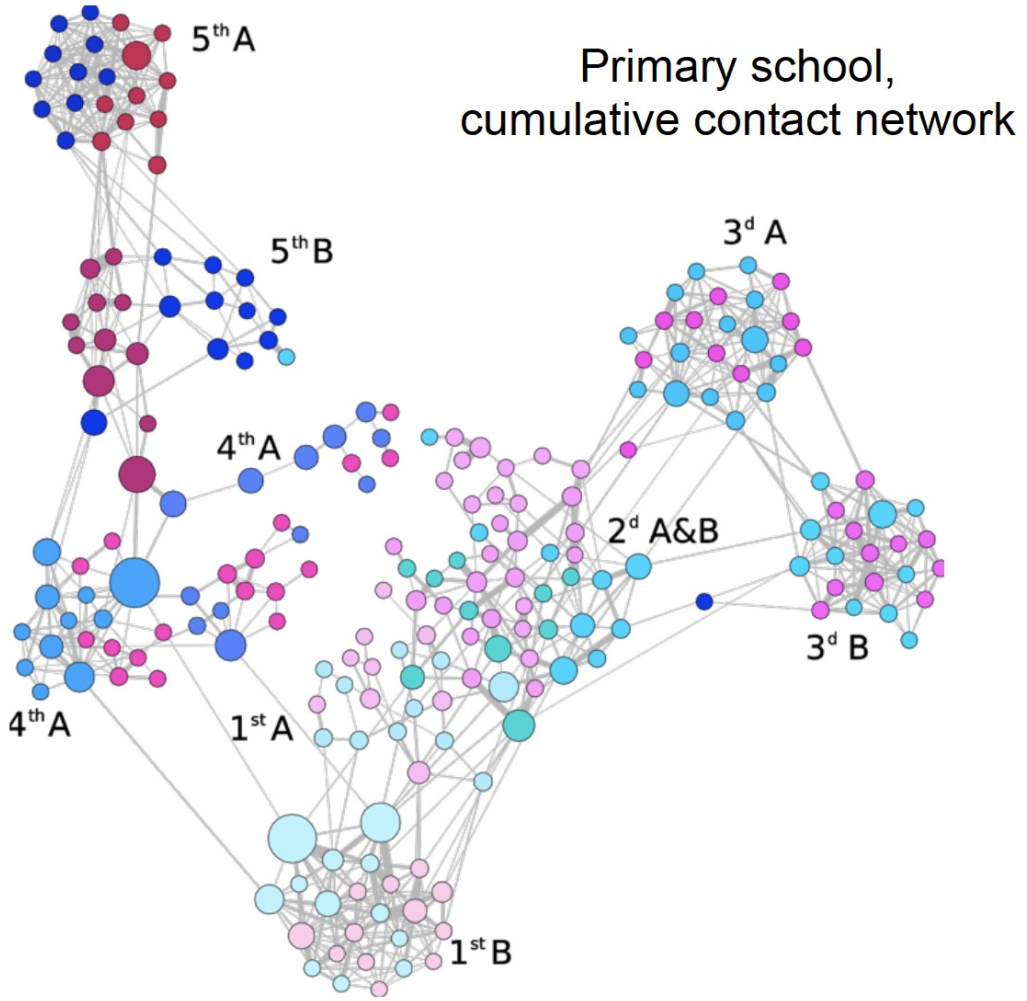
- Graph theory
- Sociology
- Communication science
- Biology
- **Transportation / Logistics**



Basic Definition

In the following, we provide definitions for some basic properties of graphs and nodes:

- The **Order** of a graph is the number of its vertices $|V|$. The **Size** of a graph is the number of its edges $|E|$.
- The **Degree** of a vertex is the number of edges that are adjacent to it. The **Neighbors** of a vertex v in a graph G is a subset of vertex $V\prime V'$ induced by all vertices adjacent to v.
- The **Neighborhood Graph** (also known as Ego Graph) of a vertex v in a graph G is the subgraph of G composed of the vertices adjacent to v and all edges connecting vertices adjacent to v.



Complete Graph and **Clique**

Definition

A **fully-connected graph** is a graph where each node is connected with any other node, e.g. any two nodes are adjacent.

A **clique**, C, in an undirected graph is defined a subset of its vertices, $C \subseteq V$, such that every two distinct vertices in the subset are adjacent. This is equivalent to the condition that the induced subgraph of G induced by C is a fully-connected graph.

- **Degree** =
- **Size** =
- **Ego Graph**

Complete Graph and Clique

Definition

A **fully-connected graph** is a graph where each node is connected with any other node, e.g. any two nodes are adjacent.

A **clique**, C, in an undirected graph is defined a subset of its vertices, C ⊆ V, such that every two distinct vertices in the subset are adjacent. This is equivalent to the condition that the induced subgraph of G induced by C is a fully-connected graph.

- **Degree** = $(|\boldsymbol{V}| - \mathbf{1})$
- **Size** = $\frac{|\boldsymbol{V}|(|\boldsymbol{V}|-\mathbf{1})}{2}$
- **Ego Graph** G'(V, E) == Graph G(V, E)

Important

the task of finding cliques of a given size n in larger graphs (clique problem) is of great interests and can be shown that is a NP-complete problem often studies in computer science.

Complete Graph and Clique

Definition

A **fully-connected graph** is a graph where each node is connected with any other node, e.g. any two nodes are adjacent.

A **clique**, C, in an undirected graph is defined a subset of its vertices, C ⊆ V, such that every two distinct vertices in the subset are adjacent. This is equivalent to the condition that the induced subgraph of G induced by C is a fully-connected graph.

- **Size** = $\frac{|\boldsymbol{V}|(|\boldsymbol{V}|-\mathbf{1})}{2}$
- **Degree** = $(|\boldsymbol{V}|-\mathbf{1})$
- **Ego Graph** G'(V, E) == Graph G(V, E)

Density of a graph

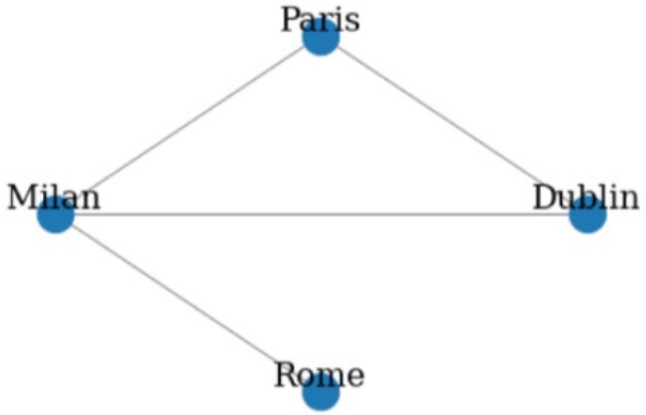
$$\boldsymbol{D} = \frac{\boldsymbol{Edges}}{\boldsymbol{Max\ Edges}} = \frac{\mathbf{2}|\boldsymbol{E}|}{|\boldsymbol{V}|(|\boldsymbol{V}|-\mathbf{1})}$$

In a **fully-connected graph**, D=1

Graph Representations

List of nodes + Edge Lists

The **Edge List** L of a graph G=(V,E) is a list of size |E| such that its element Li is a couple representing the start and the end node of the edge i. For weighted graph the list of tuple of 2-elements is replaced by list of tuple of 3-elements, where the third element of the tuple is the weight.



	Edge
1	{Milan, Dublin}
2	{Milan, Paris}
3	{Paris, Dublin}
4	{Milan, Rome}

List of nodes and neighbours for each node

The **Node-Neighbour Representation** of a graph G=(V,E) is a list of size |V| such that its element Li is a couple representing the node and a list *Ni* of all neighbours. For weighted graph the list of neighbours is replaced by a list of tuple of 2-elements, where the second element of the tuple is the weight.

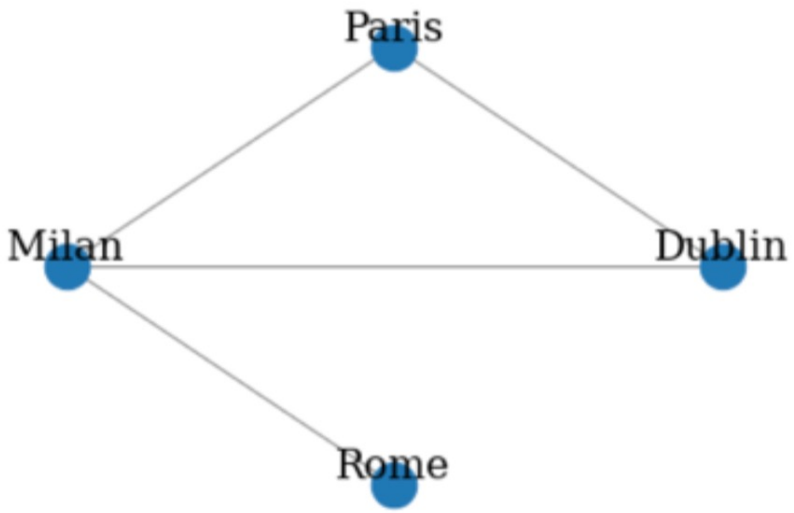
```
Dublin: ['Milan', 'Paris']
Paris: ['Milan', 'Dublin']
Rome: ['Milan']
Milan: ['Dublin', 'Paris', 'Rome']
```

Graph Representations

Adjacency Matrix

The **Adjacency Matrix** *M* of a graph *G=(V,E)* is a square matrix (|*V*| × |*V*|) matrix such that its element Mij is one when there is an edge from node *i* to node *j*, and zero when there is no edge.

$$a_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ if } (i,j) \notin E \end{cases}$$



	Milan	Paris	Dublin	Rome
Milan	0	1	1	1
Paris	1	0	1	0
Dublin	1	1	0	0
Rome	1	0	0	0

In general the adjacency matrix is a sparse matrix as the density of graphs (especially for large graphs) is low, e.g. $D \ll 1$

Graph Representations

Adjacency Matrix

We can compute the different quatitities using the adjacency matrix, in the following way:

- **Order =**
- **Degree** =
- **Size** =
- **Ego Graph**

	Milan	Paris	Dublin	Rome
Milan	0	1	1	1
Paris	1	0	1	0
Dublin	1	1	0	0
Rome	1	0	0	0

Graph Representations

Adjacency Matrix

We can compute the different quatitities using the adjacency matrix, in the following way:

- **Order =** Shape(A)[0]
- **Degree** = sum(A, axis=1)
- **Size** = sum(Degree_i) / 2
- **Ego Graph =**

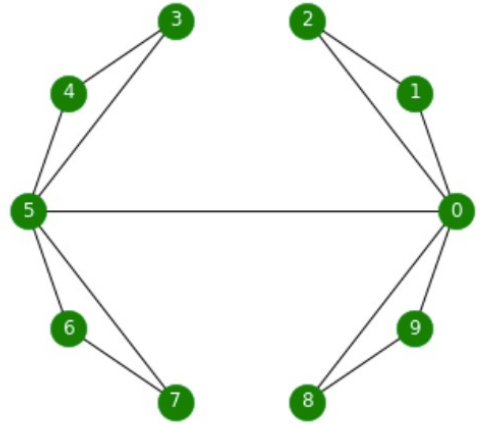
	Milan	Paris	Dublin	Rome
Milan	0	1	1	1
Paris	1	0	1	0
Dublin	1	1	0	0
Rome	1	0	0	0

```
_, idx = np.where(adj[i, :]>0)

adj[np.ix_(idx, idx)]
```

But fortunately python packages already have these metrics already implemented

Other important matrices

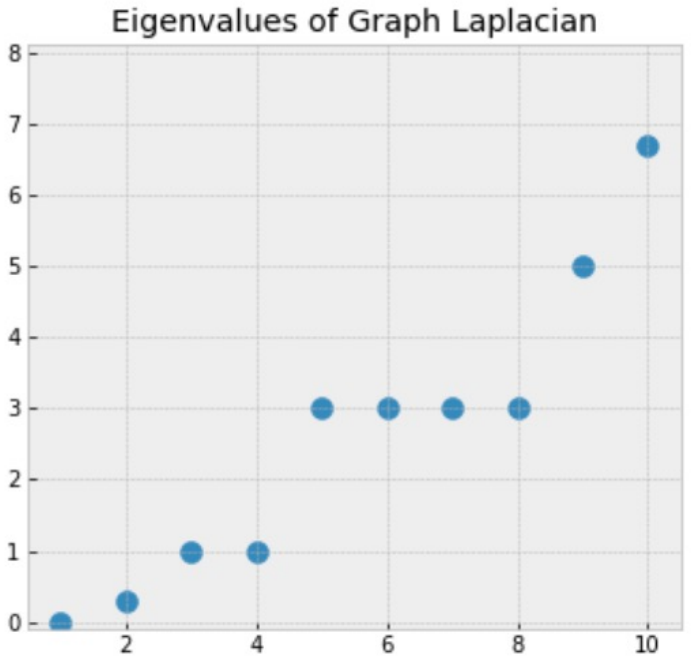


Degree Matrix $D = diag(d_i)$ d_i is the degree of the ith node

Laplacian Matrix $L = D - A$

The **Laplacian** has some important properties:

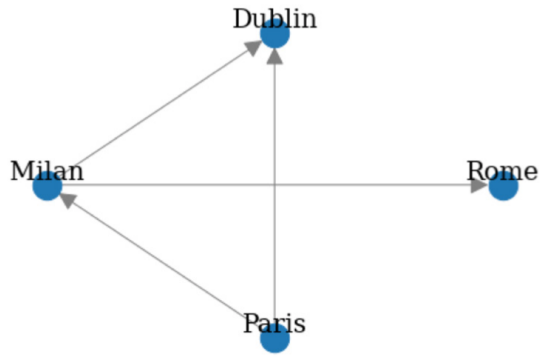
- Number of 0-eigenvalues correspond to the number of **connected components**
- The first nonzero eigenvalue is called the **spectral gap,** and provides a measure of density of the graph. If this graph was densely (fully) connected, then the spectral gap would be proportional to the number of nodes.
- The second eigenvalue is called the **Fiedler value**, and the corresponding vector is the **Fiedler vector**. The Fiedler value approximates the minimum graph cut needed to separate the graph into two connected components (e.g. a graph is connected iff Friedler value is zero). Each value in the Fiedler vector gives us information about which side of the cut that node belongs.



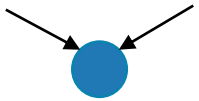
The Laplacian carries information about the structure of the graph
=> See the spectral clustering

Other types of Graphs

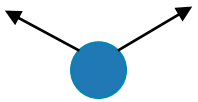
Directed Graph (DiGraph)




```
import networkx as nx
G = nx.DiGraph()
V = {'Dublin', 'Paris', 'Milan', 'Rome'}
E = [('Milan','Dublin'), ('Milan','Paris'),
('Paris','Dublin'), ('Milan','Rome')]
G.add_nodes_from(V)
G.add_edges_from(E)
```

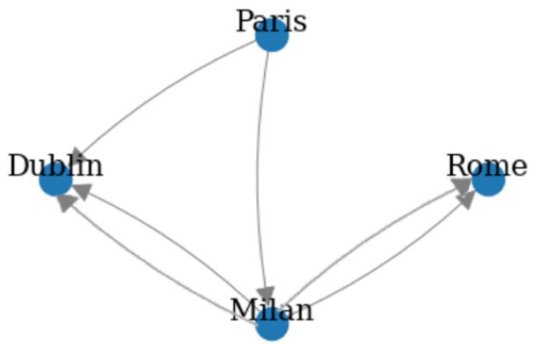


In Node Degree: `G.in_degree('Dublin')`



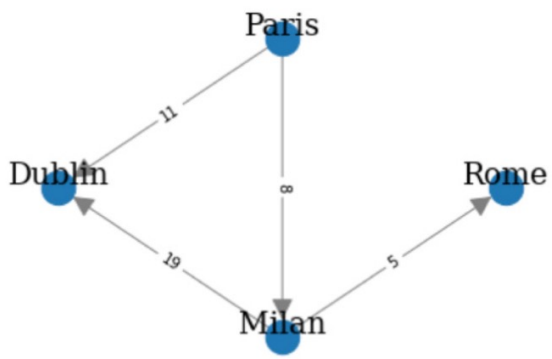
Out Node Degree: `G.out_degree('Paris')`

Multi Graph




```
dmg = nx.MultiDiGraph()
umg = nx.MultiGraph()
V = {'Dublin', 'Paris', 'Milan',
'Rome'}
E = [('Milan','Dublin'),
('Milan','Dublin'), ('Paris','Milan'),
('Paris','Dublin'), ('Milan','Rome'),
('Milan','Rome')]
dmg.add_nodes_from(V)
umg.add_nodes_from(V)
dmg.add_edges_from(E)
umg.add_edges_from(E)
```

Weidgted Graph




```
G = nx.DiGraph()
V = {'Dublin', 'Paris', 'Milan', 'Rome'}
E = [('Milan','Dublin', 19), ('Paris','Milan',
8), ('Paris','Dublin', 11), ('Milan','Rome', 5)]
G.add_nodes_from(V)
G.add_weighted_edges_from(E)
```

Getting Started with Graphs and networkx

NOTEBOOK

https://github.com/CGnal/agos-ai-course

Graph-derived metrics

Graph-derived metrics

Simple example of centrality metrics:

$$\text{Degree}_i = \text{Number of neightbour of node } i$$

We have a large list of numbers. But more in general node metrics can be studied statistically:

- **Histograms**
  Nk= number of nodes with degree k

- **Distributions**
  P(k) = Nk/N = probability that a randomly chosen node has degree k

Graph-derived metrics

Simple example of centrality metrics:

Degree_i = Number of neightbour of node i

We have a large list of numbers. But more in general node metrics can be studied statistically:

- **Histograms**
  Nk= number of nodes with degree k
- **Distributions**
  P(k) = Nk/N = probability that a randomly chosen node has degree k
- **Cumulative Distributions**
  P>(k) = probability that a randomly chosen node has degree at least k

- Homogeneous networks
  Light tails
- Heterogeneous networks
  Heavy tails

- **Statistics** (Average, Fluctuations, etc) = statistical characterization of the probability distributions

Graph-derived metrics

But in general, metrics can be

- **Local:** if they are related only to a specific node
- **Global:** if they are aggregated metrics over the entire network

Global metrics can often be derived from Local metrics by some kind of aggregation (like sum, average, harmonic average, max, etc) but it does not have to be, see the following example:

Second-order (classical) local clustering coefficient at node u.

$$C_2(u) = \frac{\#\,\triangle_u}{\#\,\wedge_u}$$

Second-order (classical) average clustering coefficient.

$$\overline{C}_2 = \frac{1}{n}\sum_u \frac{\#\,\triangle_u}{\#\,\wedge_u} = \frac{1}{n}\sum_u C_2(u)$$

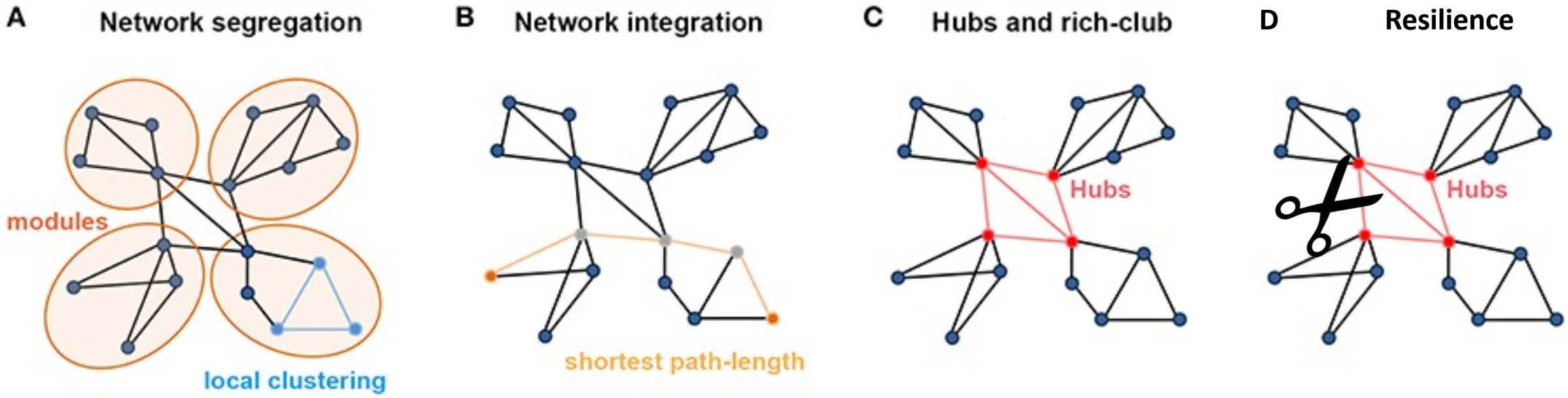
Second-order (classical) global clustering coefficient.

$$C_2 = \frac{\sum_u \#\,\triangle_u}{\sum_u \#\,\wedge_u}$$

Graph-derived metrics

We want to have a way to quantify the topological properties of graphs

A. **Network segregation.** Quantify the presence of groups of interconnected nodes, known as communities or modules, within a network
B. **Network integration.** Measure how nodes tend to be interconnected with each other.
C. **Centrality.** Assess the importance of individual nodes inside a network.
D. **Resilience.** Measure of how much a network is able to maintain and adapt its operational performance when facing failures or other adverse conditions.



Graph-derived metrics - **Segregation**

A. **Network segregation.** Quantify the presence of groups of interconnected nodes, known as communities or modules, within a network

The **clustering coefficient** is a measure of how much nodes cluster together. It is defined as the fraction of **triangles** (complete subgraph of 3 nodes and 3 edges, i.e. **clique of order** 3) around a node and is equivalent to the fraction of node's *neighbors* that are neighbors of each other.

The definition can be easily extended to clique of order n, but …

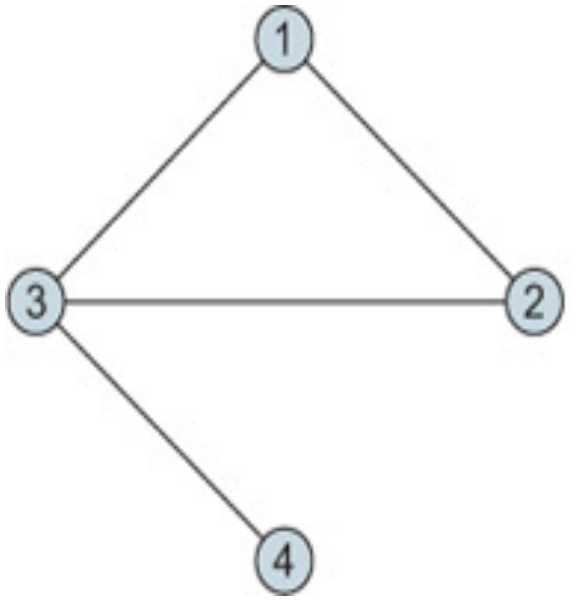
Transitivity: It can simply be defined as the ratio between the observed number of **closed triplets** (complete subgraph with 3 nodes and 3 edges) and the maximum possible number of closed triplets in the graph.

Modularity was designed to quantify the division of a network in aggregated sets of highly interconnected nodes, commonly known as **modules** or **communities**. The main idea is that networks having high modularity will show dense connections within the module and sparse connections between modules.

Graph-derived metrics - **Segregation**

Second-order (classical) global clustering coefficient. $C_2 =$

$$\frac{\sum_u \#\,\triangle_u}{\sum_u \#\,\wedge_u}$$

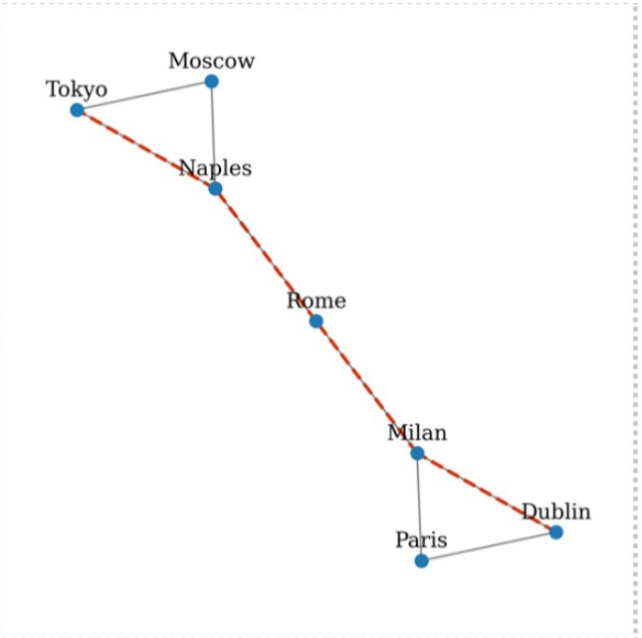


$$\frac{|\{1\text{-}2\text{-}3,\ 2\text{-}3\text{-}1,\ 3\text{-}1\text{-}2\}|}{|\{1\text{-}2\text{-}3,\ 2\text{-}3\text{-}1,\ 3\text{-}1\text{-}2,\ 1\text{-}3\text{-}4,\ 2\text{-}3\text{-}4\}|} = 0.6$$

Graph-derived metrics – Integration

B. Network integration. Measure how nodes tend to be interconnected with each other.

The concept of **distance** in a graph, is often related to the number of edges to traverse in order to reach a target node from a given source node. The set of edges connecting the node i to the node j is called **path**. When studying complex networks, we are often interested in finding the **shortest path** between two nodes, defined as l_{ij}. The distance is clearly defined only within a **connected component**.



Metrics

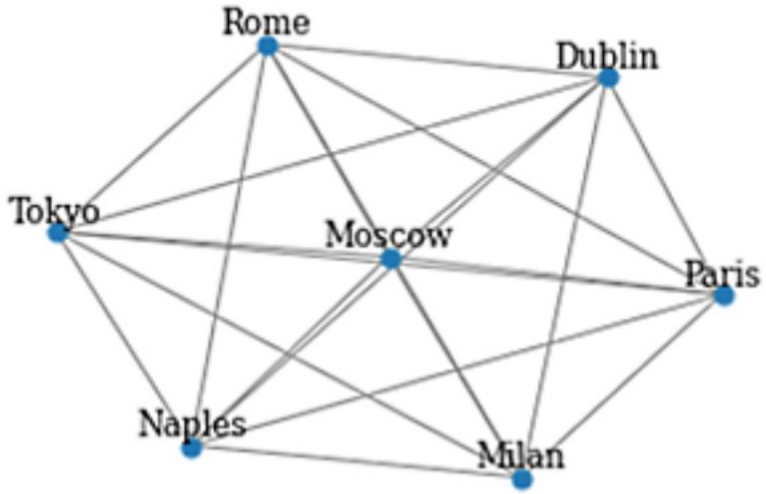
- **Diameter** $\max_{ij} size(l_{ij}) = \max_{ij} d_{ij}$
- **Characteristic Path Length** $\frac{1}{q(q-1)} \sum_{i!=j} d_{ij}$
- **Global Efficiency** $\left(\frac{1}{q(q-1)} \sum_{i,!=j} \frac{1}{d_{ij}} \right)$
- **Local Efficiency** $\frac{\mathbf{1}}{\mathbf{N}} * \sum_{\boldsymbol{i}} \left(\frac{1}{q\prime(q\prime-1)} \sum_{k,j} \frac{1}{d_{kj}} \right)_{\boldsymbol{G' induced\ by\ V_i}}$

Graph-derived metrics - **Integration**

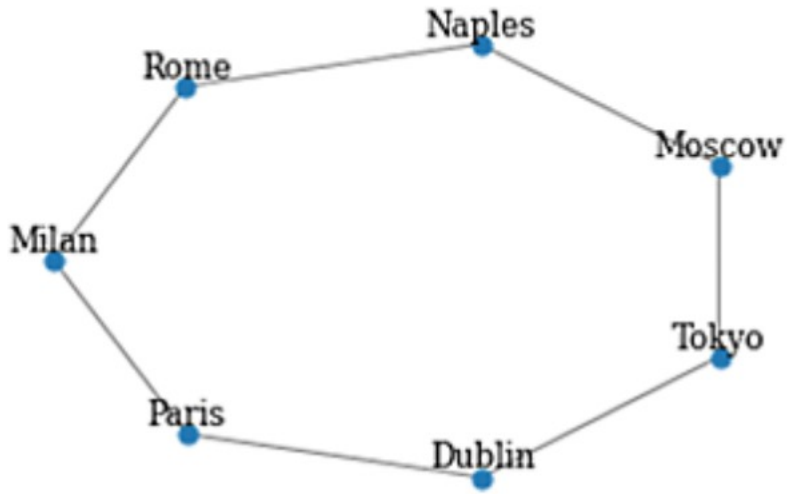
- **Global efficiency** is the average of the inverse shortest path length for all pairs of nodes.
- Let d(i,j) be the shortest path from node *i* to node *j* and *n* be the number of nodes

$$E(G) = \frac{1}{n(n-1)} \sum_{i \neq j \in G} \frac{1}{d(i,j)}$$

```
nx.global_efficiency(G)
```



Global Efficiency:1.0



Global Efficiency:0.61

Graph-derived metrics – **Centrality**

C. Centrality. Assess the importance of individual nodes inside a network.

- **Degree centrality**. It is directly connected with the *degree* of a node, measuring the number of edges *incident* on a certain node i. Note that, if the graph is *directed*, **in-degree centrality** and **out-degree centrality**

$$\frac{1}{n-1} * Degree_i$$

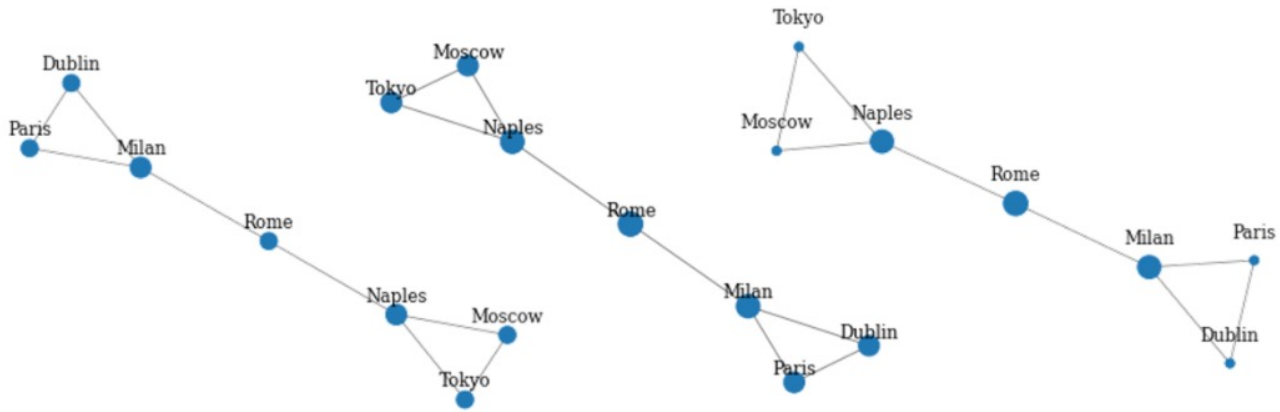
- **Closeness centrality** attempts to quantify how much a node well connected to other nodes. It refers to the inverse of the average distance of a node i to all the other nodes in the network.

$$C(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(v,u)},$$

- The **betweenness centrality** evaluates how much a node acts as a **bridge** between other nodes. Even if poorly connected, a node can be strategically connected, helping at keeping the whole network connected.

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

$\sigma(s,t)$ is the number of shortest (s,t)-paths



Degree centrality **Closeness centrality** **Betweenness centrality**

Graph-derived metrics – Centrality

C. Centrality. Assess the importance of individual nodes inside a network.

Who you are connected to matters …

- **Eigenvector centrality.** Based on the fact that connections to high-scoring nodes contribute more to the score of the node than connections to low-scoring nodes

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

Eigenvector

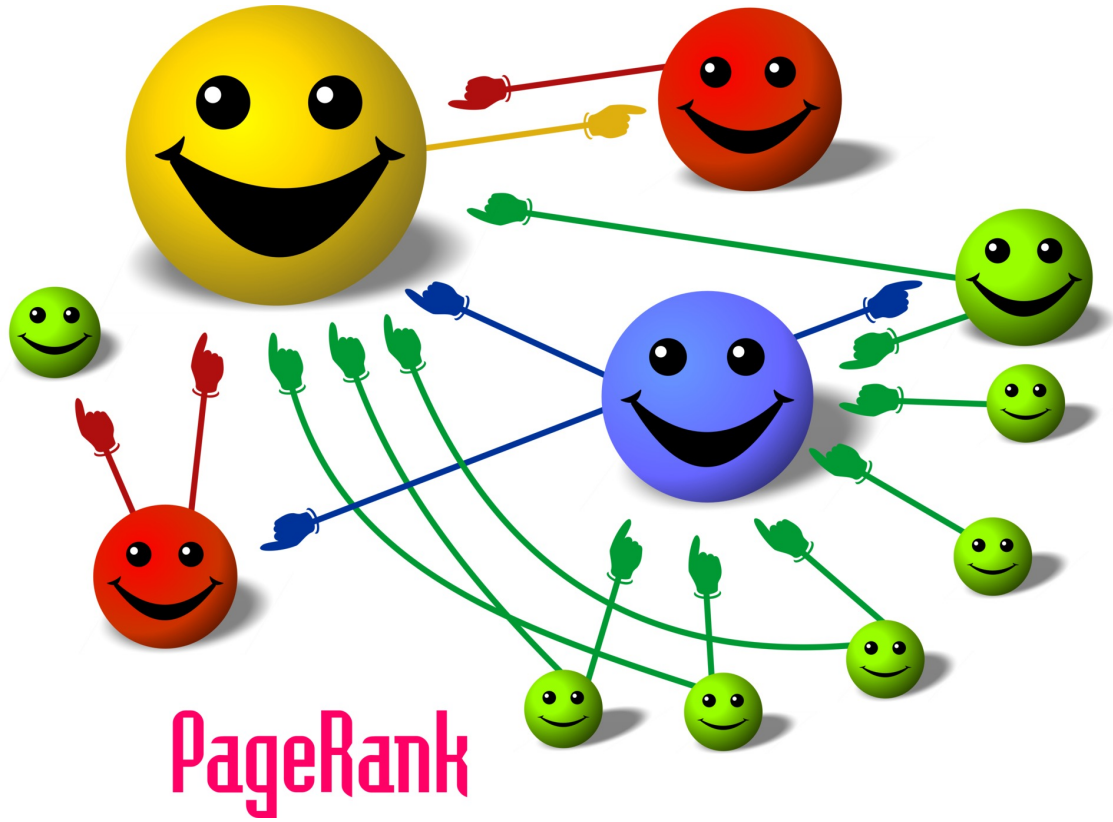
Adjacency Matrix

Eigenvalue

Centrality is the value of the n-th component for the eigenvector associated to the largest eigenvalue

- **PageRank** special case for the eigenvector centrality in the case of **directed** links. Special case of a power iterative procedure

$$PR[A] = \frac{1-d}{N} + d\left(\sum_{k=1}^{n} \frac{PR[P_k]}{C[P_k]}\right)$$



Graph-derived metrics – Resilience

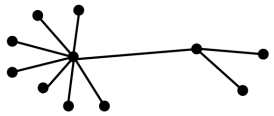
D. Resilience. Measure of how much a network is able to maintain and adapt its operational performance when facing failures or other adverse conditions.

Assortativity is used to quantify the tendency of nodes being connected to similar nodes, usually based on the degree of the nodes. There are several ways of measure such correlations. One of the most commonly used is the **Pearson correlation coefficient** between the degrees of directly connected nodes (nodes on two opposite ends of a link).

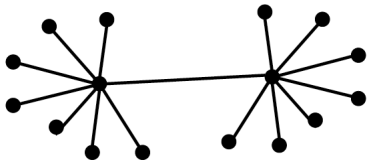
Neighbor connectivity uses the mean of the conditional probability of a node of degree k to be attached to a node of degree k'

$$\sum_{k'} k' P(k' \mid k)$$

If the function is increasing with k the network is assortative, otherwise it is dissortative.



Technological networks → **More assortative** → **Social networks**



These are **global** measure of assortativity. There also exists local measure but they are less intuitive and based on the contribution of each node to the global assortativity.

Graph-derived metrics – **Full List**

1. VertexCount
2. EdgeCount
3. VertexDegree
4. VertexInDegree
5. VertexOutDegree
6. GraphDistance
7. GraphDistanceMatrix
8. VertexEccentricity
9. GraphRadius
10. GraphDiameter
11. Connectivity Measures
12. VertexConnectivity
13. EdgeConnectivity
14. Centrality Measures
15. ClosenessCentrality
16. BetweennessCentrality
17. DegreeCentrality
18. EigenvectorCentrality
19. KatzCentrality
20. PageRankCentrality
21. HITSCentrality
22. RadialityCentrality
23. StatusCentrality
24. EdgeBetweennessCentrality
25. GraphReciprocity
26. GlobalClusteringCoefficient
27. MeanClusteringCoefficient
28. LocalClusteringCoefficient
29. GraphAssortativity
30. VertexCorrelationSimilarity
31. MeanNeighborDegree
32. MeanDegreeConnectivity
33. VertexDiceSimilarity
34. VertexJaccardSimilarity
35. VertexCosineSimilarity

GRAPH MODELS

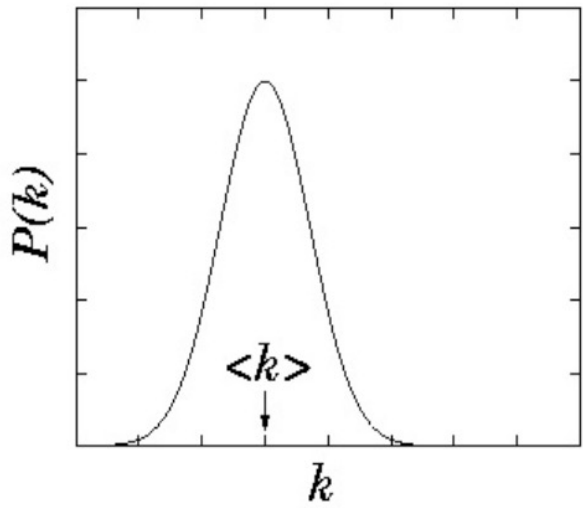
Graph Models - Erdös-Renyi (1960)

Generative Model

This represent a model for a generating random graph. Take **n** nodes and connect two nodes with probability **p**

(Average) number of edges ~ …

(Average) degree ~



The distribution of the degree of any particular vertex is ……..

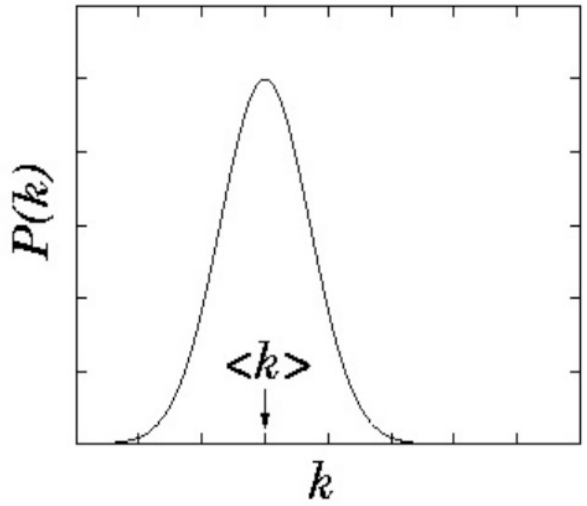
Graph Models - Erdös-Renyi (1960)

Generative Model

This represent a model for a generating random graph. Take **n** nodes and connect two nodes with probability **p**

(Average) number of edges ~ $\frac{n(n-1)}{2}p$

(Average) degree ~ $p\ (n-1)$



The distribution of the degree of any particular vertex is binomial

$$P(\deg(v) = k) = \binom{n-1}{k} p^k (1-p)^{n-1-k},$$

And for large n and fixed $n\,p = \langle k \rangle$, the distribution tends to the Poisson distribution

$$P(k) = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!}$$

Depending on p, we transit from having large number of small graphs (when p is small) or large cluster with small subgraphs

Small clustering $\langle C \rangle = p = \frac{\langle k \rangle}{N}$

Short distances $\langle l \rangle = \frac{\log(n)}{\log(p(n-1))} \sim \frac{\log(n)}{\log(\langle k \rangle)}$

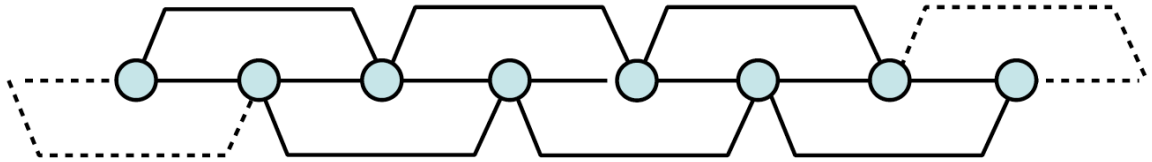
Graph Models - Watts & Strogatz (1998)

Graph with short distance

This model was used by the authors to study the behavior of **small-world networks**, that is to say, networks that resemble to some extent common social networks.

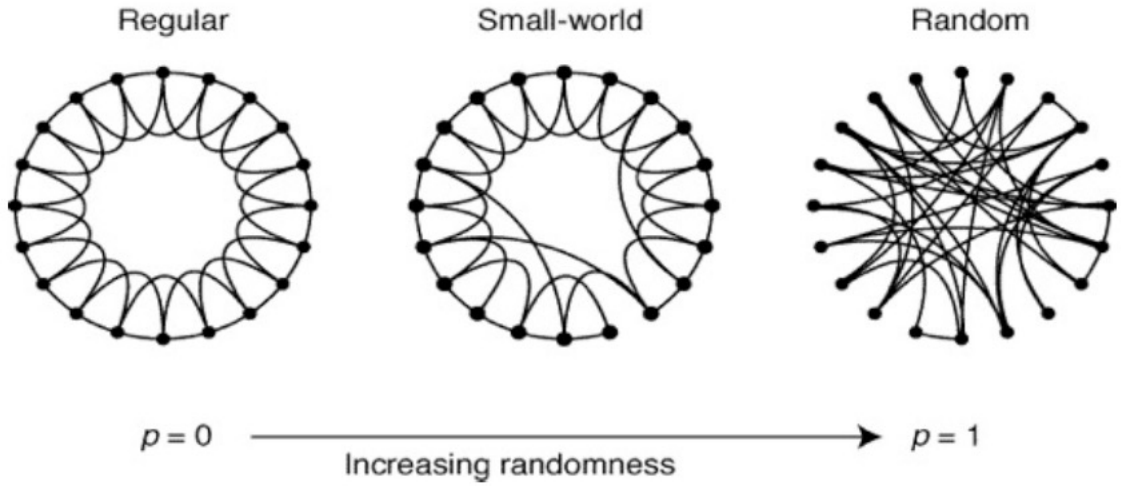
Motivation:

- random graph: short distances but no clustering
- regular structure: large clustering but large distances

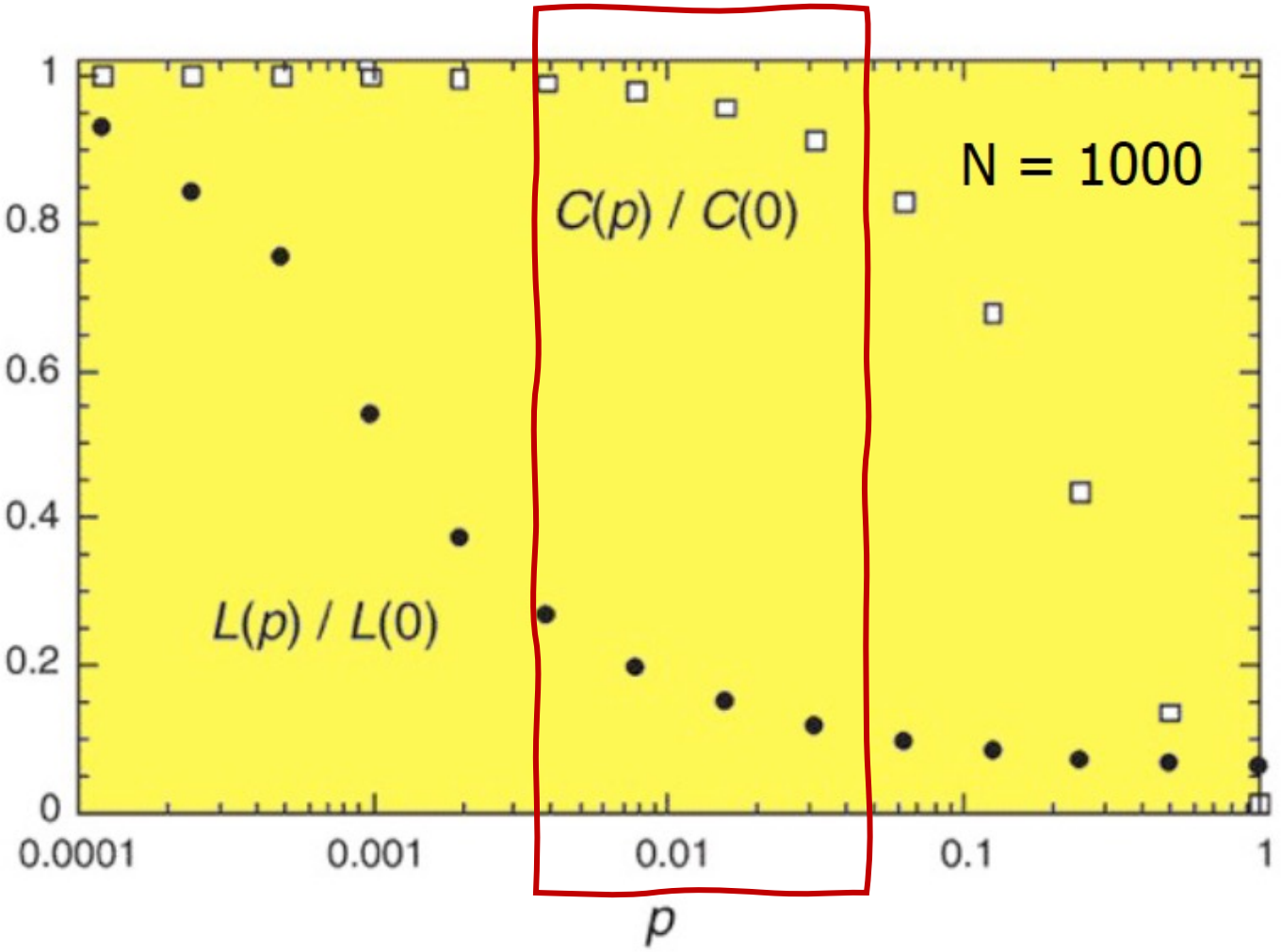


Generative Process

1. N nodes arranged in a line/circle
2. Each node is linked to its 2k neighbors on the circle, k clockwise, k anticlockwise
3. Each edge going clockwise is rewired towards a randomly chosen other node with probability p



Graph Models - Watts & Strogatz (1998)



Large clustering coefficient
Short distance (diameter)

Graph Models - Barabási-Albert (1999)

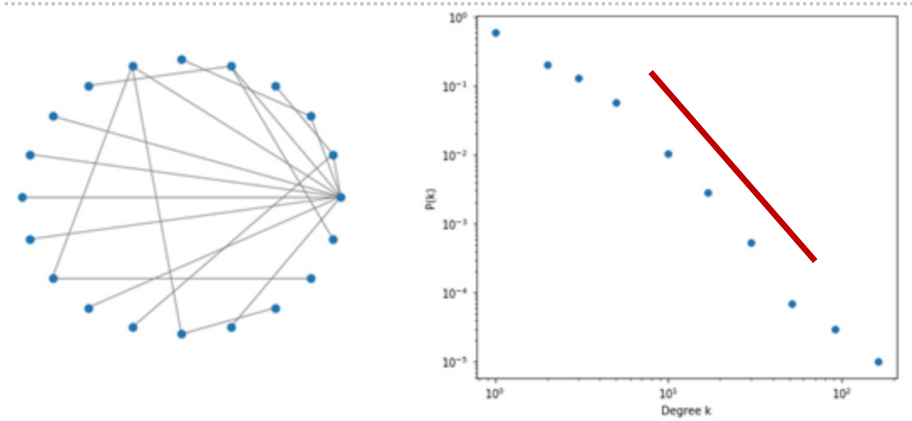
The model proposed by Albert & Barabasi is based on a generative model that allows to create random scale-free networks by using a **preferential attachment** schema as new nodes are added to the networks.

Generative Model

The underlying idea of this model is that the probability for a new node to be attached to an existing node i depends on the degree of the i-th node according to the following formula:

$$p_i = \frac{k_i}{\sum k_i}$$

Thus, nodes with a large number of edges (hubs) tend to develop even more edges, whereas nodes with few links will not develop other links (periphery). Networks generated by this model exhibit a *power-law distribution* for the connectivity (i.e. degree) between nodes, which is observed also for the Web and Scientific papers networks.



Benchmarks

However, it is often useful to already have real-data to play with. Some source in the following:

- **Networkx** (https://networkx.org/documentation/stable/reference/generators.html)
- **Network Data Repository** (http://networkrepository.com/)
- **Stanford Network Analysis Platform** (SNAP), https://snap.stanford.edu/index.html
- **Open Graph Benchmark** (https://ogb.stanford.edu/)