Complex systems and networks

github.com/martinopiaggi/polimi-notes

Table of Contents

1 Networks

A network is represented by a graph with N nodes (or vertices) and L links (or edges).

- **Undirected Networks**: No direction associated with links.
- **Directed Networks**: Links have a specified direction.
- **Weighted Networks**: Links have weights representing the strength or capacity of the connection.

1.1 Adjacency and Laplacian matrices

Representation of networks using the **adjacency** matrix A is popular. We simply put a 1 in position ij if exists a link between node i and node j, 0 otherwise. Typically A is a sparse matrix (small density).

A weighted network is described by the $N \times N$ weight matrix $W = w_{ij}$ where $w_{ij} > 0$ if the link $i \to j$ exists, $w_{ij} = 0$ otherwise.

The $N \times N$ Laplacian matrix is an alternative representation of the network, given by:

$$L = \text{diag}\,(k_1, k_2, \ldots, k_N) - A$$

It's symmetric and zero-row-sum.

1.2 Bipartite network

Bipartite networks are made up of two different classes of nodes. Each class has a certain number of nodes. Only nodes from different classes can be connected to each other.

For example, in a network of papers and authors, a paper node can only be connected to an author node, and vice versa.

To represent a bipartite network, we can use a rectangular incidence matrix called B.

We can also create a **projected network** by projecting the bipartite network onto one of the classes of nodes, denoted as S. In this projected network, the weight of a link between two nodes is determined by the number of common neighbours they have in the original network.

To obtain the weight matrix W of the projected network, follow these steps:

1. Compute $M = B^T B$, where B^T is the transpose of matrix B. This multiplication will give an intermediate matrix M.
2. Set the diagonal entries of M to zero.

This resulting matrix M will correspond to the weight matrix W of the projected network.

1.3 Strongly connected component

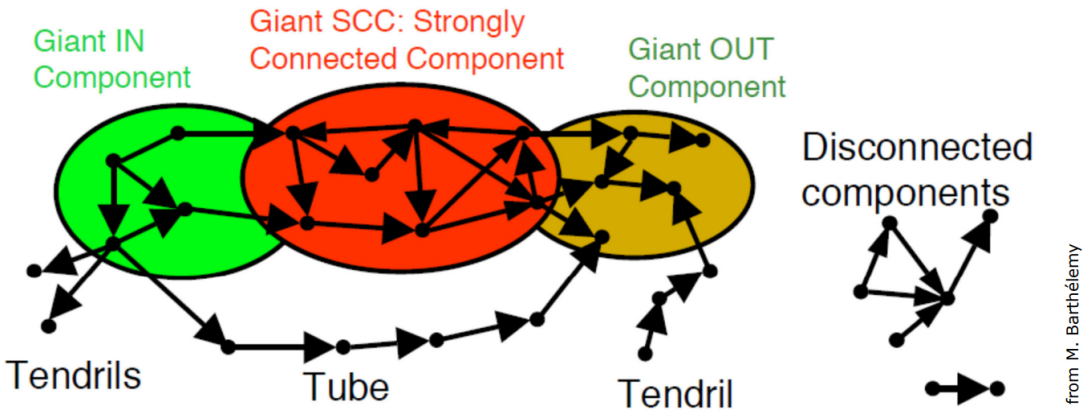
It is common to classify components based on their specific properties or the extent of their connectivity. We often refer to highly interconnected sub-sections of a graph as “Strongly Connected Components” or “Giant Out Components” indicating intensive links and relations among its nodes. At the same time, certain components might demonstrate no connectivity or weak connections, forming what we describe as “Disconnected Components”.

Algorithm:

1) For each node of the directed graph check if it’s in “communication” with each other node. Where communication means that x must have a path to y but also y must have one to x (the graph is **directed**)

After finding the **SCC** we can also identify:

- **in**: set of nodes with links that are pointing to the **SCC**
- **out**: set of nodes with links that are reachable from the **SCC**
- **tubes**: links that are connecting nodes in the in set with nodes of out set



1.4 Basic properties

1) For each node of the directed graph check if it’s in “communication” with each other node. Where communication means that x must have a path to y but also y must have one to x (the graph is **directed**)

The **Distance** is the length (in number of links) of the shortest path connecting two nodes i and j, denoted as d_{ij}. We can define:

Diameter (D):

$$D = \max(d_{ij})$$

The **Average Distance (d)**

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

And the **Efficiency** :

$$E = \langle \frac{1}{d_{ij}} \rangle = \frac{1}{N(N-1)} \sum_{i \neq j \in V} \frac{1}{d_{ij}}$$

with $\frac{1}{d_{ij}} = 0$ if path $i \to j$ does not exist:

> "An unconnected pair contributes 0 to the efficiency of the network"

1.4.1 Density

The **density** of a network:

$$\rho = \frac{L}{N(N-1)} \text{ (dir.) or } \rho = \frac{L}{N(N-1)/2} \text{ (undir.).}$$

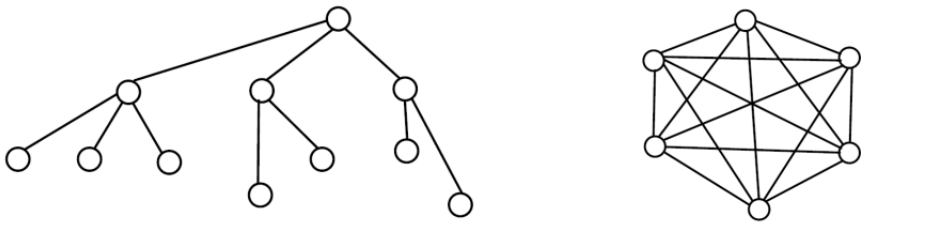
1.4.2 Clustering

Clustering (or Transitivity) Coefficient quantifies "local link density" by counting the triangles in the network. **Local Clustering Coefficient** c_i of node i:

$$c_i = \frac{e_i}{\frac{k_i(k_i-1)}{2}}$$

Where k_i is the degree of i and e_i the number of links directly connecting neighbors of i.

The **global clustering** coefficient is calculated by averaging individual node coefficients $C = \langle c_i \rangle$. Global clustering coefficient of a tree is zero (think of the triangles):



Examples:

Network	Size	Clustering coefficient	Average path length
Internet, domain level [13]	32711	0.24	3.56
Internet, router level [13]	228298	0.03	9.51

Network	Size	Clustering coefficient	Average path length
WwW [14]	153127	0.11	3.1
E-mail [15]	56969	0.03	4.95
Software [16]	1376	0.06	6.39
Electronic circuits [17]	329	0.34	3.17
Language [18]	460902	0.437	2.67
Movie actors [5, 7]	225226	0.79	3.65
Math. co-authorship [19]	70975	0.59	9.50
Food web [20, 21]	154	0.15	3.40
Metabolic system [22]	778	-	3.2

1.5 Degree Distribution in Networks

Let's start with the **Degree and Strength of a Node**:

- **Degree** k_i in undirected network is the **number** of links connected to node i.
- **Strength** s_i in a weighted network is the **total weight** of the links connected to node i.

In a directed network we can distinct **in**,**out** and **total** degree/strength.

The **Degree Distribution $P(k)$ of a network is the fraction of nodes having exactly degree k.

$$P(k) = \frac{\text{Number of nodes with degree } k}{N}, \quad \sum_k P(k) = 1$$

It is often more practical to consider the **Cumulative Degree Distribution** which is the fraction of nodes with degree $\geq k$:

$$\bar{P}(k) = \frac{\text{Number of nodes with degree } \geq k}{N} = \sum_{h=k}^{k_{\max}} P(h), \quad \bar{P}\left(k_{\min}\right) = 1$$

And the **moments of Degree Distribution**: $\langle k^r \rangle$ are:

$$< k^r >= \sum_k k^r P(k) \quad , \quad r = 1, 2, \ldots$$

The first moment $(r = 1)$ is the average degree

$$< k >= \sum_k k P(k) = \frac{1}{N}$$

which interestingly can be also computed as

$$< k >= \sum_i k_i = \frac{2L}{N}$$

1.5.1 Authorities and hubs

Homogeneous Network: All nodes have the same degree. **Heterogeneous Network (Real-World)**: Broad degree distribution, some nodes are highly connected (hubs), while most have few connections.

Said this we can talk about **authorities** and **hubs** scores which are based taking into account the different role of in-out links. The formulas of authorities and hubs score are part of a recursive process: the authority score of a node depends on the hub scores of nodes pointing to it, and the hub score of a node depends on the authority scores of the nodes it points to. This interdependency is key in networks where the importance of a node is not just a function of how many connections it has, but also how important those connections are.

Authority Score x_i :

$$x_i = \alpha \sum_j a_{ji} y_j$$

The authority score is calculated by summing the hub scores y_j of all nodes j that point to i. The adjacency matrix a_{ji} is used to know if there's a link from node $j \to i$. The factor α is a normalization constant to keep the scores from escalating too high.

Hub Score y_i:

$$y_i = \beta \sum_j a_{ij} x_j$$

The hub score y_i of a node i is computed by summing the authority scores x_j of all nodes j that node i points to. The adjacency matrix a_{ij} indicates whether there is a link from node $i \to j$. The factor β serves as a normalization constant to prevent the scores from becoming excessively large.

1.5.2 Nearest neighbors

The **degree distribution of nearest neighbours** $Q(h)$ specifies the **fraction** of nodes' neighbours having exactly degree h (=the probability that a randomly selected neighbour of a randomly selected node has degree h):

It is not $P(k)$ but it is **biased towards highest degrees**:

$$Q(h) = \frac{\text{n. of links from nodes of degree } h}{\text{n. of links from nodes of any degree}} = \frac{h(P(h)N)}{\sum_k k(P(k)N)} = \frac{hP(h)}{< k >}$$

Thus the **average degree of nearest neighbours** k_{nn} is:

$$k_{nn} = \sum_h hQ(h) = \sum_h \frac{h^2 P(h)}{< k >} = \frac{< k^2 >}{< k >} = \frac{< k >^2 + \sigma^2}{< k >} = < k > + \frac{\sigma^2}{< k >}$$

which is larger than $< k >$ provided $\sigma^2 \neq 0$ (non strictly homogeneous network).

If **variance** is not equal to zero, $Q(h)$ is always higher than $P(k)$.

This is the mathematical foundation of the **friendship paradox** which states that on average, your friends will have more friends than you do. As navigating randomly through a network is more likely to encounter nodes with many connections. This idea has applications for finding hub nodes within a network.

1.5.2.1 Correlated Networks

There is a correlation between $P(k)$ with $Q(k)$? In a degree-correlated network, the probability $P(h|k)$ that the neighbour of a node with degree k has degree h depends on k. Correlations in such a network can be captured by the average nearest neighbour degree **function**:

$$k_{nn}(k) = \sum_h hP(h|k)$$

Practically, this function is computed as:

$$k_{nn}(k) = \frac{1}{N(k)} \sum_{i|k_i=k} \frac{1}{k} \sum_j a_{ij} k_j$$

where $N(k)$ is the number of nodes with degree k. In an **assortative network**, high-degree nodes tend to connect to other high-degree nodes. Conversely, in a **disassortative network**, high-degree nodes tend to connect to low-degree nodes.

In an **assortative network**, high-degree nodes tend to connect to other high-degree nodes. Conversely, in a **disassortative network**, high-degree nodes tend to connect to low-degree nodes.

In summary, $k_{nn}(k)$ provides insight into the degree correlation patterns within a network, revealing how nodes of a certain degree tend to connect with other nodes of specific degrees.

1.6 Random walks on networks

A **random walk** is a path formed by a sequence of random steps. Random walks have many variants:

- Discrete vs continuous time
- Uniform vs non-uniform steps
- Markovian vs non-Markovian processes
- etc.

Random walks have practically unlimited applications across all scientific fields: ecology, economics, psychology, computer science, physics, chemistry, and biology.

In a unweighted network, the random walker at node i chooses an out-link $i \to j$ with uniform probability:

$$p_{ij} = \frac{a_{ij}}{k_i^{\text{out}}}$$

In a weighted network, the out-link is chosen with probability proportional to its weight:

$$p_{ij} = \frac{w_{ij}}{\sum_j w_{ij}} = \frac{w_{ij}}{s_i^{\text{out}}}$$

The **transition matrix** $P = [p_{ij}]$ is the $N \times N$ matrix which represents the probabilities of a random walker moving from node i to any node j. The probabilities are not symmetric because each row is normalized with k_i^{at}.

$\pi_{i,t}$ = state probability = probability of being in node i at time t ($\sum_i \pi_{i,t} = 1 \forall t$). $\pi_t = \begin{pmatrix} \pi_{1,t} & \pi_{2,t} & \cdots & \pi_{N,t} \end{pmatrix}$ evolves according to the Markov chain equation:

$$\pi_{t+1} = \pi_t P \quad , \quad \pi_{i,t+1} = \sum_{n=1}^{N} \pi_{n,t} p_{ni}$$

If the network is strongly connected, then:

- The transition matrix $P = [p_{ij}]$ is irreducible.
- There exists a unique stationary state probability distribution $\pi = \pi P$, which is strictly positive ($\pi_i > 0$ for all i).

$$\pi_i = \text{fraction of time spent on node } i = \text{centrality of node } i$$

Observations:

- In undirected networks, π_i is the (rescaled) node strength: $\pi_i = s_i / \sum_j s_j$.
- In directed networks, π_i is mostly correlated to the in-strength s_i^{in} (e.g., WWW).

1.6.1 Google PageRank

The solution to $\pi = \pi P$ might not be unique, positive, or the Markov chain might not even be well-defined. PageRank uses **teleportation**: at each time step, the random walker has a probability $\gamma > 0$ to jump to a randomly selected node.

$$p_{ij} \to p'_{ij} = (1 - \gamma) \frac{w_{ij}}{s_i^{\text{out}}} + \gamma \frac{1}{N}$$

A suitable value for γ is not too large (to avoid heavily modifying the network) nor too small (to prevent π_i from being too sensitive to γ). The standard (Google) value is $\gamma = 0.15$.

PageRank's revolution is that the rank emerges naturally from the self-organized internet structure. A high rank is achieved only if many other nodes point to it, creating a naturally emerged rank.

1.7 Node centrality

The **centrality** of a node is a measure of its **importance** in the network. The importance of a node can trivially be captured by the number k_i of its neighbours (or in a weighted networks by the strength s_i). Considering *central* a node which has an high number of connections is a simple measure but we can also consider other criteria.

1.7.1 Betweenness centrality

Betweenness Centrality of node i is calculated as the number of all shortest paths in the network that pass through i .

$$b_i = \sum_{j,k} \frac{\text{n. of shortest paths connecting } j,k \text{ via } i}{\text{n. of shortest paths connecting } j,k} = \sum_{j,k} \frac{n_{jk}(i)}{n_{jk}}$$

Betweenness can be a valid and very useful property to consider in the computation of centrality in some models/networks.

1.7.2 Closeness centrality

A node is considered central if it is, on average, close to all the other nodes in a network. This means it has better access to information, more direct influence on other nodes, and so on. To calculate this average distance, we can use the formula:

$$l_i = \frac{1}{N-1} \sum_j d_{ij}$$

The closeness centrality is defined as

$$c_i = \frac{1}{l_i} = \frac{N-1}{\sum_j d_{ij}}$$

If the network is directed, we need to differentiate between in-closeness and out-closeness. If the network is weighted, there are several alternative definitions available for closeness centrality.

1.7.3 Eigenvector centrality

> " I'm important if I'm friend of important people "

It's just called **eigenvector** because it's computed using eigenvectors of the adjacency matrix. The centrality γ_i is (proportional to) the sum of the centralities of the neighbours (i.e., a node is important if it relates to important nodes).

$$\gamma_i = \alpha \sum_j a_{ij}\gamma_j$$

Letting $\gamma = \begin{bmatrix} \gamma_1 & \gamma_2 & \dots & \gamma_N \end{bmatrix}^T$ and $\lambda = 1/\alpha$, we obtain the eigenvector equation:

$$A\gamma = \lambda\gamma$$

If the network is connected ($= A$ is irreducible), the centralities γ_i are given by the only solution with $\lambda > 0, \gamma_i > 0$ for all i (Frobenius-Perron theorem).

- "Quantitative Sociologists" (who is the most influential individual?)
- applications in web searching (with some modifications: Google "PageRank" which is the most important webpage?)
- another modification is Katz (or alpha-) centrality: $\gamma_i = \alpha \sum_j a_{ij}\gamma_j + \beta$ just a variant to solve some degeneration problem of the eigenvector centrality in not completely connected networks.

1.7.4 Random walker centrality

$\pi_{i,t}$ is the state probability, which means the probability of being in node i at time t :

$$\Sigma_i \pi_{i,t} = 1 \, \forall t$$

With $t \to \infty$ we can define the centrality of node π_i represents fraction of time spent on node i by a random walker (only if the network is strongly connected). Basically this quantity/metric/centrality represents the frequency to find the random walker at node i.

> "A node is important if is visited many times"

Random walker centrality is useless in **undirected** and **unweighted** networks π_i since it is the *rescaled* node degree:

$$\pi_i = \frac{k_i}{\sum_j k_j} = \frac{k_i}{k_{tot}}$$

In **undirected** but **weighted** networks:

$$\pi_i = \frac{s_i}{\sum_j s_j} = \frac{s_i}{s_{tot}}$$

In **directed** networks, π_i turns out to be mostly correlated to the in-strength s_i^{in}:

$$\pi_i = \frac{s_i^{in}}{\sum_j s_j} = \frac{s_i^{in}}{s_{tot}}$$

2 Network models

Network Models are **algorithms** that generate networks with particular properties, aiming to emulate the formation processes of real-world networks.

2.1 Random (Erdos-Renyi) networks

Two alternatives procedure:

- $G(N, L)$:
  - L number of links is specified and then extracted randomly pairs
  - For large N , the degree is **Poisson-distributed** with $\langle k \rangle \approx \frac{2L}{N}$
- $G(N, p)$:
  - Start from a graph with N nodes and **no** links but a probability p as parameter
  - Each pair is connected based on the probability p
  - The degree distribution is **binomial** $P(k) = \binom{N-1}{k} p^k (1-p)^{(N-1-k)}$
  - resulting with $\langle k \rangle = p(N-1)$

Properties of networks generated by this model are:

- The network is **homogeneous**.
- Node degrees have small fluctuations around $\langle k \rangle$.
- a **giant component** emerging when $\langle k \rangle > 1$
- an average distance that grows logarithmically with N (indicating a **small-world** effect)
- a clustering coefficient that tends to zero as N increases.

2.2 Barabási-Albert algorithm for Scale-free networks

Real networks often evolve spontaneously, rarely exhibiting a uniform/regular structure where each node consistently has the same number of connections. The Barabasi-Albert algorithm, introduced by Reika Albert and her collaborator in 2000 generates **scale-free** networks, which are characterized by a few key properties:

1. **Preferential Attachment**: New nodes are added to the network one at a time and are more likely to connect to nodes that already have a high degree.
2. **Heterogeneity**: node degrees have large fluctuations around the average degree $\langle k \rangle$ and there is no typical scale of node degree.
3. **Power-Law Degree Distribution**: $P(k) \approx k^{-\alpha}$ implies that there are a few nodes with a very high degree and many nodes with a low degree.

- $\langle k \rangle = 2m$ is the average degree distribution
- The average distance tends to $d \approx \frac{logN}{loglogN}$
- The clustering coefficient C vanishes as $C \approx \frac{(\log N)^2}{N} \to 0$

The Barabási-Albert algorithm is inspired on the growth of the World Wide Web. Scale-free networks, are found in many different areas such as biology, technology, and social networks. Examples include metabolic and protein interaction networks, again the Internet, airline routes, and collaboration networks.

The procedure of building a network according to the BA model involves the following steps:

1. **Initial Network**: Start with a small number m_0 of nodes.
2. **Addition of New Nodes**: At each time step, add a new node with m ($\leq m_0$) edges that link the new node to m different nodes already present in the network.
3. **Preferential Attachment**: The probability that a new node will be connected to node i is $P(i) = \frac{k_i}{\sum_j k_j}$, such that nodes with higher degrees have a higher probability of being selected.
4. **Network Growth**: Repeat step 2 until the network reaches the desired size.

In a Barabasi-Albert network, for $N \to \infty$, the degree distribution is such that the second moment diverges and the first moment is finite

The divergence of the second moment (indicative of the variance or the spread of the degree distribution) reflects the high degree variability in the network, with a significant difference between the most connected nodes and the average. In contrast, the first moment (the average degree) remains finite, signifying that despite the presence of highly connected hubs, the average connectivity of the network does not become infinitely large. This behavior is a hallmark of scale-free networks and is crucial for understanding the robustness and vulnerability of such networks in various contexts.

In the context of a network's degree distribution:

1. **The First Moment** is the average degree of the network. Mathematically, if k_i represents the degree of node i and N is the total number of nodes, the first moment (average degree $\langle k \rangle$) is given by:

$$\langle k \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i$$

2. **The Second Moment** is the average of the squared degrees. It provides an insight into the variance or spread of the degree distribution across the network. It is calculated as:

$$\langle k^2 \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i^2$$

In networks like the Barabási-Albert model, where the degree distribution follows a power law, the second moment tends to diverge as the network grows. This divergence is due to the presence of a few nodes with extremely high degrees (hubs), which significantly influence the average of the squared degrees.

2.2.1 Barabasi-Albert algorithms variants

The Dorogovtsev-Mendes-Samukhin (**DMS**) model introduces an additional parameter δ, modifying the **attachment probability**:

$$P(i) = \frac{k_i + \delta}{\sum_j (k_j + \delta)}$$

allowing for a tunable initial attractiveness of new nodes. This variation leads to a degree distribution

$$P(k) \sim k^{-\gamma(\delta)}$$

where γ now depends on δ.

The Holme-Kim (**HK**) incorporates triangular linking (local clustering) into the BA model, with a probability p of adding an additional edge to one of the neighbours of the new node's connection, thus increasing the network's clustering coefficient. This adaptation reflects more realistic social networks where nodes tend to form tightly knit groups or communities.

2.3 Watts-Strogatz algorithm for Small-World networks

Small World property? The average distance grows "slowly" with N:

$$d \cong \frac{logN}{log < k >}$$

As the network size increases, the average distance between nodes remains relatively small, even for large networks.

The Watts-Strogatz model is a fundamental model in network theory, particularly known for introducing the concept of 'small-world' networks.

This model bridges the gap between regular lattices and random graphs, capturing both local clustering and short average path lengths.

1. It starts with a regular lattice (ring) where each node is connected to k nearest neighbours
2. Then rewires each edge with a probability p. The rewiring process introduces randomness and decreases the network's characteristic path length.

This model demonstrates how a small amount of randomness p in a regular network can significantly reduce the path length, creating the 'small-world' phenomenon, while maintaining a high clustering coefficient, which is characteristic of many real-world networks, including social networks and neural networks.

2.4 Stochastic block-model

The Stochastic Block Model (SBM) is a "block" generalization of Erdös-Rényi networks and it's completely defined by:

- the number of nodes N
- the number of groups (blocks) B
- a partition of the nodes: the group membership b_i of each node i
- the probabilities $p_{rs} = p_{sr}$ that a node in group r is linked to a node in group s (including $r = s$)

We can then (based on probabilities on connecting nodes between different groups (or the same group) define a matrix adjacency.

It is a general, versatile model for large-scale networks, suitable to parameter identification via statistical inference techniques.

3 Mesoscale Network Analysis

Meso means that we are analyzing properties that are not in micro or macro scale but something in the middle. There are structure in the middle zone?

Mesoscale network analysis encompasses both community detection and core-periphery analysis. Here are the key points for each:

3.1 Community analysis

Community analysis, also known as **graph clustering**, involves identifying groups of nodes within a network that have strong connections within the group but weak connections to nodes outside of the group.

Fun fact: **Echo chambers** refer to situations where people are exposed only to opinions and information (through media, like they are trapped in a community) that align with their own beliefs and perspectives, thus reinforcing their existing views.

3.1.1 Modularity optimization

Modularity is a measure of the degree to which the links within a community are denser than what would be expected by chance. It quantifies the strength of connections within a community, surpassing random

link placement based on node degree alone.

$$Q = (\text{fraction of links internal to communities}) - (\text{expected fraction of such links})$$

$$Q = \frac{1}{2L} \sum_{C_h} \sum_{i,j \in C_h} \left[a_{ij} - \frac{k_i k_j}{2L} \right]$$

Finding an exact solution for optimizing modularity is very difficult to achieve in practice. However, there are several suboptimal algorithms that can be used instead. The most commonly used and efficient algorithm is the Louvain method. This method involves moving nodes to neighboring communities in order to increase modularity. After this step, a meta-network of these communities is built.

Modularity in network theory refers to the degree to which a network can be divided into distinct communities or modules. The concept is defined and interpreted as follows:

**Interpretation:

- The modularity Q measures the fraction of links within communities minus the expected fraction of such links in a null model (a random network with the same degree sequence).
- A high value of Q (close to 1) indicates strong community structure, meaning that there are dense connections within communities and sparse connections between them.
- The goal of modularity optimization is to find a partition of the network that maximizes the value of Q, revealing the underlying community structure.

Modularity is a crucial concept in understanding the structure and function of complex networks, as it helps identify subnetworks or groups within a larger network that are more densely interconnected compared to the rest of the network.

3.1.2 Communities by random walkers

The idea: random walkers tend to remain within a community due to denser internal connections. Communities should have large persistence probabilities, indicating longer times spent by random walkers within them.

Infomap, based on information theoretic coding of random paths, is a notable implementation.

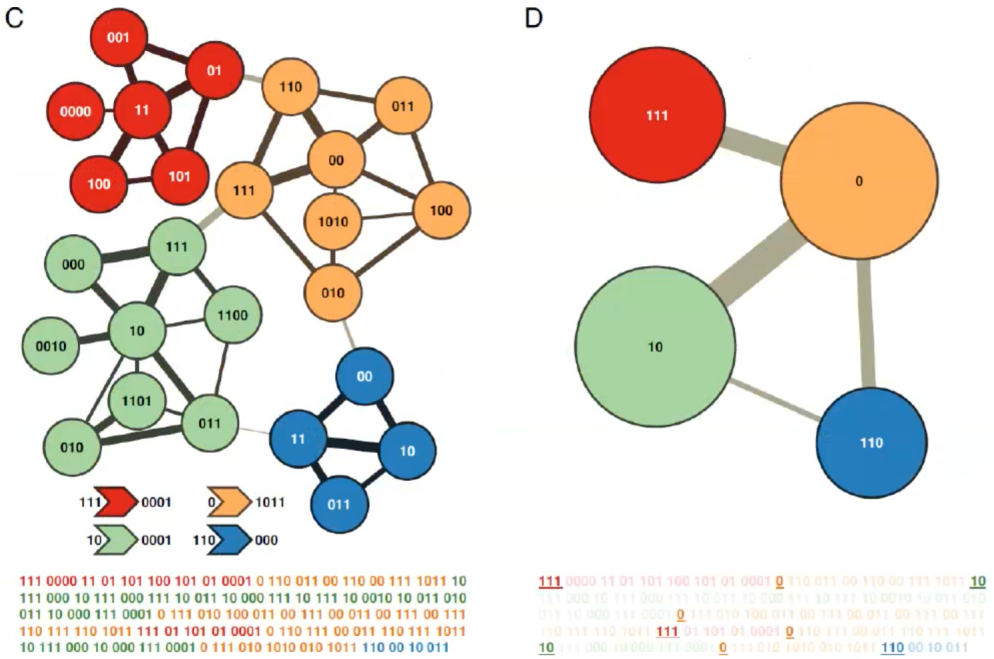
3.1.2.1 Infomap

The Infomap algorithm is a network analysis method used for detecting communities in networks. It leverages the concept of information flow on the network to reveal community structure. Here's an explanation of how it works:

1. **Random Walks as a Proxy for Information Flow**: Infomap uses random walks to mimic the flow of information in a network. The basic idea is that a random walker (an agent moving from node to node) tends to get trapped for longer periods in communities.

2. **Minimizing the Description Length of Random Walks**: The algorithm aims to find a community division that minimizes the description length of these random walks. Description length here refers to the amount of information required to describe the path of a random walker. It's a measure derived from **information theory**.
3. **Two-Level Description**: Infomap uses a two-level description of the random walk:
    - At the first level, each community is assigned a unique code.
    - At the second level, individual nodes within a community are given unique codes.
4. **Encoding the Random Walks**: During a random walk, when the walker stays within the same community, only the second-level code (node code) is used. When the walker moves to a different community, both the first-level (community code) and the second-level codes are used. This dual coding scheme tends to be more efficient when the walker spends long periods within the same community, highlighting the community structure.
5. **Optimization Process**: Infomap **iteratively** adjusts the community assignments to minimize the overall description length of the random walks. The more efficient the description (i.e., the shorter the description length), the better the community structure the algorithm has uncovered.
6. **Result**: The output of the algorithm is a partition of the network into communities where the information flow (as approximated by random walks) is efficiently described, indicating strong internal community structure and weaker between-community interactions.

Infomap is particularly effective for networks where the flow of information is a relevant characteristic, and it's widely used due to its efficiency and the quality of the communities it detects.



3.1.3 Lumped Markov Chain

The concept of "lumping" in Markov chains refers to simplifying a chain by aggregating certain states into a single state under specific conditions.

The dynamics of the random walker on an aggregate level, or "meta-network" at stationarity ($\pi_0 = \pi$), are described by the lumped Markov matrix U. Given this:

- The Partition Matrix H is used to determine which original state belongs to which aggregated state. It helps map each node to its respective community.
- The Original Transition Matrix P shows the probabilities of transitioning from one state to another in a single time step in the original Markov chain.
- The Stationary Distribution π represents the long-term probabilities of being in each original state, indicating the steady-state probabilities.

The lumped Markov matrix is mathematically defined as:

$$U = [\mathrm{diag}(\pi H)]^{-1} H' \,\mathrm{diag}(\pi) P H$$

Inside U, the u_{cd} is the probability that the random walker is in any of the nodes of $\mathbb{C}_d$ at time $t+1$, given it was in any of the nodes of $\mathbb{C}_c$ at time t.

The diagonal terms u_{cc} of the lumped Markov matrix U are known as **persistence probabilities**. They represent the fraction of time a random walker spends within the links/nodes of a community $\mathbb{C}_c$. For undirected networks, u_{cc} measures the fraction of total internal strength to total strength within a community:

$$u_{cc} = \frac{\sum_{i \in \mathbb{C}_c} s_i^{\text{int}}}{\sum_{i \in \mathbb{C}_c} s_i}$$

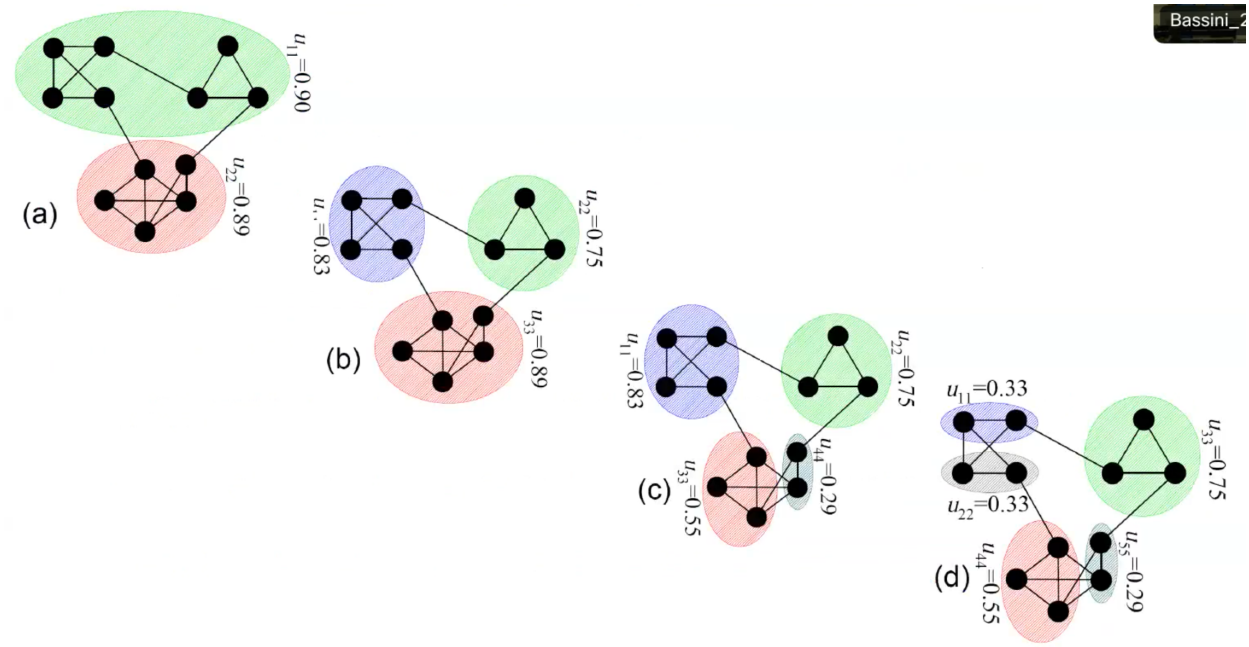
$$u_{cc} = \frac{\sum_{i \in \mathbb{C}_c} s_i^{\text{int}}}{\sum_{i \in \mathbb{C}_c} s_i}$$

or for undirected and and unweighted:

$$u_{cc} = \frac{\text{total internal degree}}{\text{total degree}}$$

A high u_{cc} value, approaching 1, indicates that the community is more of a "trap" for the random walker, consistently representing a significant community $\mathbb{C}$.

The effectiveness of this approach relies on the assumption that the random walker is in a network with a **stationary distribution**. Stationarity implies that the walker's long-term behavior is predictable and stable.



3.1.3.1 α-Communities and α-Partitions The concepts of α-communities and α-partitions provide a framework for analyzing communities in networks based on a set quality level, denoted as $0 < \alpha < 1$:

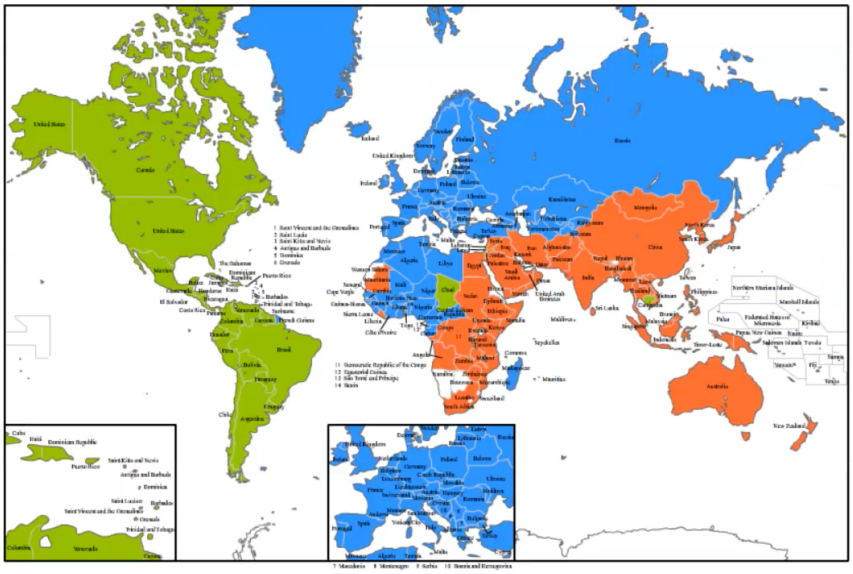
- A set $\mathbb{C}_c$ is considered an α-community if the persistence probability $u_{cc} \geq \alpha$.
- A partition $\mathbb{P}_q = \{\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_q\}$ is an α-partition if all its constituent sets $\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_q$ are α-communities, i.e., $\min_c u_{cc} \geq \alpha$.

The general algorithm is:

1. Set the quality level α.
2. Generate a set of "good" candidate partitions with varying numbers q of clusters.
3. Select the α-partition that has the largest q, indicating the finest decomposition that meets the desired quality level.

Example: the European Union emerges as a strong α-community, particularly in terms of internal commerce. The geopolitical influence of China in Africa is also evident, demonstrating the utility of community analysis in complex global economic dynamics.

$\mathbb{C}_1$ ($u_{11} = 0.77$): Europe (incl. ex-USSR countries) + half of Africa
$\mathbb{C}_2$ ($u_{22} = 0.59$): Asia + Australia + half of Africa
$\mathbb{C}_3$ ($u_{33} = 0.55$): Americas



3.2 Core-periphery analysis

The **Core-Periphery Paradigm** conceptualizes networks as a dense core with a sparsely connected periphery.

3.3 Block-Modelling

Block-modelling is a type of structure where there are central nodes that are connected to a few peripheral nodes, but the peripheral nodes are not connected to each other.

other periphery nodes..."

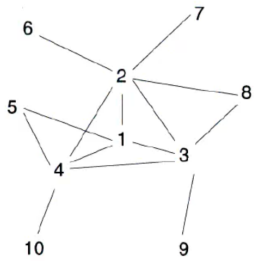


Fig. 1. A network with a core/periphery structure.

But this is a “too crude separation”.

3.4 K-shell Decomposition

k-Core decomposition is a technique used to organize a network into layers or shells based on node connectivity.

- **k-Core**: The k-core of a network is defined as the maximal subgraph S where each node has an internal degree $deg_s \geq k$. This means each node in the k-core has at least k connections within the subgraph.
- **k-Shell**: The k-shell refers to the set of nodes that belong to the k-core but not to the $(k+1)$-core. This distinction allows the network to be organized into "concentric" layers, with each layer representing a different level of connectivity.

The union of all k-shells with $k' \geq k$ forms the k-core of the network. This hierarchical decomposition helps in identifying and analysing the structure and influence of nodes within the network.

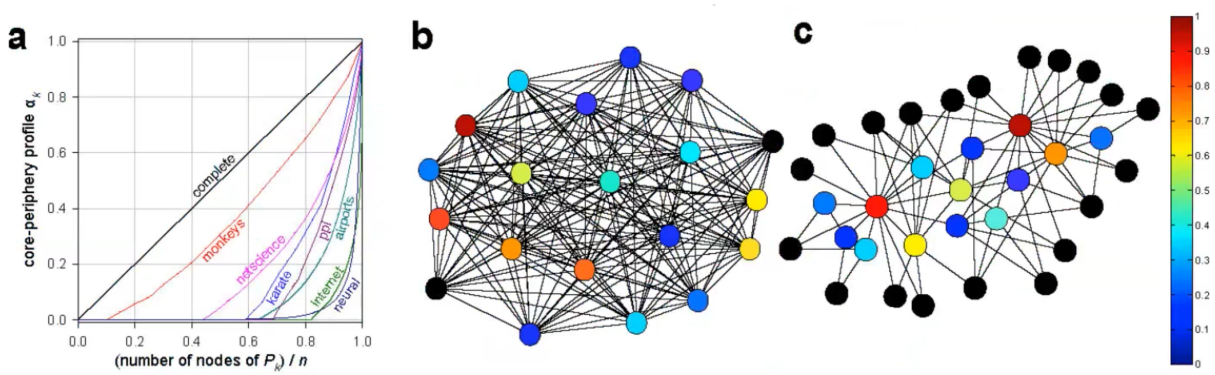
3.5 Core-Periphery Profile in Network Analysis

The Core-Periphery Profile is a heuristic procedure used to order nodes in a network from the periphery to the core. This ordering is based on the nodes' strength and their **persistence probabilities**.

The process of ordering nodes from the most peripheral to the most central involves:

1. Starting with the node i having the minimal strength.
2. Generating a sequence of sets $\{i\} = S_1 \subset S_2 \subset \cdots \subset S_N = \{1, 2, \ldots, N\}$, by adding, at each step, the node with the minimal persistence probability $\alpha_1, \alpha_2, \ldots, \alpha_N$.

The sequence $0 = \alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_N = 1$ constitutes the Core-Periphery profile, where α_k denotes the coreness of the node added at step k.



The Core-Periphery score, C, is the normalized area (within the range $[0, 1]$) between this profile and that of a complete network. The shape and growth of the Core-Periphery profile curve provide insights into the amount of nodes that are genuinely peripheral within the network.

4 Link prediction

Networks are crucial in modeling interactions observed in experiments or data processing. However, it's essential to consider that many interactions might be overlooked and some observed ones might be artificial: which means that the observed network A^O is a "noisy" observation of the actual network A^{true}.

Reconstruct A^{true} given A^O has multiple applications:

- Finding missing links, which can suggest ad-hoc experiments.
- Deleting spurious links.
- Predicting future links in time-varying networks, such as forecasting future friendships.

Different way to do quantifying structural similarity of nodes, examples:

- **Common Neighbours**: check for each pair of nodes the number of common neighbours and then sort them based on this "score". In an undirected, unweighted network with adjacency matrix a_{ij}, the number of common neighbours of nodes i and j is $(A^2)_{ij}$.
- **Katz index**: a generalization of common neighbours

$$s_{ij} = \beta A_{ij} + \beta^2 (A^2)_{ij} + \beta^3 (A^3)_{ij} + \cdots$$

- **Preferential Attachment Index** to sort the nodes $s_{ij} = k_i k_j$. The nodes with highest $s_i j$ should be connected.
- **Resource allocation** which doesn't consider just the common neighbours but also how many links has this neighbour. The rationale here is that a neighbour with a lot of links *doesn't allocate too much resources to i and j"* so it's not relevant in the similarity score.

$$s_{ij} = \Sigma_{h \in (B_i \cap B_j)} \frac{1}{k_h}$$

4.1 Recommender Systems

A similar formulation of the link prediction problem is to consider a set of users $U = \{u_1, u_2, \ldots, u_m\}$ and a set of objects $O = \{o_1, o_2, \ldots, o_n\}$.

The **Bipartite Network Representation** represent the user-object relationships $B = [b_{ij}]$ where:

$$b_{ij} = 1 \text{ if } u_i \text{ owns } o_j, \quad b_{ij} = 0 \text{ otherwise}$$

We can then define:

- **User Degree** $k(u_i) = \sum_j b_{ij}$ as Number of objects owned by user i
- **Object Degree** $k(o_j) = \sum_i b_{ij}$ as the number of users owning object j.

Assuming users like the objects they own, predict and recommend additional objects they might like.

- **Global Ranking Method**: Recommend objects with the largest degree $k(o_j)$, implying the most commonly owned objects. This approach lacks personalization.
- **Content-Based Filtering**: Define object similarity and recommend to user u_i objects most similar to those they already own. This method offers personal recommendations without utilizing network structure.

In **collaborative filtering**, the main assumption is that similar users like similar objects.

$$\hat{b}_{ij} = \frac{s_{i1}b_{1j} + s_{i2}b_{2j} + \cdots + s_{im}b_{mj}}{s_{i1} + s_{i2} + \cdots + s_{im}} = \frac{\sum_{l=1}^{m} s_{il}b_{lj}}{\sum_{l=1}^{m} s_{il}}$$

The score assigned to the (non-existing) pair u_i and o_j. So basically we are looking to the strength on the projected graph where i and l will be users. We are looking at the strength of the relationship between user i and user l weighted by a factor b_{lj} .

After found b_{ij} (the "score of recommendation" of object j to user i) we can also quantify the similarity between users:

- **topological similarity**: number of objects in common

$$s_{il} = \sum_{j=1}^{n} b_{ij}b_{lj}$$

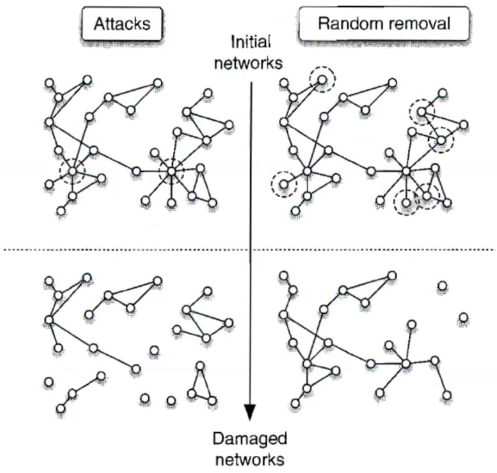
- **cosine-similarity**: each user has a score to object he owns $r_i = (r_{i1}, r_{i2}, \dots)$ and then just measure the difference between vectors:

$$s_{il} = \frac{r_i \cdot r_l}{|r_i||r_l|}$$

5 Robustness

The robustness or resiliency of a network refers to the network's ability to maintain acceptable levels of service despite network faults. Network faults are generally modelled as either:

- **Failures**: Random removal of nodes/links.
- **Attacks**: Targeted removal of crucial nodes/links (most central or most loaded).



We can measure the impact of removing a fraction f of nodes or links by looking the **loss of efficiency** after removal of all links incident to node i:

$$I_i = \frac{\Delta E_i}{E} > 0$$

Different network structures, like the Barabasi-Albert and Erdös-Rényi models, have different responses when nodes or links are removed.

- **Erdös-Rényi Networks Under Stress**: Homogeneous networks are robust.
- **Scale-Free Networks Under Stress**: while removing random nodes doesn't significantly impact connectivity, targeting high-degree nodes rapidly decays connectivity. So they are pretty **fragile** to **attacks**.

5.1 Motter and Lai model of network breakdown

Understanding the propagation of breakdown phenomena can be interesting.

- Nodes exchange units of material along the shortest path.
- At $t = 0$, node load is proportional to betweenness $b_i(0)$.
- Node capacity $C_i = (1 + \alpha)b_i(0)$, with α as the tolerance parameter.

The key part is that:

- Failure/attack leads to changes in betweenness and thus their load
- The subsequent change of loads can create other node fails since $b_i > C_i$
- These other node failures change the betweennesses b_i: the cycle repeats

At the end $G = S/N$ is used as a metric with S as the size of the largest connected component of the network after a failure or attack has occurred.

6 Spreading processes on networks

Networked Dynamical Systems involve dynamical systems hosted by each node, with interactions defined by links.

The collective behaviour of the network is typically more complex than isolated nodes and is influenced by the topological structure of the network.

6.1 Contagion and epidemics

The spread of infectious diseases, as well as the adoption of products and opinions, can be modeled using **probabilistic cellular automata** in networked system characterized by:

- **Finite State Set**: Each node i is in state s^i at time t, belonging to a set Σ (example *Susceptible, Infected, Recovered* in the SIS model we will see).
- **Local Rules**: The state of a node at the next time step depends probabilistically on its current state and the states of its neighbour (the contagion mechanism in the SIS model).

6.1.1 SIS Process in networks

The SIS (Susceptible-Infected-Susceptible) model describes the dynamics of disease spread in networks. The process includes two main transitions:

- **Susceptible to Infected (S → I)**: Occurs with probability $\beta I_i \Delta$, where β is the infection rate, I_i is the number of infected neighbours and Δ the time step.
- **Infected to Susceptible (I → S)**: Occurs with probability $\gamma\Delta$, representing the recovery rate.

In summary:

- β measures how easily the disease spreads (higher β means more contagious disease)
- γ measures how quickly individuals recover from the disease (higher γ means faster recovery).

The dynamics of epidemics are influenced by the network topology.

6.1.1.1 Homogeneous networks Using as first approximation that the degree of each node can be approximated to the average k. The equations of the rate of change of the fraction of infected individuals in the population is:

$$\frac{dy(t)}{dt} = -\gamma y(t) + \beta\langle k\rangle y(t)[1 - y(t)]$$

where γ is the recovery rate, β is the transmission rate, and $\langle k\rangle$ is the average number of neighbours.

In case of **homogeneous networks**, the epidemy outcome depends on a β respect to the threshold β_c :

$$\beta_c = \frac{\gamma}{\langle k\rangle}$$

Below this threshold, the disease dies out, while above it, the disease can persist in the population. If $\beta > \beta_c$, the epidemy reaches a non-trivial (> 0) asymptotically stable equilibrium given by:

$$y = 1 - \frac{\gamma}{\beta\langle k\rangle}$$

6.1.1.2 Heterogeneous networks In **Heterogeneous networks** (like scale-free networks) the epidemic threshold is

$$\beta_c = \frac{\gamma\langle k\rangle}{\langle k^2\rangle}$$

then it may tend to 0 for large networks $(N \to \infty)$! This means that for large networks $y(t)$ never vanishes, whatever the value of the transmission rate β and that the epidemy is able to survive with arbitrarily small transmission rate.

6.1.2 Immunization

With the immunization we are able to eradicating the epidemic $y \to 0$ and also we can define the threshold of fraction of individuals (g_c) we need to vaccinate to eradicate the population.

Homogeneous networks (random vaccination)

β is now $\beta(1-g)$ which makes the epidemic threshold equation $\beta(1-g) < \frac{\gamma}{<k>}$ which is satisfied by:

$$g > g_c = 1 - \frac{\gamma}{\beta < k >}$$

If $g > g_c$ nodes are vaccinated at random, the epidemic is eradicated $(y \to 0)$.

Heterogeneous networks (random vaccination)

In heterogeneous networks, same reasoning but with different threshold:

$$\beta(1-g) < \frac{\gamma < k >}{< k^2 >}$$

which is satisfied by:

$$g > g_c = 1 - \frac{\gamma < k >}{\beta < k^2 >}$$

If the network is too large, $< k^2 > \to \infty$ (scale-free nets with large N) then $g_c \to 1$ which means that the entire population needs to be vaccinated.

6.1.3 Models of influence propagation (“social contagion”)

These models describe the diffusion of information or behaviour in a network-structured population. This can include:

- Information (news, rumor, opinion, etc.)
- Behavior (adoption of a product/service, food/smoking habits, etc.)

The simple models are based on two states $\Sigma = \{\text{Inactive, Active}\}$. The **progressive** nature of the contagion makes that once an individual turns to active, it remains such forever.

Two problems can be formulated from this:

1. Analysis: Determine the final active set S_∞ based on a given initial set S_0
2. Optimization: Find the best initial active set S_0 that maximizes the size of final set size $|S_\infty|$ while staying within a specified budget $|S_0| = m \ll N$

We saw two simple models:

- **Linear Threshold** (LT) Model:
  - Each node i has a threshold $\theta_i > 0$ and is influenced by the incoming neighbors $N_i = \{j | a_{ji} = 1\}$ with a weight $w_{ji} > 0$.
  - If A_t is the set of Active nodes at time t, then node i becomes active at time $t + 1$ if $\sum_{j \in N_i \cap A_t} w_{ji} \geq \theta_i$
  - The model is deterministic.
- **Independent Cascade** (IC) Model
  - Each node i becoming active at time t is given a single chance to activate each of its currently inactive outgoing neighbors $N_i = \{j | a_{ij} = 1\}$.
  - The activation attempt succeeds with probability p_{ij}, and j will become Active at time $t + 1$.
  - The model is stochastic.

7 Consensus in Networked Multi-Agent Systems

The concept of "consensus" refers to the process where all agents in the network eventually reach the same state over time.

Both continuous and discrete models to specify how each single agent's state changes over time can exists.

The simplest local dynamic for an agent is represented by a network of N **integrator** agents, each hosting a linear system $\dot{x}_i(t) = u_i(t)$ (integrator because actually they continue to compute the integral of the input).

The interaction between elements in a system is proportional to the difference in their states due to its "diffusive" nature:

$$\dot{x}_i(t) = u_i(t) = \sum_{j \in V_i} (x_j(t) - x_i(t))$$

The **Laplacian** matrix plays a crucial role in consensus dynamics because its properties determine how quickly and effectively a consensus can be reached in the network.

$L = diag(k_1, k_2, ..., k_N) - A$ represents the $N \times N$ Laplacian matrix of the network, influencing the collective dynamics of the system.

The eigenvalues and eigenvectors of the Laplacian, in particular, provide insights into the network's connectivity and the speed of convergence to consensus.

In principle, infinite consensus states but only one is possible which **depends** on the **graph topology**.

7.0.1 Undirected networks

Conservation of the Sum of States: The equation $\Sigma_i \dot{x}_i = 0$ suggests that the sum of the rate of change of states over all nodes in the network is zero. This implies conservation in the system – the total "quantity" represented by the states x_i of the nodes doesn't increase or decrease over time; it's merely redistributed among the nodes.

The only feasible equilibrium state $\overline{x}$ for all nodes, where equilibrium means no node's state is changing anymore (consensus is reached), is the average of all the initial states:

$$\bar{x}_1 = \bar{x}_2 = \cdots = \bar{x}_N = \frac{1}{N} \sum_i x_i(0)$$

7.0.2 Directed networks

In **directed networks**, consensus is still achievable, but it depends on the network's connectivity. In directed networks the Laplacian matrix is built using k_i^{out} :

$$L_{ii} = k_i^{out}, L_{ij} = -a_{ij}, i \neq j$$

We found out that only if the **network is strongly connected**, there is a consensus value which is **weighted average** network. In a directed network, a consensus value is a weighted average of the initial states. This is mathematically represented as:

$$\alpha = \sum_i w_i x_i(0)$$

where $\sum_i w_i = 1$.Here, w_i are weights that depend on the network topology, and α is the weighted average.

These weights $[w_1, w_2, ..., w_N]$ are actually the components of a left eigenvector of the Laplacian matrix L corresponding to the eigenvalue $\lambda_1 = 0$.

Iff each node's in-degree k_i^{in} is equal to its out-degree k_i^{out}, α becomes the average of the initial states, expressed as:

$$\alpha = \frac{1}{N}\sum_i x_i(0)$$

Finally, the speed at which a network reaches consensus is dominated by the first sub-dominant eigenvalue of $L_S = (L + L^T)/2$,where L^{T} is the transpose of the Laplacian matrix. This eigenvalue gives an indication of how quickly the consensus state will be achieved in the network.

8 Synchronization

Synchronization is the adjustment of the rhythms of two (or more) **oscillators** due to their weak interaction.

Synchronization can happen with a very small ϵ (coupling strength) but it requires a smaller difference in frequencies.

The phase $\Phi(t)$ of an oscillator with period T is defined as:

$$\Phi(t) = \frac{2\pi t}{T} + \mathrm{const} = \omega t + \mathrm{const}$$

The oscillator is synchronized with the **forcing input** (phase synchronization, or phase locking) when

$$\Phi(t) - \Phi_u(t) = \mathrm{const}$$

Two periodic oscillators are synchronized when their frequencies are equal ($\omega_1 = \omega_2$). Same $\omega_1 = \omega_2$ means also they are in phase: $\Phi_1(t) - \Phi_2(t) = \mathrm{const}$.

We can also define **synchronization of order** $n : m$ when $n\omega_1 = m\omega_2$: which means that the oscillators counts n periods for each m periods of the forcing input:

$$n\Phi_u(t) - m\Phi(t) = \text{const}$$

8.1 Chaotic oscillators

A chaotic oscillator's path is non-periodic: it never passes through the same point twice. If it did, based on the differential equation of the Cauchy problem, it would imply that the system is periodic, which contradicts the chaotic nature of the oscillator.

First problem is to just represent the curve with a function. Three methods:

First method: We do a parameterization in a monotonic way of the trajectory:

$$\omega = \lim_{t\to\infty} \frac{\Phi(t)}{t}$$

A second method: when the attractor has a suitable geometric structure we can define a vector centered in 0, formally called “the projector of the attractor”.

$$\Phi(t) = \arctan \frac{y(t) - \tilde{y}}{x(t) - \tilde{x}}$$

where $x(t)$ and $y(t)$ are the coordinates of the oscillator and $\tilde{x}$, $\tilde{y}$ are the coordinates of a reference point. But often is not possible to define a vector (module and direction) $\Phi(t)$ which monotonically increase . A third method: to find the parametrization which defines me a phase is to define the **crossings of a suitably Poincaré section**.

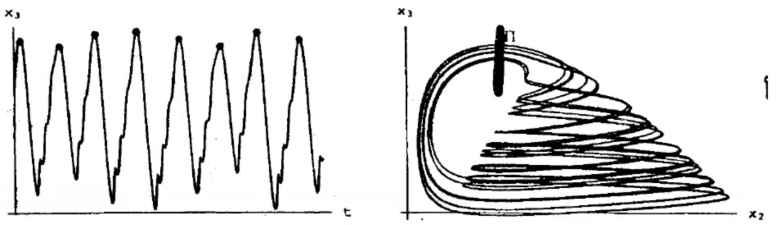
$$\Phi(t) = 2\pi \frac{t - t_k}{t_{k+1} - t_k} + 2\pi k, \quad t_k \leq t < t_{k+1}$$

$\Phi(t)$ is defined to (linearly) increase of 2π between two consecutive crossings of a suitably defined Poincaré section Π . Basically we define a section and every time the oscillator passes through that section we increase by 2π the phases.

This last method is useful when we may not fully understand a system, but by measuring and analysing a time series local maximums, we can comprehend the phase variable.

This can be beneficial in various applications. By detecting this variable’s maximum peaks, we are essentially observing its intersection with a cross section at its peak, a powerful tool. Even without deep knowledge of the system or its components, this allows us to track the system’s movement and define a monotonically increasing phase.

Example: prey-predator system (Rosenzweig-MacArthur)



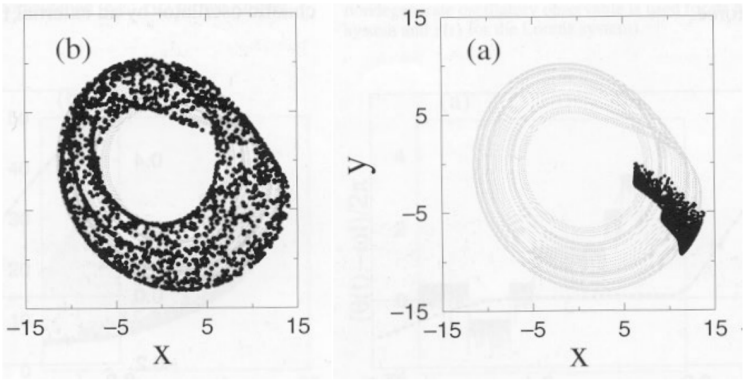
8.1.1 Synchronization of a chaotic oscillators

8.1.1.1 By a periodic forcing input The synchronization of a chaotic oscillator by a periodic forcing input. The two are considered synchronized when their phase difference is **limited**, unlike the scenario with two periodic oscillators where we wanted that their difference is constant.

$$|\Phi(t) - \Phi_u(t)| < \text{const}$$

if and only if $\Omega = \omega$, meaning the synchronization is achieved only when the chaotic oscillator’s average frequency equals the periodic oscillator’s frequency.

Stroboscopic diagrams can be used to discuss the states of synchronization, with dots representing the state value at time intervals that are multiples of $2\pi/\omega$. In an unsynchronized state, dots are scattered throughout the attractor, indicating a distributed phase between $-\pi$ and π. However, during synchronization, the dots converge within a narrow phase interval.



8.1.1.2 Between two chaotic oscillators

The two chaotic oscillators are synchronized (in phase and frequency) when

$$\left|\Phi_1(t) - \Phi_2(t)\right| < \text{const} \qquad \text{that is} \quad \omega_1 = \omega_2$$

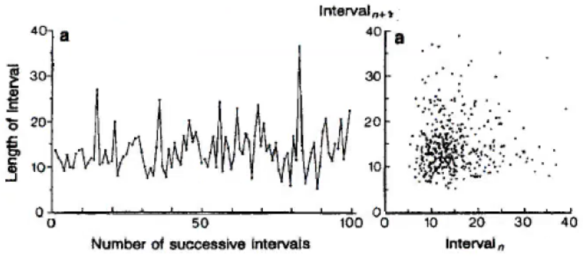
But remember that ω_i are not numbers but are defined like:

$$\omega_i = \lim_{t\to\infty} \frac{\Phi_i(t)}{t}$$

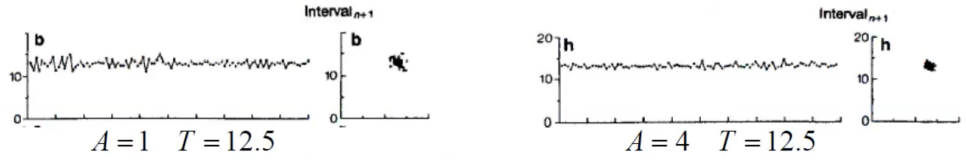
8.1.2 Halobacterium salinarium example

Example: imperfect synchronization in the locomotion of Halobacterium Salinarium. It is a **ciliated bacterium** moving in a fluid, which changes direction every few seconds.

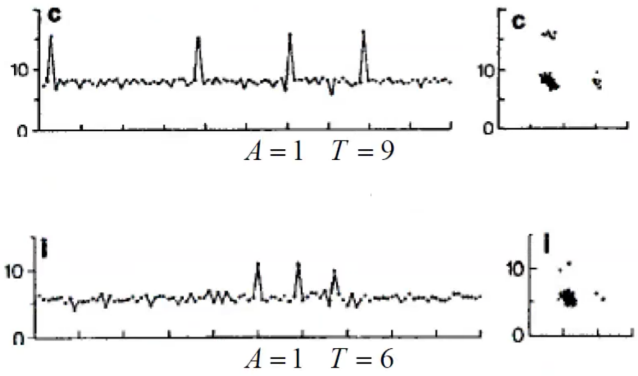
> Prof: "… and here the level of perversion of the experimenter rises, which is already (in my opinion) quite high. Let's shoot him periodically flashes of light."



Under periodic light stimuli (flash sequences) of intensity/amplitude A, the commuting intervals tend to synchronize with the period T of the stimuli.



An interesting finding was that synchronization led to an adaptation to the forcing **rhythm**, reducing both longer and shorter intervals, causing the mean commuting interval to gravitate towards the stimuli period. However, for some (A, T), some flashes **fail** to induce commutation:



For $T = 9$ and $T = 6$ we can clearly see **imperfect synchronization**: long intervals of "apparent" synchronization (=the bacterium commutes with the frequency of the input) are interrupted by sudden "phase jumps" (=the bacterium commutes only once for two flashes).

8.2 Complete synchronization

Two systems Σ_1 and Σ_2 are completely synchronized if are:

$$\lim_{t \to \infty} |x'(t) - x''(t)| = 0$$

- The definition implies that the synchronized state $x'(t) = x''(t)$ must be asymptotically stable at least locally.
- Contrasting with phase synchronization, which only requires the same average frequency (with uncorrelated amplitudes), complete synchronization necessitates perfect coincidence in both frequency and amplitude of the behaviours of the two systems.
- For achieving complete synchronization, the interaction between the systems may need to be strong, as opposed to weak interactions.
- Interestingly, complete synchronization maintains the chaotic nature of the individual systems.

More generally, the systems need not be identical but merely similar (e.g., having the same state equations but slightly different parameters). In such cases, the criterion for synchronization is relaxed to $|x'(t) - x''(t)| <$ constant, allowing for some degree of difference in their evolution.

8.3 Synchronization of networked oscillators

The **Kuramoto Model**, created by Yoshiki Kuramoto in 1984, is a key framework for studying synchronization with interconnected oscillators. The Kuramoto Model is particularly notable for its ability to describe the shift from disorder to order in a system of oscillators that are linked together, providing important information about the critical points where this transition occurs.

There is nonlinear coupling between the rotors, denoted by $V(i)$ which represents the neighbours of rotor i. The equation describing the coupling is given by:

$$\frac{\mathrm{d}\phi_i}{\mathrm{d}t} = \omega_i + K \sum_{j \in V(i)} \sin(\phi_j - \phi_i)$$

To quantify synchronization, we introduce the "order parameter" $r(t)$ which represents the centre of mass of the oscillators.

$$r(t)\mathrm{e}^{\mathrm{i}\psi(t)} = \frac{1}{N}\sum_{j=1}^{N} \mathrm{e}^{\mathrm{i}\phi_j(t)}$$

The behaviour of $r(t)$ determines the degree of synchronization:

- If $r(t) \to 0$, there is no synchronization.
- If $r(t) \to 1$, all rotors are synchronized, meaning they have identical $\frac{\mathrm{d}\phi_i}{\mathrm{d}t}$ values.
- If $r(t)$ tends to a nonzero value, a fraction of the oscillators is synchronized.

We have N oscillators on a complete network with coupling strength $K = \frac{K^0}{N}$. The dynamics of each oscillator is described by the equation $\frac{\mathrm{d}\phi_i}{\mathrm{d}t} = \omega_i + \frac{K^0}{N}\sum_j \sin(\phi_j - \phi_i)$.

By combining this with the equation $r(t)\mathrm{e}^{\mathrm{i}\psi(t)} = \frac{1}{N}\sum_{j=1}^{N} \mathrm{e}^{\mathrm{i}\phi_j(t)}$, we can derive:

$$\frac{\mathrm{d}\phi_i}{\mathrm{d}t} = \omega_i + K^0 r \sin(\psi - \phi_i)$$

Note that the coupling term depends on the "mean phase" ψ. The coupling strength, $K^0 r$ increases with r. This is called positive feedback, where the more synchronized oscillators there are, the greater the chance of capturing the remaining ones.

At the end the results are:

- $K^0 < K_c$: no synchronization $(r(t) \to 0)$
- $K^0 > K_c$: synchronization of larger and larger fractions of oscillators $(r(t) \to 1$ as $K^0 \to \infty)$
- when $r(t) \to r < 1$, the de-synchronized oscillators are those with largest detuning $|\omega_i - \Omega|$.

On complex network we can also give a value to the threshold K_c :

- Regular lattices: $K_c \to \infty$ (no synchronization)

- Small-world networks: K_c decreases with p probability of rewiring (so basically we are moving from a regular lattices to a random network)
- Heterogeneous networks: $K_c = \frac{\langle k \rangle}{\langle k^2 \rangle}$

8.3.1 Generalization of complete synchronization

Let's formulate better the problem:

- we have a system consisting of isolated nodes represented by i (where $i = 1, 2, ..., N$)
- Each node represents an identical, autonomous, n-dimensional dynamical system.
- When each node is isolated, its behaviour is (periodic or chaotic) **oscillatory**.
- The state of each node i is denoted by $x^{(i)}$ in $\mathbb{R}^n$, and its isolated dynamics are governed by the equation:

$$\dot{x} = f(x), \quad x \in \mathbb{R}^n$$

When the nodes are coupled, their dynamics change. The coupling between the nodes is linear and diffusive, described by the equation:

$$\dot{x}^{(i)} = f(x^{(i)}) + \sum_{j:a_{ij}=1} d\Big(H(x^{(j)} - x^{(i)})\Big)$$

where:

- $d \geq 0$ represents the coupling strength between the nodes.
- H is a nonnegative $n \times n$ matrix called the **diffusion profile**, which determines which variables interact among the nodes. A diagonal H matrix indicates the specific components or species that disperse and their relative dispersal rates. H represents different dispersion rates in the diffusion for each component/variable.

At synchronization all the interaction terms $d\Big(H(x^{(j)} - x^{(i)})\Big)$ vanish and $\bar{x}(t)$ must be a solution of the isolated system $\dot{x} = f(x)$.