Consensus with pinning control in complex network and application to Data Clustering

Benjamin BENTEKE LONGAU (benjamin.benteke@aims.senegal.com)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Professor Franck KALALA MUTOMBO
AIMS-Senegal and University of Lubumbashi, DRC

July 29, 2021

Master Thesis

Abstract

In this project, network based approach for data clustering is investigated. A network here is viewed as graph with nodes and links between them. Data clustering is technique group data, such that points in a same cluster are similar to each other, and less similar to points in other clusters. The main challenge is the construction of a network base on top of the dataset, and each will be represented by a node. One needs to obtain a connected and sparse network. To achieve this the single linkage technique and $k-$nearest neighbors are combined. Once the network is constructed, there exist many techniques to perform network-based clustering. We use here a consensus technique with pinning control. We consider a consensus process with pinning control on the constructed network based on consensus time, a dissimilarity matrix is built which is use for clustering, by using hierarchical clustering method. We perform some experiment on simulated and real dataset.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

Benjamin BENTEKE LONGAU, 7th January 2021

Acknowledgements

First of all, I am grateful to the Almighty God for the gift of life and for seeing me through up to this far.

I want to thank AIMS and it's funders for supporting this work, as well as my supervisor, Professor **Franck KALALA MUTOMBO** academic director at AIMS-Senegal and associated Professor at University of Lubumbashi (DRC).

I thank all tutors for their Supports both scientific and moral.

I thank my very dear mother, **Marie-Therese YOKA EALE**, who have always been there for me. I thank my sister **Vanessa LILANGA**, and my brothers **Jackson LOKOFO** and **Michaël YOKA**, for their encouragement.

Finally, I thank my classmates Arame Thiam, Faty Niane, Olive Etoga, Idriss Tondji, Idriss Nguepi, Yvan Tamdjo, Junior Kaningini, Arnaud Watusadisi, Cyrille Kesiku, Mikhaël Kibinda, Gospel Chinonso, Saliou Fall, and so many others that I could not mention who have always been there for me. Their unconditional support and encouragement has been a great help.

Contents

List of Tables

List of Figures

Acronyms

SWN	Small-world network
SFN	Scale-free network
DBSCAN	Density-based spatial clustering of applications with noise
LCC	Local clustering coefficient
GCC	Global clustering coefficient
BFS	Breadth-First-Search
DC	Degree centrality
CC	Closeness centrality
BC	Betweenness Centrality
SC	Subgraph Centrality
EE	Estrada index
BA	Barabasi-Albert
ER	Erdos-Renyi
WS	Watts-Strogatz
K-NN	K-nearest neighbors
SL	Single linkage
WL	Ward linkage
CL	Complete linkage
AL	Average linkage
LPA	Label Propagation Algorithm
RI	Rand index
ARI	Adjusted Rand index
HC	Hierarchical clustering

1. Introduction

1.1 Background

Clustering is one of the fundamental topics in unsupervised learning. It has been widely and successfully applied in data mining, pattern recognition, and many other fields (Gama, 2010; Bishop, 2006). Its goal is to partition a set of data items into homogeneous subsets. Applications of clustering include categorization of different customer groups based on purchasing patterns, categorization of documents on the World Wide web, grouping of genes and proteins that have similar functionality (Duda et al., 2012).

Network is viewed as a graphical representation of interconnected objects, each node is presented by a data and edges a link between two data items. The clustering technique that uses network as data representation is called graph clustering (or community detection). Graph clustering is one of the family methods of clustering, which tries to find disjoints partitions of a network such that the connections between nodes in the same partitions are much denser than those across different partitions and these partitions are called clusters or communities or modules. Naturally *evolved data* and Networks representing them, like social networks, form a class of network called Scale-free network (SFN) and Small-world network (SWN) (Lu and Chakraborty, 2019;Lu and Chakraborty, 2019). A *scale-free* network here stands for the distribution of network degree is exponential, with a long tail of nodes with low degree. A *small world* network stands for the distance between two randomly selected nodes is proportional to the logarithm of the number of nodes.

1.2 Problem statement

Today many companies such as insurance, sales, credit and other companies seek to understand their customers' behaviours. On this basis, they collect data from their customers and want to make advanced studies to group their customers such that the most similar ones are in the same group. The most used algorithms are K-means (Lloyd, 1982), Hierarchical clustering and others.These algorithms group individuals according to their similarities where the similarity is measured by a distance and they use vector representation (each data items is a vector) to compute these similarities. Another way to represent these data is to use the network, where a network.

The motivation for using the network approach is that today individuals are becoming much more connected, for example, two individuals are connected because they work in the same company or because they are friends on one of the social networks. These connections can bring influences into the data and we have more challenge to explore these data. In this case, using the network approach will help to understand the data and to do the clustering properly.

Graph clustering methods are usually consist of two main steps which are, the network construction from original dataset and partition of this network into sub-networks such that, each one representing a cluster. The common methods for network construction are: 1) ϵ-neighbors, 2) k-nearest neighbors (Karypis et al., 1999), 3) full connected network. These network construction techniques try to map a dataset into a network structure and at the end of the process it is possible to identify data clusters with different shapes. However, their common drawbacks are : (1) the resulting network is not necessary connected, (2) the resulting network is often denser or complete. In these cases, it's difficult to correctly divide the network into many *sub-networks* (communities). The task of finding the sub-networks in a network such that nodes in the same cluster are densely connected and node in the different clusters are less connected is called *community detection*.

There are several methods for community detection in complex networks for example *girvan-newman algorithm* (Newman, 2006) based on the removal of edges with largest betweenness centrality and at the end of the process the result can be visualised as a divisive hierarchical tree of communities. There are also set of techniques based on the *optimal modularity* (Louvain) and those are based on the *spectral* of the Laplacian associated to the network (spectral clustering). Therefore, our problem is to perform data clustering by using consensus in a network and this method is called consensus clustering.

1.3 Objectives Hypotheses Research Question

Our main objective is to understand how to construct a network from a dataset, the theory behind Consensus on top of this constructed network and how to use it in data clustering. Besides, our specific objectives are :

• to understand the theoretical consensus (with pinning control or not) and establish the major steps to follow in graph clustering process.

• to know how to perform data clustering by using consensus approach.

1.4 Research method

In this work, we present another method to construct a network from a dataset based on the combination of K-nearest neighbors (K-NN) and Single linkage (SL) in which the resulting network is connected, the connections inside a *sub-network* are much denser and connections between *sub-networks* are *sparse*, which could help us to detect communities. After network has been constructed, we consider a dynamical process on top of it, the consensus process with pinning control (Cupertino et al., 2013). In a network, a consensus occurs when all nodes reach the same state (desired target state). Then after a dissimilarity measure between node is obtained based on the consensus time (the amount total that takes a nodes to reach a desired state). The obtained dissimilarity matrix is none symmetric. We transform into a symmetric matrix and we pass it on to the agglomerative hierarchical clustering for node or data clustering.

1.5 Structure of the thesis

This thesis consists of three chapters organized as follows:

• **Chapter 2**

In this chapter, we make a review of graph and network theory as well some network characteristics and models. In the literature we show our to understand a network by visualizing degree distribution of nodes, average path length, local and global clustering coefficient. We go to the common graphs models as small-world, Barabasi-Albert and Watt-Strogatz. These models are most important because they will influence the detection of communities.

• **Chapter 3**

Chapter 3 is where we present some important notions for community detection, its traditional methods, the different properties of Laplacian of a graph and new approach for network construction from a dataset. In the literature, We present the advantages and disadvantages of network construction methods as well as community detection methods.

- **Chapter 4**

Chapter 3 is where we present the consensus clustering method and the dissimilarity matrix from the consensus-time. This method helps us to detect communities on top of network built in chapter 3. In this literature review, we first make an introduction to the consensus clustering by assuming some hypothesis in order to achieve our tasks.

- **Chapter 5**

Chapter 5 is where we present some simulations on the real datasets which show the effectiveness of the new approach.

2. Review on Networks

2.1 Konigsberg bridge problem

In Prussia, there was a city called Konigsberg (now Kaliningrad, Russia) which was set on both sides of the branched Pregel river forming two islands. These islands were connected to each other and the mainland by seven bridges (see the Figure 2.1(a)). The challenge to the citizens was to find a walk around the city that crosses each bridge exactly once (known as the Konigsberg bridge problem). In the attempt to solve the problem, Euler reformulated and represented the Konigsberg bridge problem in a way that is similar to what is referred to as a graph as shown in the Figure 2.1(b). It was then that the story of graph theory begun (Estrada, 2012b).

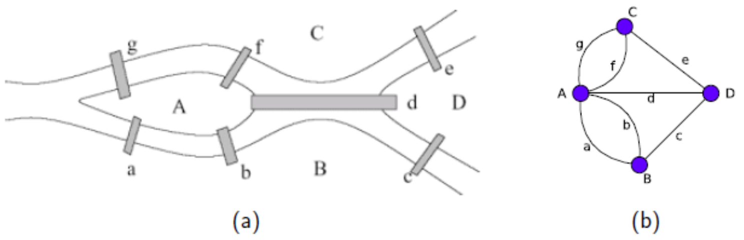


Figure 2.1: The Konigsbergs bridges. (a) a schematic diagram of the seven konigsberg bridges. (b) a graph of the konigsberg bridges. Source: (internet, e).

2.2 Basics of Graph theory

2.2.1 Definition (Undirected Graph)**.** Mathematically, we define a *undirected graph* G is a triple of (V, E, R) with: V a finite set called *vertex set*, E a finite set called *edge set* and R a map called incidence map such that $R : E \to F$ where:

1. $F = P(V) \bigcup \bar{V}$;
2. $P(V) = \{(u, v) \in V \times V | u \neq v\}$;
3. $\bar{V} = \{(u, u) \in V \times V\}$ is the diagonal of the *Cartesian product*.

For example $G = (V, E, R)$: $V = \{A, B, C, D\}$, $E = \{e_1, e_2, e_3\}$ and $R : E \to F$ such that: $R(e_1) = AC$, $R(e_1) = CB$ and $R(e_1) = DA$ So the G is represented by the Figure 2.2

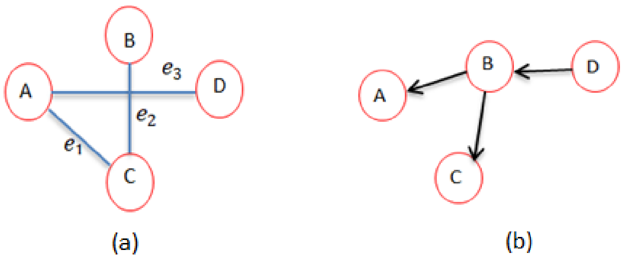


Figure 2.2: (a) an undirected graph with four vertices and three edges, (b) directed graph with four vertices and three edges

2.2.2 Definition (Directed Graph)**.** A *directed* graph also called *digraph*, is a graph having edges with direction (Newman, 2010). See the Figure 2.2(b)

2.2.3 Definition (Network)**.** A network is a diagrammatic representation of a system. It consists of nodes or vertices, which represent the entities of the system. Pairs of nodes are joined by links or edges, which represent a particular kind of interconnection between those entities. According to (Estrada, 2012b), in mathematics the study of networks is known as graph theory, In this work we'll use the term *Network* or *graph*.

2.2.4 Types of a real Network.

- In *social networks*, the nodes are the individuals or people and the links represent the friendship between them. Examples of social networks are facebook, twitter, etc.
- *Citation network:* The nodes of the network may be articles, patents, or legal case while the links are the citations;
- *Food web:* In a food web, the species are the nodes as the links represent the predation;

The figure 2.3 explain the reals network:

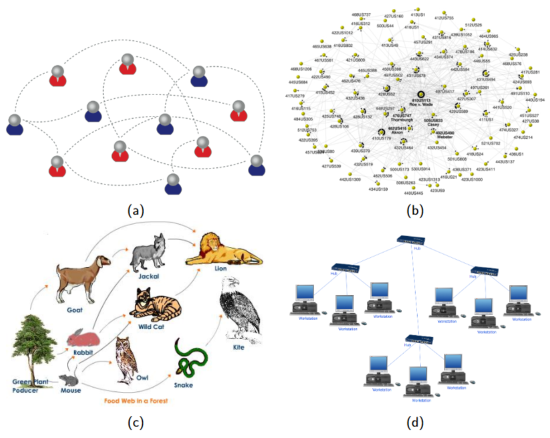


Figure 2.3: Networks in real world: (a) A social network. (b) A citation network. (c) A food web. (d) Computer network. Source: (internet, f)

2.2.5 Definition (Neighbors of a nodes)**.** The neighborhood of a vertex $v \in V$ is a set of all vertices that are adjacent to v. Mathematically, $N_G(v) = \{u \in V | (u,v) \in E\}$ (Newman, 2010).

From the graph of the Figure 2.2 we have the neighbor's set of each node:

$$N_G(A) = \{C, D\},\ N_G(B) = \{C\},\ N_G(C) = \{A, B\} \text{ and } N_G(D) = \{A\}.$$

2.2.6 Definition (Degree/in-degree/out-degree)**.** The *degree* of a vertex i, denoted as k_i , is defined as the number of edges that are adjacent to this vertex i. Or it is the number of neighbors of vertex i $(|N_G(i)|)$. We define the degree of a node i as:

$$k_i = |N_G(i)| \tag{2.2.1}$$

The definition of the degree of a node in a directed graph is separated to two concepts. The *in-degree* of a vertex i , denoted as k_i^{in} , is defined as the number of edges directed towards this vertex i. The out-degree of a vertex i , denoted as k_i^{out} , is defined as the number of edges directed away from this vertex i and of i in this case is $k_i = k_i^{in} + k_i^{out}$ the degree of vertex i (Krishnaraj et al., 2018).

The table 2.1 represent the degree of all nodes of the figure 2.2 (b) and (b) respectively:

Node i	A	B	C	D
Degree	2	1	2	1

(a) *Node's degree of figure 2.2(a)*

Node i	A	B	C	D
In-Degree	1	1	1	0
Out-Degree	0	2	0	1
Degree	1	3	1	1

(b) *Node's degree of figure 2.2(b)*

Table 2.1: Degree of nodes in case of undirected and directed network

2.2.7 Remark (Undirected Graph)**.** If $k_v = 0$, then node v is said to be *isolated* in G, and if $k_v = 1$, then v is a *leaf* of the graph.

2.2.8 Remark (Directed Graph)**.** If $k_v^{in} = 0$, v is called a *source* vertex, $k_v^{out} = 0$ v is called a *sink vertex*, and $k_v^{out} = k_v^{in}$, v is called an *isolated* vertex.

2.2.9 Definition (Minimum and maximum degree of undirected graph G)**.** The *minimum degree* and *the maximum degree* are respectively (Estrada et al., 2011):

$$k_{min}(G) = min\{k_v | v \in G\} \tag{2.2.2}$$

And

$$k_{max}(G) = max\{k_v | v \in G\} \tag{2.2.3}$$

2.2.10 Definition (Minimum and maximum degree of directed graph G)**.** Knowing that: $k_v = k_v^{out} + k_v^{in}$ for $\forall v \in G$, after this computation, we use the formulas 2.2.2 and 2.2.3 for finding $k_{min}(G)$ and $k_{min}(G)$ of a directed Graph.

2.2.11 Lemma (Handshaking Lemma)**.** For any given undirected network $G = (V, E)$, where V is the set of nodes and E the set of edges, the sum of all vertex degrees is equal to twice the number of edges (Newman, 2010):

$$\sum_{v \in V} k_v = 2|E| \tag{2.2.4}$$

Therefore:

$$|E| = \frac{1}{2} \sum_{v \in V} k_v = L \tag{2.2.5}$$

Where L is the total number of links of G. When we have a directed network, L becomes:

$$L = \sum_{v \in V} k_v^{in} = \sum_{v \in V} k_v^{out} \tag{2.2.6}$$

2.2.12 Definition (Average Degree)**.** The average degree is an important property of a network and defined as:

$$< k >= \frac{1}{N} \sum_{i=1}^{N} k_i \tag{2.2.7}$$

For directed network:

$$< k^{in} >= \frac{1}{N}\sum_{i=1}^{N} k_i^{in} =< k^{out} >= \frac{1}{N}\sum_{i=1}^{N} k_i^{out} \tag{2.2.8}$$

2.3 Degree Distribution

The degree distribution of a network is obtained in terms of the probability p_k and is defines as the probability that a node chosen uniformly at random has degree k or equivalently as the fraction of nodes in the graph having degree k (Estrada, 2012b).

For a network with N nodes the degree distribution is the normalized histogram is given by

$$p_k = \frac{N_k}{N} \tag{2.3.1}$$

Where N_k is the number of nodes of degree k such as:

$$\sum_{k=1}^{\infty} p_k \tag{2.3.2}$$

Hence the number of nodes having degree k can be obtained from the degree distribution as:

$$N_k = p_k N \tag{2.3.3}$$

The degree distribution is the most fundamental topological characterisation of a network. It assumed a central role in network theory following the discovery of *scale-free* networks.

One reason is that the calculation of most network properties requires us to know p_k. For example, the average degree of a network can be written as:

$$< k >= \sum_{k=0}^{\infty} kp_k \tag{2.3.4}$$

The Figure 2.4 describe the degree distribution of the graph in the Figure 2.2:

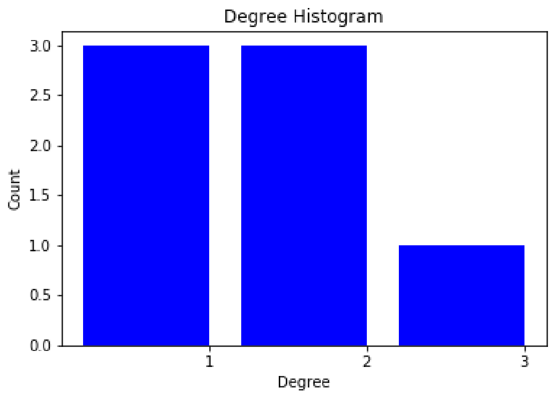


Figure 2.4: Degree distribution of the graph in the figure 2.4

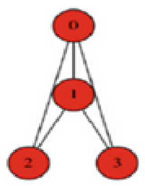
If we interpret the Figure 2.4, we notice that three nodes are degree 1, three nodes also are degree 2 and one node is degree 1. The same style for a directed, you find *in-degree distribution* and *out-degree distribution*.

2.4 Graph Representation

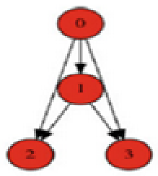
2.4.1 Adjacency Matrix.

The adjacency matrix of a graph, denoted by A, is defined as a matrix where 1 indicates the presence of an edge and 0 indicates the absence of one. More formally, an adjacency matrix is defined as (Krishnaraj et al., 2018):

$$A_{ij} = \begin{cases} 1 & \text{is an edge exists between the vertex i and the vertex j} \\ 0 & \text{otherwise} \end{cases} \tag{2.4.1}$$



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



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

Figure 2.5: (a) An undirected graph with its adjacency matrix and (b) an directed with its adjacency matrix.

2.4.2 Properties (Adjacency Matrix properties).

For any adjacency matrix we have:

(a) $A_{ii} = 0$, (b) $A_{ii} \neq 0$, (c) $A_{ij} = A_{ji}$, (d) $A_{ij} \neq A_{ji}$, (e) $|E| = \sum_{i,j=}^{n} A_{ij}$ and (f) $|E| = \frac{1}{2}\sum_{i,j=}^{n} A_{ij}$.

An *Undirected graphs* satisfy (a), (c), and (f) while *directed graphs* satisfy (a), (d) and (e).

2.4.3 Node's degree computation via Adjacency matrix.

1. **Case of Undirected graph:**

   Let's A be the adjacency matrix of graph G. And let's note $K(V)$ a vector containing the node's degrees of G. We have:

   $$K(V) = AI \tag{2.4.2}$$

   Where I is the identity matrix. For example, let's consider the graph of the figure 2.5(a), we have:

   $$\begin{pmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \\ 2 \\ 2 \end{pmatrix} \tag{2.4.3}$$

   Where k_0, k_1, k_2 and k_3 are respectively the degree of nodes $0, 1, 2$ and 3.

2. **Case of directed graph:**

   Let's A be the adjacency matrix of directed graph G.Let's define these terms:

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

     It means, the sum of the column of node i in the adjacency matrix.

- $k_i^{out} = \sum_{i=1}^{n} A_{ij}$.

  It means the sum of the row of node i in the adjacency matrix.

For example, let's consider the graph of the figure 2.5(b), we have:

- $k_0^{in} = 0+0+0+0 = 0$ and $k_0^{out} = 0+1+1+1 = 3$;
- $k_1^{in} = 1+0+0+0 = 1$ and $k_1^{out} = 0+0+1+1 = 2$;
- $k_2^{in} = 1+1+0+0 = 2$ and $k_2^{out} = 0+0+0+0 = 0$;
- $k_3^{in} = 1+1+0+0 = 2$ and $k_3^{out} = 0+0+0+0 = 0$.

2.4.4 Degree matrix.
The degree matrix is a diagonal matrix that provides information about the degree of each node in a given network $G = (V, E)$ with $n = |V|$, the degree matrix $D(G)$ is defined as (Krishnaraj et al., 2018):

$$D_{ij} = \begin{cases} k_i & \text{if i=j} \\ 0 & \text{otherwise} \end{cases} \tag{2.4.4}$$

The Degree matrix corresponds to the graph of the figure 2.5 is:

$$D(G) = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \tag{2.4.5}$$

2.4.5 Laplacian Matrix.
An other way to represent a graph G is *Laplacian matrix*. A laplacian matrix is defined as (Krishnaraj et al., 2018):

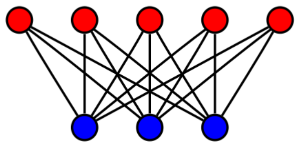
$$L = D - A. \tag{2.4.6}$$

Where D is the diagonal matrix containing the degree of each vertex i i.e $D = diag(k_1, .., k_n)$ and A is an adjacency matrix of G. In other words, the elements of L are noted l_{ij} where:

$$l_{ij} = \begin{cases} -1 & \text{if } j \in N_i \\ k_i & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \tag{2.4.7}$$

2.5 Particular Graphs

2.5.1 Definition (Bipartite Graph)**.** A Network whose nodes can be split into 2 sets L and R and every edge connects a node in L with a node in R. Mathematically A network $G = (V, E)$ is *bipartite* if the nodes can be divided into disjoint sets V_1 and V_2 such that $(u, v) \in E$ implies that $u \in V_i$, $v \in V_j$, $i \neq j$ (Newman, 2010). The figure 2.6a shows an example of *bipartite* graph:


(a) Example of Bipartite graph. Source: (internet, a

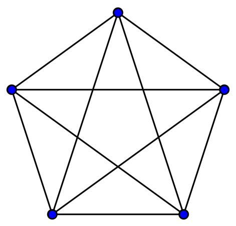

(b) A complete graph of 5 nodes with 10 edges. Source: (internet, b)

Figure 2.6: Example of a Bipartite and complete graph

2.5.2 Definition (Complete Graph (clique))**.** The maximum number of edges that a graph with $|V|$ number of vertices, as denoted by L_{max}, is given by equation 2.5.1:

$$L_{max} = \begin{cases} C^2_{|V|} = \frac{|V|(|V|-1)}{2} \text{ for an undirected graph} \\ 2C^2_{|V|} = |V|(|V|-1) \text{ for a directed graph} \end{cases} \tag{2.5.1}$$

A graph with $|E| = L_{max}$ number of edges is called a *complete graph*. So the average degree of a complete graph is $|V| - 1$. The figure 2.6b shows a *complete graph*.

2.5.3 Definition (Density)**.** Let G be a network, the density of a network is very important when want to know the sparsity of a network:

$$D(G) = \frac{|E|}{L_{max}} \tag{2.5.2}$$

2.5.4 Definition (Regular Graph)**.** This type of networks have are characterized by regular patterns in the degree distribution. Representative examples of them are:

- *All-to-all network:* an edge connects every pair of nodes;
- *Ring-shaped network:* every node is connected with its k closest neighbors;
- *Star network:* all nodes are connected to one hub node.

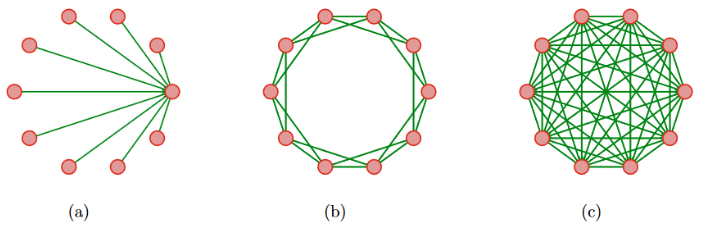


Figure 2.7: (a) Star-shaped network (b) Ring-shaped network (c) All-to-all network (internet, g).

2.5.5 Definition (Weighted Graphs)**.** These are networks in which each $link(i,j)$ has a unique weight w_{ij} . The adjacency matrix of a weighted networks can be defined as follow (Newman, 2003):

$$A_{ij} = \begin{cases} \mathrm{w}_{ij} & \text{is an edge exists between the vertex i and the vertex j} \\ 0 & \text{otherwise} \end{cases}$$

In other words, a *weighted network* is created by replacing the set of links E by the set of link weights $W = \{w_1, ..., w_m\}$.

2.5.6 Definition (Signed Network)**.** Some networks can carry information about friendship and antagonism based on *conflict* or *disagreement*. This figure 2.8 illustrate the signed network:

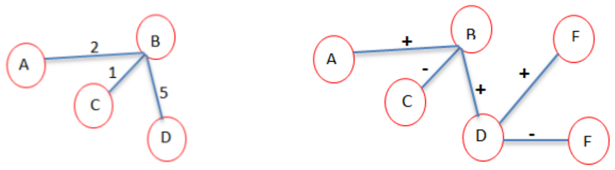


Figure 2.8: To the left an undirected Weighted network and to the right Signed network

In the left of the Figure 2.8, we can also use directed network can be also weighted. And In the right of the Figure 2.8 the edges are assigned positive or negative sign.

For example, in a transportation system for a city, the roads have weights that represent the cost of fuel incurred by using those routes while for a social network, the weights on the connections represent the frequency of communication between the two people (Newman, 2003).

In a network, the edges between nodes can be interpreted as friendship, communication and other. And the nodes also can have some attributes. The figure 2.8 explain this situation.

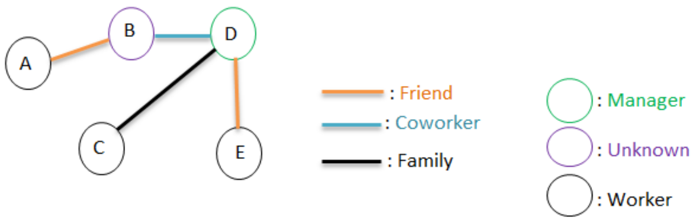


Figure 2.9: Graph with nodes and edges attributes

2.5.7 Definition (Unweighted Graph)**.** Unweighted networks are networks where the edges do not have any associated weights. The figure 2.2 is *unweighted*.

2.5.8 Definition (Self-looped Graph)**.** Self-loops are defined as edges whose source and destination vertices are the same. More formally, an edge $e \in E$ is called a *self-looped* edge if $e = (u, u)$ where $u \in V$. A graph that contains one or more self-loops is called a *self-looped graph*.

2.5.9 Definition (Multi-graph)**.** A *Multi-graph* is a graph where multiple edges may share the same source and destination vertices.

2.5.10 Definition (Simple Graph)**.** A *simple graph* is a graph that has neither loops nor multiple edges. For example the graph of the Figure 2.9.

2.6 Network Connectivity

Network connectivity refers to the ability to move from one node to another in a network. It is the ratio between route distance and geodesic distance. Connectivity can be calculated locally (for a part of the network) and globally (for the entire network). Let's take a look at some of the important metrics of network connection (Al-Taie and Kadry, 2017).

2.6.1 Definition (Triangle)**.** A triangle is simply three nodes that are connected by three edges. A triangle is called also *closed triad*.

2.6.2 Definition (Open triad)**.** Open triad is an open triangle.

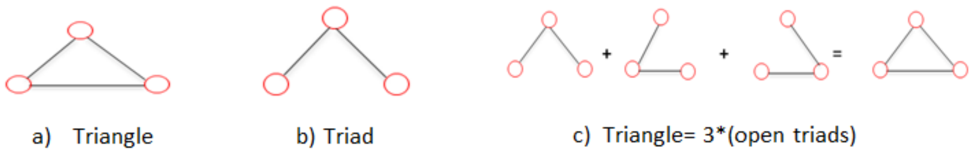


Figure 2.10: Triangle, open triangle and one triangle equal to three open open triads

2.6.3 Clustering coefficient.

Clustering coefficient is a measure of the degree to which nodes tend to cluster together. Such behaviour is more evident in real-world networks, especially social networks where nodes tend to form tightly knit groups that have relatively high density of ties among them (Estrada and Knight, 2015).

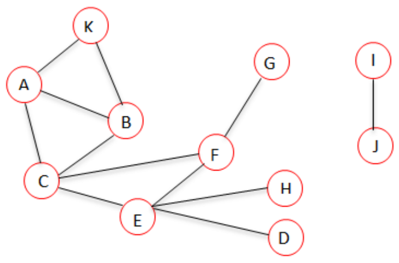
Consider three nodes in a network, say, i, j and k. Suppose i is connected to both j and k (two neighbors of a node will be neighbors themselves), then the likelihood that j and k are also connected is what is known as the *clustering coefficient*. In other words, clustering coefficient measures the density of triangles in a network. The value of clustering coefficient is in the interval $[01]$. There are two types of clustering coefficients namely, the *local* and the *global* clustering coefficients.

2.6.4 Definition (Local clustering Coefficient Local clustering coefficient (LCC))**.** Local clustering coefficient measures the degree to which the nodes in a network tend to cluster together. For a node i with degree k_i , it is defined as :

$$LCC(i) = \frac{\Delta_i}{\frac{k_i(k_i-1)}{2}} = \frac{2\Delta_i}{k_i(k_i-1)} \tag{2.6.1}$$

where Δ_i is the number of triangles in G having node via one of its vertices. In other words, Δ_i is the number of neighbors of node i who are neighbors.

The table 2.12(b) summarize the LCC for each node of the figure 2.12(a)



(a) A network of 11 vertices and 12 edges.

Node i	A	B	C	D	E	F	G	H	I	J	K
LCC(i)	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	0	$\frac{1}{6}$	$\frac{1}{3}$	0	0	0	0	1

(b) Local clustering coefficient of each node

Figure 2.12: Local clustering coefficient computation

We have some remarks:

- Local clustering coefficient belongs to $[0, 1]$;
- If we have a node such as $0 \leq k_i < 2$, then $LCC(i) = 0$ because they don't belong in any triangle (open or closed);
- If a node has a small degree, then LCC of this node will be high;
- The nodes with higher degree have a small LCC;
- LCC measure the probability that any two neighbors of a vertex i are connected.

2.6.5 Definition (Global Clustering coefficient Global clustering coefficient (GCC))**.** The *global clustering coefficient* is concerned with the density of triangles of nodes in a network. There two ways to compute *global clustering coefficient*, by computing the average of local clustering for its vertices we call it *average clustering coefficient* and another by using the *transitivity* of the a network.

1. *Average clustering coefficient:*

   The first approach to compute the global clustering coefficient of graph is to compute the *Average of the local clustering coefficient* from LCC of its nodes. Since LCC of a node is high if has a small degree if not low, so *Average clustering coefficient* will be high if most of the nodes have a small degree, if not low. The average clustering coefficient are define as (Opsahl and Panzarasa, 2009):

   $$\bar{C} = \frac{1}{n}\sum_{i=1}^{n} LCC(i) \tag{2.6.2}$$

   Let compute the average clustering coefficient of the network of the figure **??**:

   $$\bar{C} = \frac{1}{11}(\frac{2}{3} + \frac{2}{3} + \frac{1}{3} + 0 + \frac{1}{6} + \frac{1}{3} + 0 + 0 + 0 + 0 + 1) = 0.28787 \tag{2.6.3}$$

   This value means the probability that two nodes of the randomly selected network are connected to each other.

2. *Transitivity:*

   Transitivity is a network property that refers to the extent to which a relation between two nodes is transitive. It is a very important measure in social networks but less important in other types of networks. In social networks, the term transitivity reflects the friend-of-a-friend concept. It is sometimes used as a synonym of whole network clustering coefficient. For example let consider the figure 2.12(a), the node C is connected to E, E is connected to D. That means that C is connected to D?

   $$Transitivity(G) = \frac{2 * \text{number of triangles in the network}}{\text{number of connected triples in the network}} \tag{2.6.4}$$

The transitivity of the network of the figure 2.12 (a) is $transitivity(G) = 0.409$

The *Transitivity* is based on high degree nodes. It is high when a network has many nodes of high degree while *Average clustering coefficient* is based on high LCC nodes.

2.6.6 Distance Measures.

2.6.7 Definition (Walk)**.** A *walk* of length l is any sequence of (not necessarily different) nodes v_1, v_2,..., v_l, v_{l+1} such that there is a link from v_i to v_{i+1}, for each $i = 1, 2, ..., l$ (Cvetkovic et al., 1997).

2.6.8 Definition (Path)**.** A path of length l between v_1 and v_{i+1} is a walk of length l in which all the nodes (and all the edges) are distinct. The *length* of a path from a node v_1 to v_{i+1} is the number of edges between v_1 and v_{i+1} (or the small number of steps) (Cvetkovic et al., 1997).

Among all the paths between v_1 and v_{l+1} the ones having the minimum length are called *shortest-paths*.

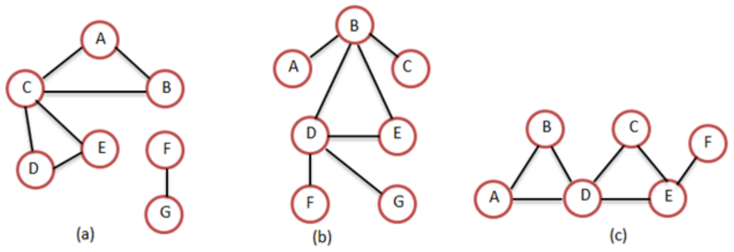


Figure 2.13: Networks

Let's consider the figure 2.13(b), $A - B - D - E$ is a path form A to E with *path length* $l = 3$ and $G - D - F$ is a path form G to F with *path length* $l = 2$.

2.6.9 Definition (Distance between two nodes)**.** The distance between two nodes is the length of the shortest path between them. In the case when u and v are in different connected components of the network, the distance between them is set to infinite, $d(u, v) = \infty$. It is also known as geodesic distance. It is possible to have more than one shortest paths between a pair of nodes (Wang and Chen, 2003).

For example, the graph of the figure 2.13(a), we can compute the distance between some nodes: $d(A, B) = 1$, $d(A, G) = \infty$ and $d(A, E) = 2$.

2.6.10 Definition (Distance Matrix)**.** In an undirected connected network the *distance matrix* D is such the element are the distance between any pair of node in the network. The distance matrix is square and the distances between a node v and any other node in the network are given at the $v-$th row or column of D. In the case of a directed network, the distance matrix is not necessarily symmetric and can contain entries equal to *infinite*.

One way to know the number of possible paths of length k between two nodes i and j is to raise the network adjacency matrix to the power k. And the value $A^k[ij]$ is the possible number of paths of length k between i and j. Raising the adjacency matrix to the power k is not easy when the network is large, that's why we will introduce the *Breadth-First-Search algorithm*

2.6.11 Breadth-First-Search Breadth-First-Search (BFS).

The idea would be to see the distance from one node i to all other nodes. In the presence of a small network, this is easy to calculate but when the network is large, it becomes very difficult. Hence the Breadth-first-search BFS algorithm is used to solve this problem. The figure 2.14 explain the steps of BFS algorithm, if we want to find the distance from A to all nodes of the graph.

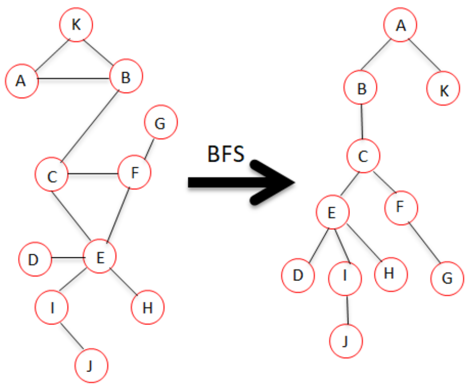


Figure 2.14: BFS illustration

We see that, $d(A, J) = 5$, $d(A, D) = 4$ and So on.

2.6.12 Definition (Average Path Length)**.** The *average path length* of a network is the average number of steps along the shortest paths for all possible pairs of network nodes. Let G = (V;E) be a graph the average path length L_G is defined by:

$$L_G = \frac{1}{n(n-1)} \sum_{i,j \in V, i \neq j} d_{ij} \tag{2.6.5}$$

where d_{ij} is the shortest path between node i and j and n is the total number of nodes in G. The value of L_G determines the size of a network and helps to determine the efficiency of information flow or disease spread over a network (Wang and Chen, 2003).

2.6.13 Definition (Eccentricity)**.** The *eccentricity* of a node i is the largest distance between i and all others nodes (Wang and Chen, 2003).

$$e(i) = max_{i,j \in V}\{d(i, j)\} \tag{2.6.6}$$

where $d(u, v)$ is the distance between u and v.

2.6.14 Definition (Diameter)**.** The *diameter* of a graph G is the maximum distance of the nodes between any pair of nodes (Wang and Chen, 2003).

$$diam(G) = max_{i,j \in V}\{d_{ij}\} \tag{2.6.7}$$

For a disconnected network, the diameter is undefined and therefore, for such a case, we take the efficiency of a network which is a finite value given by:

$$\bar{e} = \frac{1}{n(n-1)} \sum_{i,j \in V, i \neq j} \frac{1}{d_{ij}} \tag{2.6.8}$$

2.6.15 Definition (Radius)**.** The *Radius* of the network is the minimum eccentricity of the nodes.

$$Rad(G) = min_{i \in V}\{e(i)\} \tag{2.6.9}$$

2.6.16 Definition (Center)**.** a node is called *central* if its *eccentricity* is equal to the *radius* of the network. the centre of the graph C(G) is the set of all central nodes.

$$C(G) = \{i \in V(G) : e(i) = Rad(G)\} \tag{2.6.10}$$

2.6.17 Definition (Periphery)**.** A *periphery* of a graph G is the set of nodes that have *eccentricity* equal to the *diameter* of the graph.

$$P(G) = \{i \in V : e(i) = Diam(G)\} \tag{2.6.11}$$

Let consider the network in the figure 2.2 summarize these definitions:

Node i	A	B	C	D	E	F	G
Ecc(i)	3	2	3	2	2	3	3

(a) *The eccentricities of the node the network in the figure 2.13 (a)*

Radius	Diameter	Periphery	Center
2	3	A,C,F,G	B,D,E

(b) *Diameter, Radius, Periphery and Center of network in the figure 2.13(a)*

Table 2.2: Eccentricity, radius, diameter, periphery and center

2.7 Connected Components

2.7.1 Definition (Connected graph)**.** An undirected graph is *connected* if, for every pair of node, there is a path between them. For example, the graph of the figure 2.13(b) and 2.13(c) are connected.

2.7.2 Definition (Connected component)**.** A connected components of an undirected graph is subset of nodes such as:

- Every node in the subset has a path to every other nodes;
- No other node has a path to any node in the subset.

2.8 Network Measures

The *Centrality* identify the most important nodes in a particular (Qi et al., 2012): Influential nodes in a social network, A central node is important or powerfull, Nodes that discriminate information to many nodes or prevent epidemics, Hubs in a transportation network, Important pages on the web, node that prevent the network from breaking, etc.

2.8.1 Definition (Degree centrality (DC))**.** Degree centrality is the number of edges incident to a node. It is the simplest centrality measure and it is highly effective in determining the most important node especially in social network. However, a weakness of the degree centrality measure is that it only captures information about the local structure around a node leaving out the global structure (Qi et al., 2012). Mathematically, the degree centrality $CD(i)$ of a node i is given by:

$$CD(i) = \sum_j A_{ij} \tag{2.8.1}$$

For comparison purposes, we normalise the degree centrality by dividing it by the maximum centrality value possible; $n-1$. The *degree centrality* of a network is the maximum *degree centrality* of its nodes.

2.8.2 Definition (Closeness centrality (CC)). *Closeness centrality* is a centrality measure in which the importance of a node is determined by how close the given node is to its neighbors. In other words, how easily a node is reachable. This centrality helps in determining how fast it is to get to other nodes from a particular node and so closeness centrality can provide information regarding how fast an epidemic or information can spread over a given network. Closeness centrality captures information regarding the global network structure (Qi et al., 2012). It is defined as the inverse sum of shortest distances from the focal node i to all other nodes in a network, that is:

$$CC(i) = \frac{1}{\sum_{i \neq j} d_{ij}} \tag{2.8.2}$$

where d_{ij} is the shortest distance between nodes i and j. The normalised closeness centrality is given by:

$$CC'(i) = (n-1)CC(i) \tag{2.8.3}$$

2.8.3 Definition (Betweenness Centrality (BC)). The *Betweenness centrality* quantifies degree to which a node or edges occurs on the shortest path between all the other pair of nodes in a network. We define the betweenness centrality as (Qi et al., 2012):

$$C_B(i) = \sum_{s,t} \frac{\sigma_{s,t}(i)}{\sigma_{s,t}} \tag{2.8.4}$$

Where:

- $\sigma_{s,t}$ is the number of shortest path between nodes s and t;
- $\sigma_{s,t}(i)$ is the number of shortest path between nodes s and t that pass through node i.

A node i has high *betweenness centrality* if it participate in many of the shortest path of nodes s and t.

2.8.4 Definition (Subgraph Centrality (SC)). This type of centrality was presented by Ernesto and Rodriguez-Velazquez. With this centrality measure, the importance of a node is characterized by its participation in all subgraphs in a network. We define the subgraph centrality $CS(i)$ of a node i as:

$$C_S(i) = \sum_{k=0}^{\infty} \frac{u_k(i,i)}{k!} = \sum_{k=0}^{\infty} \frac{A^k(i,i)}{k!} = (e^A)_{ii} \tag{2.8.5}$$

where $u_k(i,i)$ is the number of closed k-walks that vertex i participates in the network, and A is the adjacency matrix of the network. It is sometimes desirable to normalize the subgraph centrality of a node by the sum:

$$EE(G) = \sum_{i=1}^{N} C_S(i) = \sum_{i=1}^{N} (e^A)_{ii} = Tr(e^A) = \sum_{i=1}^{N} (e^{\lambda_i}) \tag{2.8.6}$$

of all the subgraph centralities. The quantity $EE(G)$ is known as the Estrada index Estrada index (EE) of the graph G.

2.9 Network Properties

2.9.1 Small-world network.

A network is said to have the small-world property if, for a fixed average degree, the average path length

Node i	Degree	Closeness	Betweenness	Subgraph
A	0.4	0.555	0	0.13
B	0.6	0.6250	0.45	0.1911
C	0.6	0.71	0.10	0.1942
D	0.8	0.8333	0.35	0.2528
E	0.6	0.7143	0.4	0.16
F	0.2	0.4545	0	0.0685

Table 2.3: *The eccentricities of the node the network in the figure 2.13*

between pairs of nodes increases logarithmically with the number of nodes n (Newman, 2003):

$$L \propto \log n \tag{2.9.1}$$

Where L is the average path length. This property can be interpreted as propagation efficiency: spreading on the network remains relatively fast even if the network grows.

In other words, *Small world* stands for, small *average path length* and *high global clustering coefficient*. The figure 2.18 illustrate a *small world network.*

2.9.2 Scale-Free work.

Research shows that most of the real-world networks roughly follow a power-law degree distribution (Albert and Barabási, 2002). In this distribution, the probability of finding a node with degree k decreases as a negative power of degree k. This implies that in such networks, it is less likely to find a node with high degree:

$$p(k) \propto k^{-\gamma}, \text{ with } 2 \leq \gamma \leq 3 \tag{2.9.2}$$

The formula 2.9.2 means that, $\exists C \in \mathbb{R}^+ - \{0\}$ such that:

$$p(k) = Ck^{-\gamma}, \text{ with } 2 \leq \gamma \leq 3 \tag{2.9.3}$$

By using a logarithmic scale, the plot of Equation 2.9.3 becomes: $lnp(k) = -\gamma ln(k) + lnC$, with a slope equal to $-\gamma$ and an intercept equal to lnC as illustrated in figure 2.17. However, we observe that the part that corresponds to high degrees (tail of the distribution) is very noisy. In order to overcome this problem, one of the solutions is to consider the cumulative distribution function, which is defined:

$$p(k) = \sum_{k'=k}^{\infty} p(k') \tag{2.9.4}$$

which represents the probability of randomly choosing a node with degree k or greater (Estrada et al., 2011).

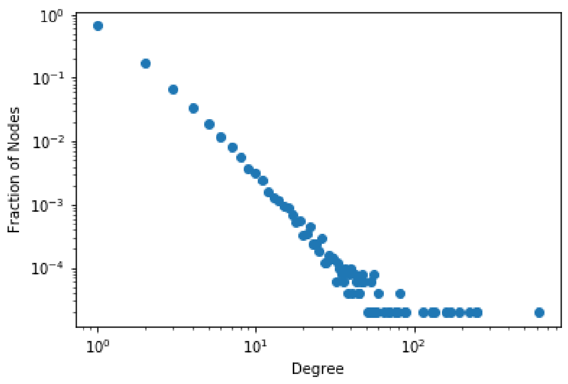


Figure 2.15: Power law distribution

2.10 Network Model

2.10.1 Random models of networks.

In the study of real-world networks (such as protein-protein interaction, transport network, food web, etc.), we observe different topological structures of these networks and it is of great importance to understand the mechanisms that are responsible for such structures. We therefore discuss some of the models that mimic real-world networks.

2.10.2 Erdos-Renyi (ER) model.

This type of network is completely random, and any graph composed by a set V is equally likely. There is also a fixed probability for a link to appear in the graph. This network is generated with the following algorithm (Erdős and Rényi, 1960):

- Initialize the network N nodes, all of them isolated;
- Add an edge with probability $p \in]0, 1[$ between a pair of nodes.

It is useful to notice that if $p = 0$, the nodes are completely *isolated* and when $p = 1$ we will obtain a complete network (all-to-all network).

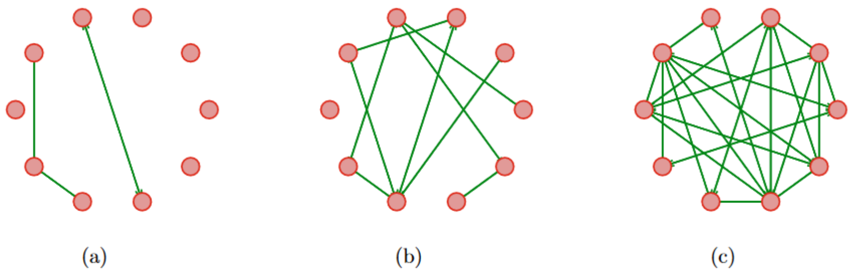


Figure 2.16: Erdos-Renyi network (a) $p = 0.1$ (b) $p = 0.2$ (c) $p = 0.5$.

2.10.3 Barabasi-Albert (BA) model.

Starting from a number M_0 of arbitrarily linked nodes:

- *Growth:* at each step of the algorithm, add one node with a number $M \leq M_0$ of edges connecting this node to any other previously attached to the network;

- *Preferential attachment model (PAM):* a new node is attached to node i according to the probability:

$$P_{attach}(i) = \frac{k_i}{\sum_j k_j} \tag{2.10.1}$$

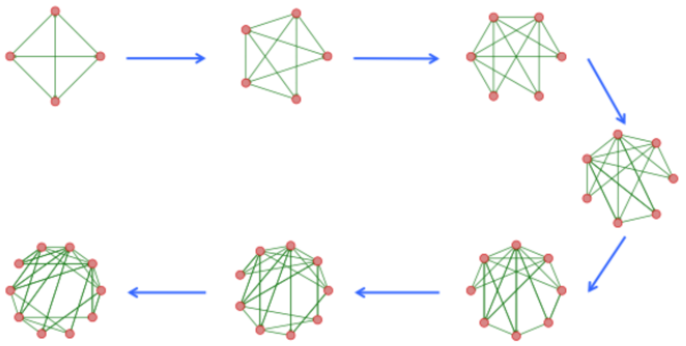


Figure 2.17: Scale free network generation.

2.10.4 Watts-Strogatz (WS) model.

- Start with a ring-shaped network of N nodes, connected with its $2K$ neighbors, with $K > 0$;
- Select a randomly-chosen pair of connected nodes. Keep the beginning of the edge and disconnect its end with probability $p \in]0, 1[$. Then, reconnect it to other randomly chosen node.

The rewiring process is done to each pair nodes only once.

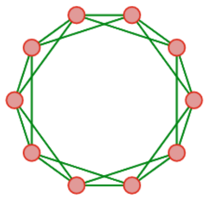


Figure 2.18: Small World network for $p = 0.1$: Watts-Strogatz (internet, i)

The *network topology* determines how the nodes are connected in the network. And the *regular graph* and the random model of networks are some kind of *network topology*. It can help us to generate a networks having some appropriate properties.

3. Community detection

Clustering is one of the fundamental topics in unsupervised learning. It has been widely and successfully applied in data mining, pattern recognition, and many other fields. Its goal is to group data, such that points in a same cluster are similar to each other, and less similar to points in other clusters. *Graph clustering* or *Graph partitioning* is a key branch of clustering, which tries to find disjoint partitions of graph nodes such that the connections between nodes within the same partition are much denser than those across different clusters. Some real application of graph clustering are: *community detection*, *image segmentation*, and other.

Graph clustering methods usually consist of two mains steps: construction of a network from dataset, and partition of the network into sub-networks, each one representing a data cluster. In a network, the concept of clusters or communities is a densely connected group of vertices. The connections among different clusters are sparse.

Community detection is very important for some reasons, such as: Identifying communities and their boundaries allows for a classification of vertices, according to their structural position in the communities. So a central vertices i.e sharing a large number of edges with the other group partners, may have an important function of *control* and *stability* with in the group. The boundaries between communities play an important role of mediation and lead the relationships and exchanges between different communities (Csermely, 2008). Its problem to find K *communities* or *modules* that are tightly connected with one other than with outside vertices.

Community detection can have concrete applications such as: identifying clusters of customers with similar interests in the network of purchase relationships between customers and products of online retailers (like, e.g.,www.amazon.com) enables one to setup efficient recommendation systems (Reddy et al., 2002).

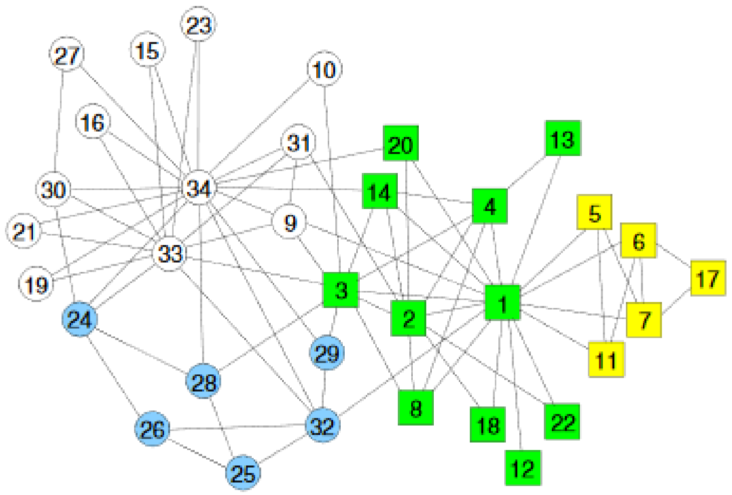


Figure 3.1: Network with four communities, vertices having the same color attributes belong to the same community. Source: (internet, d)

There are some problems to deal before starting community detection such as (Rosvall and Bergstrom, 2008):

- The choice of the graph to use especially a directed or undirected graph? Which means you've to know the interactions (relationships) between elements of a system need, it could be reciprocal or not. For example in the Figure 2.3 we can cite the predator-prey relationships in food webs which not reciprocal. Using the directed graph in community detection is hard task because the matrix associated to directed graph (Adjacent, Laplacian) are asymmetrical, so spectral analysis is much more complex. Therefore few techniques use directed graph;
- In many real networks, some vertices belong to more than one group, this problem is calling *overlapping* and we use the term *core* in stead of to use *partition*.

In this chapter we consider *disjoints communities* i.e., a vertex cannot belongs to different communities.

3.1 Elements of community detection

Let G be a n graph and C its subgraph with n_C vertices. And let $v \in C$, we have these definitions:

3.1.1 Definition (internal degree)**.** We define, *internal degree* of a node $v \in C$, as the number of edges connecting v to other vertices of C and we note k_v^{int} Fortunato, 2010).

3.1.2 Definition (External degree)**.** We define, *external degree* of a node $v \in C$, as the number of edges connecting v to other vertices who don't belong to C and we note k_v^{ext} Fortunato, 2010).

If $k_v^{ext} = 0$, then the vertex has neighbors only within C, which is likely to be a good cluster for v and $k_v^{int} = 0$, then instead, the vertex is disjoint from C and it should be better assigned to a different cluster. And we have (Fortunato, 2010):

1. $k_C^{int} = \sum_{v \in C} k_v^{int}$;
2. $k_C^{ext} = \sum_{v \in C} k_v^{ext}$

So, we have:

$$K_C = k_C^{int} + k_C^{ext} \tag{3.1.1}$$

K_C stands for the total degree of C.

3.1.3 Definition (Intra-cluster density)**.** We define the *intra-cluster density* δ_{int} of C as the ration between the number of internal edges of C and the number of all possible internal edges (Fortunato, 2010):

$$\delta_{int}(C) = \frac{\text{Number of internal edges of C}}{n_C(n_C - 1)} \tag{3.1.2}$$

3.1.4 Definition (inter-cluster density)**.** We define *inter-cluster density* δ_{ext} of C as the ratio between the number of edges running from the vertices of C to the rest of the graph and the maximum number of inter-cluster edges possible (Fortunato, 2010):

$$\delta_{ext}(C) = \frac{\text{Number of internal edges of C}}{n_C(n - n_C)} \tag{3.1.3}$$

Let consider $\delta(G)$ the *average* link of the graph G. If we consider C as a subgraph of G, so for C to be a community, we expect that:

The $\delta_{int}(C)$ to be larger than $\delta(G)$ and the $\delta_{ext}(C)$ has to be much smaller than $\delta(G)$. The main goal of clustering algorithms is to find the best tradeoff between a large value of $\delta_{int}(C)$ and a small value of $\delta_{ext}(C)$. A direct way is to *maximize* the sum of the differences:

$$\sum_{C} (\delta_{int}(C) - \delta_{ext}(C)) \tag{3.1.4}$$

over all clusters the partitions (Mancoridis et al., 1998).

Another property that require a community is the *connectedness*, it stands for there is always a path between each pair of its vertices running only through vertices of C.

3.1.5 Definition (Modularity)**.** Let $G = (V, E)$ be a undirected and unweighted graph. The *modularity* measure has been presented by Newman and Girvan (Wan et al., 2008) to assess the quality of a network partition $S = \{S_1, S_2, ..., S_k\}$. Defined by:

$$Q(S) = \frac{1}{2|E|} \sum_{i,j} \left(A_{ij} - \frac{k_u.k_v}{2|E|} \right) .\delta_{ij} \tag{3.1.5}$$

where A_{ij} the adjacency matrix value between i and j, k_i and k_j represent respectively the node degree of i and j. $\delta_{ij} = 1$ if i and j belong to the same clusters, 0 otherwise. $-1 \leq Q \leq 1$ and $Q > 0.3$ means the significant community structure.

The optimal clustering of graph maximizes the *modularity*.

We distinguish some types of methods of community detection among which we can cite: The traditional methods (graph partitioning, hierarchical clustering, partitional clustering and spectral clustering), The modularity-based methods, dynamical algorithms, etc... In this chapter we will study the traditional methods and the modularity-based methods (Fortunato, 2009).

Before continue, let recall some basics on linear algebra, distance notion and Laplacian matrix properties.

3.2 Similarity, Dissimilarity and Distance Matrix

Similarity and dissimilarity express the degree of coincidence or divergence between two elements of a given domain. Thus, it is reasonable to treat them as functions since the objective is to measure or calculate this value between any two elements of the domain (Orozco and Belanche, 2005).

In this section, we suppose that the attributes in x_i and x_j are all numerical. They are called feature vectors and have an arbitrary dimension of $p > 0$ and let's $X = \{x_1, ..., x_n\}$ be set of elements).

3.2.1 Definition (Similarity)**.**

A similarity function, is a function s that has any pair (x_i, x_j) that has a value in $\mathbb{R}_+$ $s : X \times X \longrightarrow \mathbb{R}_+$ such as (Silva and Zhao, 2016):

1. $s(x_i, x_j) = s(x_j, x_i) \geq 0$, for $i, j \in \mathbb{N}$; (Symmetry property)
2. $s_{max} = s(x_i, x_i) \geq s(x_i, x_j)$.

The larger the measurement, the more similar the points are.

3.2.2 Definition (Dissimilarity)**.** A Dissimilarity function is a function d that has any pair (x_i, x_j) that has a value in $\mathbb{R}_+$ $d : X \times X \longrightarrow \mathbb{R}_+$ such as (Silva and Zhao, 2016):

1. $d(x_i, x_j) = d(x_j, x_i) \geq 0$, for $i, j \in \mathbb{N}$; (Symmetry property)
2. $d(x_i, x_j) = 0 \Longrightarrow x_i = x_j$.

The lower the measurement, the more similar the points are.

It should be noted here that the shift from the similarity index (s) to the dual notion of dissimilarity (d) is obvious. Given that s_{max} the similarity of an individual with himself ($s_{max} = 1$ in the case of a normalized similarity), it is sufficient to set :

$$\forall x_i, x_j \in X, d(x_i, x_j) = s_{max} - s(x_i, x_j) \tag{3.2.1}$$

3.2.3 Definition (Distance)**.**

A distance is a dissimilarity d which additionally verifies the triangular inequality (Silva and Zhao, 2016):

$$\forall x_i, x_j, x_k \in X, d(x_i, x_k) \leq d(x_i, x_j) + d(x_j, x_k) \tag{3.2.2}$$

Using a distance depends on the types of data we are measuring.

1. **Example of distance:**

   Measurement of the distance $d(x_1, x_2)$ between two points x_1 and x_2 (Silva and Zhao, 2016):

   - *Minkowsky Distance:*

   $$d(x_1, x_2) = (\sum_{j=1}^{p} \mid x_{1,j} - x_{2,j} \mid^q)^{\frac{1}{q}} \tag{3.2.3}$$

   - *Euclidian Distance:* $q = 2$

   $$d(x_1, x_2) = \sqrt{\sum_{j=1}^{p} \mid x_{1,j} - x_{2,j} \mid^2} = \sqrt{(x_1 - x_2)^t (x_1 - x_2)} \tag{3.2.4}$$

   - *Manhattan Distance*: $q = 1$

   $$d(x_1, x_2) = \sum_{j=1}^{p} \mid x_{1,j} - x_{2,j} \mid \tag{3.2.5}$$

     with W diagonal weighting matrix.

   - *Mahalanobis Distance*: if the correlated variables become too important, the Euclidean distance can be normalized by covariance matrix.

   $$d^2(x_1, x_2) = (x_1 - x_2)^t C_X^{-1} (x_1 - x_2) \tag{3.2.6}$$

     with C_X is $p \times p$ variance-covariance (variance) matrix.

     And $cov(x, y) = cov(y, x) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$ so C_X is a symmetric matrix. When the variables are normalized (the standard deviation of each variable is 1), we call C_X the *correlation matrix*.

   - *Cosine Similarity:*

   $$cos(x_i, x_j) = \frac{\sum_{j=1}^{p} x_{ik} x_{jk}}{\sqrt{\sum_{k=1}^{p} x_{ik}^2} \sqrt{\sum_{k=1}^{p} x_{jk}^2}} \tag{3.2.7}$$

3.3 Network construction

There are several popular constructions to transform a given dataset $X = \{x_1, ..., x_n\}$ of data items with pairwise similarity matrix S and the degree matrix D of the graph. The goal when constructing similarity graphs is to model the local neighborhood relationships between the data points. Moreover, most of the constructions below lead to a sparse representation of the data, which has computational advantages. In this section, we explain these three methods (Silva and Zhao, 2016):

1. *ϵ-neighborhood graph:* In This method, we connect all data items i and j whose pairwise distances $s(i, j)$ are smaller than ϵ. Since the distances between all connected points are in the same scale (at most ϵ), weighting the edges of the graph will be useless. Hence, the ϵ-neighborhood graph is usually considered as an unweighted graph. Most of the time, we use Euclidian Distance.

2. *k-Nearest Neighbors Graph:* In This method, use k-Nearest Neighbors to connect vertex i with data items j if j is among the k-nearest neighbors of i. The k-Nearest Neighbors Graph method present some issues, in which we can say: The nearest neighbors are not symmetric that's means if there is a data item i which has j as a nearest neighbor, it is not necessary that i is a nearest neighbor of j. Thus, we'll have a directed graph which is a problem as we don't know what similarity between two data items means in that case. In this case, we have two ways to make this graph undirected:

    1. Ignore the directions of the edges, that's we connect i and j with an undirected edge if i is among the k-nearest neighbors of j or if j is among the k-nearest neighbors of i . The resulting graph is what is usually called the k-nearest neighbor graph.

    2. choice is to connect vertices i and j if both i is among the k-nearest neighbors of j and j is among the k-nearest neighbors of v_i . The resulting graph is called the mutual k-nearest neighbor graph. In both cases, after connecting the appropriate vertices we weight the edges by the similarity of the adjacent points.

3. *Fully connected graph:* In this method, the idea is, to construct this network, we simply connect all points with each other, and we weight all edges by similarity s_{ij}. This graph should model the local neighborhood relationships, thus similarity functions such as Gaussian similarity function are used.

    $$s_{ij} = \exp\left(\frac{-||x_i - x_j||^2}{2\sigma^2}\right) \tag{3.3.1}$$

    The Hyper-parameter σ controls the width of the neighborhoods, similar to the parameter ϵ in case of the ϵ-neighborhood graph. Thus, when we create an adjacency matrix for any of these graphs, $A_{ij} \sim 1$ when the points are close and $A_{ij} \to 1$ if the points are far apart and $||x_i - x_j||$ is the Euclidean norm between x_i and x_j (Silva and Zhao, 2016).

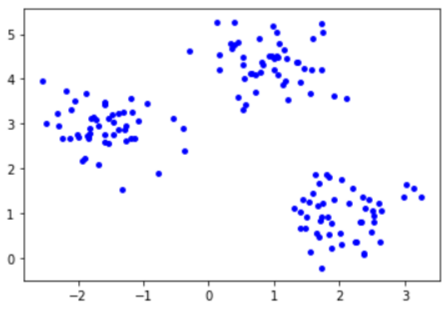
3.3.1 Network construction by K-nearest neighbors K-NN with Single linkage SL.

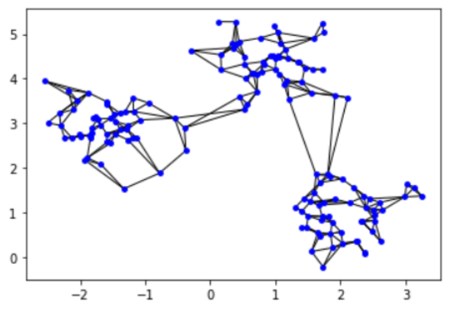
In other way to solve the problems of lack of sparsity (dense) and disconnected network that represent the techniques of section 3.3, we present the method proposed by (Cupertino et al., 2013), where the construction of a network based on *single-linkage* (Sibson, 1973) clustering heuristic and k NN that is capable of constructing *connected* and *sparse* networks which, at the same time, tend to keep the cluster structure of the original dataset.

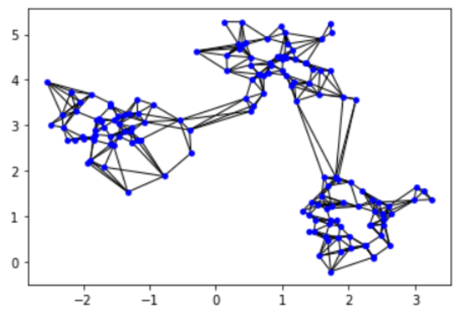
Algorithm 1: Network construction based on the single linkage clustering heuristic

```
Input: $X$ unlabeled dataset of size $n$;
Result: A sparse network
hyparameters:
$k$: number of nodes to connect;
$\gamma$: threshold value.
(1) start with each vertex $i$ into a different group $G_i$ ($n$ initial groups);
(2) calculate the distances among all groups by using a distance measure (Euclidian distance);
Repeat:
(3) find two closest groups (by single linkage) and denote them $G_1$ and $G_2$;
(4) calculate the average distance among vertices inside each group $G_1$ and $G_2$ and denote them
    by $d_1$ and $d_2$, respectively;
(5) select the $k-$ most similar pairs of vertices that connect $G_1$ and $G_2$, and create an edge
    between each selected pair if its dissimilarity is smaller than a threshold defined as:
    $d_{th} = \gamma max(d_1, d_2)$, where $\gamma > 0$. This step joins $G_1$ and $G_2$ into a larger group;
(6) Update the adjacency matrix by calculating the dissimilarities between the group formed in
    step $5$ and all other groups;
(7) If the number of groups is larger than one, return to step 3.
```

3.3.2 Example (Illustrative of network construction)**.**


(a) An artificial dataset with 2 variables, 150 instances, 3 center and the standard deviation of each cluster is 0.5.


(b) Networks constructed with $k = 3$


(c) Networks constructed with $k = 5$

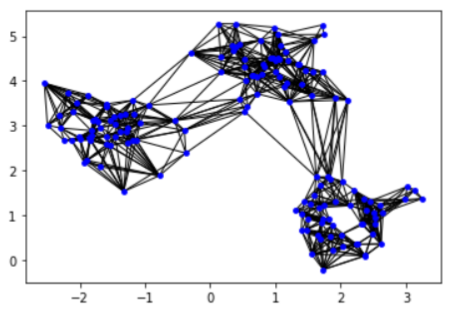

(d) Networks constructed with $k = 15$

Figure 3.2: Results of the network construction algorithm using an artificial dataset. In all cases $\gamma = 3$

As an example of the proposed algorithm for network construction, the figure 3.2, shows the results for an artificial dataset composed of three clusters of different sizes and densities. In this example three different values of k are used 3, 5 and 15. It can be observed that for all cases it results in a connected network with a fair distribution of edges among clusters, that is, the connections inside a cluster are dense while the connections inter-clusters are *sparse*.

3.3.3 Remark (Remark)**.**

- In using this method, the interest is that, when two groups are being connected the connection starts by the closest vertices, that is, the border vertices of each group. We'll use them for connecting the groups and it is necessary to define how many vertices will be connected;
- As it can be noticed in figure 3.2, k is responsible for reinforcement intra-cluster connections, while keeping inter-cluster connections *sparse*. Thus, parameter k is used for model selection.

3.4 Hierarchical clustering

Hierarchical clustering (HC) seeks to partition data by building a hierarchical tree in which each node can develop into several sub-nodes such that at each level, corresponds a data partition. And his hierarchical tree is called *Dendrogram*. HC techniques may be subdivided into *agglomerative* methods, which proceed by a series of successive fusions of the n individuals into groups, and *divisive* methods, which separate the n individuals successively into finer groupings (Everitt et al., 2011).

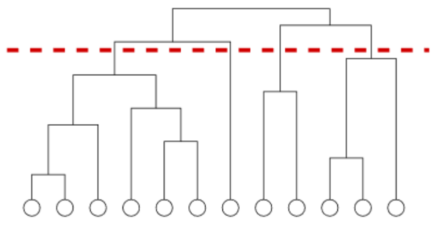


Figure 3.3: Dendrogram illustration (Fortunato, 2010).

3.4.1 Definition (Hierarchy index)**.** Hierarchy index is a monotone function that is:

$$i : H \to \mathbb{R}^+, A \subset B \longrightarrow i(A) < i(B) \tag{3.4.1}$$

$i(A)$ represents the level at which the elements of A are aggregated for the first time.

3.4.2 Definition (Agglomerative and Divisive method)**.** *Agglomerative* procedures are probably the most widely used of the hierarchical methods. They produce a series of partitions of the data: the first consists of n single member *clusters*; the last consists of a single group containing all n individuals. In other words, it is a *bottom up* approach (Everitt et al., 2011) while *Diviside:* is a *top down* approach: all observations start in one cluster, and splits are performed recursively as one moves down the *hierarchy* (Everitt et al., 2011).

3.4.3 Hierarchical clustering Algorithm by Agglomerative.

```
Algorithm 2: Hierarchical clustering Algorithm by Agglomerative
Input: $X$ ($n$ individuals, $p$ variables);
Result: A partition of individuals characterized
(1) Compute the distance matrix;
(2) Let each data point be a cluster ;
Repeat:
(1) Merge the two closest clusters;
(2) Update the distance matrix;
Until: only a single cluster remains.
```

3.5 Linkage Criterion

Before performing *hierarchical clustering* (HC), it is required to determine the proximity matrix containing the distance between each point using a distance function. Then, the matrix is updated to display the distance between each cluster. The following three methods differ in how the distance between each cluster is measured. So the common techniques to do this computation it is: *complete linkage*, *average linkage* and *ward (weighted linkage)* (Jain and Dubes, 1988).

Let X be a dataset, A and B its clusters.

3.5.1 Definition (Single linkage (SL))**.** Based on the distance between the two nearest points of each cluster (Boberg and Salakoski, 1997).

$$d_{min}(A,B) = min\{d(a,b); a \in A, b \in B\} \tag{3.5.1}$$

3.5.2 Definition (Complete linkage Complete linkage (CL))**.** Based on the distance between the two furthest points of the two clusters (Boberg and Salakoski, 1997).

$$d_{max}(A,B) = max\{d(a,b); a \in A, b \in B\} \tag{3.5.2}$$

3.5.3 Definition (Average linkage Average linkage (AL))**.** Based on the average distance between the two clusters (Chehreghani et al., 2008).

$$d_{avrg}(A,B) = \frac{1}{|A|.|B|} \sum_{a \in A, b \in B} d(a,b) \tag{3.5.3}$$

3.5.4 Definition (Weight linkage or ward Ward linkage (WL))**.** Ward stipulates that two clusters are merged where the loss of between inertia is lower. This will promote a merger that results in the lowest heterogeneity (Jain and Dubes, 1988). The loss of inertia on merging is:

$$d_{weight}(A,B) = \frac{P_A.P_B}{P_A + P_B} \sum_{a \in A, b \in B} d(a,b) \tag{3.5.4}$$

where P_A and P_B represent respectively the proportion of data in clusters A and B:

$$P_A = \tfrac{|A|}{|X|} \text{ and } P_B = \frac{|B|}{|X|}.$$

The Figure 3.4 explain well the linkage criterion.

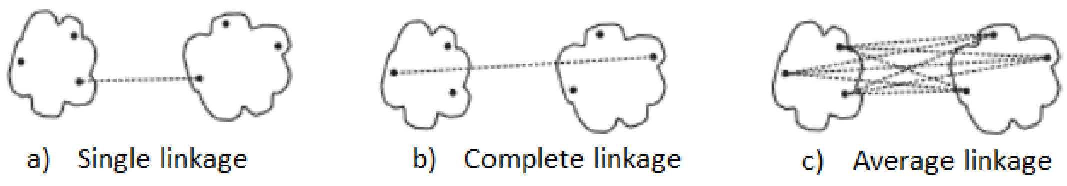


Figure 3.4: Single, complete and average linkage illustration

3.5.5 Advantages.
Hierarchical methods have a number of advantages, among them:

1. They do not impose the fixing of a priori the number of classes;
2. They apply to any data set by defining the measure of similarity or dissimilarity between objects;
3. The reading of the *dendrogram* provides the optimal number of classes.

3.5.6 Disadvantages.
Hierarchical methods have a number of drawbacks, including the following:

1. They are very expensive in terms of computing time;
2. In them, it is impossible to make improvements to the classes (Tree of nodes) once they have been built. This is due to the static assignment of data (that is, once data is assigned to a class, it remains permanently in the class during the construction of the hierarchy);
3. They have a high computational complexity: if n is the number of data, then it is of the order$O(n^2)$ for the simple link and $O(n^2log(n))$ for the complete and average link.

3.6 Community detection methods

3.6.1 Normalized cut.
Graph partitioning divide a graph into subgraphs such that we minimize the number of edges that connect different subgraphs. A useful measure that we minimize is the *normalized cut*. So, we need to define a *cut* and volume given $V_1, V_2 \subset V$.

3.6.2 Definition (Cut)**.** The $Cut(V_1, V_2)$ be the number of edges that connect a node in V_1 to a node in V_2.

3.6.3 Definition (Volume)**.** The volume of a set V_1 of nodes, denoted $Vol(V_1)$, to be the number of edges with at least one end in V_1.

The *normalized cut* for V_1 and V_2 is given by:

$$NCut(V_1, V_2) = \frac{Cut(V_1, V_2)}{Vol(V_1)} + \frac{Cut(V_1, V_2)}{Vol(V_2)} \tag{3.6.1}$$

So, the best partition has the smaller *normalized cut*.

3.6.4 Example.

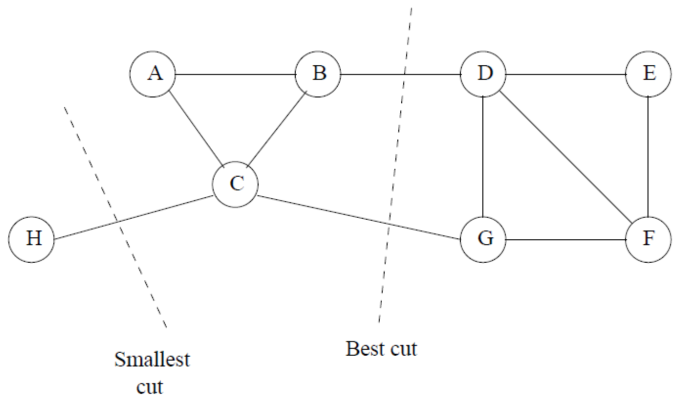


Figure 3.5: Network with 8 nodes and 11 edges

$V_1 = \{H\}$ and $V_2 = \{A, B, C, D, E, F, G\}$. So, $Cut(V_1, V_2) = 1$, $vol(V_1) = 1$ and $vol(V_2) = 11$. So the normalized cut is:

$$NCut(V_1, V_2) = \frac{1}{1} + \frac{1}{11} = 1.09 \tag{3.6.2}$$

Let take another partition: $V_1 = \{A, B, C, H\}$ and $V_2 = \{D, E, F, G\}$. So, $Cut(V_1, V_2) = 2$, $vol(V_1) = 6$ and $vol(V_2) = 7$. So the normalized cut is:

$$NCut(V_1, V_2) = \frac{1}{1} + \frac{1}{11} = 0.62 \tag{3.6.3}$$

This method oblige that the clusters have the same size. This condition is not general possible for a real-world problem.

3.6.5 Spectral clustering.

Given an undirected weighted graph $G = (V, E)$, where $V = \{v_1, ..., v_n\}$ is the set of nodes and $E = \{e_{ij}\}$ is the set of edges. And let S be the similarity matrix between nodes of G e.i., $\forall i, j \in V$ we have:

$$s(i, j) = s(j, i) \geq 0 \tag{3.6.4}$$

The similarity matrix S of a graph G will weights the score between nodes in the graph, in which we will learn same ways to construct it. In this case the adjacency matrix A is replaced by the similarity matrix S, so the degree k_i of a node i becomes:

$$k_i = \sum_{j=1}^{n} s(i, j) \tag{3.6.5}$$

Spectral clustering includes all techniques whose purpose is to partition a dataset into a clusters by using the eigenvectors of matrices such as: Similarity matrix S or Laplacian matrix L. In particular, the objects could be points in some Euclidian space, or the vertices of a graph.

The main goal of spectral clustering is to transform a dataset of items into a set of points in space, whose coordinate are elements of eigenvectors, and this set of point is then clustered by standard clustering methods, like *k-means*.

3.6.6 Normalized spectral clustering Algorithm.

In spectral clustering, all the algorithms are almost the same, they differ in the Laplacian choice and also in the choice of the similarity matrix learning technique. Other algorithms, such as that in (Shi and Malik, 2000) use the unnormalized Laplacian L. In this section, we present only the one proposed by Ng, Jordan, and Weiss (Ng et al., 2002) which uses the *normalized Laplacian* L. Given a set of points $X = \{x_1, ..., x_n\}$ in $\in \mathbb{R}^l$ that we want to cluster into k subsets:

```
Algorithm 3: Normalized spectral clustering algorithm
Input: $X$ an unlabeled dataset of size $n$
Result: Cluster for each node
(1) Form the affinity matrix $A \in \mathbb{R}^{nxn}$ defined by $A_{ij} = \exp(\frac{-||x_i - x_j||^2}{2\sigma^2})$ if $i \neq j$ and $A_{ii} = 1$.
    Where $x_i$ and $x_j$ are data items of $X$;
(2) Define D to be the diagonal matrix whose $(i, i)$-element is the sum of $A's$ $i-th$ row, and
    construct the matrix $L = D^{-1/2} A D^{-1/2}$;
(3) Find $x_1, x_2, ..., x_k$ , the $k$ largest eigenvectors of $L$ (chosen to be orthogonal to each other in
    the case of repeated eigenvalues), and form the matrix $X = [x_1 x_2 ... x_k] \in \mathbb{R}^{nxk}$ by stacking the
    eigenvectors in columns;
(4) Form the matrix $Y$ from $X$ by renormalizing each of $X's$ rows to have unit length (i.e
    $Y_{ij} = \frac{X_{ij}}{\sum_j (X_{ij}^2)^{\frac{1}{2}}}$;
(5) Treating each row of $Y$ as a point in $\mathbb{R}^k$, cluster them into $k$ clusters via K-means or any other
    algorithm (that attempts to minimize distortion);
(6) Finally, assign the original point $S_i$ to cluster$j$ if and only if row $i$ of the matrix $Y$ was assigned
    to cluster $j$.
```

Here, the scaling parameter σ^2 controls how rapidly the affinity A_{ij} falls off with the distance between s_i and s_j, and we will later describe a method for choosing it automatically.

The main question is that, why not apply clustering directly to the initial data based on the data similarity matrix? The reason is that, the change in data representation driven by eigenvectors makes the cluster properties of the initial dataset much more obvious. In this case, spectral clustering is able to separate data items that could not be resolved by applying directly k-means clustering, for instance, as the latter tends to deliver convex sets of points (Fortunato, 2010).

Advantages: Does not make strong assumptions on the statistics of the clusters. Clustering techniques like *K-Means* Clustering assume that the points assigned to a cluster are spherical about the cluster center. This is a strong assumption to make, and may not always be relevant. In such cases, spectral clustering helps create more accurate clusters.

Disadvantages: Computationally expensive for large datasets.

3.6.7 Girvan-Newman Algorithm.

Girvan-Newman algorithm is a divisive hierarchical clustering that works by removing edges. Its main steps are:

1. Create a graph of N nodes and its edges;
2. Compute the betweenness of each node in the graph;
3. remove all the edge (s) with the highest betweenness;
4. recalculate the betweenness of all the edges that got affected by the removal of edges;

5. Now repeat 3 and 4 until no edge remain in the graph .
6. Gives a hierarchical decomposition of the network.

where the *edge betweenness* is the number of shortest paths passing through the edges. The logical way of finding betweenness edge for a given edge is to use Breadth first search BFS.

Although its simplicity, *Girvan-Newman* algorithm provides some drawbacks:

- not efficient with a large number of nodes;
- difficulty to detect communities in a large and complex network.

3.6.8 Label Propagation Algorithm Label Propagation Algorithm (LPA).

LPA is a method introduced by (Raghavan et al., 2007). The main steps of LPA are:

1. Initialize labels on all nodes;
2. Randomized node order;
3. For every node replace its label with occurring with the highest frequency among neighbors;
4. If ever node has a label that the maximum number of their neighbor have, then stop the algorithm.

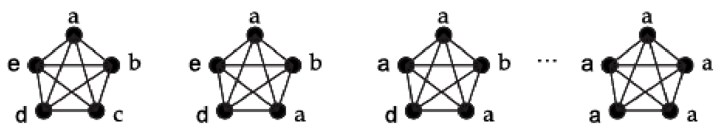


Figure 3.6: Nodes are updated one by one as we move from left to right (Raghavan et al., 2007).

3.6.9 Louvain.

The Louvain algorithm one of the hierarchical clustering technique for community detection is an algorithm for detecting communities in networks. It maximizes a *modularity* score for each community. The modularity quantifies the quality of an assignment of nodes to communities. This stands for evaluating how much are densely connected the nodes within a community.

The main steps of *Louvain* algorithm are:

- Start with every node in its own community
- **Phase 1:** *Modularity Optimization*
  - Order the nodes and do the following for each node i:
    - Move i to the community of neighbor j that leads to maximum ΔQ;
    - if $\Delta Q < 0$ the i remains in its current community.
  - Repeatedly cycle through all nodes until $\Delta Q = 0$
- **Phase 2:** *Community Aggregation*
  - Create a weighted network from *Phase 1*:
    - Nodes= communities from *Phase 1*;
    - *Edges weights*= sum of weights of edges between communities;

    * Edges within a community become 2 self-loops.

- Repeat *Phase 1*/ *Phase 2* to resulting network, and so until $\Delta Q = 0$.

The maximization of Modularity is NP-hard to optimize (Brandes et al., 2007).

3.7 Clustering metric evaluations

We want to compare the result of a clustering algorithm $C = \{C_1, ..., C_k\}$ and $P = \{P_1, ..., P_k\}$. Let's recall these concepts:

3.7.1 Definition (Rand index (RI)). The RI is a clustering metric which allows us to evaluate the quality of a clustering for a given dataset D with n data points. Suppose that P and Q are partitions of D into k and l subsets respectively (Rand, 1971). With $P = \{P_1, P_2, ..., P_k\}$ and $Q = \{Q_1, Q_2, ..., Q_l\}$.

Let's define this concepts:

- a: the number of pairs of data points in D that are in the *same* subset in P and in the *same* subset in Q;
- b: the number of pairs of data points in D that are in the *different* subset in P and in the *different* subset in Q;
- c: the number of pairs of data points in D that are in the *same* subset in P and in the *different* subset in Q;
- d: the number of pairs of data points in D that are in the *different* subset in P and in the *same* subset in Q;

The formula of *Rand index* is given by the Equation 3.7.1:

$$RI = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \tag{3.7.1}$$

The *Rand Index* is always between 0 and $+1$.

3.7.2 Definition (Adjusted Rand index (ARI)). Basically, the ARI is the *Rand index* corrected for chance (Hubert and Arabie, 1985):

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\frac{\sum_i \binom{a_i}{2}\sum_j \binom{a_j}{2}}{\binom{n}{2}}\right]}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{a_j}{2}\right] - \left[\frac{\sum_i \binom{a_i}{2}\sum_j \binom{a_j}{2}}{\binom{n}{2}}\right]} \tag{3.7.2}$$

where $n_{ij} = |P_i \cap Q_j|$ are the number of elements that are in both cluster $P_i \in P$ and cluster $Q_j \in Q$, $a_i = \sum_{j=1}^{l} n_{ij}$, $a_j = \sum_{i=1}^{k} n_{ij}$, $i \in \{1, ..., k\}$ and $j \in \{1, ..., l\}$, such that $\sum_{i,j}^{n_{ij}} = n$

For some algorithms, we use modularity of the partition or normalized cut. In this project, we'll use *consensus clustering* in order to detect communities into a network.

4. Consensus with pinning control

In this chapter, we introduce the notion of consensus using the local interactions between nodes. To reach consensus in a complex network, there are several methods, the simplest is to consider that all agents or a nodes update its state at a given time according to the state of its direct neighbours, and the consensus is reached when all nodes reach the average value of their initial states. In this chapter, we present another efficient technique to reach consensus by controlling a fraction of nodes in the network, this technique is called pinning control. we present a review of the dynamic system.

4.1 Overview on Dynamical System

4.1.1 Linear Dynamical Systems.
Dynamical systems whose variables are linked by linear functions are called linear systems. The temporal evolution of a continuous linear system characterized by n system variables is generated by a set of n ordinary differential equations (Boccaletti et al., 2018):

$$\begin{cases} \dot{x}_1 = a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \\ \dot{x}_2 = a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \\ \vdots \\ \dot{x}_n = a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n \end{cases} \tag{4.1.1}$$

where $x_i = x_i(t)$ are the time-dependent system variables $\dot{x} = \frac{dx}{dt}$ are their time derivatives, and a_{ij} are constant coefficients.

The equation 4.1.1 can be written as:

$$\dot{X}(t) = AX(t) \tag{4.1.2}$$

where $x = (x_1, x_2, .., x_n)$ is an n-dimensional vector, and A is a nxn constant matrix.

4.1.2 Definition (Trajectory or Orbit)**.** A *Trajectory* of an integral curve x is the set of $\{x(t)|t \in \mathbb{R}\} \subset \mathbb{R}^n$.

When the system dynamics are defined in terms of discrete times, i.e., when the current state of the system is iteratively determined by the previous one, the dynamics are instead described by difference equations, or iterative maps. The variables exhibit a mapping form, as x varies in discrete steps:

$$x_{i+1} = Ax_i \tag{4.1.3}$$

The eigenvalues λ represent powers of the exponential components of the solution, and the eigenvectors are their coefficients.

4.1.3 Nonlinear Dynamical Systems.
If a system is characterized by variables that depend nonlinearly on each other, the motion can become very complex. Such systems are called nonlinear dynamical systems. Mathematically, a nonlinear continuous dynamical system is described by:

$$\begin{cases} \dot{x}_1 = F_1(x_1, x_2, ..., x_n) \\ \dot{x}_2 = F_2(x_1, x_2, ..., x_n) \\ \vdots \\ \dot{x}_n = F_n(x_1, x_2, ..., x_n) \end{cases} \tag{4.1.4}$$

where F_i are functions that couple the variables among them. If at least one of these functions is nonlinear, the system in Equation 4.1.3 is said to be nonlinear. In vector form, a nonlinear dynamical system can be described as:

$$\dot{x}(t) = F(x(t)) \tag{4.1.5}$$

where $F = (F_1, F_2, ..., F_n)$ is a vector function: $\mathbb{R}^n \longrightarrow \mathbb{R}^n$.

The time evolution of the system describes a trajectory, or orbit, in the Euclidean space of the n variables $x \in \mathbb{R}^n$, or phase space. Each point in the phase space represents a unique state of the system.

In the case of three-dimensional systems, one can visualize directly the trajectory in three coordinates (x_1, x_2, x_3), while for systems with $n > 3$, visualization of the orbit is only possible by means of projections of the phase space on planes (or hyperplanes) of two or three of the system's variables.

Since at any given time the system state is described by a vector defined by the functions and parameters in Equation 4.1.3, the system evolution is *deterministic*.

4.1.4 Definition (Autonomous and Nonautonomous Systems)**.** A dynamical system that contains a time-dependent function is called *nonautonomous*; otherwise, the system is said to be *autonomous*.

4.1.5 Stabilitity of Dynamical System.

The information about the stability of the solutions of dynamical system is vital for understanding the mechanisms responsible for the system behavior. We can use briefly *linear stability* and *Lyapunov exponent*.

The *stability* according to *Poisson* means that, after a while, the phase *trajectory* returns to an arbitrarily small neighbourhood of the initial point $x(0)$. We can say nothing about the behavior of neighboring trajectories, initially close to $x(t)$. In practical problems we are often interested in another property of stability, associated with a small perturbation of a given trajectory. Depending on the temporal dynamics of the perturbation, we distinguish stability according to *Lyapunov* and *asymptotic stability* (Anishchenko et al., 2014).

4.1.6 Definition (Equilibrium State)**.** Let Ω be an open set. An autonomous differential equation is of the form:

$$\dot{x} = f(x) \tag{4.1.6}$$

Where $x(0) = x_0 \in \Omega$. Given $T > 0$, we define:

$$x(.) : [0, T] \longrightarrow \mathbb{R}^n \tag{4.1.7}$$

Such that $x(0) = x_0$ and $\dot{x}(t) = f(x(t))\ \forall t \in [0, T]$.

A point $\bar{x} \in \Omega$ is called an *Equilibrium state* or *Critical point* of Equation 4.1.6 if $f(\bar{x}) = 0$ i.e $\bar{x} = \bar{x}(t)$ is a constant $\forall t \in [0, T]$.

4.1.7 Definition (Stable Point)**.** An equilibrium state $\bar{x}$ of 4.1.5 is *Stable* if $\forall \epsilon > 0$, $\exists \delta > 0$ such that: $||x_0 - \bar{x}|| < \delta \Longrightarrow ||x(t) - \bar{x}|| < \epsilon$. Otherwise $\bar{x}$ is *Unstable*.

The *stability of linear system* $\dot{X}(t) = AX(t)$. $\bar{x}$ is an equilibrium state of and is:

- *Asymptotically stable* if and only if all eigenvalues of A have negative real part $(\lambda < 0)$;
- *Stable* if all eigenvalues of A have non positive $(\lambda \leq 0)$ real part;
- *Unstable* if there is an eigenvalue with positive real part.

If the *stability nonLinear System* e.g., $\dot{X}(t) = F(X(t))$, by linearizing, equation **??** becomes:

$$\dot{X} = F(\bar{x}) + J_F(\bar{x})(x - \bar{x}) \tag{4.1.8}$$

The linearization of $F(x)$ is the first-order term of its *taylor expansion* around the specific equilibrium point $\bar{x}$ whose stability properties have to assessed. Where $J_F(\bar{x})$ is the *Jacobian* matrix of $F(x)$ evaluate at $\bar{x}$.

4.2 Synchronization Networks

The origin of the word synchronization is a Greek root which means *to occur at the same time*. The original meaning of synchronization has been maintained up to now in the colloquial use of this word, as agreement or correlation in time of different processes (Hornby, 1974).

The Synchronization is commonly understood as a collective state of coupled systems. Before describing mathematically synchronization, let's define what mean a *coupled system*.

4.2.1 Definition (coupled system)**.** An N-dimensional dynamical system is called uncoupled if it can be written in $n < N$ (n number of oscillators) *independent dynamical* subsystems with total dimension equal to n.

The general form for an uncoupled system, when we have just two oscillators ($m = 2$), the Equation 4.2.2 represents an uncoupled system:

$$\begin{cases} \dot{x} = F(x) \\ \dot{y} = G(y) \end{cases} \tag{4.2.1}$$

where $x \in \mathbb{R}^k$, $y \in \mathbb{R}^l$, and the total dimension of the coupled system is $N = k + l$. We call $\mathbb{R}^k$ and $\mathbb{R}^l$ the space state or phase space for two oscillators. For example: A we call a coupled system, when we have an interactions between the oscillators, that means (case of two oscillators):

$$\begin{cases} \dot{x} = F(x) \\ \dot{y} = G(y) + K(x, y) \end{cases} \tag{4.2.2}$$

where $K(x, y)$ is a nonzero function of x and y, called also coupling *coupling function*. This coupling scheme is often referred to as *master–slave* configuration, where the subsystem x is called *master* or *drive* and the subsystem y is called *slave* or *response*. In some region of the phase space $\mathbb{R}^{n+m}$ the behavior of the slave is influenced by the behavior of the master, while the driving system is completely independent of the response system. And we call this configuration a *unidirectional* coupled system.

$$\begin{cases} \dot{x} = F(x) + K_1(y, x) \\ \dot{y} = G(y) + K_2(x, y) \end{cases} \tag{4.2.3}$$

where $K_1(y, x)$ and $K_2(x, y)$ are nonzero functions of x and y. $K_1 = K_2$ in case of symmetric coupling, that means each subsystem influences the other in some region of the phase space $\mathbb{R}^{n+m}$. And we call this configuration a *bidirectional* coupled system.

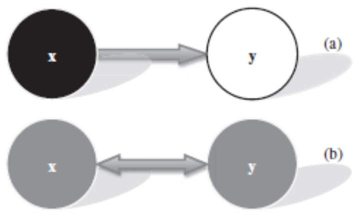


Figure 4.1: The two possible configurations for two coupled oscillators. (a) Unidirectional coupling, or master–slave configuration. (b) Bidirectional coupling. In the former case, the master and slave oscillators are shown, respectively, in black and white. In the latter, the two oscillators, which mutually combine their influence and interaction, are shown in gray (Boccaletti et al., 2018)

.

The Figure 4.1(a) describes the unidirectional from the coupled system 4.2.2. And Figure 4.1(b) describes the bidirectional coupled system.

The coupling function can be expressed in several ways, the popular way is called *diffusive coupling*:

$$K(x, y) = (R(x) - y) \tag{4.2.4}$$

where σ is the coupling strength and $R(x)$ is the *response function*. When $\sigma = 0$, the dynamical system is uncoupled i.e., the systems follow their own dynamics (Boccaletti et al., 2018)

4.2.2 Definition (Synchronization networks)**.** A synchronization networks refers to a networks connecting oscillators. In simple words, an oscillators are nodes that emit a signal with somewhat regular frequency, and are also able to receive. Synchronization networks is called also networks of coupled dynamical system.

4.2.3 Complete synchronization.

The general case can be described by considering two coupled nonlinear systems, given by:

$$\begin{cases} \dot{x} = f(x) + C_x u(x, y) \\ \dot{y} = g(y) + C_y v(x, y) \end{cases} \tag{4.2.5}$$

where $x \in \mathbb{R}^k$ and $y \in \mathbb{R}^l$ are vector variables, f and g are vector function, u and v are coupling functions, C_x and C_y are are matrices of coupling coefficients of appropriate dimension. If a coupling is symmetric, the: $u = -v$ and $C_x = C_y$. And $C_x = 0$ in case of An undirected coupling.

4.2.4 Definition (Measure of complete synchronization)**.** The coupled system of Equations 4.2.5 is said to be completely synchronized if all their state variables asymptotically match, i.e.,

$$\lim_{t \to \infty} ||x_i(t) - x_j(t)|| = 0 \tag{4.2.6}$$

For all i and j. The Equation 4.2.6 is rigorously possible for identical system $(f = g)$. The *complete synchronization*

4.2.5 Example. coupling function and synchronization

To illustrate the fundamental of synchronization and coupling function, let's consider an example of two coupled phase oscillators: (Kuramoto, 1978)

$$\begin{cases} \dot{\phi}_1 = f(\phi_1) + \epsilon_1 sin(\phi_2 - \phi_1) \\ \dot{\phi}_2 = g(\phi_2) + \epsilon_2 sin(\phi_1 - \phi_2) \end{cases} \tag{4.2.7}$$

The system 4.2.7 becomes:

$$\dot{\Phi} = [g(\phi_2) - f(\phi_1)] - (\epsilon_1 + \epsilon_2)sin(\Phi) \tag{4.2.8}$$

where $\Phi = \phi_2 - \phi_1$, ϵ_1 and ϵ_2 are the coupling strength parameters and and the coupling functions of between these two oscillators is *sinusoidal*.

4.2.6 Definition (synchronization error)**.** The simplest way to detect complete synchronization is the direct comparison of the state variables of the interacting systems x and $\bar{x}$, by calculating the synchronization error defined as:

$$e(t) = x(t) - \bar{x} \tag{4.2.9}$$

$e(t)$ is the distance between $x(t)$ and the fixed point $\bar{x}$, and as we assume that we are in the neighborhood of $\bar{x}$, the magnitude $e(t)$ is small. In the synchronization, $\bar{x}$ is the *target state*.

4.3 Information Consensus and the Graph Laplacian

Let's $G = (V, E)$ be a n-nodes simple and self-loofs undirected unweighted network.

4.3.1 Definition (Consensus)**.** In a network of coupled dynamical systems, the consensus means reaching an agreement regarding a certain quantity of interest that depends on the states of all agents (Olfati-Saber et al., 2007).

A *consensus* occurs in a network of coupled dynamical systems when a common state is reached by all vertices. For solving a consensus problem, we must to represent each node by an oscillator and now the consensus can be implemented by synchronization (Chen et al., 2009).

For simplify of presentation we have assumed that all agents are in one-dimensional space state, e.i., $x(t) \in \mathbb{R}$, $\forall i = 1, ..., n$. However, all results hereafter are still valid for high-dimensional space state by introducing of the *Kronecker product*.

The basic idea of consensus is that each agent updates its own state based on the states of its direct neighbors. And at the end of the process, all agents reach a common value (Kocarev, 2013).

Let's G be a distributed system of agents interconnected. Each agent i has an information state $x_i(t)$ that changes over time. We're interested in the follow dynamic that allow all agents to reach a consensus regarding their states:

$$\dot{x}_i(t) = f_i(t) \tag{4.3.1}$$

Where $\dot{x}_i(t)$, $f_i(t) \in \mathbb{R}$ and $\dot{x}_i(t)$ represents a dynamic of vertex i.

Telling that, each agent reach a common state $(\bar{x})$ stands for:

$$x_1 = x_2 = ... = x_n = \bar{x}, \text{ where } f(\bar{x}) = 0 \tag{4.3.2}$$

Where the common state is interpreting as an equilibrium state of the system.

The set of direct neighbors of vertex i is described by.

$$N_i = \{j \in V | a_{ij} \neq 0\} \tag{4.3.3}$$

The Equation 4.3.3 stands for, agent i accounts for the state of agent j if there is a link from j to i. At each time, node i carries out its update based on its *local interaction* with its *direct neighbors*.

And in this case, the continuous evolution rule for the state of each vertex i is defined by the following equation (Estrada, 2012a):

$$\dot{x}_i(t) = \sum_{j \in N_i} a_{ij}[x_j(t) - x_i(t)] \tag{4.3.4}$$

Where $A = (a_{ij})_{n \times n}$ denotes the outer-coupling between the nodes of the whole network. If node i can obtain information from node j, then $a_{ij} = 1$; otherwise, $a_{ij} = 0$. Such as $|x_j(t) - x_i(t)| \longrightarrow 0$ as $t \longrightarrow 0$.

The linear system represented by Equation 4.3.4 is the distributed *consensus algorithm* proposed in (Olfati-Saber and Murray, 2004). This system converges to a common state via local interactions. Assuming that the network is undirected, the sum of the states of all vertices is invariant and the consensus is reached asymptotically, thus the collective decision α is equal to the average of the initial states of all vertices, that is:

$$\alpha = \frac{1}{n} \sum_{i=1}^{n} x(0) \tag{4.3.5}$$

The Consensus algorithms with such a property of invariant sum of the states of the vertices are called *average-consensus algorithms* (Saber and Murray, 2003).

4.3.2 Proposition. The continuous evolution rule defined by 4.3.4 $\dot{x}_i(t) = \sum_{j \in N_i} a_{ij}[x_j(t) - x_i(t)]$ can be written as:

$$\dot{x} = -Lx \tag{4.3.6}$$

with $L = D - A$ where L, D and A are respectively Laplacian, Degree matrix and adjacency of G.

let's x is the vector containing the states of all nodes, such that, by simple notation, $\dot{x}(t) = x$

$$\begin{aligned}
\dot{x}_i(t) &= \sum_{j \in N_j} a_{ij}[x_j(t) - x_i(t)] \\
&= [\sum_{j \in N_j} a_{ij}.x_j(t) - x_i(t) \sum_{j \in N_j} a_{ij}] \\
&= [\sum_{j \in N_j} a_{ij}x_j(t) - k_i.x_i(t)] \\
&= [\sum_{j \in N_j} a_{ij} - k_i.\delta_{ij}]x_j(t) \\
&= -[k_i.\delta_{ij} - \sum_{j \in N_j} a_{ij}]x_j(t) \\
&= -\sum_{j \in N_j} l_{ij}x_j(t)
\end{aligned}$$

We note the dynamical system by:

$$\dot{x} = -Lx \tag{4.3.7}$$

where L is symmetric when the network is undirected and asymmetric when the network is directed and we call L *Laplacian* because of the *heat equation*:

$$\dot{x} + k\Delta x = 0 \tag{4.3.8}$$

where $k > 0$. The Equation 4.3.7 and 4.3.8 are the same when $k = 1$. That's why we call it *Laplacian*.

4.3.3 Proposition (Consensus in Discrete-time)**.** The Consensus in Discrete-time is given by the Equation 4.3.9

$$x(t+1) = Px(t) \tag{4.3.9}$$

The discrete-time version of equation 4.3.4 has the form:

$$\frac{x(t+1)-x(t)}{\epsilon} = \sum_{j\in N_i} a_{ij}[x_j(t)-x_i(t)]$$

$$\begin{aligned} x_i(t+1) &= x_i(t) + \epsilon \sum_{j\in N_i} a_{ij}[x_j(t)-x_i(t)] \\ &= x_i(t) - \epsilon \sum_{j\in N_i} l_{ij}x_j(t) \end{aligned}$$

From 4.3.7 we have:

$$\frac{x(t+1)-x(t)}{\epsilon} = -Lx(t)$$

$$\begin{aligned} x(t+1) &= x(t) - \epsilon Lx(t) \\ &= (I-\epsilon L)x(t) \end{aligned}$$

$$x(t+1) = Px(t) \tag{4.3.10}$$

where $P = (I-\epsilon L)$ (Estrada, 2012a), $\epsilon > 0$ the step-size and I is the $n\times n$ identity matrix.

One of the approaches to achieve a consensus in a dynamical complex network is called *pinning control*.

4.4 Synchronization of complex networks via pinning control

4.4.1 Definition (Pinning control)**.** The *pinning control* is a feedback control strategy for synchronization and consensus of a complex dynamical network. Where we add a virtual node (s) called *leader* or *pinner* and its desired trajectory (Grigoriev et al., 1997).

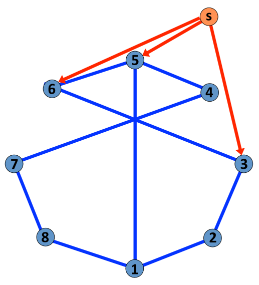


Figure 4.2: The pinner (the red node) directly controls only a subset of the network nodes (blue nodes). The control action propagates to the rest of the network by means of the interconnections (the blue lines) among the network nodes (internet, c).

We must to look at the convergence of *pinning control*. The sufficient conditions to guarantee the convergence of *complex dynamical network* locally and globally by pinning only one *node* (Chen et al., 2007).

Let's G be a complex network of n identical linearly and diffusively coupled oscillators in one dimensional space state. The continuous time evolution 4.3.4 becomes:

$$\begin{aligned}\dot{x}_i(t) &= f(x_i(t)) - \sigma \sum_{j=1}^{n} a_{ij} H[x_j(t) - x_i(t)] \\ &= f(x_i(t)) - \sigma \sum_{j=1}^{n} l_{ij} H[x_j(t)]\end{aligned}$$

$$\dot{x}_i(t) = f(x_i(t)) - \sigma \sum_{j=1}^{n} l_{ij} H[x_j(t)] \tag{4.4.1}$$

where $i = 1, ..., n$ and (Li et al., 2005):

- $x_i \in \mathbb{R}$ is the state of node i;
- $\sigma > 0$ represents the coupling strength. It represents the weight of connection between two nodes. $\sigma_{ij} > 0$ if $(i, j) \in E$ otherwise 0;
- With $t \in [0, \infty[$;
- H indicates inner-coupling between the elements of the node itself. so $H(x) = x$.
- $L = (l_{ij})_{n \times n}$ is the *Laplacian* matrix, in other word the outer coupling matrix linking coupled variables.

In this project, we simplify the task, by using an *unweighted* graph (Adjacency matrix instead of weight matrix). Because it will be complex to explore in each time a *weighted* graph due to the evolving of $\sigma_{ij}(t)$. So, we set $\sigma_{ij}(t) = 1$ for all edges (i, j).

Control a network stands for drive or force the states associated to its vertices in an homogeneous state called desired trajectory state or target state.

The goal of *pinning control* is to select a part of the vertices from network 4.4.1 to control, such that the states of all the vertices in the network can be synchronize to the target state. By applying the pinning control strategy in the network 4.4.1 we have this Equation 4.4.2 (Cupertino et al., 2013):

$$\dot{x}_i(t) = f(x_i(t)) - \sigma \sum_{j=1}^{n} l_{ij} H[x_j(t)] + u_i(t) \tag{4.4.2}$$

The network 4.4.2 is called a *controlled network* of n dynamical vertices.

where $u_i(t) = g_i\{H[s(t) - x_i(t)]\}$ is the local feedback controller, g_i quantifies the pinning control gain for vertex i, e.i., $g_i \neq 0$ if a vertex is pinned and if not $g_i = 0$ and $s(t)$ is a target state of the network which satisfies:

$$\dot{s}(t) = f(s(t)), \; s(0) = s_0 \tag{4.4.3}$$

The *target state* $s(t)$ is an *equilibrium state* of the node system.

In this approach, the task is to drive the dynamical complex network to $s(t)$ as $t \to \infty$ by pinning some vertices.

The problem of *pinning control* for synchronization of network 4.4.1 is achieved when:

$$\lim_{t\to\infty} |s(t) - x_i(t)| = 0, \ \forall i. \tag{4.4.4}$$

4.4.2 Theorem (Discrete-time for pinning control)**.** *From the Equation 4.4.2 the pinning control can be reduced to the consensus problem in the presence of pinned vertices with a reference fixed trajectory $s(t) = \bar{x}$. Such as $f[x_i(t)] = \beta x_i(t)$ with $\beta > 0$ and H the internal coupling function $H(x) = x$. The discrete-time equation for pinning control is (Cupertino et al., 2013):*

$$x_i(t+1) = \alpha x_i(t) - \epsilon \sum_{j=1}^{N} l_{ij} x_j(t) + \epsilon u_i(t) \tag{4.4.5}$$

where $u_i(t) = g_i[\bar{x} - x_i(t)]$ and $\epsilon > 0$ is the step size. For the controlled vertex, $g_i = z > 0$ is the control gain and for all other vertices $g_i = 0$. $\bar{x} = 0$, $g_1 = z > 0$ and $g = 0$ for other vertices. Therefore, $u_i(t) = -zx_(t)$.

By considering the *discrete-time* form, we must assure that the convergence is achieved within a time $t < \infty$. One of the advantages of using *consensus clustering* is that, it leads to a *stable partitions*.

The matrix associated to the Equation 4.4.5 is given by:

$$A = \begin{bmatrix} \alpha - \epsilon(l_{11} + z) & \epsilon l_{12} & \dots & \epsilon l_{1n} \\ \epsilon l_{21} & \alpha - \epsilon l_{22} & \dots & \epsilon l_{2n} \\ \vdots & \vdots & & \vdots \\ \epsilon l_{n1} & \epsilon l_{n2} & \dots & \alpha - \epsilon l_{nn} \end{bmatrix} \tag{4.4.6}$$

such as:

$$X(t+1) = AX \tag{4.4.7}$$

The discussion is about the stability of the system 4.4.5, based on our *equilibrium point* (target state). Many researchers use *Lyapunov function* of the system in order to determine whether an equilibrium point is stable or not. In this project, we use the approach in (Cupertino et al., 2013). The main idea is to show that the system is stable if and only if $\rho(A) < 1$ where $\rho(A)$ is the *spectral radius* of A. Since we know the entries of A, we can confine them by using the parameters of the system, namely: ϵ, α and z. The interval values of these parameters are provided in the theorem 4.4.3.

4.4.3 Theorem (Convergence of pinned discrete-time system)**.** *Consider a pinned discrete-time system with n coupled elements (in which only a single element is pinned) whose dynamics are governed by Equation (4.4.5). The origin of the system is asymptotically stable for any initial condition $x(0)$ if and only if, the following constraints are verified (Cupertino et al., 2013):*

1. $0 < \epsilon < \dfrac{C}{|W_{max}|}$;
2. $\epsilon l_{max} - C < \alpha < \epsilon l_{min} + C$
3. $\dfrac{\alpha}{\epsilon} l_{min} - \dfrac{C}{\epsilon} < z < \dfrac{\alpha}{\epsilon} l_{max} + \dfrac{C}{\epsilon}$

where W_{max} is the maximum edge weight (maximum value in the weight matrix), l_{min} and l_{max} are the smallest and the largest vertex degrees, respectively. According to (Zhan, 2005):

$$C = \begin{cases} \dfrac{\sqrt{2}}{n} & \text{if } n \text{ is even} \\ \dfrac{2}{\sqrt{2n^2 - 1}} & \text{if } n \text{ is odd} \end{cases} \tag{4.4.8}$$

Where n is the number of vertices in the network. Proof in (Cupertino et al., 2013).

4.4.4 Remark.

- ϵ is the responsible for the *speed of convergence*. A greater value of ϵ will imply the faster convergence of the system. If the system converges too fast, all dissimilarities between pairs of vertices may be very similar. In this case, the consensus measure will be inappropriate for clustering, because in the clustering task one expects low dissimilarity between close vertices and high dissimilarity between far vertices (Cupertino et al., 2013).
- α points in the same direction as ϵ.

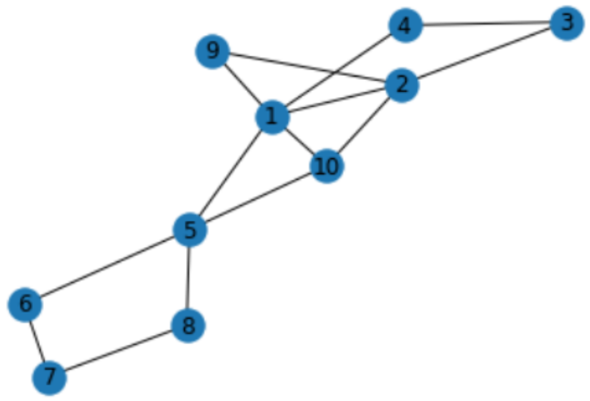
For having a high clustering accuracy, we must set ϵ and α to a small values.

4.4.5 Definition (Dissimilarity measure from consensus time)**.**

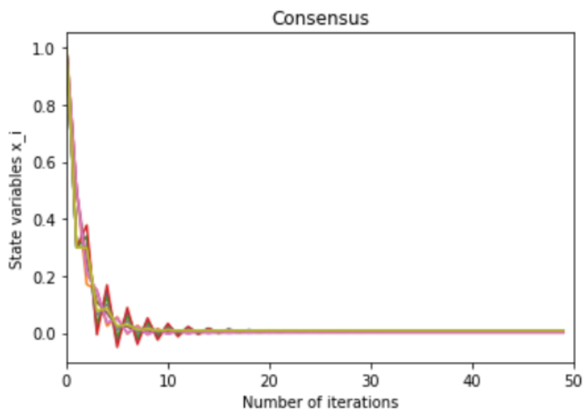
Remind that, in a dynamical complex network, the *consensus* occurs in the presence of a *pinned* vertex i with an arbitrary fixed trajectory, $x_i = \bar{x}$ when each j vertex of the network reach $x_i = \bar{x}$, and we call *consensus time* the total amount of time that takes a vertex j to reach the $x_i = \bar{x}$. This total amount of time is not necessary the same every nodes. In this case a *dissimilarity measure* among vertices on a network can be directly derived from the consensus time concept. In other words, when we've a discrete system, *dissimilarity measure* is the number of time steps d_{ji} for all nodes (Cupertino et al., 2013).

4.4.6 Example (Illustrative example of consensus and calculation of dissimilarity matrix)**.**

Given a network of 8 vertices.The task is to drive the network in the figure 5.1a in the desired state $\bar{x}$ at the different and finite times by pinning a single vertex due to the equation 4.4.5. In our case we pinn the vertex 1. So we set, $x_1(0) = \bar{x} = 0$ and $g_1 = z > 0$. All other vertices $x_2(0) = x_3(0) = x_4(0) = x_5(0) = x_6(0) = x_7(0) = x_8(0) = x_9(0) = x_{10} = 1$. And $\alpha = 0.5$, $\epsilon = z = 0.2$ and $e = 10^{-3}$ as a discretization error.



(a) Small network consisting of 10 vertices to illustrate the consensus and the calculation of dissimilarity.



(b) State evolution of the network in (a). All vertices reach a consensus in the presence of a pinned vertex $x_1 = \bar{x} = 0$

Figure 4.3: Network and convergence of each node

The calculation of the dissimilarity matrix between the nodes is done by pinning each node and you calculate the total amount of times that take another nodes to reach the state of the pinned node. And at the end of the process, we will have the dissimilarity matrix non-symmetric by the fact that if a node i

is controlling, suppose that a node j is not pinned and if j takes the time t_j for achieving consensus (the state of node i). And in turn when controlling j, i can take t_i to reach consensus, usually t_i different to t_j. This dissimilarity will be none symmetric and we need to make this matrix symmetric, which will be discussed in the next subsection. The dissimilarity matrix is associated to the network 4.3b(a):

$$D = \begin{pmatrix} 0 & 19 & 18 & 19 & 19 & 18 & 14 & 18 & 18 & 9 \\ 18 & 0 & 15 & 17 & 19 & 15 & 10 & 15 & 14 & 9 \\ 18 & 15 & 0 & 12 & 15 & 10 & 9 & 10 & 10 & 10 \\ 19 & 17 & 12 & 0 & 18 & 13 & 10 & 13 & 11 & 10 \\ 19 & 19 & 16 & 18 & 0 & 15 & 12 & 15 & 15 & 9 \\ 19 & 13 & 10 & 12 & 15 & 0 & 9 & 9 & 10 & 10 \\ 10 & 11 & 9 & 9 & 12 & 9 & 0 & 9 & 10 & 9 \\ 19 & 13 & 10 & 12 & 15 & 9 & 9 & 0 & 10 & 10 \\ 17 & 15 & 10 & 12 & 15 & 10 & 10 & 10 & 0 & 9 \\ 10 & 9 & 10 & 10 & 9 & 10 & 9 & 10 & 9 & 0 \end{pmatrix}$$

As we see, this matrix is not symmetric.

4.5 Data clustering via consensus-time

The next step after the network construction by the algorithm proposed in the section 3.3.1 is cluster (or community) detection. And we use this network for the calculation of consensus time dissimilarity (see definition 4.4.5). The basic characterization of clusters is that, for the same cluster, it takes vertices which reach a common state in a similar time, we can see the value of time in matrix D (Cupertino et al., 2013). Knowing that the dissimilarity matrix D provided by consensus time is not *symmetric*, so D must be transformed into a symmetric matrix because most of the clustering algorithm require a symmetric matrix. A simple approach to do that is to take the average dissimilarity between dij and d_{ji}. Therefore, the symmetric matrix associated to D, that we note D_s is given by:

$$D_S = (D + D^T)/2 \tag{4.5.1}$$

After this, we perform *hierarchical clustering* on D_s and after we detect the clusters.

We summarize the steps of clusters detection by using the proposed dissimilarity measure are described as follows (Cupertino et al., 2013):

1. Calculate the dissimilarity matrix D as described in definition (4.4.5);
2. Transform D into D_s, according to Eq.(4.5.1);
3. Applying hierarchical clustering on D_S;
4. Choose the partition from the dendrogram.

One of the advantages of using *consensus clustering* is that, it leads to a stable partitions (Kwak et al., 2009).

5. Experimental results

In this chapter, we present the simulations in some real datasets in term to experiment this new approach of network-based clustering. The table 5.1 provide the information about the datasets that we use. From these datasets we know in advance the number of clusters and all of them are labeled (Frank et al., 2011).

Dataset	number of observations	number of attributes	number of classes
Iris	150	4	3
Wine	178	13	3

Table 5.1: *Information about real datasets used for simulation from these datasets*

During the experimentation, we set $x_1(0) = \bar{x} = 0$ and $g_1 = z > 0$. All other vertices $x_2(0) = x_3(0) = x_4(0) = x_5(0) = x_6(0) = x_7(0) = ... = x_n(0) = 1$, where n equal to the number of observations of datasets. And $\alpha = 0.05$, $\epsilon = 0.002 = z = 0.2$ and $e = 10^{-3}$. For the network construction we set $\gamma = 3$ for each dataset and k is varying in $[1, 15]$. The table 5.2 represents the result of the experiments of this new approach and we use *adjusted rand index ARI* for clustering evaluation and *rand index RI*:

Dataset	Consensus + WL	Consensus + AL	Consensus + CL	Consensus + SL
Iris	$0.94(3)$	$0.96\ (3)$	$0.93\ (2)$	$0.15\ (2)$
Wine	$0.57\ (5)$	$0.59\ (5)$	$0.59\ (2)$	$0.19\ (3)$

Table 5.2: *Models evaluation . The value in bracket is the value of k*

The Table 5.3 summarizes the results of classical clustering on these two datasets:

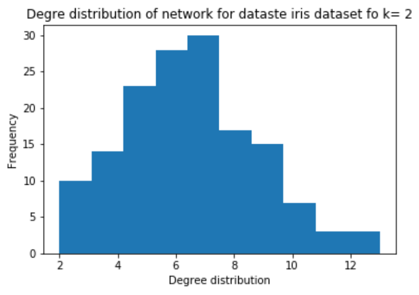
Dataset	K-means	Spectral clustering	HC +SL	HC+ AL	HC+CL	HC+ WL
Iris	0.84	0.33	0.36	0.81	0.71	0.81
Wine	0.37	0.25	0.056	0.22	0.08	0.01

Table 5.3: *Classical algorithm of clustering*

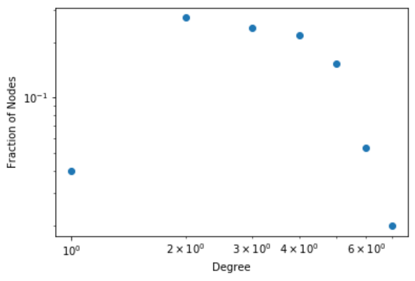
We present the characteristics of network, in Table 5.4

	K= 2	k= 3	k= 5	k= 15
Diameter	27	23	18	10
Average path length	11	8.8	6.73	4.08
transitivity (average clustering)	$0.33(0.45)$	$0.4(0.48)$	$0.48(0.54)$	$0.63(0.66)$
Density	0.022	0.30	0.04	0.1
Radius	14	12	9	5
Average degree	4.54	4.54	3.41	3.14

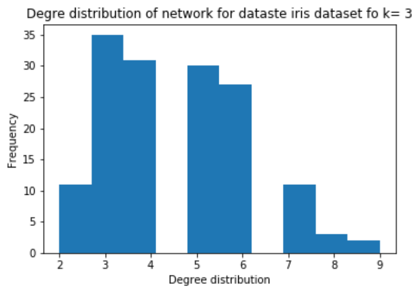
Table 5.4: *Dataset Iris network characteristics*



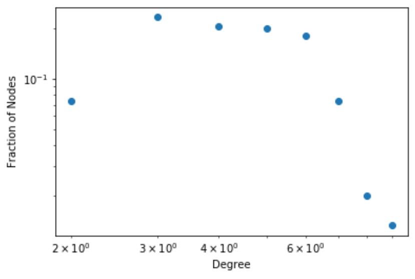
(a) Degree distribution for dataset iris network for $k = 2$



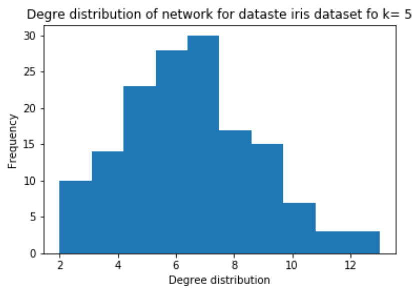
(b) log scale plot for $k = 3$



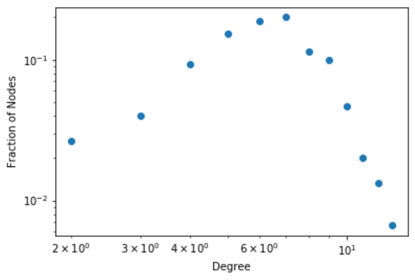
(c) Degree distribution for dataset iris network for $k = 3$



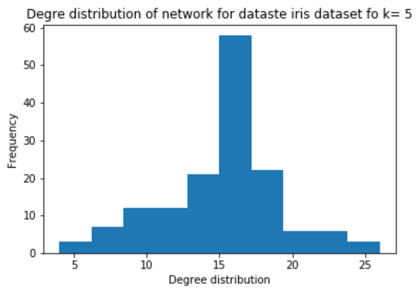
(d) log scale plot for $k = 3$



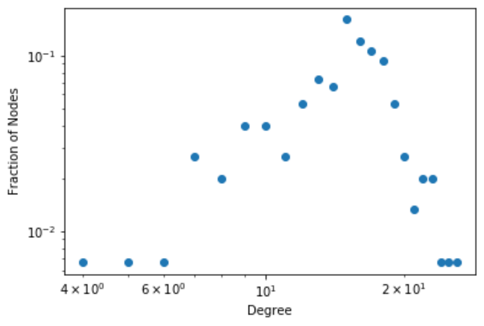
(e) Degree distribution for dataset iris network for $k = 5$



(f) log scale plot for $k = 5$



(g) Degree distribution for dataset iris network for $k = 15$



(h) log scale plot for $k = 15$

Figure 5.1: Network degree distribution and power law plotting

The networks constructed by K-NN with SL with some values of k seem either as scale free or small world. According to the experiments on Iris dataset, considering several values of k, we notice that the higher k is, the more the degree distribution of nodes tend to follows *binomial law*.

6. Conclusion and perspectives

In this work we have explored a new network-based clustering method proposed in (Cupertino et al., 2013). A method consisting of two main steps : (i) network construction and cluster detection. As we noticed, this new approach of network construction, provide a connected network and the sparsity of sub-networks, which help to detect communities in the network. The characteristics of connectedness and sparsity improve the clustering accuracy. (ii) then, we detect the cluster by using a dissimilarity measure dissimilarity which is, the total amount of time that takes a vertex to reach a consensus in the presence of a pinned vertex. A dissimilarity is constructed from the consensus time, and we perform the hierarchical clustering with it. The simulations on some real datasets showed that, the proposed method run well in the presence of clusters of different shapes and different size, and it is better than the classical method i.e.: K-means, Spectral clustering and hierarchical clustering.

The consensus dynamics considered is based on the interaction between nearest neighbors, only direct interactions are taken into account. On way to extend the mentioned dynamics is to consider also interactions between non connected nodes such that the force of interaction decrease as the distance between no connected node increases. We have to generalized the standard combinatorial Laplacian $L = D - A$ to a more general Laplacian to take into account long range interactions between nodes. This generalised Laplacian has been introduced by E. Estrada (Estrada, 2012a) and showed that consensus is reached faster when this generalized Laplacian is used. Our future will be to use that generalised Laplacian for the consensus equation and see if we can get bet clustering.

References

Rafig Agaev and Pavel Chebotarev. On the spectra of nonsymmetric laplacian matrices. *arXiv preprint math/0508176*, 2005.

Charu C Aggarwal and ChengXiang Zhai. A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer, 2012.

Doaa Mahmoud Al-Saafin. *Spectral Radius, Numerical Radius and Unitarily Invariant Norm Inequalities in Hilbert Space*. PhD thesis, Zarqa University, 2016.

Mohammed Zuhair Al-Taie and Seifedine Kadry. *Python for graph and network analysis*. Springer, 2017.

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

Vadim S Anishchenko, Tatiana Vadivasova, and Galina Strelkova. Deterministic nonlinear systems. *A Short Course. Switzerland: Springer International Publishing*, 294, 2014.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Jorma Boberg and Tapio Salakoski. Representative noise-free complete-link classification with application to protein structures. *Pattern recognition*, 30(3):467–482, 1997.

Stefano Boccaletti, Alexander N Pisarchik, Charo I Del Genio, and Andreas Amann. *Synchronization: from coupled systems to complex networks*. Cambridge University Press, 2018.

Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE transactions on knowledge and data engineering*, 20(2):172–188, 2007.

Morteza Haghir Chehreghani, Hassan Abolhassani, and Mostafa Haghir Chehreghani. Improving density-based methods for hierarchical clustering of web pages. *Data & Knowledge Engineering*, 67(1):30–50, 2008.

Fei Chen, Zengqiang Chen, Linying Xiang, Zhongxin Liu, and Zhuzhi Yuan. Reaching a consensus via pinning control. *Automatica*, 45(5):1215–1220, 2009.

Tianping Chen, Xiwei Liu, and Wenlian Lu. Pinning complex networks by a single controller. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(6):1317–1326, 2007.

Peter Csermely. Creative elements: network-based predictions of active centres in proteins and cellular and social networks. *Trends in biochemical sciences*, 33(12):569–576, 2008.

Thiago H Cupertino, Jean Huertas, and Liang Zhao. Data clustering using controlled consensus in complex networks. *Neurocomputing*, 118:132–140, 2013.

Dragos Cvetkovic, Dragoš M Cvetković, Peter Rowlinson, and Slobodan Simic. *Eigenspaces of graphs*. Number 66. Cambridge University Press, 1997.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

Ernesto Estrada. Path laplacian matrices: introduction and application to the analysis of consensus in networks. *Linear algebra and its applications*, 436(9):3373–3391, 2012a.

Ernesto Estrada. *The structure of complex networks: theory and applications*. Oxford University Press, 2012b.

Ernesto Estrada and Philip A Knight. *A first course in network theory*. Oxford University Press, USA, 2015.

Ernesto Estrada, Franck Kalala-Mutombo, and Alba Valverde-Colmeiro. Epidemic spreading in networks with nonrandom long-range interactions. *Physical Review E*, 84(3):036110, 2011.

Brian S Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. *Cluster analysis*. John Wiley & Sons, 2011.

Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

S Fortunato. Community detection in graphs/santo fortunato. *Physics Reports*, 2009.

Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

Andrew Frank, Arthur Asuncion, et al. Uci machine learning repository, 2010. *URL http://archive. ics. uci. edu/ml*, 15:22, 2011.

Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*, volume 20. Siam, 2007.

RO Grigoriev, MC Cross, and HG Schuster. Pinning control of spatiotemporal chaos. *Physical Review Letters*, 79(15):2795, 1997.

Frank J Hall. The adjacency matrix, standard laplacian, and normalized laplacian, and some eigenvalue interlacing results. *Department of Mathematics and Statistics, Georgia State University, Atlanta*, 16, 2010.

He Hao and Prabir Barooah. Improving convergence rate of distributed consensus through asymmetric weights. In *2012 American Control Conference (ACC)*, pages 787–792. IEEE, 2012.

AS Hornby. Oxford advanced learners dictionary of current english: Revised and updated, 1974.

Peg Howland and Haesun Park. Cluster-preserving dimension reduction methods for efficient classification of text data. In *Survey of Text Mining*, pages 3–23. Springer, 2004.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

internet. Bipartite. Wikipedia, the Free Encyclopedia, ://www.wikiwand.com/en/Bipartite$_graph, a$.

internet. Watt strogatz. Wikipedia, the Free Encyclopedia, ://www.researchgate.net/publication/264837172$_{R}ural-urban_{m}igration_{d}ecision_{m}aking_{p}rocesses_{A_w}hole_{a}nd_{p}erso$
$1, b$.

internet. Watt strogatz, c.

internet. Karate club. Wikipedia, the Free Encyclopedia, http://nexus.igraph.org/api/datasetinfo?id=1&format=html., Accessed April 2020d.

internet. Konigsberg. Wikipedia, the Free Encyclopedia, https://www.google.com/search?q=koningsberg+problem&tbm=isch&ved=2ahUKEwjJkPiBx-DpAhWLw4UKHcnwCK0Q2-cCegQIABAA&oq=koningsberg+problem&gs_lcp=CgNpbWcQAzoECAAQQzoCCAA6BAgAEBM6CAgAEAUQHhATOgYIABAeEBN
img&ei=JOzUXsnuFouHlwTJ4aPoCg&bih=900&biw=1215&client=ubuntu&gl=za&hs=xZp#imgrc=KTbbQfPyX4dfAM, Accessed April 2020e.

internet. Watt strogatz. Wikipedia, the Free Encyclopedia, https://www.google.com/search?q=internet+network&client=ubuntu&hs=xZp&channel=fs&gl=za&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiZsrLOueTMAhVMCcAKHS8AysQAUIBygB&biw=1215&bih=900#channel=fs&gl=za&tbm=isch&q=computer+network&imgrc=_, Accessed April 2020f.

internet. Watt strogatz. Wikipedia, the Free Encyclopedia, https://www.researchgate.net/figure/The-random-rewiring-procedure-of-the-Watts-Strogatz-fig2_264837172, Accessed April 2020g.

internet. Watt strogatz. Wikipedia, the Free Encyclopedia, https://www.wikiwand.com/en/Bipartite_graph, Accessed may 2020h.

internet. Watt strogatz. Wikipedia, the Free Encyclopedia, https://www.researchgate.net/figure/The-random-rewiring-procedure-of-the-Watts-Strogatz-fig2_264837172, Accessed may 2020i.

Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

Hansi Jiang and Carl Meyer. Relations between adjacency and modularity graph partitioning. *arXiv preprint arXiv:1505.03481*, 2015.

George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

Leonard Kaufmann and Peter J Rousseeuw. Finding groups in data: an introduction to cluster analysis. *New York: Jonh Wiley*, 1990.

Ljupco Kocarev. *Consensus and synchronization in complex networks*. Springer, 2013.

PM Krishnaraj, Ankith Mohan, and KG Srinivasa. *Practical Social Network Analysis with Python*. Springer, 2018.

Marzena Kryszkiewicz and Łukasz Skonieczny. Faster clustering with dbscan. In *Intelligent Information Processing and Web Mining*, pages 605–614. Springer, 2005.

Yoshiki Kuramoto. Diffusion-induced chaos in reaction systems. *Progress of Theoretical Physics Supplement*, 64:346–367, 1978.

Haewoon Kwak, Young-Ho Eom, Yoonchan Choi, Hawoong Jeong, and Sue Moon. Consistent community identification in complex networks. *arXiv preprint arXiv:0910.1508*, 2009.

Changpin Li, Weigang Sun, and Daolin Xu. Synchronization of complex dynamical networks with nonlinear inner-coupling functions and time delays. *Progress of Theoretical Physics*, 114(4):749–761, 2005.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Yu-Ching Lu and Goutam Chakraborty. Definition and goal of graph clustering-motivation to explore a new algorithm. In *2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST)*, pages 1–6. IEEE, 2019.

Spiros Mancoridis, Brian S Mitchell, Chris Rorres, Y Chen, and Emden R Gansner. Using automatic clustering to produce high-level system organizations of source code. In *Proceedings. 6th International Workshop on Program Comprehension. IWPC'98 (Cat. No. 98TB100242)*, pages 45–52. IEEE, 1998.

M Newman. Networks: an introduction: Oup oxford. 2010.

Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

Reza Olfati-Saber and Richard M Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, 49(9):1520–1533, 2004.

Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social networks*, 31(2):155–163, 2009.

Jorge Orozco and Lluıs Belanche. Towards a mathematical framework for similarity and dissimilarity. Technical report, Technical Report, University of Sevilla, 2005.

Julio-Omar Palacio-Niño and Fernando Berzal. Evaluation metrics for unsupervised learning algorithms. *arXiv preprint arXiv:1905.05667*, 2019.

Xingqin Qi, Eddie Fuller, Qin Wu, Yezhou Wu, and Cun-Quan Zhang. Laplacian centrality: A new centrality measure for weighted networks. *Information Sciences*, 194:240–253, 2012.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

P Krishna Reddy, Masaru Kitsuregawa, P Sreekanth, and S Srinivasa Rao. A graph based approach to extract a neighborhood customer community for collaborative filtering. In *International Workshop on Databases in Networked Information Systems*, pages 188–200. Springer, 2002.

Bradley Stuart Rees. *Ego-Based Overlapping CommunitiesDetection: A New Paradigm*. Florida Institute of Technology, 2015.

Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

Reza Olfati Saber and Richard M Murray. Consensus protocols for networks of dynamic agents. 2003.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

Thiago Christiano Silva and Liang Zhao. *Machine learning in complex networks*, volume 2016. Springer, 2016.

Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4): 585–591, 1997.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Li Wan, Jianxin Liao, and Xiaomin Zhu. Cdpm: Finding and evaluating community structure in social networks. In *International Conference on Advanced Data Mining and Applications*, pages 620–627. Springer, 2008.

Xiao Fan Wang and Guanrong Chen. Complex networks: small-world, scale-free and beyond. *IEEE circuits and systems magazine*, 3(1):6–20, 2003.

Wei Xing, Qingling Zhang, and Qiyi Wang. A trace bound for a general square matrix product. *IEEE Transactions on Automatic Control*, 45(8):1563–1569, 2000.

Xingzhi Zhan. Extremal eigenvalues of real symmetric matrices with entries in an interval. *SIAM journal on matrix analysis and applications*, 27(3):851–860, 2005.

Baodong Zheng and Liancheng Wang. Spectral radius and infinity norm of matrices. *Journal of mathematical analysis and applications*, 346(1):243–250, 2008.

G Zielke. Horn, ra; johnson, cr, matrix analysis. cambridge etc., cambridge university press 1985. xiii, 561 s.,£ 35.00. isbn 0-521-30586-1. *Zeitschrift Angewandte Mathematik und Mechanik*, 67:212–212, 1987.