Trabajo Fin de Grado

Mathematical Optimization and Social Networks

Presented by:
Lourdes Díaz Mena

Supervisors:
Dr. Rafael Blanquero Bravo, Universidad de Sevilla
Dr. Emilio Carrizosa Priego, Universidad de Sevilla

UNIVERSIDAD D SEVILLA

June 22, 2015

Contents

Chapter 1

Introduction

In recent years, social networks have become a vital part of the everyday life of most people. In 2014, 73% of the Spanish population actively used social networks, Facebook being the most used. Besides the important role they play in the relations between people, social networks can be used as a free showcase to advertise any product or service. Hence, companies use them more every day as a mean to publicize their products. On the other hand, social networks allow any company to extract data about how its customers are, to study what they like, and thus improve sales. Therefore, the analysis of social networks is currently extremely important, as much from the sociological viewpoint, as economic and political. Being aware of its importance, we have used in this work graph theory, seeing a social network as a set of nodes and their relationships.

The work is structured in four more chapters. In Chapter 2 we define some concepts that we need along the work. In addition, we study the shortest paths problem between nodes using three different methods. In Chapter 3 we introduce some probabilistic models for social networks. These models are the Erdös-Rényi-Gilbert model, the Exponential model and the Barabási-Albert model, which are used to generate networks. In Chapter 4 we study some centrality concepts for social networks like degree centrality or closeness centrality. These measures of centrality allow us to know the relative importance of a node in a graph. Finally, in Chapter 5 we introduce some concepts of association like k-clique or k-club, as well as integer programming formulations to find the maximum according to these concepts in a graph.

Chapter 2

Basic concepts

Definition 2.1 *A graph is a pair* $G \equiv G(N, E)$ *where* N *is a finite set whose elements are called nodes and* E *is a subset of* $N \times N$ *whose elements are called edges.*

Definition 2.2 *A network or valuated graph is a graph where each edge* $(i, j) \in E$ *has associated a weight* $c_{ij} \in \mathbb{R}$.

We are going to study the concept of graph, so for each edge $(i, j) \in E$ $c_{ij} = 1$.

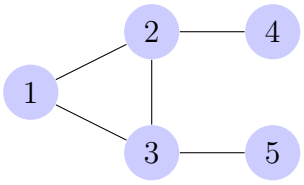


Figure 2.1: Graph T.

Definition 2.3 *The adjacency matrix* A *of a graph* G *with* n *nodes is an* $n \times n$ *matrix where:*

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

For instance, for the graph T given in Figure 2.1, we obtain the adjacency matrix A:

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

Different types of graph exits. A directed graph is a graph where each edge is directed from one node to another. For example, Twitter is a directed network, since you can follow another user which does not follow you. In contrast, a graph where the edges are bidirectional is called undirected graph. For example, Facebook is a undirected network, because if you are friend to another user, the user is friend to you too.

While the terminology of network, nodes and edges is standard in Statistics and Operations Research, in Social Sciences the corresponding terminology is *actors* for nodes and *ties* for edges.

In what follows, we are going to consider G as an undirected graph with a set of nodes N, a set of edges E, A its adjacency matrix and $n = |N|$.

Definition 2.4 *A graph G is complete if for every pairs of nodes $i, j \in N$ one has $(i, j) \in E$.*

Definition 2.5 *A path in a graph is a sequence of edges which connects a sequence of nodes.*

Definition 2.6 *A connected component of an undirected graph is a subgraph in which any two nodes are connected by a path.*

As an illustration, in Figure 2.2 one can see a graph with three connected components.

Definition 2.7 *The distance between two nodes i and j in a graph G "$d_G(i, j)$" is the length of a shortest path between them. This is also known as the geodesic distance.*

For example, the distance matrix of the graph T (Figure 2.1) is the following one:

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

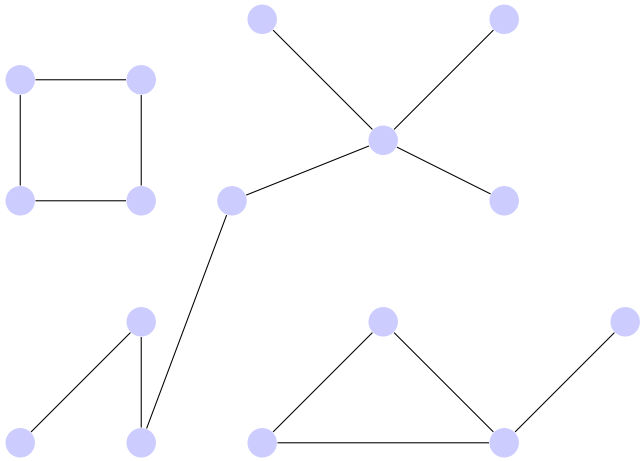


Figure 2.2: Graph formed by three connected components

Definition 2.8 *The diameter of a graph G is the maximum distance between any two nodes in the graph:*

$$diam(G) = \max_{i,j \in N} d_G(i,j)$$

For example, in graph T in Figure 2.1 one has $diam(T) = 2$.

2.1 Shortest paths

We are going to study three methods to find the shortest path between two nodes in the graph G, analyzed in the following subsections.

2.1.1 Linear program

Given two nodes s and t, our aim is to find the shortest path P between them. The formulation considers originally directed graphs, so we define $E' = \{(i,j) : (j,i) \in E\}$, and, for each $(i,j) \in E \cup E'$, we consider the binary variables x_{ij}:

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ belongs to } P \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Then the shortest path problem can be formulated as a linear program as follows, [1]:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E\cup E'} x_{ij} \\
\text{subject to} \quad & \sum_{(s,j)\in E\cup E'} x_{sj} - \sum_{(j,s)\in E\cup E'} x_{js} = 1 \\
& \sum_{(t,j)\in E\cup E'} x_{tj} - \sum_{(j,t)\in E\cup E'} x_{jt} = -1 \qquad (2.2) \\
& \sum_{(i,j)\in E\cup E'} x_{ij} - \sum_{(j,i)\in E\cup E'} x_{ji} = 0 \qquad \forall i \in N:\ i \neq s,t \\
& x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \cup E'
\end{aligned}
$$

Since the matrix of constraints is totally unimodular, i.e. a matrix for which every square non-singular submatrix is a square integer matrix having determinant +1 or 1 (see [2]), the integrality constraints can be replaced by their continuous relaxations $0 \leq x_{ij} \leq 1\ \forall (i,j) \in E \cup E'$.

As an illustration, we are going to compute the shortest path between nodes 3 and 9 in the graph S (Figure 2.3).

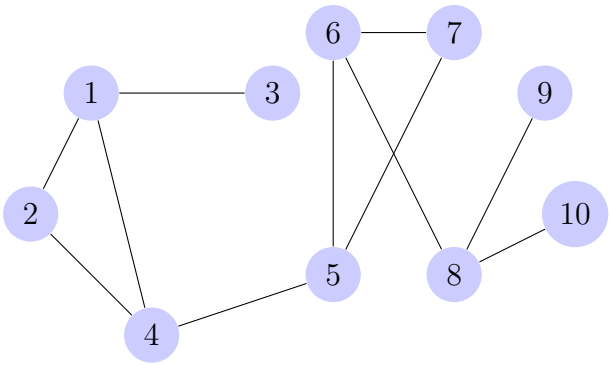


Figure 2.3: Graph S.

This problem is written in `AMPL`, a modeling language for mathematical optimization (see [3]), as shown in Table 2.1.

```
param n;

param s;

param t;

set nodes:= 1..n;

set Adjacent within {n1 in nodes, n2 in nodes:  n1<>n2};

var x{nodes, nodes}>=0,<=1;

minimize f:  sum{(i,j) in Adjacent} x[i,j];

subject to r1:

 sum{(s,j) in Adjacent} x[s,j]- sum{(j,s) in Adjacent} x[j,s]=1;

subject to r2:

 sum{(t,j) in Adjacent} x[t,j]- sum{(j,t) in Adjacent} x[j,t]=-1;

subject to r3{i in nodes:  i<>s and i<>t}:

 sum{(i,j) in Adjacent} x[i,j] -sum{(j,i) in Adjacent} x[j,i]=0;
```

Table 2.1: Code of shortest path problem between nodes s and t.

The optimal solution for this problem provided by the solver `Gurobi` ([4]) is shown in Table 2.2.

2.1.2 Dijkstra Algorithm

This algorithm allows us to obtain the shortest path between a node s and the other nodes in the graph. All edges in G must have nonnegative weights c_{ij} (for us $c_{ij} = 1\ \forall i, j$) and the graph must be connected.

We assign a label $[d_j, p_j]$ for every node j, where d_j is the known distance between s and j at the moment and p_j is the predecessor node of j in the shortest path between s and j. These labels can be temporary or permanent.

$$x_{ij} = \begin{cases} 1 & if\ (i,j) = (3,1),(1,4),(4,5),(5,6),(6,8),(8,9) \\ 0 & otherwise \end{cases}$$

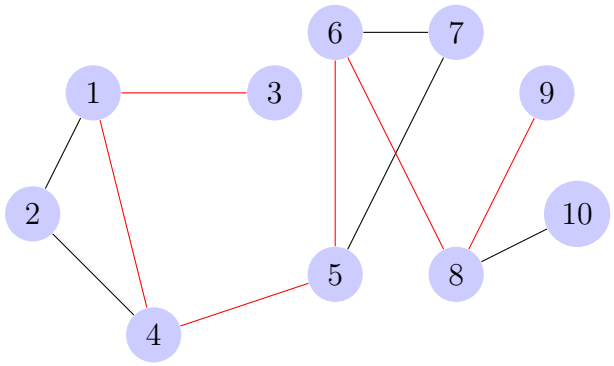


Table 2.2: Result.

A node with a permanent label is an explored node.

The pseudocode of Dijkstra algorithm is shown in Table 2.3 (see [2]).

1. Initiation:
A. Assign the label [0,-] to s and mark it as an explored node.
B. Assign the label [∞,-] to all other nodes.

2. Update the labels:
A. Let m be the last explored node.
B. $\forall (m,j) \in E$ with j unexplored:
if $t_j = d_m + c_{mj} < d_j$, assign the label $[t_j, m]$ to j.

3. Explore the nodes:
A. Find the unexplored node with less d_j and mark it as an explored node.

4. Exit test:
A. If every node has been explored: END.
B. Otherwise: go to step 2.

Table 2.3: Dijkstra algorithm.

We apply the Dijkstra algorithm to the graph S (Figure 2.3) to obtain the

shortest paths between the node 3 and all other nodes, see Table 2.4. We can observe, for example, that the distance between nodes 3 and 9 is 6 and the shortest path between them is $P = \{(3,1),(1,4),(4,5),(5,6),(6,8),(8,9)\}$, as we have just obtained with the linear program.

2.1.3 Floyd Algorithm

This algorithm allows us to obtain the shortest path between each pair of nodes in G.

In each iteration k, we determine the matrices $D^k = ((d_{ij}^k))$ and $P^k = ((p_{ij}^k))$ where:

- d_{ij}^k: length of the shortest path between i and j through only the k first nodes.
- p_{ij}^k: predecessor node of j in the shortest path between i and j through only the k first nodes.

Finally, $D = D^n$ is the distance matrix of G, and $P = P^n$ is the matrix with predecessor nodes in the shortest paths.

The pseudocode of the Floyd algorithm is shown in Table 2.5 (see [2]).

1. Initiation:

A. $\forall i, j = 1, .., n$:

$$d_{ij}^0 = \begin{cases} 0, & \text{if } i = j \\ c_{ij}, & \text{if } (i,j) \in E \\ \infty, & \text{otherwise} \end{cases}$$

$$p_{ij}^0 = \begin{cases} i, & \text{if } (i,j) \in E \\ -, & \text{otherwise} \end{cases}$$

B. $k = 0$

2. Main loop:

A. If $d_{ij}^k < 0$ for any (i, j): STOP (there is a cycle with negative longitude.)

B. $k = k + 1$. If $k > n$, END.

C. $\forall i, j = 1, .., n$:

$$\begin{cases} d_{ij}^k = d_{ik}^{k-1} + d_{kj}^{k-1} \text{ and } p_{ij}^k = k & \text{if } d_{ik}^{k-1} + d_{kj}^{k-1} < d_{ij}^{k-1} \\ d_{ij}^k = d_{ij}^{k-1} \text{ and } p_{ij}^k = p_{ij}^{k-1} & \text{otherwise} \end{cases}$$

Table 2.5: Floyd algorithm.

We apply this algorithm to the graph T (Figure 2.1), see Table 2.6. We can observe that D^5 is the distance matrix of T and, for example, the shortest path between the nodes 4 and 5 is $S = \{(4, 2), (2, 3), (3, 5)\}$.

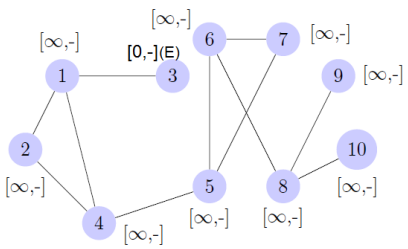
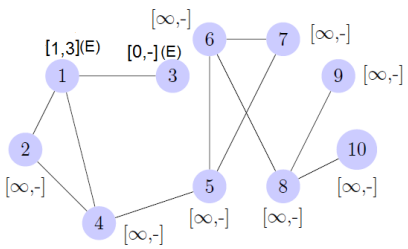
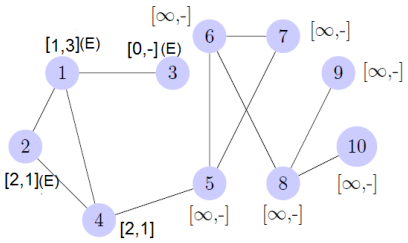
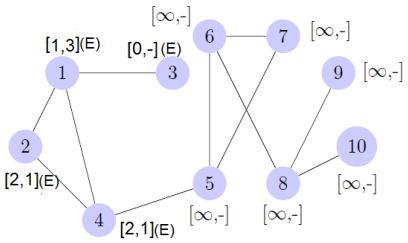
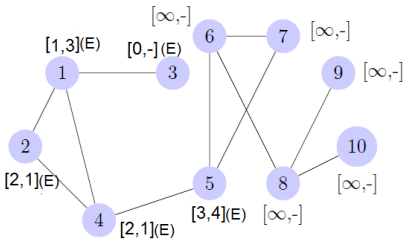
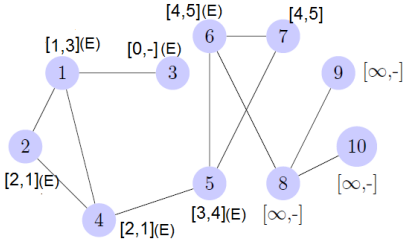
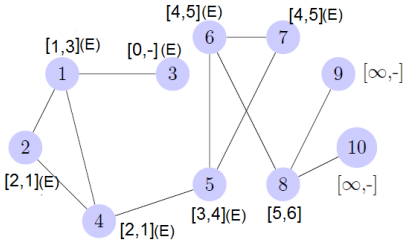
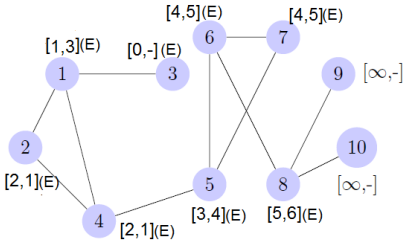
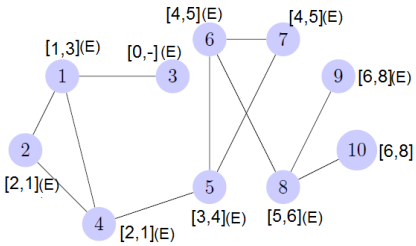
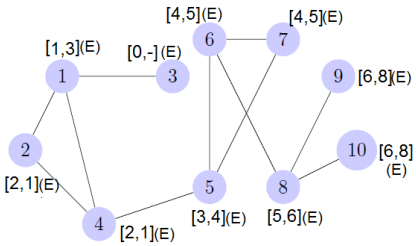
	
	$t_1 = 1 < \infty = d_1$
	
$t_2 = 2 < \infty = d_2$, $t_4 = 2 < \infty = d_4$	$t_4 = 3 > 2 = d_4$
	
$t_5 = 3 < \infty = d_5$	$t_6 = 4 < \infty = d_6$, $t_7 = 4 < \infty = d_7$
	
$t_7 = 5 > 4 = d_7$, $t_8 = 5 < \infty = d_8$	7 doesn't have any unexplored adjacent node
	
$t_9 = 6 < \infty = d_9$, $t_{10} = 6 < \infty = d_{10}$	9 doesn't have any unexplored adjacent node

Table 2.4: Example of applying Dijkstra algorithm to the graph S in Figure 2.3.

$D^0 = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & 1 & 1 & \infty \\ 1 & 1 & 0 & \infty & 1 \\ \infty & 1 & \infty & 0 & \infty \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$	$P^0 = \begin{bmatrix} - & 1 & 1 & - & - \\ 2 & - & 2 & 2 & - \\ 3 & 3 & - & - & 3 \\ - & 4 & - & - & - \\ - & - & 5 & - & - \end{bmatrix}$
$D^1 = \begin{bmatrix} 0 & 1 & 1 & \infty & \infty \\ 1 & 0 & 1 & 1 & \infty \\ 1 & 1 & 0 & \infty & 1 \\ \infty & 1 & \infty & 0 & \infty \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$	$P^1 = \begin{bmatrix} - & 1 & 1 & - & - \\ 2 & - & 2 & 2 & - \\ 3 & 3 & - & - & 3 \\ - & 4 & - & - & - \\ - & - & 5 & - & - \end{bmatrix}$
$D^2 = \begin{bmatrix} 0 & 1 & 1 & 2 & \infty \\ 1 & 0 & 1 & 1 & \infty \\ 1 & 1 & 0 & 2 & 1 \\ 2 & 1 & 2 & 0 & \infty \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$	$P^2 = \begin{bmatrix} - & 1 & 1 & 2 & - \\ 2 & - & 2 & 2 & - \\ 3 & 3 & - & 2 & 3 \\ 2 & 4 & 2 & - & - \\ - & - & 5 & - & - \end{bmatrix}$
$D^3 = \begin{bmatrix} 0 & 1 & 1 & 2 & 2 \\ 1 & 0 & 1 & 1 & 2 \\ 1 & 1 & 0 & 2 & 1 \\ 2 & 1 & 2 & 0 & 3 \\ 2 & 2 & 1 & 3 & 0 \end{bmatrix}$	$P^3 = \begin{bmatrix} - & 1 & 1 & 2 & 3 \\ 2 & - & 2 & 2 & 3 \\ 3 & 3 & - & 2 & 3 \\ 2 & 4 & 2 & - & 3 \\ 3 & 3 & 5 & 3 & - \end{bmatrix}$
$D^4 = \begin{bmatrix} 0 & 1 & 1 & 2 & 2 \\ 1 & 0 & 1 & 1 & 2 \\ 1 & 1 & 0 & 2 & 1 \\ 2 & 1 & 2 & 0 & 3 \\ 2 & 2 & 1 & 3 & 0 \end{bmatrix}$	$P^4 = \begin{bmatrix} - & 1 & 1 & 2 & 3 \\ 2 & - & 2 & 2 & 3 \\ 3 & 3 & - & 2 & 3 \\ 2 & 4 & 2 & - & 3 \\ 3 & 3 & 5 & 3 & - \end{bmatrix}$
$D^5 = \begin{bmatrix} 0 & 1 & 1 & 2 & 2 \\ 1 & 0 & 1 & 1 & 2 \\ 1 & 1 & 0 & 2 & 1 \\ 2 & 1 & 2 & 0 & 3 \\ 2 & 2 & 1 & 3 & 0 \end{bmatrix}$	$P^5 = \begin{bmatrix} - & 1 & 1 & 2 & 3 \\ 2 & - & 2 & 2 & 3 \\ 3 & 3 & - & 2 & 3 \\ 2 & 4 & 2 & - & 3 \\ 3 & 3 & 5 & 3 & - \end{bmatrix}$

Table 2.6: Example of applying Floyd algorithm to the graph T in Figure 2.1.

Chapter 3

Probabilistic models for social networks

In this chapter we summarize important probabilistic models for social networks. Probabilistic models allow us to make inferences about whether certain network substructures are more commonly observed in the network than might be expected by chance, or to understand a network evolution.

3.1 The Erdös-Rényi-Gilbert Random Graph Model

Given a network G, the probability of an edge between each pair of nodes equal to p, independently of the other edge. It is known as the network model $G(n,p)$, e.g., see [5] and [6].

From a probabilistic viewpoint, if A denotes the (random) adjacency matrix associated with a network $G = G(N,p)$ generated by Erdös-Rényi-Gilbert model, then A has the probability function given by:

$$P(A = \tilde{A}|p) = \prod_{i \neq j, i<j} p^{\tilde{a}_{ij}}(1-p)^{1-\tilde{a}_{ij}} \tag{3.1}$$

Property 3.1 *The number of edges follows a binomial distribution with parameters* $\binom{n}{2}$ *and* p, $Bi\left(\binom{n}{2}, p\right)$*:*

$$P(G(n,p) \text{ has } e \text{ edges } |p) = \binom{\binom{n}{2}}{e} p^e (1-p)^{\binom{n}{2}-e} \qquad e = 0, .., \binom{n}{2} \tag{3.2}$$

So the expected number of edges is $p\binom{n}{2}$.

Property 3.2 *Let $G(N;p)$ be a network following the Erdös-Rényi-Gilbert model. For nodes i, j, $i \neq j$ let E_i and E_j denote the degree of nodes i and j, i.e. the number of edges connected to them. Then,*

$$cov(E_i, E_j) = p(1-p)$$

$$\rho(E_i, E_j) = \frac{1}{n-1}$$

In particular, $\rho(E_i, E_j) \longrightarrow 0$ when $n \longrightarrow \infty$.

Proof:

Consider the Bernoulli variable I_{kl} as follows:

$$I_{kl} = \begin{cases} 1 & \text{if } a_{kl} = 1 \\ 0 & \text{otherwise} \end{cases}$$

It follows that

$$E_i = \sum_{l \neq i} I_{il}, \;\; E_j = \sum_{l' \neq j} I_{jl'}$$

and hence

$$var(E_i) = \sum_{l \neq i} var(I_{il}) + 2 \cancelto{0}{\sum_{l \neq i, l<k} cov(I_{il}, I_{ik})} = (n-1)p(1-p) = var(E_j)$$

$$cov(E_i, E_j) = cov(I_{ij} + \sum_{l \neq i,j} I_{il}, I_{ij} + \sum_{l' \neq j,i} I_{j,l'}) = cov(I_{ij}, I_{ij}) + \cancelto{0}{cov(I_{ij}, \sum_{l' \neq j,i} I_{j,l'})} +$$

$$+ \cancelto{0}{cov(\sum_{l \neq i,j} I_{il}, I_{ij})} + \cancelto{0}{cov(\sum_{l \neq i,j} I_{il}, \sum_{l' \neq j,i} I_{j,l'})} = var(I_{ij}) = p(1-p)$$

$$\rho(E_i, E_j) = \frac{p(p-1)}{\sqrt{(n-1)p(1-p)}\sqrt{(n-1)p(1-p)}} = \frac{p(1-p)}{(n-1)p(1-p)} = \frac{1}{n-1}$$

□

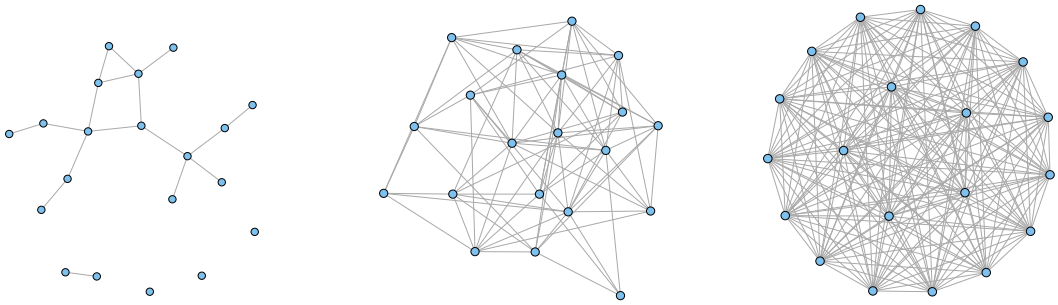
Proposition 3.1 *Let $\lambda = pn$,*

- *If $\lambda < 1$: a graph in $G(n,p)$ will have no connected components of size larger than $O(\log n)$, almost surely as $n \longrightarrow \infty$.*

- *If $\lambda = 1$: a graph in $G(n,p)$ will have a largest component whose size is of $O(n^{2/3})$, almost surely as $n \longrightarrow \infty$.*
- *If $\lambda > 1$: a graph in $G(n,p)$ will have a unique "giant" component containing a positive fraction of the nodes, almost surely as $n \longrightarrow \infty$. No other component will contain more than $O(\log n)$ nodes, almost surely as $n \longrightarrow \infty$.*

  For a proof, see [7].

There is a huge number of mathematical papers that study the Erdös-Rényi-Gilbert model. But few of them are relevant for the actual statistical analysis of networks because the model imposes that every node has approximately the same number of neighbors and we can find few real networks with such simple structure.



n=20, p=0.1 n=20, p=0.5 n=20, p=1

Figure 3.1: Graphs generated by the binomial model of Erdös-Rényi-Gilbert.

Some instances of the binomial model of Erdös-Rényi-Gilbert can be seen in Figure 3.1. These graphs have been generated with the library `igraph` ([22]) in `R` and using the function `erdos.renyi.game`.

3.2 Exponential Random Graph Models

An exponential random graph model is a parametric model of random graph whose adjacency matrix A has the following probability function:

$$P(A = \tilde{A}) = \left(\frac{1}{\kappa}\right) \exp\left\{\sum_{Y} \eta_Y g_Y(\tilde{A})\right\} \tag{3.3}$$

where:

- The summation is over all configurations Y. A configuration is a set of nodes and a subset of edges among them, such that, if a set of edges represents a configuration in the model, then any subset of possible edges is also a configuration.
- $g_Y(\tilde{A})$ is a statistic corresponding to configuration Y defined as $g_Y(\tilde{A}) = \prod_{\tilde{a}_{ij} \in Y} \tilde{a}_{ij}$. Thus, $g_Y(\tilde{A}) = 1$ if the configuration Y is observed in $\tilde{A}$, and is 0 otherwise.
- κ is a normalizing quantity.
- η_Y is a parameter corresponding to configuration of type Y.

See more in [8] and [9].

The Markov random graphs are a particular sub-class of exponential random graph models, introduced by Frank and Strauss (see [10]), in which configurations consist of edges such that any pair of edges share a node. See Figure 3.2.

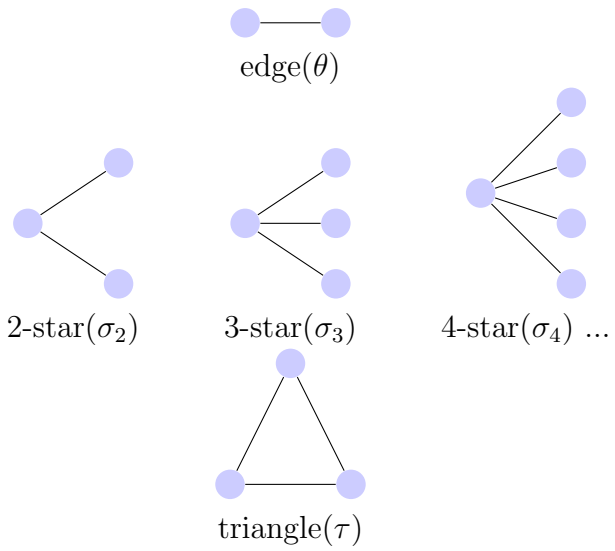


Figure 3.2: Configurations and parameters for non-directed Markov random graphs

For example, in a Markov random graph, the probability of observing a particular non-directed network $\tilde{A}$ with edges, 2-stars, 3-stars and triangles is given by the equation (3.4).

$$P(A = \tilde{A}) = \left(\frac{1}{\kappa}\right) \exp(\theta L(\tilde{A}) + \sigma_2 S_2(\tilde{A}) + \sigma_3 S_3(\tilde{A}) + \tau T(\tilde{A})) \qquad (3.4)$$

where $L(\tilde{A})$ is the numbers of edges in $\tilde{A}$, $S_2(\tilde{A})$ and $S_3(\tilde{A})$ are the number of 2-star and 3-star, respectively in the network $\tilde{A}$, and $T(\tilde{A})$ is the number of triangles in $\tilde{A}$.

3.3 Barabási-Albert Model

The Barabási-Alber (BA) model is used for generating scale-free networks, i.e, networks where the probability that a node has k edges follows a power law, see [11].

The Erdös-Rényi-Gilbert Model or Exponential models, which we have already explained, lack two aspect that most real-world networks have. These models assume that the network has a fixed numbers of nodes (n), in contrast with most real-world networks formed by the continuous addition of new nodes to the network. On the other hand, these models assume that the probability that two nodes are connected is random and uniform; on the contrary, most real networks show preferential attachment. A new node usually has higher probability to be connected to a node that already has a larger number of connections.

The BA model incorporates these two aspects. The model is defined in two steps:

1. The network starts with a small number (n_0) of nodes. At every timestep (t) we add a node with m ($\leq n_0$) edges (connected to a node that is already present in the network).

2. A new node (j) will be connected to a node i with a probability that depends on the number of edges of i (k_i):

$$P((j, i) \in E) = \frac{k_i}{\Sigma_{j \in N_{t-1}} k_j}$$

where N_{t-1} is the set of nodes which are already present in the network.

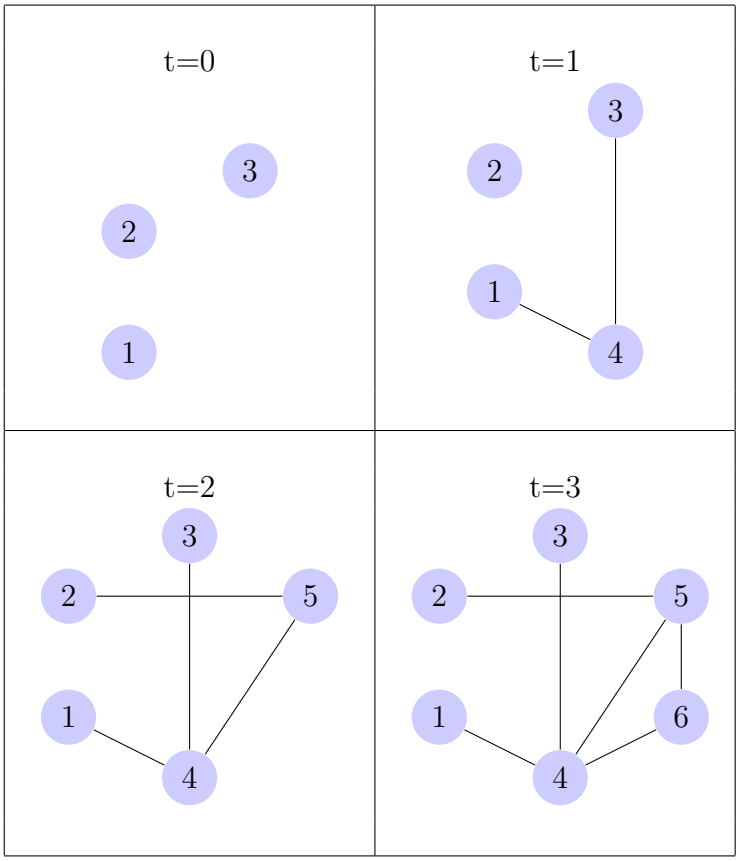


Table 3.1: Example of a Barabási-Albert graph

After T timesteps we obtain a random network with $n = T + n_0$ nodes and mT edges.

As an illustration, in Table 3.1 one can see a graph generated by the Barabási-Albert model.

Chapter 4

Centrality concepts

In the study of Social Networks, an important objective is to know who are the most important or influential users in a network.

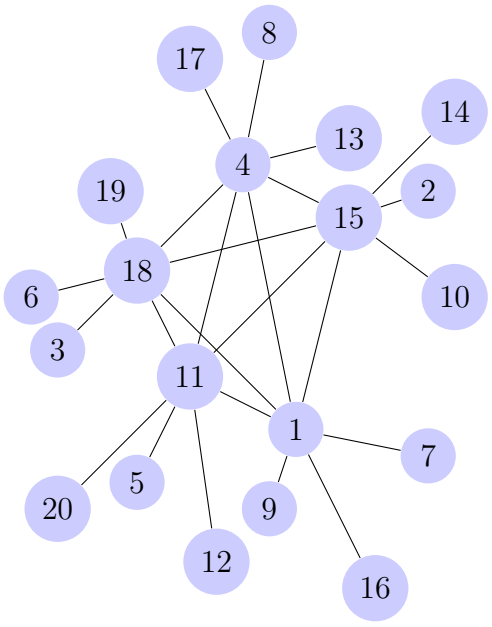


Figure 4.1: Graph G

In Figure 4.1 it seems logical to think that the nodes $1, 4, 11, 15, 18$ are the most important ones in the network G because of their number of links. The process of counting how many links each node in a network has leads to the concept of *degree centrality*. But how can we know who is connected to important nodes? Or what if the network is not neatly partitioned into core and periphery like happens in Figure 4.1? In this chapter we introduce

some measures of centrality in a network, which allow us to know the relative importance of a node in a graph.

4.1 Centrality Measures

We are going to study different centrality measures of a graph G. See more in [17].

Definition 4.1 *The degree centrality of a node i is equal to the number of edges connected to the node:*

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

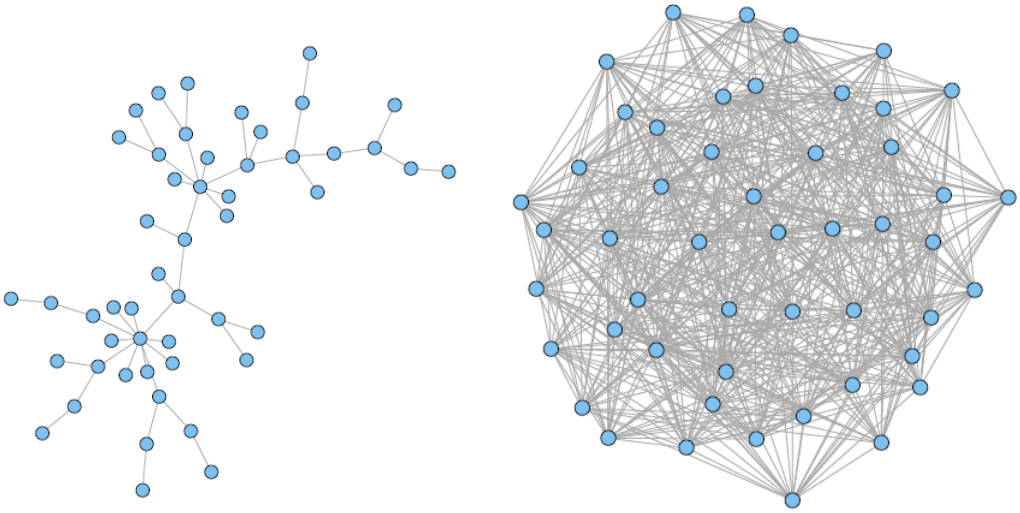
We can standardize the measure to compare with other networks. To do this, we divide by the number of possible edges a node can have, namely, $n-1$:

$$C^*_{D_i} = \frac{1}{n-1} \sum_{j=1}^{n} a_{ij}$$

As an illustration, we are going to study the degree centrality of a Barabási-Albert random graph S with 50 nodes, of a Erdös-Rényi-Gilbert random graph T with 50 nodes and probability 0.5 and of a graph U usually found in the literature. In Figure 4.2 and 4.3 the three so obtained graphs are represented.

To calculate the degree centrality we can use the function `degree` of the library `igraph` in `R`.

In Figure 4.4 the standardized degree centrality of each node in S is shown. We see that there are two nodes (1 and 6) whose degree centrality is clearly bigger than the remaining ones, and can be identified as the most important ones in the graph. In Figure 4.5 (left) we depict the standardized degree centrality of the nodes in T. It is seen that the degree centrality is more or less homogeneous, and then this measure can not identify the most relevant nodes, if any. The standardized degree centrality of the nodes in the graph U is shown in Figure 4.5 (right). As in graph S happened, we can observed that there are two nodes (7 and 9) whose degree centrality is bigger than the all other ones, so we can identify the nodes 7 and 9 as the most important ones in the graph U.



Graph S Graph T

Figure 4.2: Barabási-Albert random graph S and Erdös-Rényi-Gilbert random graph T

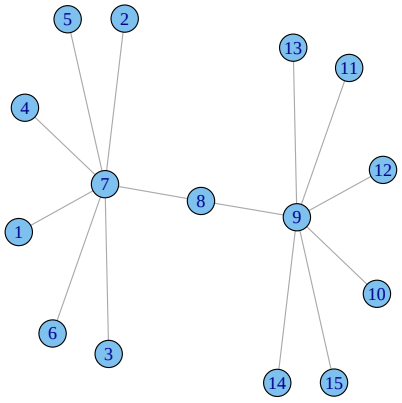


Figure 4.3: Graph U

A complementary viewpoint is represented in Figure 4.6, which shows the histograms of the standardized degree centrality of the graphs S (left) and T (right), and in Figure 4.7 which shows the histograms of the standardized

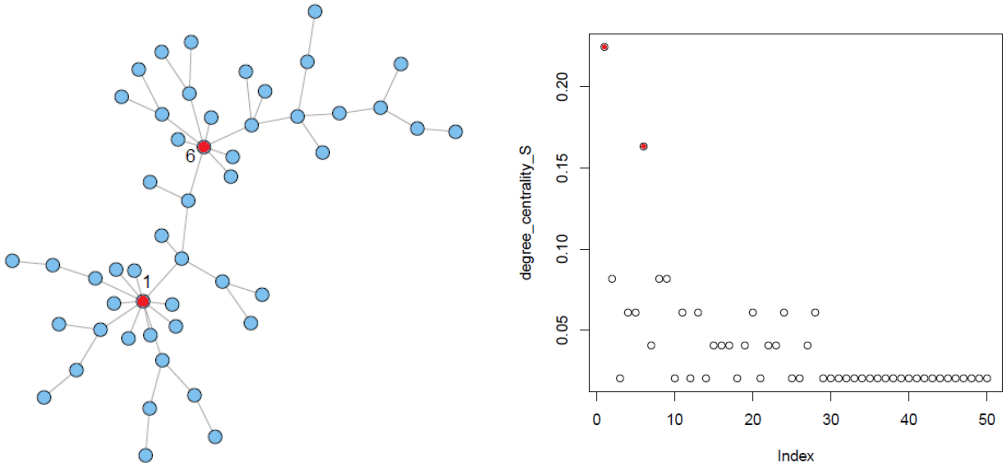


Figure 4.4: Representations of the degree centrality of the graph S

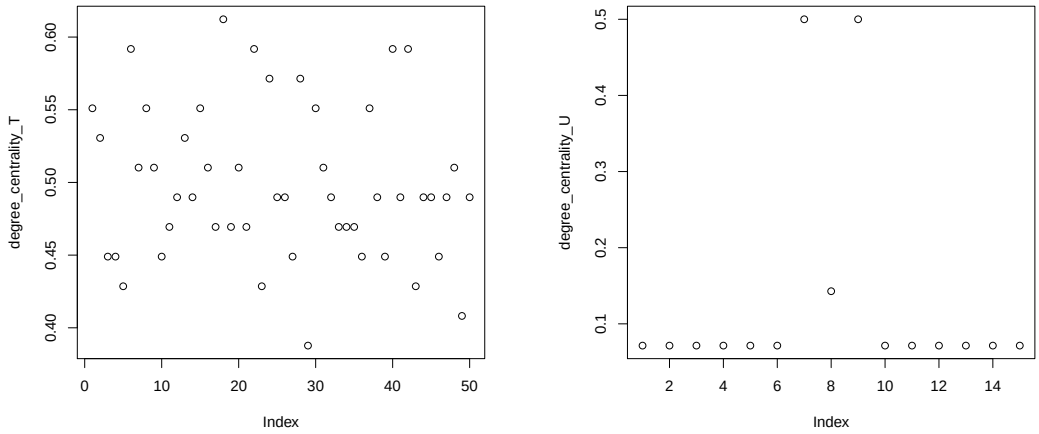


Figure 4.5: Representations of the degree centrality of the graphs T (left) and U (right)

degree centrality of the graph U. We can observe that the graph S has a lot of nodes with low degree centrality and few of them with high degree centrality. The degree centrality of the nodes of S follows a power law, as we expected. On the contrary, the histogram of T (Figure 4.6 right) has a gaussian like shape. Finally, in the histogram of U (Figure 4.7) we observe that the majority of the nodes has low degree centrality.

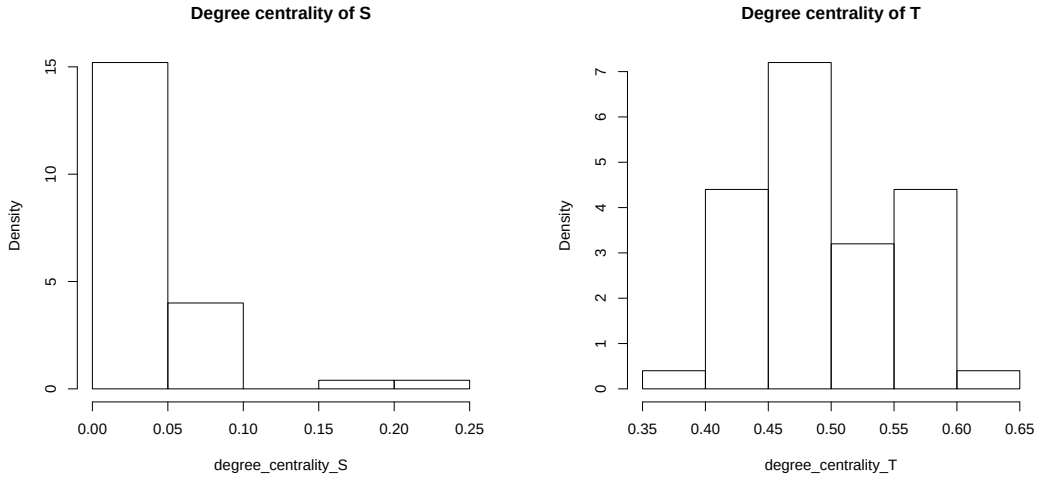


Figure 4.6: Histograms of the distribution of the degree centrality of S (left) and T (right)

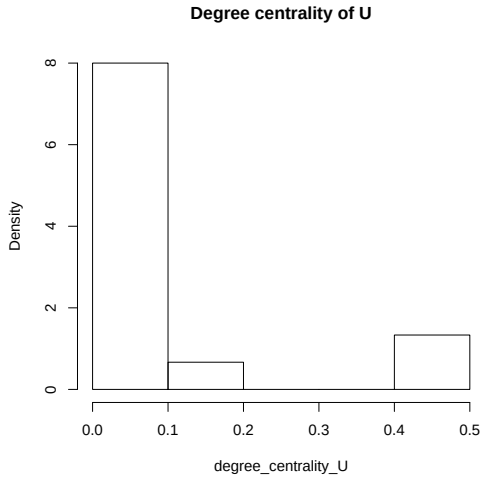


Figure 4.7: Histogram of the distribution of the degree centrality of U

Definition 4.2 *The closeness centrality measures the closeness of node i relative to all other nodes in a graph:*

$$C_{C_i} = \left[\sum_{j=1}^{n} d(i,j)\right]^{-1}$$

C_{C_i} will reach its maximum value, namely, $(n-1)^{-1}$, if i is adjacent to

all other nodes in the graph.

Closeness centrality can be standardized multiplying it by $n-1$:

$$C^*_{C_i} = (n-1)\left[\sum_{j=1}^{n} d(i,j)\right]^{-1}$$

$C^*_{C_i}$ is an index which ranges between 0 and 1. In this case, $C^*_{C_i} = 1$ if i is adjacent to all other nodes in the graph. The inverse of the standardized closeness centrality $C^*_{C_i}$ is the average path length between i and all other nodes.

We are going to study the closeness centrality in graphs S and T of Figure 4.2 and in graph U of Figure 4.3. To do this, we use the function `closeness` of the library `igraph` in `R`. See the histograms of the distribution of the standardized closeness centrality of these graphs in Figure 4.8 and Figure 4.9.

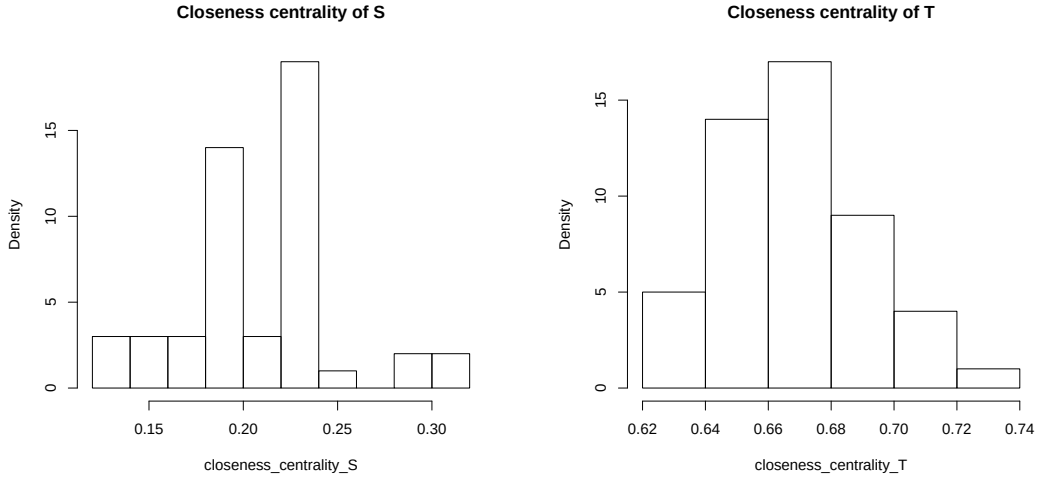


Figure 4.8: Histograms of closeness centrality of graphs S (right) and T (left)

We observe that the closeness centrality are bigger in all the nodes of the graph T than in graphs S and U. So, for example, if we want to expand a piece of information from a node to all other ones, it would be faster in graph T than in graph S or U.

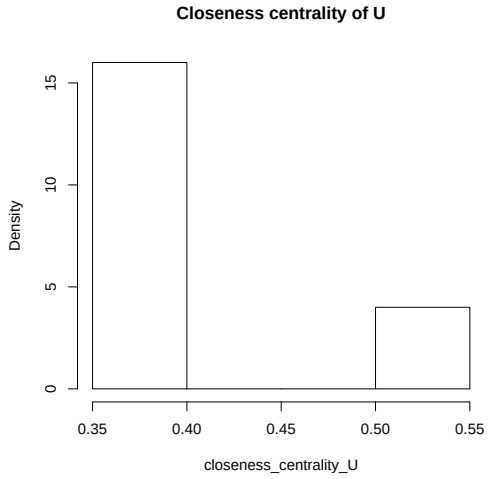


Figure 4.9: Histogram of closeness centrality of graphs U

Definition 4.3 *The eigenvector centrality of the nodes of a graph is defined as the vector* $(x_1, \ldots, x_n)$*, solution of:*

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} a_{ij} x_j$$

where λ *is a constant value.*

The problem in the previous definition can be rewritten in vector notation as follows:

$$Ax = \lambda x$$

which is an eigenvector problem.

There will be many different eigenvalues λ for which an eigenvector solution exists. However, a requirement in this contest is that all values of x must be positive, and by the Perron-Frobenious theorem, [16], λ only can be the greatest eigenvalue.

The value in eigenvector of a node is dependent of the value in eigenvector centrality of its adjacent nodes. So, if a node is connected to many nodes which are well connected, it has a high value in eigenvector centrality. In Social Networks a node with high value in eigenvector centrality is an influential node.

We are going to study the eigenvector centrality of the graphs S and T of Figure 4.2 and of the graph U of Figure 4.3.

In order to do that, we use the function `evcent` of the library `igraph` in `R` and we obtain the eigenvector x_U:

$$x_U = [0.35, 0.35, 0.35, 0.35, 0.35, 0.35, 1, 0.71, 1, 0.35, 0.35, 0.35, 0.35, 0.35, 0.35]$$

Observe that the nodes 7 and 9 have the value 1 in the eigenvector centrality. Moreover, the node 8 has a high value in the eigenvector centrality though it is only linked with two nodes. This is due to the fact that those two nodes are central nodes. Hence, the nodes 7, 8 and 9 are the influential nodes in the graph.

In graph S (Figure 4.10) there is a node (node with colour red) with value 1 in the eigenvector centrality. The remaining ones have low value (less than 0.42) in the eigenvector centrality. Hence, the red node is the only important node in S.

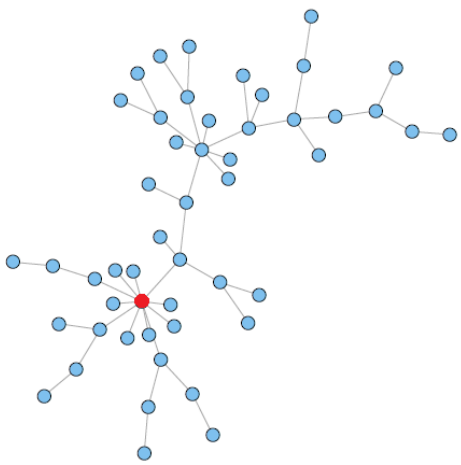


Figure 4.10: Graphs S

In graph T (Figure 4.2 right) all nodes have the eigenvector centrality in the range $[0.65, 1]$. So we can not know who are the most important nodes in the graph with this measure.

Definition 4.4 *The betweenness centrality of a node i of a graph is defined as follows:*

$$C_{B_i} = \sum_{j<k,\ j,k\neq i} \frac{g_{jik}}{g_{jk}}$$

where g_{jk} is the number of shortest paths between j and k, and g_{jik} is the number of shortest paths between j and k that contain i.

Betweenness centrality is a centrality measure which quantifies the number of times that a node acts as a bridge in a shortest path between two other nodes.

We are going to see an example. We study the betweenness centrality of the node 4 of the graph W of Figure 4.11. See Table 4.1.

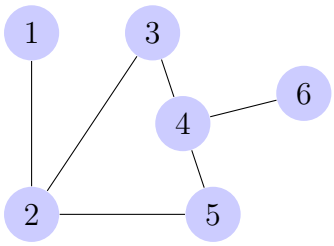


Figure 4.11: Graph W

j	k	shortest Paths	$\frac{g_{j4k}}{g_{jk}}$
1	2	$\{1,2\}$	0
1	3	$\{1,2,3\}$	0
1	5	$\{1,2,5\}$	0
1	6	$\{1,2,3,4,6\},\{1,2,5,4,6\}$	1
2	3	$\{2,3\}$	0
2	5	$\{2,5\}$	0
2	6	$\{2,3,4,6\},\{2,5,4,6\}$	1
3	5	$\{3,4,5\},\{3,2,5\}$	0.5
3	6	$\{3,4,6\}$	1
5	6	$\{5,4,6\}$	1
			$C_{B_4} = 4.5$

Table 4.1: Betweenness centrality of node 4 of graph W

The betweenness centrality can be standardized by dividing through the number of pairs of nodes not including i:

$$C_2^{n-1} = \frac{(n-1)!}{2!(n-1-2)!} = \frac{(n-1)(n-2)}{2}$$

Hence, the standardized betweenness centrality has the following shape:

$$C_{B_i}^* = \frac{C_{B_i}}{C_2^{n-1}} = \frac{2}{(n-1)(n-2)} \sum_{j<k,\ j,k\neq i} \frac{g_{jik}}{g_{jk}}$$

For example, the node 4 of the graph W (Figure 4.11) has the following standardized betweenness centrality:

$$C_{B_4}^* = \frac{C_{B_4}}{C_2^5} = \frac{4.5}{10} = 0.45$$

We can study the betweenness centrality of the nodes of a graph with the function `betweenness` of the library `igraph` in `R`. We use this function to study this measure of the graphs T, S and U (Figures 4.2 and 4.3).

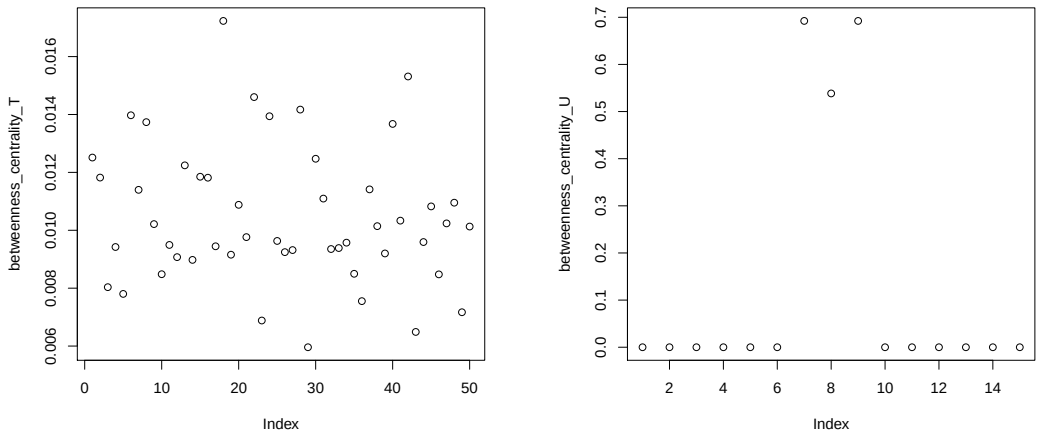


Figure 4.12: Representations of the betweenness centrality of the graphs T (left) and U (right)

In Figure 4.12 (left) the standardized betweenness centrality of T is shown. It is seen that the betweenness centrality is more or less homogeneous and moreover all nodes have low betweenness centrality. Hence, we can not identify who is the most important node with this measure. We depict the standardized betweenness centrality of graph U in Figure 4.12

(right). We observe that all the nodes have betweenness centrality equal to 0 except nodes 7, 8 and 9 whose betweenness centrality is bigger than 0.54. So, we can identify these nodes as the most relevant ones in U.

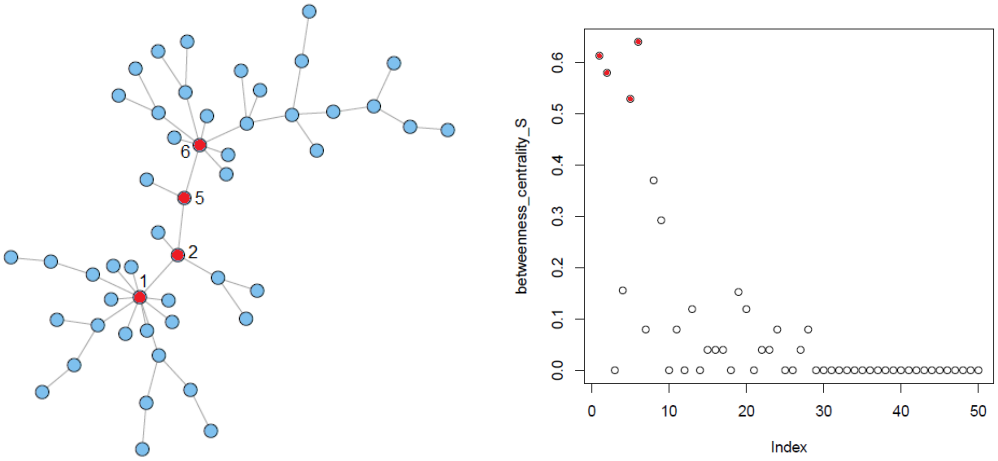


Figure 4.13: Representations of the betweenness centrality of the graph S

Figure 4.13 (right) shows the standardized betweenness centrality of the graph S. We observe that the nodes 1, 2, 5 and 6 have betweenness centrality bigger than the remaining ones, so these nodes can be identified as the most important ones in S.

Chapter 5

Association concepts

In the actual study of social networks there is a need for useful concepts to represent closely knit groups. Many of these can be developed with the help of the theory of graphs. A well-known concept is the clique: a group in which all its members are in contact with each other or are friends.

In this chapter we will introduce four different cluster concepts of graphs: k-cliques, k-clans, k-club and k-plex.

5.1 Cluster concepts

Consider an undirected graph $G = G(N, E)$. Given a subset of nodes $S \subseteq N$, $G(S)$ denotes the subgraph induced by S on G, $G(S) = (S, S \times S \cap E)$.

Definition 5.1 *A clique of a graph G is a subset of nodes T such that for all pairs of nodes $u, v \in T$: $(u, v) \in E$, i.e., its induced subgraph $G(T)$ is a complete graph.*

The problem of finding maximal cliques in a graph, i.e., find complete subgraphs that are not contained in any other complete subgraph, is in general a hard problem. To do this, it is usually used backtracking algorithms that apply branch-and-bound techniques to cut off branches that can not lead to a clique. The Bron-Kerbosch algorithm is an example [12].

Luce introduced the concept of "k-clique" (see [13]), given by the following definition

Definition 5.2 *A k-clique L of a graph G is a maximal subgraph of G such that for all pairs of nodes u, v of L: $d_G(u, v) \leq k$.*

In other words, a k-clique L is a set of nodes in which any two nodes are a distance of at most k from each other in G, and no other node in the graph is of distance k or less from every other node in L. For example, in the graph Z the subset of nodes $C_1 = \{1, 2, 4, 5, 6\}$ forms a 2-clique.

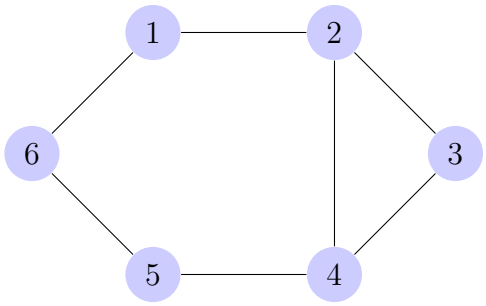


Figure 5.1: Graph Z

By definition, if two nodes $u, v \in N$ belong to a k-clique L, then $d_G(u, v) \leq k$, however this does not imply that $d_{G(L)}(u, v) \leq k$. So the diameter of L may be larger than k. For example, in the graph Z the subset of nodes $C_2 = \{1, 2, 3, 4, 5\}$ forms a 2-clique, although the diameter of $G(C_2)$ is 3.

Hence, the concept of k-clique lacks the requirement of "tightness" in the group of nodes corresponding of a k-clique, while this requirement is essential to applications in Social Networks. Because of this, Alba introduced the concept of a "sociometric clique" (see [14]), which was later renamed to "k-clan" by Mokken.

Definition 5.3 *A k-clique L is called k-clan if the diameter of the induced subgraph $G(L)$ is no more than k.*

So for example C_1 also forms a 2-clan.

In 1979, Mokken defined the concept of "k-club" (see [15]):

Definition 5.4 *A maximal subset of nodes $D \subseteq N$ is a k-club if the diameter of the induced subgraph $G(D)$ is at most k.*

Hence for a k-club D of G the following clauses are true:

1. $\forall u, v \in D$: $d_{G(D)}(u, v) \leq k$

2. Because of the maximality of D, $\forall w \in G - D$, $\exists u \in D$: $d_{G(u,w)} > k$

The 2-clubs of the graph Z are $D_1 = \{1, 2, 3, 4\}$, $D_2 = \{2, 3, 4, 5\}$ and $D_3 = C_1$.

By definition a k-clan is a k-clique of diameter k. But how is a k-clique related to a k-club? See Proposition 5.1 (proof in [15]).

Proposition 5.1 *Every k-club N of a graph G is contained in some k-clique L of G.*

Let $N(u)$ and $deg_G(u)$ denote respectively the set of neighbors of a node $u \in N$ ($N(u) = \{v : (u, v) \in E\}$) and the number of neighbors of u in G. Let $L[u]$ denote the closed neighborhood of a node u, $N[u] = \{u\} \cup N(u)$.

Definition 5.5 *A subset of nodes $S \subseteq N$ is a k-plex if $deg_{G(S)}(v) = \mid N(v) \cap S \mid \geq \mid S \mid -k$, $\forall v \in S$, (see [20]).*

For example the subset of nodes $E_2 = \{2, 3, 4\}$ forms a 1-plex.

Seidman and Foster proposed an equivalent characterization of k-plexes (proof in [21]):

Proposition 5.2 *$S \subseteq N$ is a k-plex if and only if for any subset of k nodes, $\{v_1, .., v_k\} \subset S$: $S = \bigcup_{i=1}^{k} N[v_i]$.*

5.2 Integer programming formulations

This section presents integer programming (IP) formulations for the maximum k-clique, maximum k-club and maximum k-plex problems. In order to do that, consider for every $i \in N$, the binary variable x_i equal to 1 if and only if i belongs to the solution.

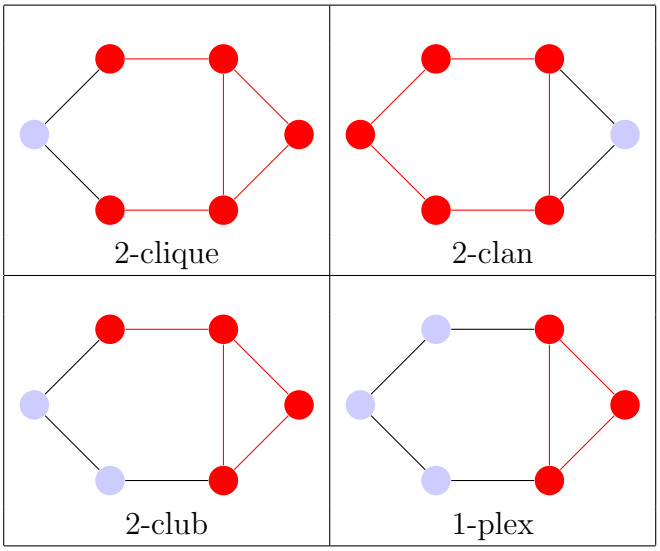


Figure 5.2: Examples of clusters

5.2.1 Maximum k-clique problem

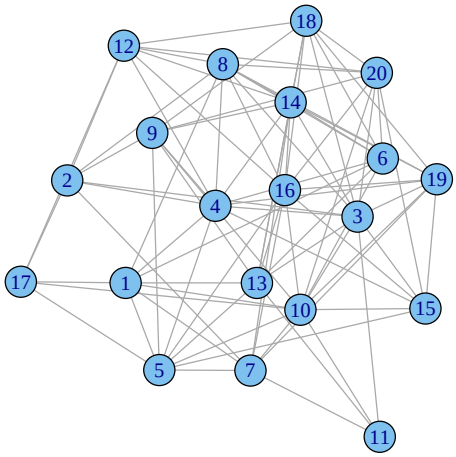
Consider the following formulation for the k-clique problem, [18]:

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in N} x_i \\ \text{subject to} \quad & x_i + x_j \leq 1 + \frac{k}{d_G(i,j)} \quad \forall i,j \in V : i < j \\ & x_i \in \{0,1\} \qquad\qquad\qquad \forall i \in V \end{aligned} \tag{5.1}$$

The constraints expres the fact that if two actors have $d_G(i,j) > k$, they can not be simultaneously included in a k-clique.

We are going to see a example. We consider a set of twenty actors in a social network and the relationship among themselves. It is represented by the Erdös-Réngy-Gilbert random graph E with probability 0.5 which was introduced in Section 2.1 (Figure 5.3).

The matrix of distances of the graph E, d_E, has been calculated with the program `Grafos` ([23]).




```
[1,] . . . 1 1 1 1 1 . 1 . . 1 . . . 1 . . .
[2,] . . 1 1 . . 1 1 1 . . 1 . . . . 1 . . .
[3,] . 1 . 1 . . . 1 . 1 1 . 1 1 1 . . 1 1 1
[4,] 1 1 1 . 1 1 . 1 1 . . 1 . 1 1 . . . 1 .
[5,] 1 . . 1 . . 1 . 1 1 . . 1 . 1 1 1 . . .
[6,] 1 . . 1 . . . 1 . 1 . . 1 1 . . . 1 . 1
[7,] 1 1 . . 1 . . . . 1 1 . . 1 . 1 . . 1 .
[8,] 1 1 1 1 . 1 . . . . . 1 . 1 . 1 . . 1 1
[9,] . 1 . 1 1 . . . . 1 . . 1 1 . . . 1 . 1
[10,] 1 . 1 . 1 1 1 . 1 . 1 . . . 1 1 1 . 1 1
[11,] . . 1 . . . 1 . . 1 . . 1 . . . . . . .
[12,] . 1 . 1 . . . 1 . . . . . 1 . 1 1 1 . 1
[13,] 1 . 1 . 1 1 . . 1 . 1 . . . 1 . 1 . 1 . .
[14,] . . 1 1 . 1 1 1 1 . . 1 1 . . 1 . 1 1 .
[15,] . . 1 1 1 . . . . 1 . . . . . 1 . . 1 1
[16,] . . . . 1 . 1 1 . 1 . 1 1 1 1 . . 1 1 1
[17,] 1 1 . . 1 . . . . 1 . 1 . . . . . . . .
[18,] . . 1 . . 1 . . 1 . . 1 1 1 . 1 . . 1 1
[19,] . . 1 1 . . 1 1 . 1 . . . 1 1 1 . 1 . .
[20,] . . 1 . . 1 . 1 1 1 . 1 . . 1 1 . 1 . .
```

Figure 5.3: Network E and its adjacency matrix which has been obtained with the library `igraph` in `R` and using the function `get.adjacency`

$$d_E = \begin{pmatrix}
0 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 2 & 2 & 1 & 2 & 2 & 2 & 1 & 2 & 2 & 2 \\
2 & 0 & 1 & 1 & 2 & 2 & 1 & 1 & 1 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 2 \\
2 & 1 & 0 & 1 & 2 & 2 & 2 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 2 & 2 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 2 & 1 & 1 & 2 & 2 & 1 & 2 & 1 & 1 & 2 & 2 & 2 & 1 & 2 \\
1 & 2 & 2 & 1 & 0 & 2 & 1 & 2 & 1 & 1 & 2 & 2 & 1 & 2 & 1 & 1 & 1 & 2 & 2 & 2 \\
1 & 2 & 2 & 1 & 2 & 0 & 2 & 1 & 2 & 1 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 1 & 2 & 1 \\
1 & 1 & 2 & 2 & 1 & 2 & 0 & 2 & 2 & 1 & 1 & 2 & 2 & 1 & 2 & 1 & 2 & 2 & 1 & 2 \\
1 & 1 & 1 & 1 & 2 & 1 & 2 & 0 & 2 & 2 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 2 & 1 & 1 \\
2 & 1 & 2 & 1 & 1 & 2 & 2 & 2 & 0 & 1 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 1 & 2 & 1 \\
1 & 2 & 1 & 2 & 1 & 1 & 1 & 2 & 1 & 0 & 1 & 2 & 2 & 2 & 1 & 1 & 1 & 2 & 1 & 1 \\
2 & 2 & 1 & 2 & 2 & 2 & 1 & 2 & 2 & 1 & 0 & 3 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
2 & 1 & 2 & 1 & 2 & 2 & 2 & 1 & 2 & 2 & 3 & 0 & 2 & 1 & 2 & 1 & 1 & 1 & 2 & 1 \\
1 & 2 & 1 & 2 & 1 & 1 & 2 & 2 & 1 & 2 & 1 & 2 & 0 & 1 & 2 & 1 & 2 & 1 & 2 & 2 \\
2 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 2 & 1 & 1 & 0 & 2 & 1 & 2 & 1 & 1 & 2 \\
2 & 2 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 0 & 1 & 2 & 2 & 1 & 1 \\
2 & 2 & 2 & 2 & 1 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 & 1 & 0 & 2 & 1 & 1 & 1 \\
1 & 1 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 1 & 2 & 1 & 2 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\
2 & 2 & 1 & 2 & 2 & 1 & 2 & 2 & 1 & 2 & 2 & 1 & 1 & 1 & 2 & 1 & 2 & 0 & 1 & 1 \\
2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 2 & 2 & 1 & 1 & 1 & 2 & 1 & 0 & 2 \\
2 & 2 & 1 & 2 & 2 & 1 & 2 & 1 & 1 & 1 & 2 & 1 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 0
\end{pmatrix}$$

This problem is written in `AMPL`, as shown in Table 5.1.

```
param n;

param k;

param d{1..n,1..n};

var x{1..n} binary;

maximize f:  sum{i in 1..n} x[i];

subject to r1{j in 1..n, i in 1..j-1}:  x[i]+x[j]<=1+k/d[i,j];
```

Table 5.1: Code of maximum k-clique problem

The diameter is the largest element in the distance matrix d_E, so E has diameter 3. Hence we can search the maximum 1-clique, 2-clique and 3-clique. The optimal solutions for these problems provided by the solver `Gurobi` are given in Table 5.2.

$k = 1$
$x_i = \begin{cases} 1 & \text{if } i = 3, 4, 8, 14, 19 \\ 0 & \text{everywhere else} \end{cases}$
$k = 2$
$x_i = \begin{cases} 0 & \text{if } i = 12 \\ 1 & \text{everywhere else} \end{cases}$
$k = 3$
$x_i = \begin{cases} 1 & \text{if } i = 1, 2, 3, 4, 5, 6... \\ 0 & \text{never} \end{cases}$

Table 5.2: Results

So, the maximum 1-clique of E is $E(S)$ where $S = \{3, 4, 8, 14, 19\}$. In

others words, $E(S)$ is the maximum maximal subgraph of E in which every two nodes are adjacent in E. See the Figure 5.4.

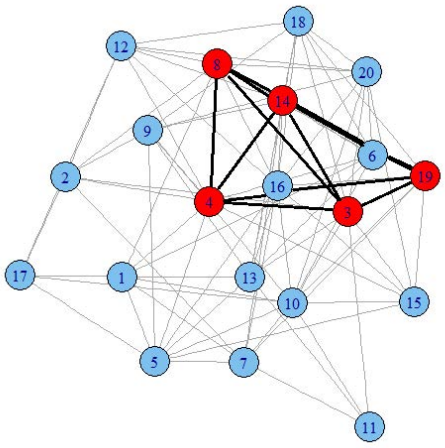


Figure 5.4: Maximum 1-clique of E

The subset V formed by all the nodes excluding number 12 forms the maximum 2-clique, i.e., every two nodes of V either are adjacent in E or are adjacent of a common node in E.

Logically E is the maximum 3-clique because its diameter is three.

We have observed that the nodes 11 and 12 are the only pair of nodes whose distance between them is 3. Hence we study if the maximum 2-clique problem has multiple solutions. For it, we impose the constraint `r2` which makes the node 12 belongs to the solution. See the code in Table 5.3.

The optimal solution for this problem provided by the solver `Gurobi` is given in Table 5.4.

The induced subgraph $E(W)$ where $W = N\backslash\{12\}$ is a 2-clique and this problem has the same objective value than the previous one. Then $E(W)$ is also a maximum 2-clique.

```
param n=20;

param k;

param d{1..n,1..n};

var x{1..n} binary;

maximize f:  sum{i in 1..n} x[i];

subject to r1{j in 1..n, i in 1..j-1}:  x[i]+x[j]<=1+k/d[i,j];

subject to r2:  x[12]=1;
```

Table 5.3: Code of the maximum k-clique problem in which we impose that the node 12 belongs to the solution

$k=2$
$x_i = \begin{cases} 0 & \text{if } i = 11 \\ 1 & \text{everywhere else} \end{cases}$

Table 5.4: Optimal solution of the problem in Table 5.3

5.2.2 Maximum k-club problem

For any two nodes $i, j \in N$, let C_{ij}^k be the sets of all chains of length at most k linking i and j. Denoted by N_t the nodes set of a chain t. Let y_t be an auxiliary binary variable associated with every chain $t \in \bigcup_{i,j \in N} C_{ij}^k = C$. Then the maximum k-club problem can be formulated as an integer linear program as follows, [19]:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i \in N} x_i \\
\text{subject to} \quad & x_i + x_j \leq 1 + \textstyle\sum_{t \in C_{ij}^k} y_t && \forall (i,j) \notin E,\ C_{ij}^k \neq \varnothing \\
& y_t \leq x_r && \forall t \in C,\ \forall r \in N_t \\
& x_i + x_j \leq 1 && \forall (i,j) \notin E,\ C_{ij}^k = \varnothing \\
& x_i \in \{0,1\} && \forall i \in N \\
& y_t \in \{0,1\} && \forall t \in C
\end{aligned}
\tag{5.2}
$$

The constraints ensure that if two nodes i and j are not linked by an edge but are linked at least by a chain of length at most k, then both nodes can belong to the solution if and only if $y_t = 1$ for at least one chain $t \in C$. Also, if $y_t = 1$ then all nodes of the chain t belong to the solution. Finally, if two nodes are not linked neither by an edge nor by a chain of length at most k, can not simultaneously belong to the solution.

We are going to study the maximum k-club in the same graph E of the previous subsection, see Figure 5.3.

Table 5.5 shows this problem which is written in `AMPL`.

The maximum 1-clique is a complete subgraph, so its diameter is 1. Then the maximum 1-clique problem and the maximum 1-club problem are the same. In addition, the diameter of E is 3, so the maximum 3-club is E. Hence, we only study the maximum 2-club problem, which optimal solution provided by the solver `Gurobi` is given in (5.3).

$$
x_i = \begin{cases} 0 & \text{if } i = 11 \\ 1 & \text{everywhere else} \end{cases}
\tag{5.3}
$$

The induced subgraph $E(W)$ is also the maximum 2-club.

```
param n;

param NC;

set nodes := 1..n;

set Pairs = {n1 in nodes, n2 in nodes:  n1<n2};

set Existchain within {n1 in nodes, n2 in nodes:  n1<>n2};

set Adjacent within {n1 in nodes, n2 in nodes:  n1<>n2};

set C{Existchain};

set T := 1..NC;

set P{T};

var x{nodes} binary;

var y{T} binary;

maximize f:  sum {i in nodes} x[i];

subject to r1 {(i,j) in Existchain diff Adjacent}:

 sum {t in C[i,j]} y[t]>=x[i]+x[j]-1;

subject to r2 {t in T, r in P[t]}:  y[t]<=x[r];

subject to r3 {(i,j) in Pairs diff Existchain diff Adjacent}:

 x[i]+x[j]<=1;
```

Table 5.5: Code of maximum k-club problem

We are going to check if this problem has multiple solutions as happen with the 2-clique problem. For it, we impose the new constraint `r4`:

`x[11]=1` which makes the node 11 belongs to the solution. The solver `Gurobi` gives us the following solution:

$$x_i = \begin{cases} 0 & \text{if } i = 12, 17 \\ 1 & \text{everywhere else} \end{cases} \tag{5.4}$$

Remark that the objective value is lower in this problem, so $E(W)$ is the only optimal solution for the maximum 2-club problem.

5.2.3 Maximum k-plex problem

Let $\bar{d}_i = |N \setminus N[i]|$ denote the degree of the vertex i in the complement graph $\bar{G} = \bar{G}(V, \bar{E})$. We can considerer the maximum k-plex problem as an integer linear program as follows, [21]:

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in N} x_i \\ \text{subject to} \quad & \sum_{j \in N \setminus N[i]} x_j \leq (k-1)x_i + \bar{d}_i(1 - x_i) \quad \forall i \in V \\ & x_i \in \{0, 1\} \qquad\qquad\qquad\qquad\quad \forall i \in V \end{aligned} \tag{5.5}$$

Each constraint ensures that if a node is in a k-plex, then it has at most $k - 1$ non-neighbors inside the k-plex.

The maximum 1-plex problem is the same than the maximum 1-clique problem. Hence, we are going to study the maximum 2-plex and 3-plex of the graph E.

The maximum k-plex problem is written in `AMPL`, as shown in Table 5.6.

The optimal solutions for the maximum 2-plex and 3-plex problems provided by `Gurobi` are given in Table 5.7.

Then the maximum 2-plex of E is the induced subgraph $E(G)$ where $G = \{4, 8, 12, 14, 16, 19\}$. In others words, G is the maximum subset of

```
param n;

param k;

set nodes := 1..n;

set N{nodes};

param d{nodes};

var x{nodes} binary;

maximize f:  sum{i in nodes} x[i];

subject to r1{i in nodes}:

 sum{j in nodes diff N[i]} x[j] <= (k-1)*x[i]+d[i]*(1-x[i]);
```

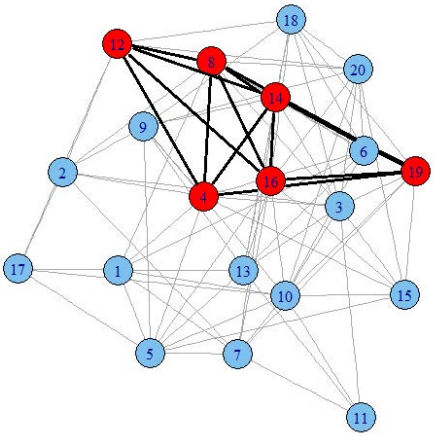
Table 5.6: Code of the maximum k-plex problem

$k = 2$
$x_i = \begin{cases} 1 & \text{if } i = 4, 8, 12, 14, 16, 19 \\ 0 & \text{everywhere else} \end{cases}$
$k = 3$
$x_i = \begin{cases} 1 & \text{if } i = 3, 8, 12, 14, 16, 18, 19, 20 \\ 0 & \text{everywhere else} \end{cases}$

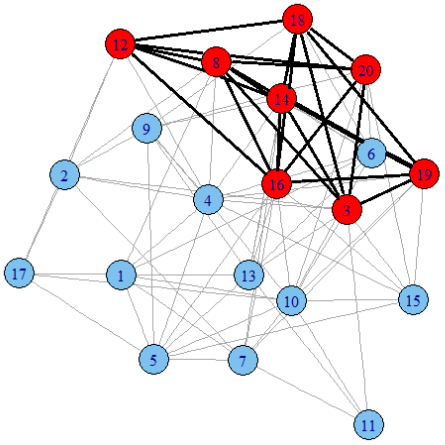
Table 5.7: Results

nodes which forms a induced subgraph where every node has at most 1 non-neighbor.

$E(H)$, where $H = \{3, 8, 12, 14, 16, 18, 19, 20\}$, is the maximum 3-plex of E, i.e., H is the maximum subset of nodes where every node has at most 2 non-neighbors in $E(H)$.



maximum 2-plex



maximum 3-plex

Bibliography

[1] Dimitri P. Bertsekas, Network Optimization. Continuous and Discrete Models. Athena Scientific, 1998.

[2] Mokhtar S. Bazaraa, John J. Jarvis and Hanif D. Sherali. Programación lineal y flujo en redes. Limusa, 1998.

[3] Robert Fourer, David M. Gay, and Brian W. Kernighan. AMPL: A Modeling Language for Mathematical Programming. Duxbury Thomson, 2003.

[4] Zonghao Gu, Edward Rothberg and Robert Bixby. Gurobi Optimization. http://www.gurobi.com.

[5] Anna Goldenberg, Alice X. Zheng, Stephen E. Fienberg and Edoardo M. Airoldi. A survey of statistical network models. arXiv: 0912.5410v1,2009.

[6] Paul Erdös and Alfréd Rényi. On Random Graph, I. Publicationes Mathematicae, 6: 290-297, 1959.

[7] Béla Bollobás. Random graphs. Cambridge University Press, 2nd edition, 2001.

[8] Garry Robins, Pip Pattison, Yuval Kalish and Dean Lusher. An introduction to exponential random graph (p*) models for social networks. Social Networks, 29: 173-191, 2007.

[9] Garry Robins, Tom Snijders, Peng Wang, Mark Handcock and Philippa Pattison. Recent developments in exponential random graph (p*) models for social networks. Social Networks, 29: 192-215, 2007.

[10] Ove Frank and David Strauss. Markov graphs. Journal of the American Statistical Association, 81: 832-842, 1986.

[11] Albert-László Barabási, Réka Albert, Hawoong Jeong. Mean-field theory for scale-free random networks. Physica A, 272: 173-187, 1999.

[12] Coen Bron and Joep Kerboscht. Finding All Cliques of an Undirected Graph [H]. Communications of the ACM, 16(9): 575-579, 1973.

[13] R. Ducan Luce. Connectivity and generalized cliques in sociometric group structure. Psychometrika, 15(2): 169-190, 1950.

[14] Richard D. Alba. A graph-theoretic definition of a sociometric clique. Journal of Mathematical Sociology, 3: 113-126, 1973.

[15] Robert J. Mokken. Cliques, clubs and clans. Quality and Quantity, 13: 161-173, 1979.

[16] Alberto Barobia and Ujué R. Tras. A geometric proof of the Perron-Frobenius theorem. Revista matemtica de la Universidad Complutense, 5: 57-63, 1992.

[17] Ian A. McCulloh, Helen L. Armstrong and Anthony N. Johnson. Social network analysis: with applications. Jonh Wiley and Sons, 2013.

[18] Balabhaskar Balasundaram, Sergiy Butenko and Svyatoslav Trukhanov. Novel Approaches for Analyzing Biological Networks. Journal of Combinatorial Optimization, 10: 23-39, 2005.

[19] Jean-Marie Bourjolly, Gilbert Laporte and Gilles Pesant. An exact algorithm for the maximum k-club problem in an undirected graph. European Journal of Operational Research, 138: 21-28, 2002.

[20] Balabhaskar Balasundaram, Sergiy Butenko and Illya V. Hicks. Clique Relaxations in Social Network Analysis: The maximum k-plex problem. Operations Research, 59(1): 133-142, 2011.

[21] Stephen B. Seidman and Brian L. Foster. A graph-theoric generalization of the clique concept. J.Mathematical Sociology, 6: 139-154, 1978.

[22] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. InterJournal, Complex Systems, 2006. http://igraph.org.

[23] Alejandro Rodríguez Villalobos. Grafos: Software para la construcción, edición y análisis de grafos.