CIMAT Centro de Investigación en Matemáticas, A.C.

A MODEL OF USERS INTERACTION IN A SOCIAL NETWORK USING RANDOM GRAPHS

T E S I S

Que para obtener el grado de

Maestro en Ciencias

con Especialidad en

Computación y Matemáticas Industriales

Presenta

José Guillermo Herrera Ramírez

Director de Tesis:

Dr. Octavio Arizmendi Echegaray

Co-Director de Tesis:

Dr. Juan Antonio Pichardo Corpus

Autorización de la versión final

Guanajuato, Gto., 9 de agosto de 2018

Dedicado

A mis padres:

María del Pilar Ramírez Juárez
José Alberto Herrera Escutia

A mis hermanos:

Tishbe Pilarh Herrera Ramírez
Angel Alberto Herrera Ramírez

Agradecimientos

A mi papá José Alberto Herrera Escutia, a mi mamá María del Pilar Ramírez Juárez, a mi hermana Tishbe Pilarh Herrera Ramírez y a mi hermano Angel Alberto Herrera Ramírez, por apoyarme toda la vida, por su cariño, por ser un ejemplo a seguir y por estar conmigo en la distancia. Los amo.

A mis profesores. Al Dr. Jean-Bernard Hayet quien fue mi tutor durante mi primer año de la maestría, al Dr. Johan Jozef Lode Van Horebeek por las enseñanzas durante sus cursos, a la Dra. Lilia Leticia Ramírez Ramírez por los comentarios que hizo al revisar este trabajo de tesis. Y especialmente, al Dr. Octavio Arizmendi Echegaray y al Dr. Juan Antonio Pichardo Corpus, por sus consejos, por guiarme en el desarrollo de esta tesis, y por enseñarme los aspectos teóricos y prácticos que fueron necesarios.

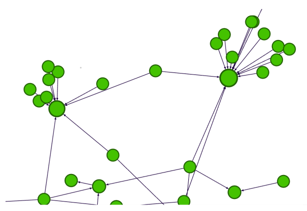
A mis compañeros de la maestría, de los otros programas del CIMAT, del laboratorio de robótica, y a los Fisiconautas de la BUAP, por su amistad y compañía. A todas las personas dentro y fuera del CIMAT que me han mostrado su afecto desde que llegué a Guanajuato. Y finalmente al CONACyT por la beca que me fue otorgada para estudiar la maestría.

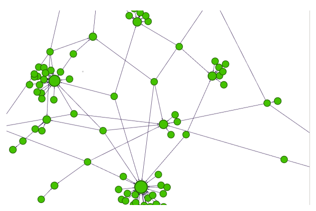
Contents

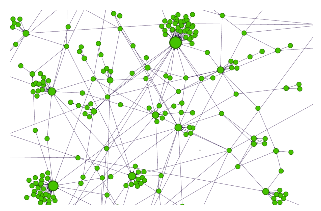
Introduction

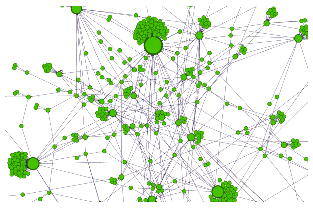
Twitter is an online social network, specifically, it is a micro-blogging composed of short messages on any topic. In these messages the users can interact with other users. Thus, due to the interaction of users is significant in different areas such as marketing and polity, we try to model this behavior using random graphs.

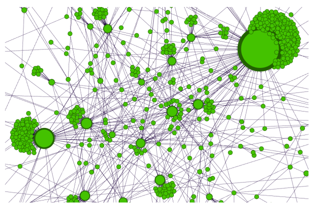
The objective of this work is to model the interaction of users in a social network through time. We are interested in describing things like: How many links are added at every step and how they appear at every step. If there are differences between the links from the current nodes and from the new nodes. Which is the distribution for in-degree and out-degree through time. What is the value of the biggest in-degree and out-degree. Even more, how is the network at the end of the process. In other words, we want to know how the network grows. Figure 1 shows visually what we want to model. We want to know how to pass from 1(a) to 1(b), then from 1(b) to 1(c), and so on. Then we want to develop a model with a similar behavior.


(a) Network 1


(b) Network 2


(c) Network 3


(d) Network 4


(e) Network 5

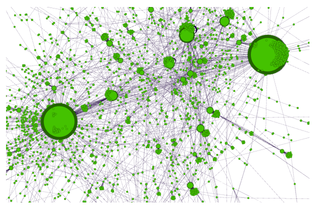

(f) Network 6

Figure 1: Evolution of a network over time.

The thesis is divided in 5 chapters. Chapter 1 is dedicated to explain the graph and network theory needed to understand this thesis. We show two models to generate random graphs. The first one is the model proposed by Erdös and Rényi (ER), the second one is the model proposed by Barabási and Albert (BA). Furthermore, we will see what is a scale free graph and its relation with a graph generated with the BA model. Chapter 2 is dedicated to explain the source and properties of the data we will try to model. First, we explain how to build the networks. Then, we show an analysis of those networks in order to construct a representative model. Chapter 3 is dedicated to develop the proposed model, explain its parameters and how to set them for a data set. Besides, we talk about how we compare the real network with our modeled network. Chapter 4 is dedicated to present the results of setting the parameters and comparing the real data with the modeled network. In addition, we present a simple result of how we can use the proposed model in order to predict information in the future. Finally, in Chapter 5 we present the conclusions and the future work.

Chapter 1

Random Graphs

In this chapter we present the theoretical framework to study graph and network theory. Next, we show the model proposed by Erdös and Rényi in 1959 and 1960 to construct random graphs. Later, we study scale-free random graphs and the model proposed by Barabási and Albert in 1999 to construct them.

1.1 Graph and Network Theory

The first example that introduces graph theory was proposed by Leonard Euler in 18th century. He solved the problem known as the Seven Bridges of Kögnisberg. In Figure 1.1 we can see a representation of the 7 bridges that existed in that city in 1736, the parcels of land are labelled by A, B, C and D. This problem ask about the possibility of visiting all the city crossing every bridge once and only once.

The solution was given with a graph like in Figure 1.2 where the parcel of lands are the vertices and the bridges are the edges. Due to Euler's solution, nowadays we call an *Eularian trail* a trail in a graph which crosses every edge exactly once. We will next define some basic aspects of graph theory.

Definition 1.1. *A* ***graph*** *is a pair $G = (V, E)$ of sets such that $E \subseteq V \times V$; the elements of V are called* ***vertices*** *and the elements of E are called* ***edges****.*

If the set of edges E is symmetric (i.e. $(u, v) \in E \Rightarrow (v, u) \in E$) it is said that G is an *undirected graph.* On the other hand, if E is nonsymmetric it is said that G is a *directed graph* or *digraph.* Also, it is said that a graph

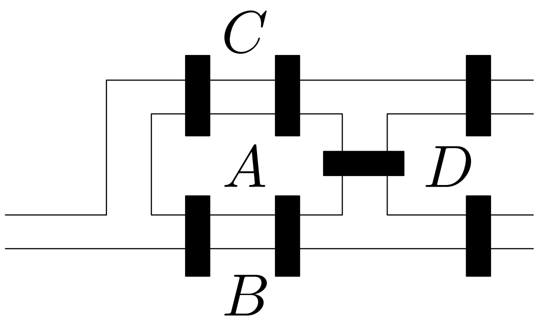


Figure 1.1: The Seven Bridges of Kögnisberg.

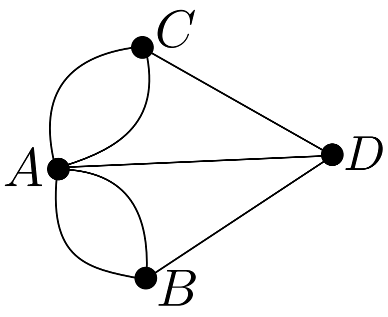


Figure 1.2: Graph of the Seven Bridges of Kögnisberg.

without self loops is a *simple graph.* In this work we use simple digraphs to represent the data, but we just say *graph* to refer to them unless we specify else. If we work with different graphs at the same time, we denote the set of vertices and the set of edges of G by $V(G)$ and $E(G)$ respectively.

Definition 1.2. *A **path** in a graph G is a non-empty graph $P = (V, E)$ of the form $V = \{x_0, x_1, ..., x_k\}$ and $E = \{(x_0, x_1), (x_1, x_2), ..., (x_{k-1}, x_k)\}$, where the x_i are all distinct.*

The number of edges of a path is its *length.* We often refer to a path by a natural sequence of its vertices, $x_0, x_1, ..., x_k$, and we say that x_0 and x_k are linked by a path.

Definition 1.3. *A graph G is **connected** if it is non-empty and any two of its vertices are linked by a path in the underlying undirected graph of G.*

If the graph is disconnected we are interested in the connected components. If the connected component with more vertices is unique and it is much larger than the other components we call it the *giant component*, sometimes denoted by GC.

Definition 1.4. *Two vertices v and w of G are* ***adjacent*** *or* ***neighbours*** *if (v, w) is an edge of G. We denote the set of neighbours of a vertex $v \in G$ by $N(v)$.*

Definition 1.5. *The* ***degree*** *of a vertex v is the number $|N(v)|$. We denote the degree of v by $d(v)$.*

In other words, the degree of a vertex is the number of edges related to it. In a directed graph the *in-degree* of v es the number of edges that it is incident from, and the *out-degree* is the number of edges that it is incident to. We denote the in-degree and out-degree of a vertex $v \in G$ by $d_{in}(v)$ and $d_{out}(v)$, respetively.

Definition 1.6. *The* ***minimum degree*** *and the* ***maximum degree*** *of a graph are denoted by δ and Δ respetively. More precisely,*

$$\delta := \min\{d(v)|v \in V\}$$
$$\Delta := \max\{d(v)|v \in V\}$$

Definition 1.7. *The* ***average degree*** *of a graph is denoted by $\bar{d}$, and it is defined as*

$$\bar{d} := \frac{1}{|V|}\sum_{v \in V} d(v)$$

In a directed graph we denote by δ_{in} the *minimum in-degree*, Δ_{in} the *maximum in-degree*, $\bar{d}_{in}$ the *average in-degree*, δ_{out} the *minimum out-degree*, Δ_{out} the *maximum out-degree* and $\bar{d}_{out}$ the *average out-degree.* It is satisfied that $\delta \leq \bar{d} \leq \Delta$, $\delta_{in} \leq \bar{d}_{in} \leq \Delta_{in}$ and $\delta_{out} \leq \bar{d}_{out} \leq \Delta_{out}$. Also, it can be proved that $\bar{d}_{in} = \bar{d}_{out}$.

Definition 1.8. *The* ***empirical degree distribution*** *of a graph G is denoted by p_k. It gives the probability that a randomly selected vertex in the graph has degree k. Thus*

$$p_k = \frac{|\{v \in G \text{ such that } d(v) = k\}|}{|V|}.$$

In general, it is named just as the *degree distribution.* In addition, p_k is almost always identified with an approximating continuous distribution. In directed graphs there are the analogous *in-degree distribution* and *out-degree distribution.* Poisson, power law and log-normal are degree distributions that usually appear in graphs.

Sometimes, we need to describe a graph with matrices to be helped by linear algebra. Here we present two informative matrices related to a graph.

Definition 1.9. *The* ***adjacency matrix*** *of a grah G is denoted by A, and it is defined as*

$$A_{ij} := \begin{cases} 1 & \textit{if } (v_i, v_j) \in E \\ 0 & \textit{otherwise.} \end{cases}$$

Let's note that if G is an undirected graph, A is a symmetric matrix. On the other hand, if G is a directed graph, it has non-symmetric adjacency matrix.

Definition 1.10. *The* ***laplacian matrix*** *of a grah G is denoted by L, and it is defined as $L := D - A$ where*

$$D_{ij} := \begin{cases} d(v_i) & \textit{if } i = j \\ 0 & \textit{otherwise.} \end{cases}$$

If we work with different graphs at the same time, we denote the adjacency matrix and the laplacian matrix of G by A_G and L_G respectively.

Let's note that for every graph, the laplacian matrix sum zero by row. Thus, $L\mathbf{1} = 0 = 0\mathbf{1}$, where $\mathbf{1} = (1, \ldots, 1)^t$. It means that $\mathbf{1}$ is an eigenvector and 0 is an eigenvalue of L.

Finally, what is the difference between *graph* and *network*? In general terms they can be thought as the same. However, we will do a distinction as Barabási does in [4]. The network terminology refers to real systems, and we want to recall the idea that two different networks can be represented by the same graph. When we refer to a *network* we will use the words *node* and *link* instead of *vertex* and *edge.* Since in this work we model interaction of users, the *network* terminology will highlight the fact that the users or nodes are more than simple dots, even more, the nodes are connected or linked. Also, with this distinction we can talk about the graphical representation of the network just as a *graph.*

To find complementary definitions and a larger introduction to graph and network theory we recommend [4], [9], [12] and [17].

1.2 Erdös-Rényi Model

In real life we can find many data that can be studied with graph and network theory. In fact, sometimes the data could be analyzed if it is part of a random process, thus we create *random graphs*. For example, we can describe probabilistic rules to create edges in a graph with a fixed number of vertices.

Erdös and Rényi proposed a model for random graphs in 1959. This model chooses uniformly a graph among all the graphs with n vertices and M edges and it is denoted by $G(n, M)$ [10]. Afterwards, in 1960, they proposed a new model. In this model we start with n isolated vertices. Then, for every pair of vertices and with probability $p > 0$ we add an edge between them, independently of the other vertices [11]. We denote this model by ER and the graph generated by $G(n, p)$.

This model has two important characteristics. The degree distribution of the graph generated by the ER model follows a Poisson distribution asintotically in n. Also, if we think in many graphs $G(n, p)$ with a fixed value of n, but with differents values of p, we will find a *phase transition*. In the regime that $p \in (0, \frac{1}{n})$ we have tiny clusters with a small number of edges. On the other hand, if $p \in (\frac{1}{n}, 1)$ the size of the largest component is significant. In this regime numerous isolated components coexist with the giant component. The point $p = \frac{1}{n}$ is called the critical point. It is important to note that, since the distribution in the ER model is Poisson, then the average degree at this point is $d = np = 1$. These and some more characteristics of this model can be found in [4], [10] ,[11] and [21].

An example with the ER model using $n = 200$ and $p = 0.3$ was simulated. In Figure 1.3 the graph created is shown, the size of the vertices is proportional to its degree. In Figure 1.4 the degree distribution of the graph is shown. As expected, it looks like a Poisson distribution with mean 60.

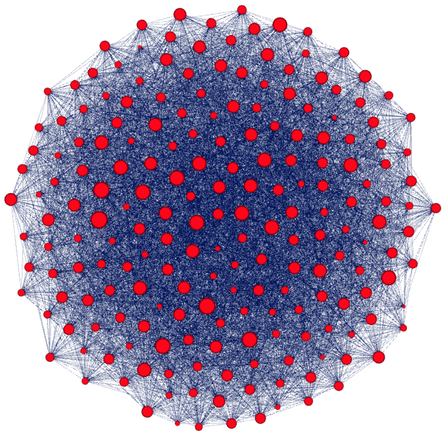


Figure 1.3: Example of a random graph generated by the ER model with $n = 200$ and $p = 0.3$.

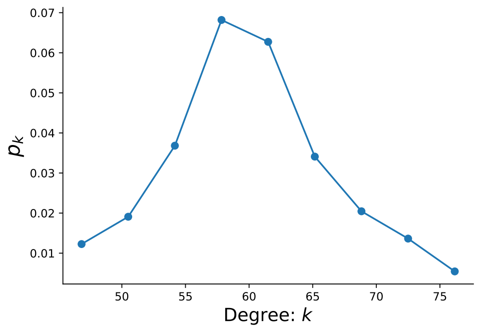


Figure 1.4: Degree distribution of the graph in Figure 1.3.

1.3 Scale-free Random Graphs

In real-world many networks are different to networks modeled by the *ER model*. As an example, let's see the structure of a network generated by 300,000 nodes and 1.5 million links. The nodes in this network are documents in the WWW and the edges are links among them. Different to ER model this network has nodes with very large in-degree, some nodes have more than 50 links, and a few other have more than 500 links [4].

As we can imagine, the in-degree distribution in this WWW network does not follow a Poisson distribution like in an ER model. However, if we plot the data on a log-log scale we can see that the distribution looks like a straight line. In Figure 1.5, we can see in gray the in-degree and the out-degree distribution in a log-log scale, in purple the line that fits the data, and in green the Poisson distribution with average $\bar{d}_{in} = \bar{d}_{out} = 4.6$, this average comes from the data.

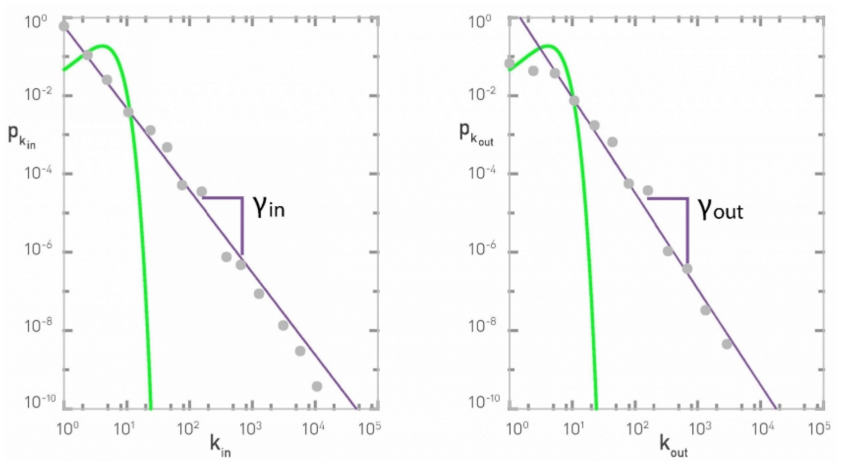


Figure 1.5: Left side shows the line that fits the in-degree distribution and the right side shows the line that fits the out-degree distribution in a log-log scale. Image from [4].

Because of the straight line in Figure 1.5, the degree distribution p_k seems to follow an inverse of a power of k. This is known as a *power law distribution.* Note that if p_k is similar to $k^{-\lambda}$ then $\log(p_k)$ is similar to $-\lambda \log(k)$, this explain the straight line.

Supose that there exists a normalizing constant C and a constant exponent λ that no depend on k, such that $p_k = Ck^{-\lambda}$. Then

$$\int_{k_{min}}^{\infty} p_k dk = 1$$

where k_{min} is the minimum degree which fits the line. Thus

$$\int_{k_{min}}^{\infty} Ck^{-\lambda} dk = 1 \Rightarrow C = \frac{1}{\int_{k_{min}}^{\infty} k^{-\lambda} dk},$$

therefore,

$$C = -(1-\lambda)k_{min}^{(\lambda-1)},$$

and hence,

$$p_k = (\lambda-1)k_{min}^{(\lambda-1)}k^{-\lambda}.$$

This is a power law distribution with degree exponent λ. It is common to use this continuous distribution instead of a discrete distribution. Note that if we want to work with a discrete function, the value of C is complicated to compute. In the discrete framework

$$C = \frac{1}{\sum_{k=1}^{\infty} k^{-\lambda}}.$$

The denominator is called the Riemann-zeta function and it is not easy to work with it.

It is said that this distribution is *invariant of scale* because for $T \in \mathbb{N}$

$$\begin{aligned} p_{Tk} &= (\lambda-1)k_{min}^{(\lambda-1)}(Tk)^{-\lambda} \\ &= T^{-\lambda}(\lambda-1)k_{min}^{(\lambda-1)}k^{-\lambda} \\ &= T^{-\lambda}p_k. \end{aligned}$$

We have that, clearly $P_{1000} < P_{10}$ but, $P_{1000} \propto P_{10}$ and it means, in a discret framework, that selecting randomly a vertex, the probability that this vertex has degree 1000 is proportional to the probability of this vertex has degree 10.

This distribution is interesting also because if we calculate its the second moment

$$< k^2 >= \int_{k_{min}}^{k_{max}} k^2 p_k dk = (\lambda-1)k_{min}^{\lambda-1}\left(\frac{k_{max}^{(3-\lambda)} - k_{min}^{(3-\lambda)}}{3-\lambda}\right)$$

where k_{max} is the maximum degree which fits the line. We can note that when $k_{max} \to \infty$, if $\lambda \leq 3$, then $< k^2 >$ diverges. Denoting by N the number of vertices in the graph, Barabási argues that

"The divergence of $< k^2 >$ for large N indicates that the fluctuations around the average can be arbitrary large. This means that when we randomly choose a node, we do not know what to expect: The selected node's degree could be tiny or arbitrarily large. Hence networks with $\lambda \leq 3$ do not have meaningful internal scale, but are ***scale-free****" [4].*

In other words, in a scale-free graph it is completely possible to find vertices with small and vertices with very large degree, which is very different to ER model. A scale-free graph is a graph whose degree distribution follows a power law.

1.4 Barabási-Albert Model

In 1999 Albert-László Barabási and Réka Albert proposed a model in [1] that generate scale-free random graphs. This model is called the *BA model* and it is based in the idea that the rich gets richer. Thus, when we add a vertex to the graph it is more probably linked to the vertices with the biggest degrees. With this in mind, this model generates graphs only with two steps: growth and preferential attachment. Here we describe the simplest version of this model.

To get a graph with N vertices, start with 2 vertices, v_0 and v_1, linked by one edge, $j = 0$ and do $N - 2$ times these 2 steps:

1. **Growth.** Add the vertex v_{j+2}.
2. **Preferential attachment.**
    - Add the edge (v_{j+2}, v_i) choosing v_i, with probability
    $$p = \frac{d(v_i)}{\sum_{i=0}^{j+1} d(v_i)}$$
    where $d(v_i)$ denote the degree of vertex v_i, and $i \in \{0, ..., j+1\}$.
    - Do $j = j + 1$.

At the end of the process, this model generates a scale-free random graph with degree exponent $\lambda = 3$, the proof can be consulted in [4]. Specifically, it generates a connected graph without loops, a tree.

As we said, this is the simplest version of the model. In fact, in [1] the description of the model is different. First, they said that we could start with a graph with n vertices, but they do not say which is the initial configuration of these vertices. So, if we do not have edges, the probability described above does not make sense. Second, they propose that we could add m edges instead of one edge to every new vertex, but they does not say if this process is adding one by one changing probabilities or all simultaneously. Even more, they do not say if we could generate a multigraph, because if the m edges are independent that could happen.

It is important to say that the model we presented above does not have these problems. However, in 2001, Bollobás, Riordan, Spencer and Tusnády investigate the model rigurosly and proposed the *Linearized Chord Diagram* in [5] to solve the problems. In [21] we can find a general description of the preferential attachment model based in BA model but considering just one vertex with one self-loop as the initial configuration, and with the posibility of get multilinks.

This model generates scale-free random graphs based in the preferential attachment idea, but we have to say that there exists models that generate scale-free random graphs based in other properties of the graph.

An example with the BA model using n = 200 was simulated. In Figure 1.6 the graph built is shown, the size of the vertices is proportional to its degree. As we can see, this model generates trees. In Figure 1.7 the degree distribution of the graph in log-log scale is shown. As expected, it looks like a straight line.

Non-linear Preferential Attachment

The preferential attachment described in step 2 of the BA model can also be written as follows: Add the new edge choosing v_i, with probability

$$p = \frac{d(v_i)^\alpha}{\sum_{i=0}^{j+1} d(v_i)^\alpha}$$

where $\alpha = 1$. With this fixed value of α, we say that the preferential attachment is linear, and we obtain scale-free graphs. However, we can work with differents values of α. For $\alpha \in (0, 1)$ we say that the preferential attachment is sublinear. And, for $\alpha > 1$ we say that the preferential attachment is superlinear. In sublinear preferential attachment, the distribution of p is

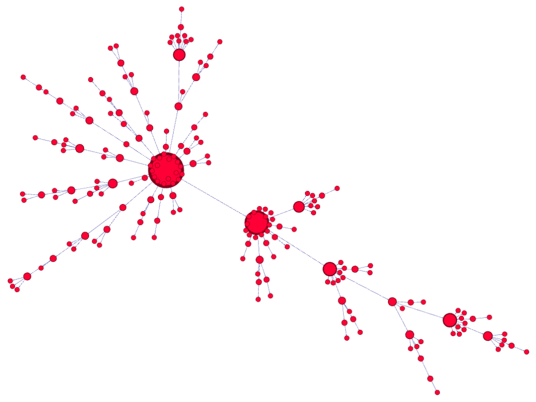


Figure 1.6: Example of scale-free random graph generated by the BA model with $n = 200$.

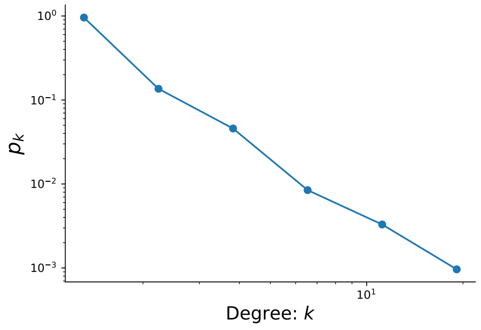


Figure 1.7: Degree distribution in log-log scale of the graph in Figure 1.6.

more similar to a uniform distribution. That's why, in this case the degree values follows the stretched exponential distribution. In superlinear preferential attachment the rich-gets-richer process is accelerated. Thus, we get a few vertices connected with almost all the rest vertices. Complementary information and more results can be consulted in [4], [7], [20].

We simulated an example with non-linear preferential attachment using $n = 200$ and $\alpha = 0.5$. In Figure 1.8(a) the graph built is shown, the size of the vertices is proportional to its degree. In addition, an example with non-linear preferential attachment using $n = 200$ and $\alpha = 1.5$ was simulated. In Figure 1.8(b) the graph built is shown, the size of the vertices is proportional to its degree. For these graphs, the degree histogram is not very informative, however we can visualize the maximum degree Δ for several simulations. Figure 1.9 shows an histogram with the maximum degree Δ for 100 simulations in sublinear model ($\alpha = 0.5$), linear model ($\alpha = 1.0$) and superlinear model ($\alpha = 1.5$), using $n = 250$. Clearly, they return values of Δ in different ranks and the dispersion is also very different.

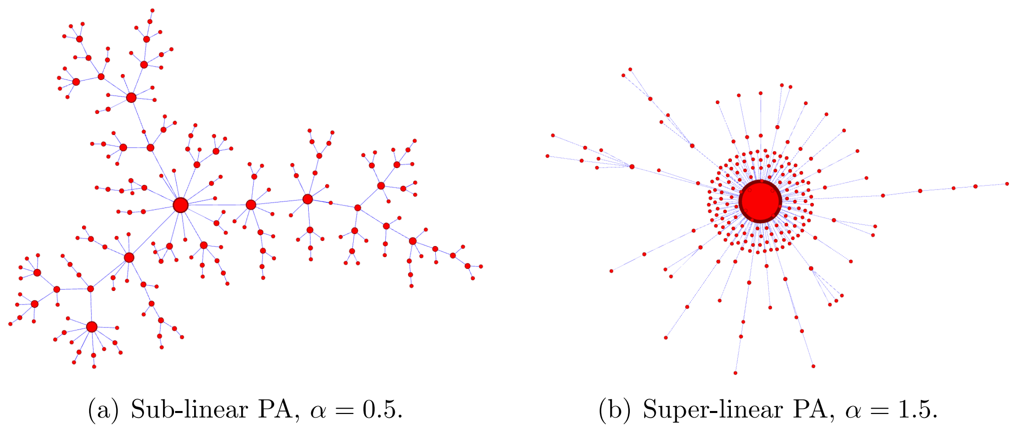

(a) Sub-linear PA, $\alpha = 0.5$. (b) Super-linear PA, $\alpha = 1.5$.

Figure 1.8: Examples of non-linear preferential attachment using $n = 200$.

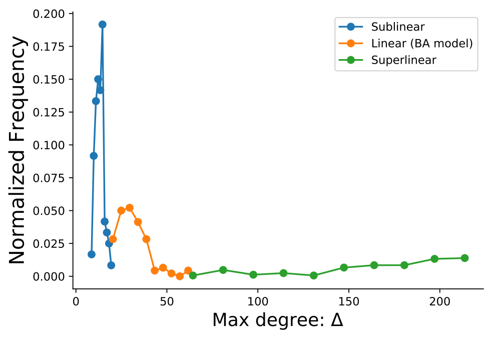


Figure 1.9: Maximum degree Δ for 100 simulations using $n = 250$ and $\alpha \in \{0.5, 1, 1.5\}$.

Chapter 2

Users Interaction Network in Twitter

This chapter is dedicated to explain the source and properties of the data we will try to model. In the first section we describe how the data was obtained and how to pass from the data to networks. Afterwards, in the second section we detail significant features in the networks in order to create a model that fits the data.

2.1 Build the Network

Twitter is an online social network, specifically, it is a micro-blogging composed of short messages on any topic. The word *tweet* refers to the action of post a message as well as to the message itself. In every tweet, users can do *retweets* [1] and *mentions* [2]. Besides, in the same tweet they can employ *hashtags*[3] in order to clasify his message in a topic.

Twitter has an Application Programming Interfaces (API) to access to its data. In this work we used the package Tweepy to use the API from Python. The data we use are tweets collected by streaming, this means that we collect tweets in real time. This is useful to download a high volume of twitter messages.

[1]A retweet is a repeated tweet from other user.

[2]A mention is a cite to another user preceded by @.

[3]A hashtag is a word or phrase without spaces preceded by a #.

All tweets we collected were downloaded with a fixed query [4]. We use as query a set of hashtags where the people are talking about the same topic. The list with all these tweets is called a *conversation*.

From every single tweet we get a lot of information: name of user who did the tweet, description of the user, friends count, followers count, text in the tweet, if it is a retweet, a list of users mention, a list of hashtags in the text or in the retweet, the exactly hour in wich the tweet was created, etc. However, to generate the users interaction network we focus on mentions and retweets. This network has as nodes all users that interact in the conversation even if they interact indirectly. The users who interact indirectly are the users who were mentioned in the conversation and they never response. The users who interact directly are the users who really tweet in the conversation. If u, v and w are users in the conversation we add the directed link (u, v) if one of these cases occurs:

1. u writes a tweet in which v is mentioned.
2. u does a retweet of the tweet of v.
3. w mentions v in a tweet, and u does a retweet to the tweet of w.

Let's note that in case 3 some links were added because of case 1 and case 2. In case 3, first the link (w, v) was added , then, 2 links were added at the same instant, (u, w) and (u, v). It is not strange to add more than one link at the same moment. For example, u can write a tweet in which he mentions v and w. Thus, at the same moment the links (u, v) and (u, w) are added to the network. It is important to note that, if a node is added to the network there is it because it has an interaction with itself or with another user in the conversation. It means that the only way that a component in the graph has just one node, is if it interacts just with itself.

These cases show how to add links in the network, however we used a new variable: time. Given that we have the exact time in which each tweet was made, we have the exact time in which each link was added. Hence, for every conversation analyzed, we have a list of edges, with a source and a target, besides the time of creation of that link. The list of edges described above is used to create a sequence $G_1, G_2, ..., G_{100}$ of networks. Although the time

[4]A query is a list of search terms to download. For example, user names, hashtags, key words, etc.

is continuous it is enough to work with 100 networks in order to analyze the interactions of users through time. We refer to G_t as the graph at time t.

For every conversation, we divide the total time in 100 windows of time uniformly. G_1 is the network such that the links are the edges in the list in the first window of time. G_2 is the network such that the links are the edges in the list until the second window of time, and so on. At the end of the process we generate 100 networks such that for every $t \in \{1, ..., 99\}$, $V(G_t) \subseteq V(G_{t+1})$ and $E(G_t) \subseteq E(G_{t+1})$.

2.2 Real Networks Analysis

To propose a model we analyze different data sets. However, in this section we just present the analysis of 3 of them. The proposed model will be compare with these data sets in Chapter 4.

The first data set we present was downloaded on Jun 29th, 2017 at 13:35 hours. It was during the FIFA Confederations Cup. The query used was just one hashtag: *#MexicovsAlemania*. This data set is conformed by 13645 tweets and they generate a network with 6909 nodes and 7936 links. The second data set we present was downloaded on Sep 19th, 2017 at 16:34 hours. It was few hours after the earthquake occured in Puebla. The query used was a list of hashtags: *#sismo*, *#fuerzamexico*, *#prayformexico* and *#ayudaciudadana*. This data set is conformed by 8000 tweets and they generate a network with 7966 nodes and 7869 links. The third data set we present was downloaded on Jun 12th, 2018 at 21:10 hours. It was during the third presidential debate in Mexico. The query used was two hashtags: *#DebateINE* and *#DebatePresidencial2018*. This data set is conformed by 10000 tweets and they generate a network with 5677 nodes and 7308 links. The networks of this data sets will be called: Soccer network, Earthquake network and Debate network, respectively.

A feature that we use to analyze the sequence of networks is the number of links. We note that the number of total links that has the network over time can be approximated with a linear regression. In Figure 2.1 we show the number of total links over time and its linear adjustment for each network. In 4.2(b) and 4.2(c) the residual sum of squares (RSS) is almost 0.

After that, we analyze the in-degree and out-degree distributions over time. We realize that all of them looks like heavy tailed distributions. The in-degree and out-degree distributions for the networks at different times t,

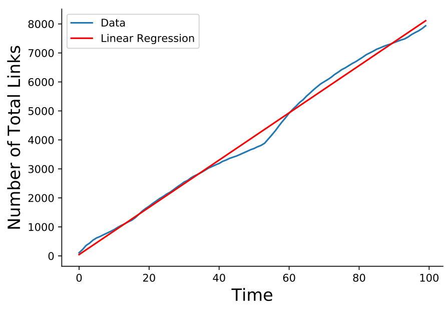
$t \in \{25, 50, 75, 100\}$, is shown in log-log scale in Figure 2.2.

Let's remember that we talked about how the giant component emerges in ER random graphs in Chapter 1. We analyze if there exists a GC by looking at the proportion of nodes and links in the larger component of the networks. Thus, we realize that the GC exist and it emerges soon in the process. Figure 2.3 shows the proportion of nodes and links in the GC for the networks over time. Moreover, we analyze the underlying undirected graphs and we contrast the average degree vs the proportion of nodes in the GC. In Figure 2.4, the contrast of these features shows the fact that the average degree is always less than 1, opposite to the ER model.

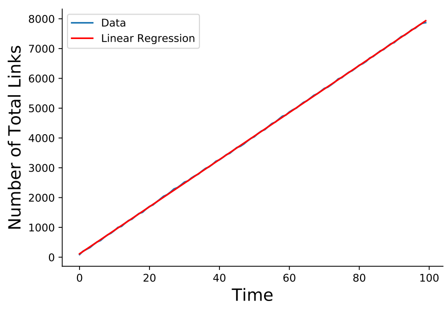
Before continuing we have to make a clarification. For every step $t \in \{1, 2, ..., 100\}$, we call *new nodes* and *new links* at time t, to the nodes and links added to the network at that time. For every step $t \in \{2, ..., 100\}$, we call *current nodes* and *current links* at time t, to all nodes and all links that the network has at that time, but that are not part of the new nodes or new links. It means, for a sequence of networks $G_1, G_2, ..., G_{100}$, G_1 has new nodes and new links but does not have current nodes or current links, and, for $G_1, G_2, ..., G_{100}$ the number of nodes is equal to sum the number of current nodes plus the number of new nodes. Thus, we observe the activity of the nodes added in the network at every step. We analyze what is the proportion of the new links that come from the new nodes. A normalized histogram with the proportion of new links in new nodes for the networks is shown in Figure 2.5. It is clear that most of the new links are from the new nodes, the rest of the links are from the current nodes.

We know that many of the new links are from new nodes, but now we need to know how many links are added by every new node. To analyze this idea we observe the average out-degree in new nodes and we found that the mean is nearly to 1. Finally, we analize how many of the new links go to the new nodes. We found that the average in-degree in new nodes is nearly 0. Figure 2.6 shows the average out-degree in new nodes and the average in-degree in new nodes for each network. If we analyze both ideas, we find that most of the links from the new nodes go to the current nodes. This last observation in many senses resembles the degree dynamics in the Barabási-Albert model [5]. Hence, we will model the users interaction network with a modification of it.

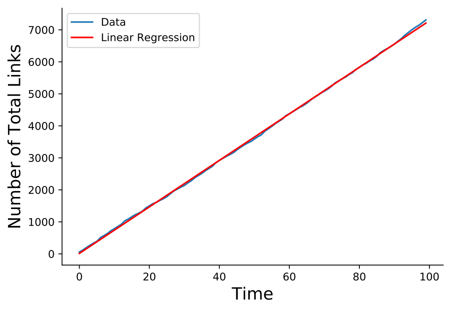
[5]There is a larger description of the degree dynamics in the Barabási-Albert model in Chapter 5, Section 5.4 of [4].



(a) Soccer Network

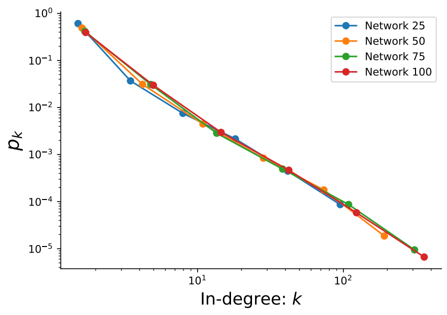


(b) Earthquake Network

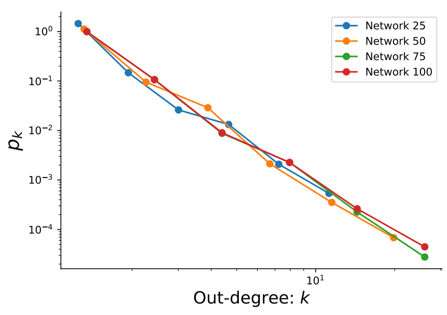


(c) Debate Network

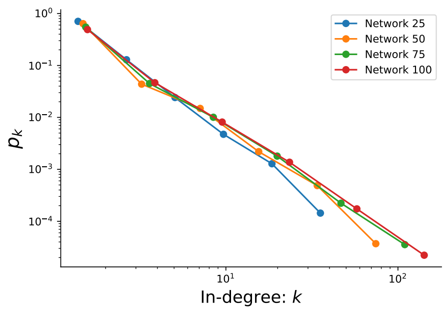
Figure 2.1: Number of total edges over time.



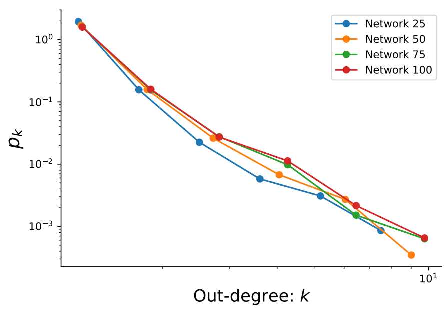
(a) Soccer Network



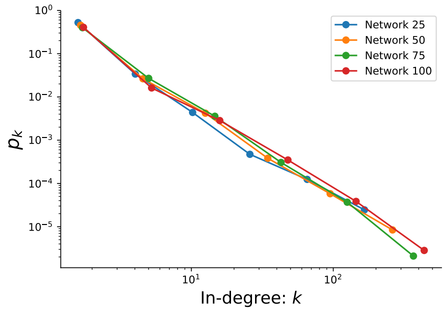
(b) Soccer Network



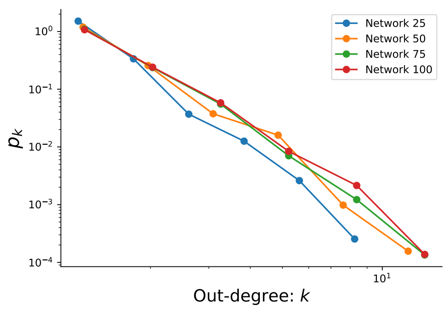
(c) Earthquake Network



(d) Earthquake Network

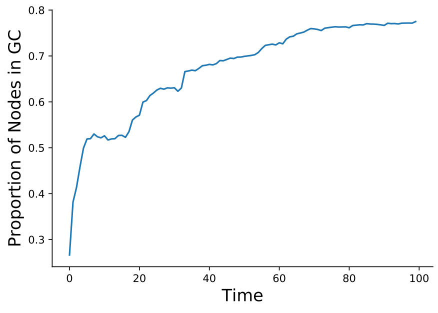


(e) Debate Network

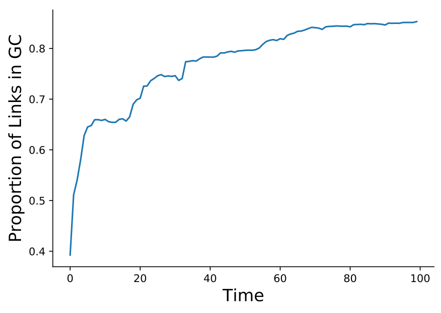


(f) Debate Network

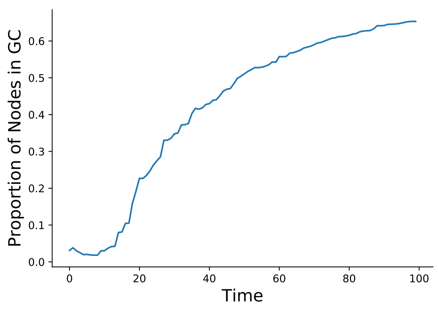
Figure 2.2: In-degree and out-degree distributions at different times.



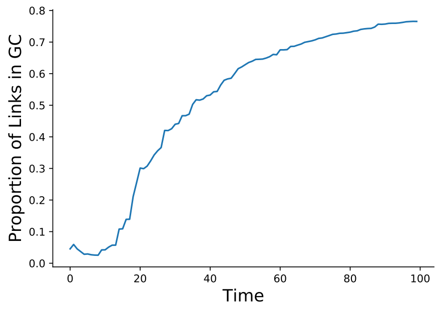
(a) Soccer Network



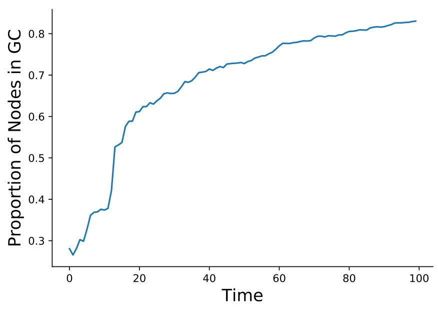
(b) Soccer Network



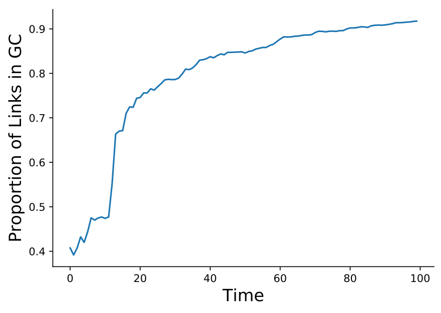
(c) Earthquake Network



(d) Earthquake Network

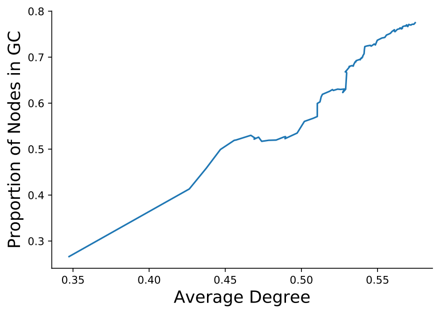


(e) Debate Network

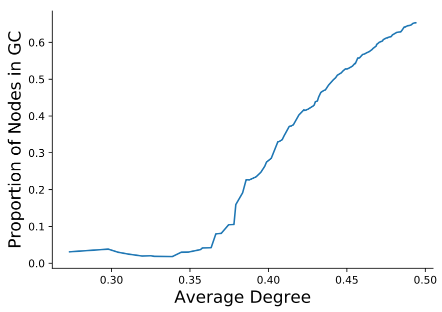


(f) Debate Network

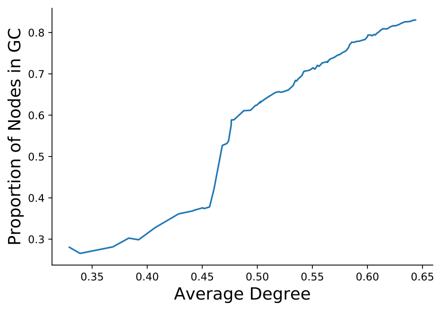
Figure 2.3: Proportion of nodes and links in the giant component over time.



(a) Soccer Network

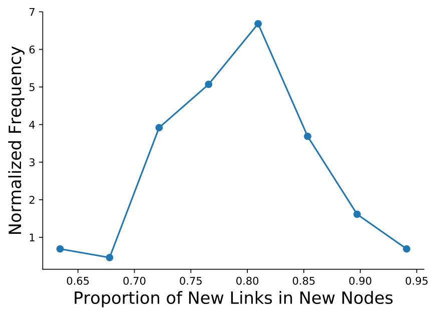


(b) Earthquake Network

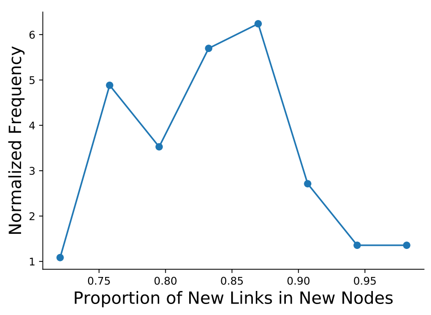


(c) Debate Network

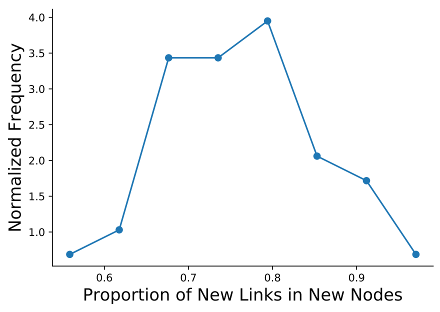
Figure 2.4: Average degree vs proportion of nodes in the GC over time.



(a) Soccer Network

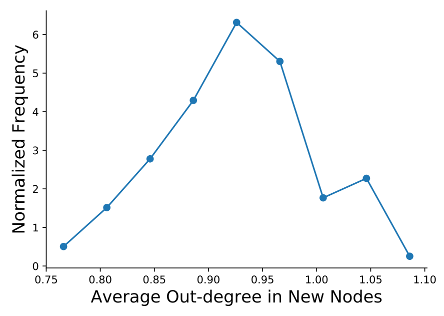


(b) Earthquake Network

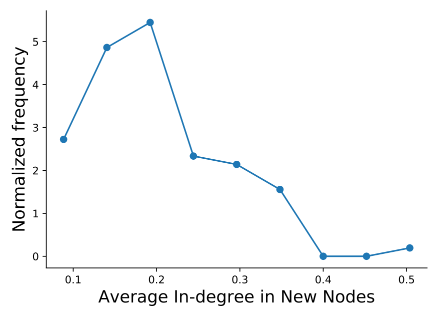


(c) Debate Network

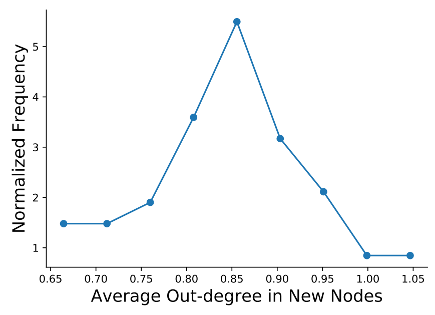
Figure 2.5: Proportion of new links added by the new nodes over time.



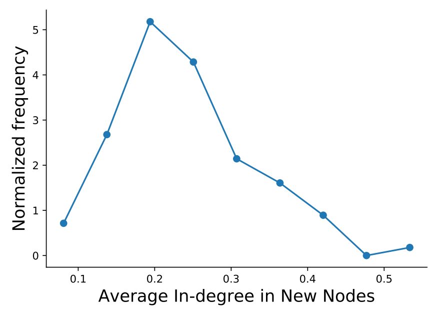
(a) Soccer Network



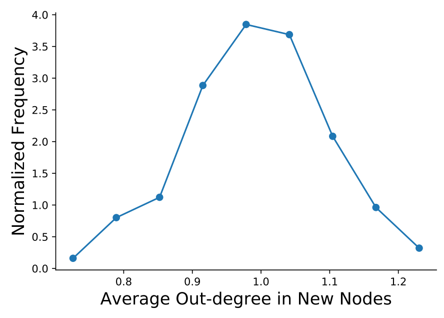
(b) Soccer Network



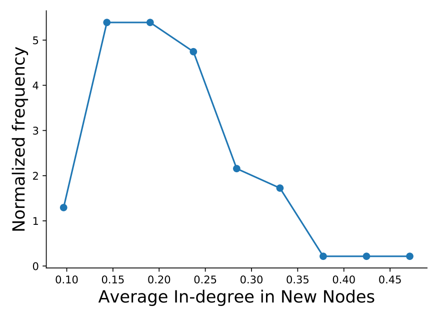
(c) Earthquake Network



(d) Earthquake Network



(e) Debate Network



(f) Debate Network

Figure 2.6: Average out-degree and average in-degree in the new nodes over time.

Chapter 3

Users Interaction Network Model

In this chapter we present a proposed model of interaction of users in a social network through time. Next we describe how we set the parameters to each data set. Finally we explain how we compare the real network with our modeled network.

3.1 Proposed Model

This model is based in some facts we described in Chapter 2. Because of the number of edges through time fits lineary in real data, then, in the proposed model we try to add, at every step, a fixed number of edges. Also, we divide the vertices in current vertices and new vertices as in the real networks. Moreover, we add new edges from current vertices and from new vertices to resemble the behavior of the real networks.

In real data, we found that the new vertices which are being added at every step have average in-degree close to 0 and average out-degree close to 1. Thus, to keep it simple, the new vertices in the model always have in-degree 0, it means that all new edges, from the current vertices and from the new vertices, will be directed to the current vertices. In addition, with the same idea of having a simple model, the out degree of every new vertex will be 0 or 1. Finally, because the real networks seem to have heavy tailed in-degree and out-degree distribution, different to an ER graph, we use preferential attachment in both directions to create new edges. Let's remember that the linear preferential attachment induces a heavy-tailed distribution, specifically, a power law.

With all these characteristics we considered different models. However, we present only the model that fits better to the data. It produces a sequence of graphs denoted by $(G_t)_{t\geq 1}$. The sequence of graphs is such that, for every $t \in \{1, ..., N-1\}$, $V(G_t) \subseteq V(G_{t+1})$ and $E(G_t) \subseteq E(G_{t+1})$.

In this model we need an initial network, and a list with the number of vertices wich will be added at every step. We use exaclty the initial network and the list that comes from the data. Besides, in this model we have 5 parameters $[\theta, \rho, \alpha_{in}, \alpha_{out}, \beta_{in}]$, where $\theta \in \mathbb{N}$ and $\rho, \alpha_{in}, \alpha_{out}, \beta_{in} \in \mathbb{R}$. The parameters θ and ρ are related with how many edges are generated for the current and new vertices. The number of edges that will be added at every step is denoted by θ. $\rho \in (0,1)$ is a value such that the floor of $(1-\rho)\theta$, denoted by $\lfloor (1-\rho)\theta \rfloor$, represents the number of new edges generated at every step in the current vertices. The parameters $\alpha_{out}, \alpha_{in}$ and β_{in} are exponents related to the preferential attachment. Specifically, $\alpha_{in} \in (0,1)$ and $\alpha_{out} \in (0,2)$ affects only at the current vertices and $\beta_{in} \in (1, 1.5)$ affects only at the new vertices. These intervals were fixed by empirical exploration.

In this model we have 3 updates for every step: *current preferential attachment*, *growth* and *new preferential attachment*. Let $l_t = \sum_{s=1}^{t} n_s$ the number of vertices at time t. Start with $t = 2$, a graph G_1 such that $|V(G_1)| = n_1 \neq 0$, and a list $[n_2, ..., n_N]$ with the number of vertices wich will be added at every step. Do $N-1$ times these 3 updates:

1. **Current preferential attachment.** Create a set of edges, E_t, such that $|E_t| = \lfloor (1-\rho)\theta \rfloor$. Every edge (v_i, v_j) in E_t is made independently choosing v_i in the current vertices, $\{v_1, v_2, ..., v_{l_{t-1}}\}$, with probability

$$p_{\alpha_{out}} = \frac{[d_{out}(v_i)+1]^{\alpha_{out}}}{\sum_{w=1}^{l_{t-1}} [d_{out}(v_w)+1]^{\alpha_{out}}},$$

and choosing v_j also in the current vertices with probability

$$p_{\alpha_{in}} = \frac{[d_{in}(v_j)+1]^{\alpha_{in}}}{\sum_{w=1}^{l_{t-1}} [d_{in}(v_w)+1]^{\alpha_{in}}},$$

where α_{out} and α_{in} allow non-linear preferential attachment.

2. **Growth.** Create a set of new vertices $V_t = \{v_{l_{t-1}+1}, ..., v_{l_{t-1}+n_t} = v_{l_t}\}$.

3. **New preferential attachment.**

   - Create a set of edges, F_t, such that $|F_t| = \min\{\lfloor \rho\theta \rfloor, n_t\}$. Every edge (v_i, v_j) in F_t is made choosing v_i in V_t without replacement and choosing v_j in the current vertices with probability

   $$p_{\beta_{in}} = \frac{[d_{in}(v_j)+1]^{\beta_{in}}}{\sum_{w=0}^{n_1}[d_{in}(v_w)+1]^{\beta_{in}}},$$

   where β_{in} allow non-linear preferential attachment.
   - Add to the network the sets E_t, V_t, and F_t.
   - Do $t = t + 1$.

Let's note that we do not add the new vertices and new edges to the network until the last update. Doing it in this way we avoid to chage the probabilities $p_{\alpha_{out}}$, $p_{\alpha_{in}}$ and $p_{\beta_{in}}$ with every new vertex or edge added.

3.2 Setting Parameters

The proposed model has 5 parameters $[\theta, \rho, \alpha_{in}, \alpha_{out}, \beta_{in}]$. We are interested in getting good enough values of them for every data set in order to describe its behavior.

The values of θ and ρ are easier to be fixed than the others. For every data set, the value of θ is chosen as the slope in the linear regression with the number of total links that has the network over time. To choose $\rho \in (0, 1)$ such that $\lfloor (1-\rho)\theta \rfloor$ represents the number of new links generated at every step in the current nodes, we use ρ as the mean of the proportion of new links in new nodes.

The values of α_{in}, α_{out} and β_{in} are more difficult to choose. However, we know that they are related with the in-degree and out-degree distribution. Thus, to set these parameters we create a cost function based in an error of the values of in-degree and out-degree in the network, and we use *Differential Evolution* (DE) to minimize it. The parameters θ and ρ are fixed during the differential evolution. In fact, we use a modification of DE in order to take advantage of the model. In addition, because of the random process, the error generated is random, so the "cost function" we use will not be literally a mathematical function. For the same parameters it could return differents results. However, it is a function in a computational sense, and it is called *fitness function.*

Differential Evolution

Differential Evolution (DE) is a stochastic method developed by Price and Storn in 1995 in [18]. It has the ability to minimize non-differentiable cost functions. DE is easy to implement, even more, the implementation of DE could be parallelizable. This is very important in case of one evaluation of the fitness function takes from minutes to hours. This method uses just 2 parameters to generate new solutions, $CR \in [0,1]$ and $F \in (0,1)$.

We present the DE method like in [19] with 4 steps: Initialization, Mutation, Crossover and Selection.

1. **Initialization.** Choose randomly N_p vectors $x_{0,0}, x_{1,0}, ..., x_{N_p-1,0}$. We call every vector an *individual* and the set with these vectors the *population*. The 1st subindex is the identifier of the individual in the population. Due to DE is an iterative search method the 2nd subindex is the number of iteration. Every iterarion is called a *generation*.

   The initial population should cover the entire parameter space. Given the dimension of the parameter space, d, let's denote every parameter of an individual with 3 subindexes thus

   $$x_{i,g} = [x_{0,i,g}, x_{1,i,g}, ..., x_{d-1,i,g}]^T.$$

   After this point we proceed to do the next 3 steps for a fixed number of generations or until we have convergence in the fitness function.

2. **Mutation.** The idea of this step is to mutate the population to produce a new one with N_p trial individuals, called *mutants*. It is used a *differential mutation*, it means that mutants are created as a linear combination of randomly individuals in the population.

   Specifically, for every $i \in \{0, ..., N_p - 1\}$ select randomly different indexes $r_1, r_2, r_3 \in \{0, ..., N_p - 1\} \setminus \{i\}$, then, create a mutant individual $v_{i,g+1}$ according to

   $$v_{i,g+1} = x_{r_1,g} + F(x_{r_2,g} - x_{r_3,g}),$$

   where $F \in (0,1)$ is a scale factor that controls the rate at which the population evolves. In case of the mutant individual $v_{i,g+1}$ is not in the parameter space we generate a new one randomly.

3. **Crossover.** DE uses an *uniform crossover* in order to increase the diversity of the population. In this step DE crosses the population at time g with the N_p mutant individuals created in step 2. DE made the crossover in the order in which the mutants were created.

   Every individual crossed $u_{i,g}$ is defined by

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & r(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{j,i,g} & \text{otherwise,} \end{cases}$$

   where $r(0,1)$ represents a random value in $(0,1)$, j_{rand} is a random index in $\{0,1,...,d-1\}$ and $CR \in [0,1]$ is a defined value called *crossover probability.* The CR value controls the fraction of parameters copied from the mutants.

4. **Selection.** Finally, a comparasion is done between every individial $x_{i,g}$ in the population and its correspondent individual crossed $u_{i,g}$. The elements with the lowest fitness function value pass to the next generation. It is

$$x_{i,g+1} = \begin{cases} u_{i,g} & \text{if } f(u_{i,g}) \leq f(x_{i,g}) \\ x_{i,g} & \text{otherwise.} \end{cases}$$

Let's remember that the proposed model produces a sequence of N random graphs. Thus, if we do some simulations with the same set of parameters, we obtain different graphs. Let's denote the graphs obtained by the real data as $(G_t(D))_{t\geq 1}$ and the graphs generated by the model as $(G_t(M))_{t\geq 1}$. We define the error in a simulation as

$$\begin{aligned} e(\alpha_{in}, \alpha_{out}, \beta_{in}) = (0.25)&\frac{|\Delta_{in}(G_N(M)) - \Delta_{in}(G_N(D))|}{|\Delta_{in}(G_N(D))|} \\ + (0.25)&\frac{||\varphi_{in}(G_N(M)) - \varphi_{in}(G_N(D))||}{||\varphi_{in}(G_N(D))||} \\ + (0.25)&\frac{|\Delta_{out}(G_N(M)) - \Delta_{out}(G_N(D))|}{|\Delta_{out}(G_N(D))|} \\ + (0.25)&\frac{||\varphi_{out}(G_N(M)) - \varphi_{out}(G_N(D))||}{||\varphi_{out}(G_N(D))||} \end{aligned}$$

where $\Delta_{in}(G_N(M))$ is the maximum in-degree in $G_N(M)$, and $\varphi_{in}(G_N(M))$ is a sorted vector with the in-degree values of $G_N(M)$. Analogously for the out-degree values and for the graphs obtained with the data. Note that every term in this error is weighted with the same value. The first term is nearly to 0 if the in-degree values in $G_N(M)$ and $G_N(D)$ are in the same rank. The second term is nearly to 0 if the values of in-degree are similar 1 by 1. The third and fourth terms are analogous for the out-degree values. To sum up, this error is nearly to 0 if the in-degree and out-degree values in $G_N(M)$ and $G_N(D)$ are similar.

The fitness function f is defined as

$$f(\alpha_{in}, \alpha_{out}, \beta_{in}) = \min\{e_1(\alpha_{in}, \alpha_{out}, \beta_{in}), ..., e_S(\alpha_{in}, \alpha_{out}, \beta_{in})\}$$

where S is the number of simulations done with the fixed parameters and every $e_i(\alpha_{in}, \alpha_{out}, \beta_{in})$ is the error in the simulation i. This fitness function allows to evaluate the model S times and pick the best. In practice we use $S = 2$ and $N = 100$.

With this fitness function, for a fixed number of generations, we can apply the differential evolution method and choose the best parameters. However, we add the variable time in order to take advantage of the model. We note that if a modeled graph in the middle of the process is not similiar to its respective graph with real data, then, at the end of the process the modeled graph is not similar to its respective graph with real data. This happens because of the way in which the graphs grow in the model. Hence, instead of generating a sequence of 100 graphs in every simulation, we generate the sequence with the first 25 graphs, then we generate the sequence with the first 50 graphs, then we generate the sequence with the first 75 graphs and finally, we generate the complete sequence of 100 graphs as explained below.

1. Apply DE with 11 generations. In this case the model generates a sequence of 25 graphs.

2. Apply DE with 9 generations using the final population in step 1 as the initial population. In this case the model generates a sequence of 50 graphs.

3. Apply DE with 7 generations using the final population in step 2 as the initial population. In this case the model generates a sequence of 75 graphs.

4. Apply DE with 5 generations using the final population in step 3 as the initial population. In this case the model generates a complete sequence of 100 graphs.

Generating a complete sequence of 100 graphs with the proposed model is very slow. However, this modification of DE allows to generate a more complete sequence of graphs restricting to solutions that fit better through time. At the end of the modified DE we will have evaluated the fitness function in candidate solutions for more than 30 generations.

3.3 Graphs Comparison

Distance Between Graphs

The first tool we use to compare graphs is a distance between them. A distance gives a quantity idea of how similar are the graphs in a single value. The distance we explain is called *cut distance* and it is based in the *cut norm* described by Frieze and Kannan in 1999 in [13]. Let A be an $n \times n$ matrix, the *cut norm* of A is defined by

$$||A||_{\square} = \frac{1}{n^2} \max_{S,T \subseteq [n]} \left| \sum_{i \in S, j \in T} A_{ij} \right| .$$

The cut distance was introduced in [6] by C. Borgs, J. T. Chayes, L. Lovász, V. T. Sós and K. Vesztergombi in 2008. This distance between graphs reflects global structural similarity. It can be defined even for two graphs with possibly different number of vertices. However, the cut distance is more informative for graphs in which the number of edges is closer than the maximal number of edges, dense graphs. If we do not have preliminary information about the density of the graphs this distance should not be the unique way to compare two graphs. For example if we compare a graph constructed by real data G with a graph modeled G_M, both non-dense graphs. Then the cut distance from G to G_M should be close to 0. On the other hand, if this distance is not close to zero we could guarantee that the graphs are different.

We describe the cut distance as in [15]. First, we introduce a distance between two graphs on the same set of vertices. After, a distance between two graphs on the same number of vertices but labeled in a different way. Finally, the distance between two arbitrary graphs.

Same Set of Vertices. To define this distance we must to start by defining some notation. Let be $G = (V, E)$ a graph and sets $S, T \subseteq V(G)$. We denote by $e_G(S, T)$ the numbers of edges such that point from a vertex in S to a vertex in T. Thus, given G and G' on the same set of vertices

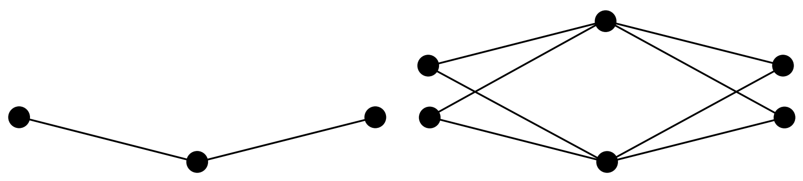
$$d_\square(G, G') = \max_{S,T\subseteq V(G)} \frac{|e_G(S, T) - e_{G'}(S, T)|}{|E|^2}.$$

Same Number of Vertices. Let's note that for two graphs which have the same number of vertices they can have labeled the vertices in a different way. In this case we just have to use a combinatorial idea. We look for the minimum value of the distance describe above but relabeling the vertices. Thus, given G and G' on the same number of vertices but labeled in a different way

$$\widehat{\delta}_\square(G, G') = \min_{\widehat{G},\widehat{G}'} d_\square(\widehat{G}, \widehat{G}'),$$

where $\widehat{G}$ denote all the posible labelings of G.

Arbitrary graphs. We next describe how to get a distance between two graphs using previous ideas. Given a graph G and $k \in \mathbb{N}$, we define $G[k]$ as the graph conceived by G but replacing every vertex by other k vertices linked exactly like their predecessors were. As an example, if we have a graph G like in Figure 3.1(a), then $G[2]$ is like in Figure 3.1(b).



(a) G (b) $G[2]$

Figure 3.1: An example of G and its respective $G[2]$

Let's note that if we have graph G and G' such that $|V(G)| = n$ and $|V(G')| = n'$, then $G[n']$ and $G'[n]$ have the same number of vertices. Thus, to define a distance between G and G' it is used $k \in \mathbb{N}$ to compare $G[kn']$ and $G'[kn]$.

Finally, the *cut distance* between two graphs is defined in terms of the distance $\widehat{\delta}_\Box$ as

$$\delta_\Box(G, G') = \lim_{k\to\infty} \widehat{\delta}_\Box(G[km], G'[kn])$$

This distance is called *cut distance* due to the fact that in the case of two graphs on the same set of vertces the distance between them is similar to calculate a *cut norm*. Specifically

$$d_\Box(G, G') = ||A_G - A_{G'}||_\Box.$$

If we analyze the final definition for two arbitrary graphs, we can note two things. First, if G is directed the definition is the same. Second, this distance is hard to compute due to the max-min process. In fact, just to compute the distance between two graphs with the same set of vertices, it is not easy. In [2] Noga Alon and Assaf Naor proved that the problem of approximating the cut-norm is MAX SNP hard. However, they proposed an efficient approximation algorithm.

In practice, we just need to compute the distance between two graphs with the same number of vertices. However, we use a function in the art state found in the package *cutnorm*. This function, implemented by Ping-Ko Chiu, Peter Diao and Olewasanmi Koyejo, computes an approximation to the cutnorm based on some techniques detailed in [2] and [22]. The function returns two values using different methods, we just present the results with one of these methods.

Analysis of Heavy Tailed Distributions

We explained in last section that the cut distance gives information about global structure. However, we want also information about particular features in the graphs over time. Thus, we analyze the in-degree and out-degree distributions of some graphs generated with the proposed model at different times. Due to we use a preferential attachment model the first idea to analyze the distributions is by fitting a power law. Let's remember that a power law distribution p_k with degree exponent λ is defined as

$$p_k = (\lambda - 1)k_{min}^{(\lambda-1)}k^{-\lambda},$$

where k_{min} is a fixed value.

It is important to note that a straight line in an histogram in log-log scale is not sufficient to assure that a distribution is a power law. However, a visual inspection of the data in this scale helps to decide if the data follows a heavy tailed distribution.

In general, the small values of the data does not fit very good in a power law. That is why in order to determine what portion of the data could be fitted with a power law the value of k_{min} must be estimated. Clearly, if k_{min} is large, less data are fitted and we can not assure that the fit is good enough.

To estimate the value of λ is used the maximum likelihood estimator (MLE) for the continuous power law. In [16] is proved that for a data list with positive values $k_1, ..., k_n$ the MLE is

$$\hat{\lambda} = 1 + n \left[\sum_{i=1}^{n} \ln \frac{k_i}{k_{min}} \right]^{-1}$$

where $k_{min} \geq k_i$. Thus, we have to choose the value of k_{min} where the power law begins. The optimal value of k_{min} can be found as in [8] by creating a power law distribution starting from every value in the data list, then compute the Kolmogorov-Smirnov distance between the data and the fit and select the fit with the smallest distance. This distance is defined as

$$D = \max_{k \geq k_{min}} |S(k) - P(k)|$$

where $S(k)$ and $P(k)$ are the cumulative distribution functions of the data list and the power law respectively.

Fitting a power law with these methods it is not sufficent to conclude that this distribution is the best description. As we said, if the k_{min} is large, we are not fitting a good proportion of the data. Even more, the same data set can be fitted with other distributions in an optimal way. Thus, J. Alstott, E. Bullmore and D. Plenz proposed in [3] that we must do a comparative test with other distributions. In practice we use the package *powerlaw* for Python in order to analyze heavy tailed distributions. In this package is implemented the method for fit a power law distribution getting λ and k_{min} as we explained. Also, a method is implemented to compare an exponential distribution fitted vs a power law fitted, and a power law fitted vs a lognormal fitted. In addition, this package has the option to show the data and the fits in a log-log plot. Complementary information and more methods of this package can be consulted in [3]. A similar package for R is presented in [14].

Chapter 4

Results

In this chapter we present the results of setting the parameters in the proposed model for each data set. Also, we present the results of compare the real networks with the networks modeled. In addition, we present a simple result of how can we use the proposed model to predict information in the future.

We recall that in this model we have 5 parameters $[\theta, \rho, \alpha_{in}, \alpha_{out}, \beta_{in}]$. We said in Chapter 3 that the value of θ is chosen as the slope in the linear regression with the number of total links that the real networks have over time. Also, we choose the value of ρ as the mean of the proportion of new links in new nodes. The values of these parameters for every network are shown in Table 4.1. Although we present the results just for 3 networks, we observe in general that ρ is close to 0.8. On the other hand, we found very different values of θ in other networks.

	θ	ρ
Soccer network	81.45	0.79
Earthquake network	78.98	0.83
Debate network	72.70	0.76

Table 4.1: Parameters θ and ρ for every network.

Also, we create a fitness function and we use a modification of differential evolution in order to fit the values of α_{in}, α_{out} and β_{in}. The values of CR and F in the DE method were calibrated with some experimental results for differents fitness functions, number of generations, and number of individuals.

At the end we executed the method once with the parameters $CR = 0.7$ and $F = 0.3$. Table 4.2 shows the values of these parameters for every network. Empirically, we had found that the value of β_{in} affects directly the value of Δ_{in}, also in other networks β_{in} is close to 1.3. On the other hand, the values of α_{in} and α_{out} affects the in-degree and out-degree distributions itself and they are more susceptible to the fitness function.

	α_{in}	α_{out}	β_{in}
Soccer network	0.417	1.707	1.352
Earthquake network	0.499	1.639	1.370
Debate network	0.235	1.460	1.285

Table 4.2: Parameters α_{in}, α_{out} and β_{in} for every network.

With the 5 parameters fitted, we do the comparison of the real networks and the networks modeled with these parameters. We use the networks with the minimum fitness at the end of the DE method in order to compare. In Table 4.3 we compare the number of links at the end of the process, it means, the value of $E(G_{100})$ for the real network and the modeled network. Clearly, the modeled network has less links. After, we inspect the number of links over time. We want to deduce if the model also fits lineary the number of links. Figure 4.1 shows the number of links over time for each data set, its linear regression and the number of links over time for the model. We observe that the model also fits lineary the number of links over time. However, using the same θ in each step it is not enough. This single value it is not robust to imitate the behavior of the number of links over time.

	Data	Model
Soccer network	7936	7275
Earthquake network	7869	7166
Debate network	7308	6681

Table 4.3: Number of links.

The number of links added for new nodes are related with the value of $\theta\rho$. Let's remember that we choose ρ as the mean of the proportion of new links in new nodes. Table 4.4 shows the values of this mean for every network with real data and with the proposed model. Besides, Figure 4.2 shows an histogram for this proportion in the real sequence of networks, and it shows an histogram for this proportion in the modeled sequence of networks. As expected, the means are similar even though the distributions are different. We can try to fit a distribution in the data. However, using just a single value helps to keep a simple model.

	Data	Model
Soccer network	0.79	0.77
Earthquake network	0.83	0.82
Debate network	0.76	0.75

Table 4.4: Mean of proportion of new links in new nodes.

Another feature we can highlight is that at every step, the new nodes in the modeled sequence of networks have average out-degree close 1, because of the new nodes in the real sequence of networks have average out-degree close to 1. Besides, the new nodes in the modeled sequence of networks have average in-degree 0, because of the new nodes in real sequence of networks have average in-degree close to 0. Figure 4.3 shows an histogram for the average out-degree in the real sequence of networks and in the modeled sequence of networks. Table 4.5 shows the mean of these values for every network. In this case, we have something similar, the distributions are different but the mean is close. Hence, it is a good idea that the new nodes in the model have out-degree at most 1 and out in-degree 0 in order to keep a simple model.

	$\bar{d}_{out}$(Data)	$\bar{d}_{out}$(Model)	$\bar{d}_{in}$(Data)	$\bar{d}_{in}$(Model)
Soccer network	0.92	0.88	0.19	0
Earthquake network	0.84	0.81	0.24	0
Debate network	0.99	0.93	0.21	0

Table 4.5: Mean of average out-degree and mean of average in-degree in new nodes.

After, we compare the global structural similarity between the real network and two random graphs. We generate a random graph G_{BA}, using the Barabási-Albert model, with the same number of nodes as in the real network

G and we compute the cut distance between them $\delta_\square(G_{BA}, G)$. In addition, if we denote by G_M the network generated at the end of the proposed model, we compute the cut distance $\delta_\square(G_M, G)$. Table 4.6 shows the values of the cut distance for every network. As expected, both distance are close to 0 because they are not dense graphs. However, the cut distance in our model is smaller, even in an order of magnitude. We suppose the cut distance in the proposed model is smaller for different reasons, but the main idea is that the real network is not a undirected tree as in a BA random graph.

	$\delta_\square(G_{BA}, G)$	$\delta_\square(G_M, G)$
Soccer network	6.41e-6	7.84e-7
Earthquake network	2.2e-06	6.35e-7
Debate network	1.02e-6	6.22e-7

Table 4.6: Cut distance between real networks and random graphs of two models.

We analyze the in-degree and out-degree distributions over time in real networks. We had realize that all of them look like heavy tailed distributions but visually it is not sufficent to assure that. We compute a p-value in order to compare if the data fit better with an exponential distribution or with a power law distribution. For every data set, and for every in-degree distribution in the networks at $t = 25, 50, 75, 100$, the p-value was less than 0.05. Thus, we can say that a power law is a better fit than the exponential. It does not mean that the data follow exactly a power law. However, we can say that the in-degree distributions are heavy tailed distributions. After that, we compare the data between a power law distribution and a lognormal distribution. In almost all of the cases, the p-value was more than 0.05. Thus, neither distribution is a significantly stronger fit. On the other hand, the out-degree distributions in many cases neither distribution is a significantly stronger fit. Even though the test is not conclusive respect to those fits, we keep the power law fit for two important reasons. First, the power law is a simple fit because it needs just one parameter contrary to the lognormal distribution which needs two. Second, and maybe the most important, the proposed model is a modification of the BA model and the power law is a distribution that can be deduce from that model. Figures 4.4 and 4.5 show the in-degree and out-degree distribution, respectively at $t = 50$ and $t = 100$, in log-log scale. These figures show the probability density function (pdf) for the real network and for the modeled networks, also it shows their power law fit and their lognormal fit. Figure 4.4 evidences some facts. For the real network and for the modeled network the power law fit is similar to the lognormal fit. The k_{min} for every power law fitted is small, between 2 and 4, so we can assure that the fit is good enough. The distributions between the data and the model are close. Even more, they are close at different times. On the other hand, Figure 4.5 evidences that for the real network and for the modeled network the power law fit is similar to the lognormal fit but neither distribution is good enough to model the real data. Also, the distributions between the data and the model are not always close.

The distributions that fit the data are importat, but, the exact values of Δ_{in} and Δ_{out} can be more important. Let's remember that every node represent an user. Thus, the value of Δ_{in} gives information about the user who is more mentioned during the conversation. The value of Δ_{out} gives information about the user who more mention different users during the conversation. Tables 4.7 and 4.8 show the values of Δ_{in} and Δ_{out} for every network at time $t = 50$ and $t = 100$ respectively. As expected, the values between the real data and the proposed model are close, this is due to the fitness function used in the DE method.

	Δ_{in}(Data)	Δ_{in}(Model)	Δ_{out}(Data)	Δ_{out}(Model)
Soccer network	308	218	26	24
Earthquake network	110	90	11	9
Debate network	434	293	15	13

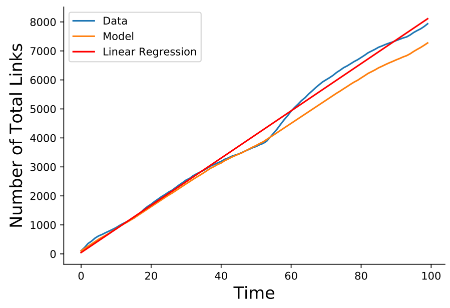
Table 4.7: Maximum in-degree and maximum out-degree at $t = 50$.

	Δ_{in}(Data)	Δ_{in}(Model)	Δ_{out}(Data)	Δ_{out}(Model)
Soccer network	612	594	35	35
Earthquake network	223	223	12	11
Debate network	760	721	17	17

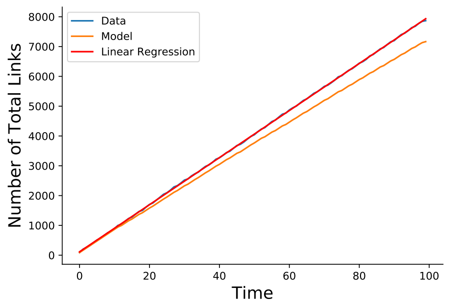
Table 4.8: Maximum in-degree and maximum out-degree at $t = 100$.

Besides, we sort the degree values in the real data and in the model and we generate a scatter plot in log-log scale to visualize a relation 1 to 1 in all the network. Figure 4.6 shows the scatter plot for the in-degree values in the networks at time $t = 50$ and $t = 100$. Figure 4.7 shows the scatter plot for the out-degree values in the networks at time $t = 50$ and $t = 100$. Both figures evidences that the dots are close to the identity due to the fitness function used. Even more, they are close at different times. Also, it shows, by the same reason, that the last dot is close to the identity even if the rest of the dots are not so close. In addition, we can note in Figure 4.6 that in general for small values the dots are under the identity and for large values the dots are above the identity. Furthermore, Figure 4.7 shows that in general the dots are above the identity. Clearly, the results for the in-degree values are better than for the out-degree values. The fact that we use two parameters that affect directly the in-degree values and just one that affects the out-degree values, could be the reason of this behavior.

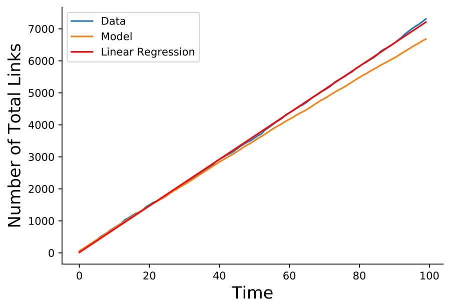
Another features we can study in the evolution of the graphs are the proportion of nodes and links in the giant component. Figure 4.8 shows these proportions over time for each graph. We can note that the GC emerges soon in both process, but in the proposed model is faster. Clearly this model is not the best to explain these features. However, both have a similar increasing trend. After, we analyze the underlying undirected graphs and we contrast the average degree vs the proportion of nodes in the GC. Figure 4.9 shows that this feature neither is very good explained with the proposed model. However, we can note that in the real data as in the model the average degree is always less than 1, opposite to the ER model. In addition, all these features are related. Note that if in the model the proportion of links in the GC decrease over time, then the proportion of nodes in the GC also decrease. Besides, if we keep the average degree, the contrast of the average degree vs the proportion of nodes could be more similar to the reality.



(a) Soccer Network

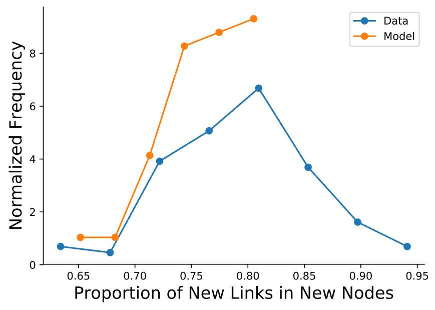


(b) Earthquake Network

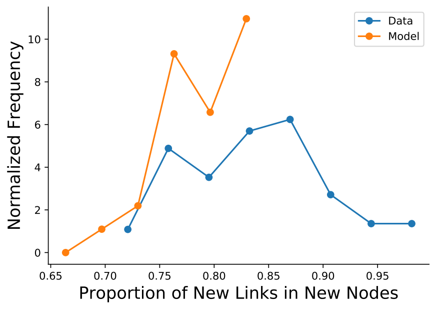


(c) Debate Network

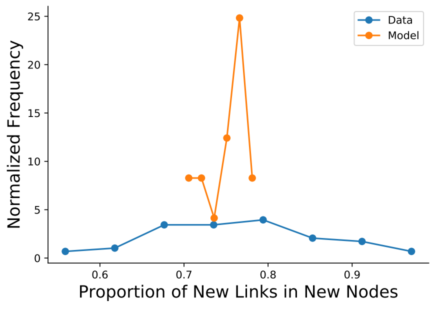
Figure 4.1: Number of total links over time.



(a) Soccer Network

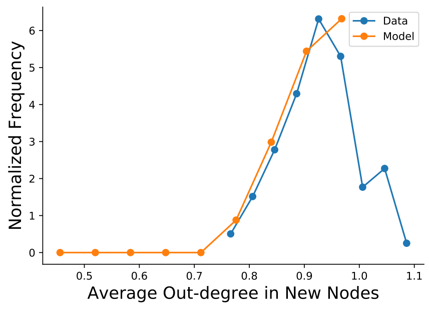


(b) Earthquake Network

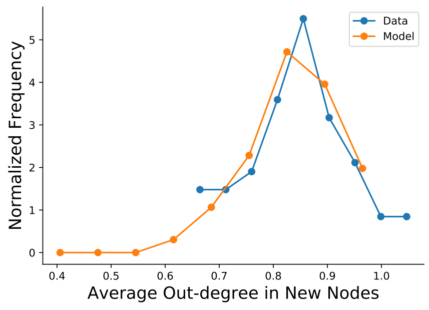


(c) Debate Network

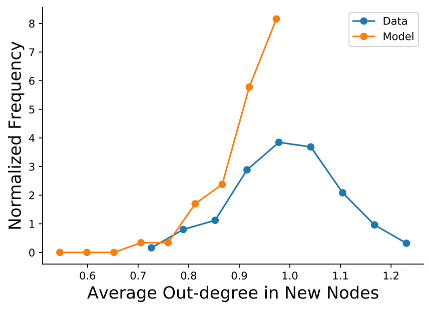
Figure 4.2: Proportion of New Links in New Nodes.



(a) Soccer Network

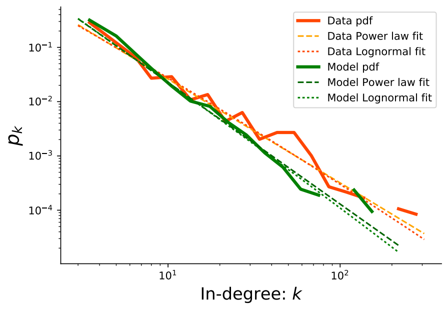


(b) Earthquake Network

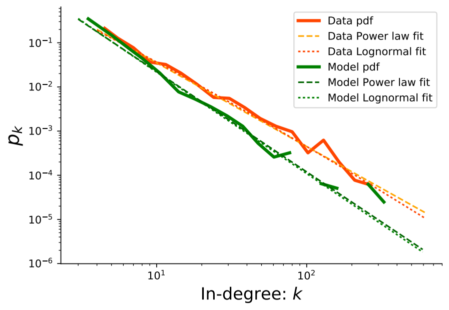


(c) Debate Network

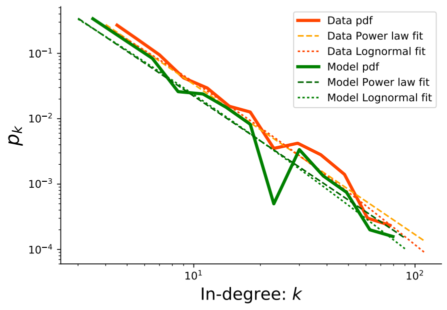
Figure 4.3: Average out-degree in the new nodes over time.



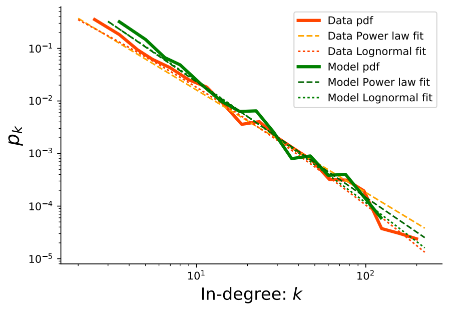
(a) Soccer Network



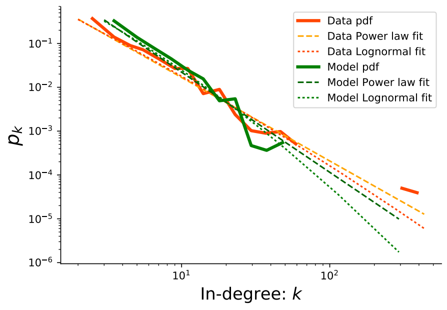
(b) Soccer Network



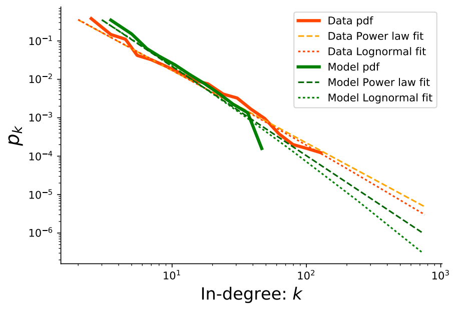
(c) Earthquake Network



(d) Earthquake Network

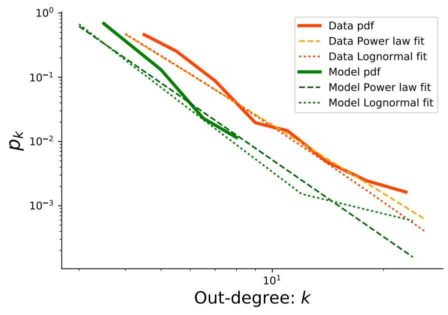


(e) Debate Network

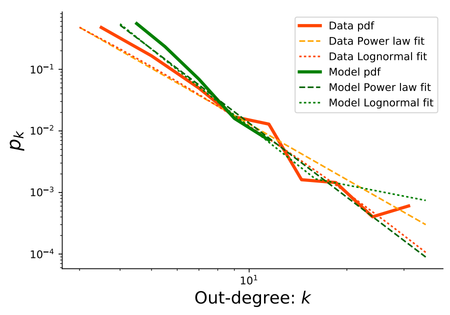


(f) Debate Network

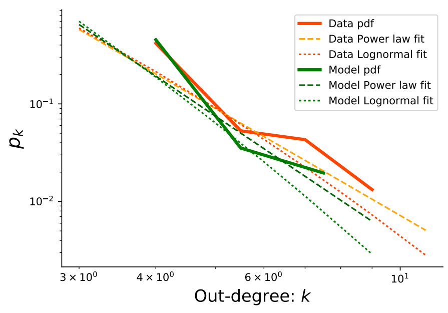
Figure 4.4: In-degree distributions at $t = 50$ and $t = 100$.



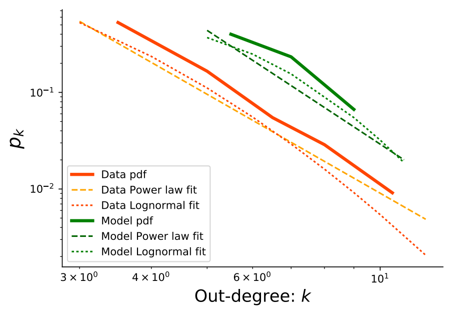
(a) Soccer Network



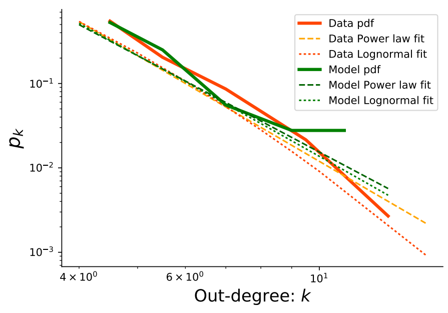
(b) Soccer Network



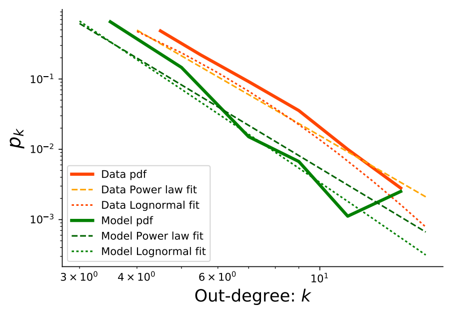
(c) Earthquake Network



(d) Earthquake Network

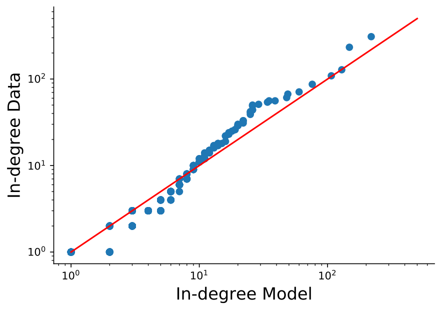


(e) Debate Network

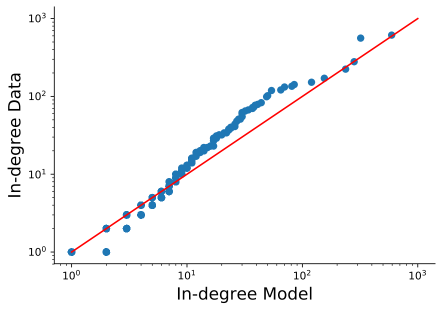


(f) Debate Network

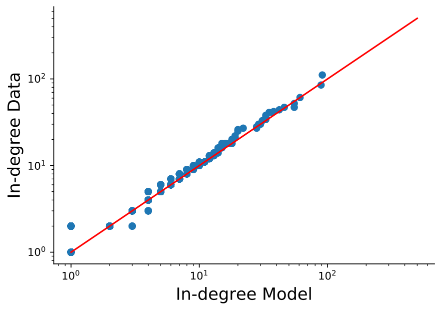
Figure 4.5: Out-degree distributions at $t = 50$ and $t = 100$.



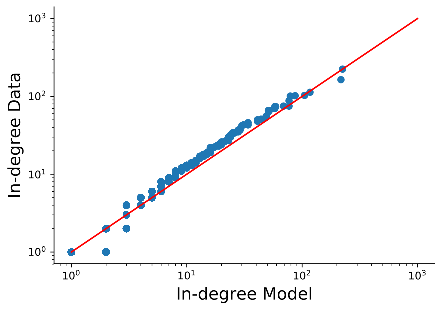
(a) Soccer Network



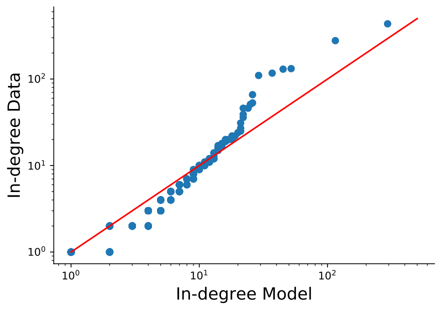
(b) Soccer Network



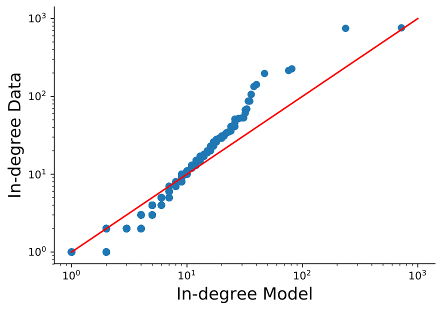
(c) Earthquake Network



(d) Earthquake Network

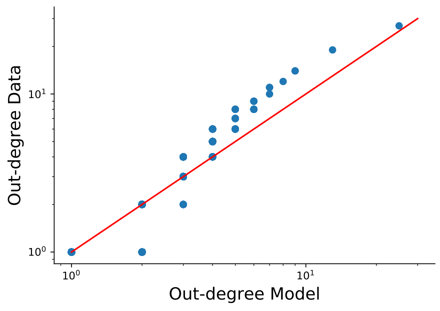


(e) Debate Network

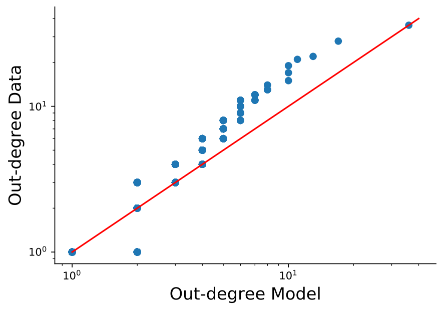


(f) Debate Network

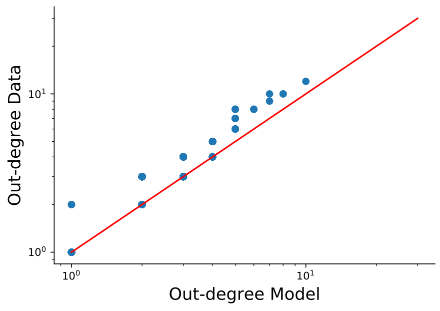
Figure 4.6: In-degree scatter plot at $t = 50$ and $t = 100$.



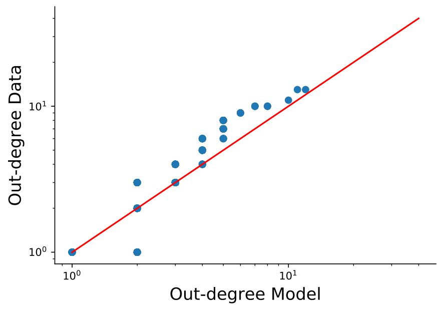
(a) Soccer Network



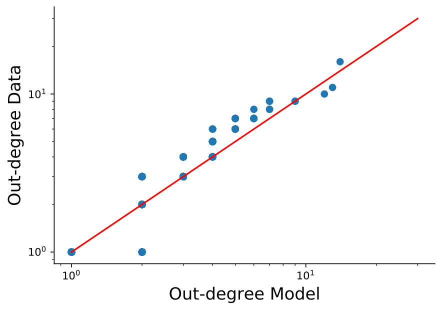
(b) Soccer Network



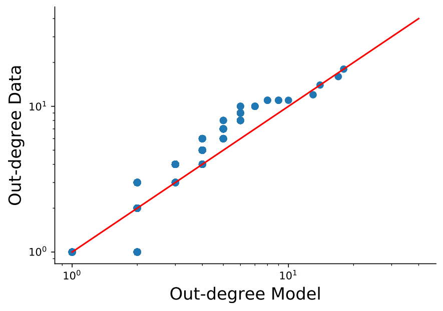
(c) Earthquake Network



(d) Earthquake Network

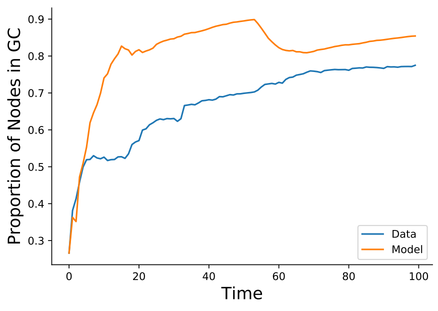


(e) Debate Network

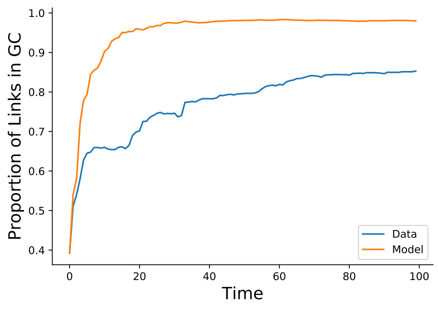


(f) Debate Network

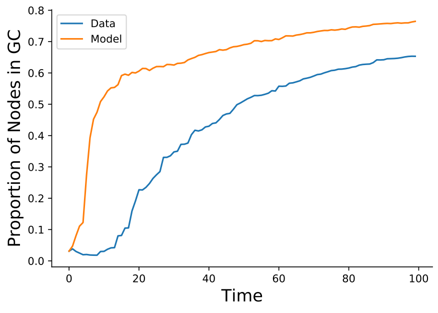
Figure 4.7: Out-degree scatter plot at $t = 50$ and $t = 100$.



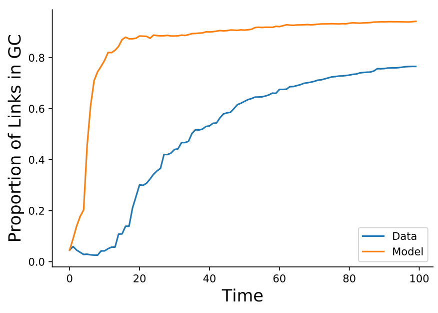
(a) Soccer Network



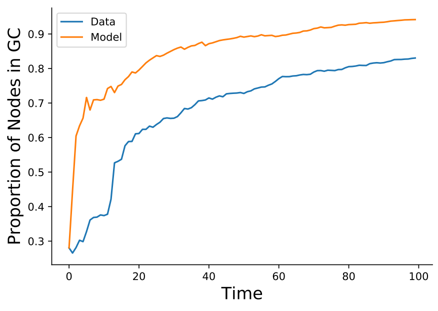
(b) Soccer Network



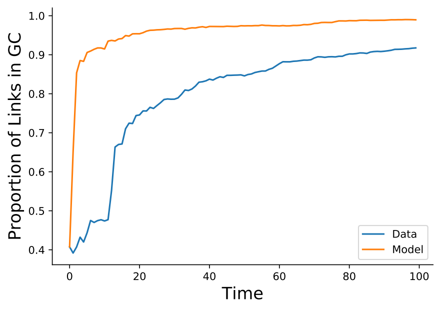
(c) Earthquake Network



(d) Earthquake Network

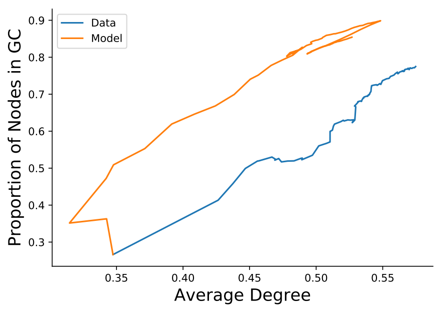


(e) Debate Network

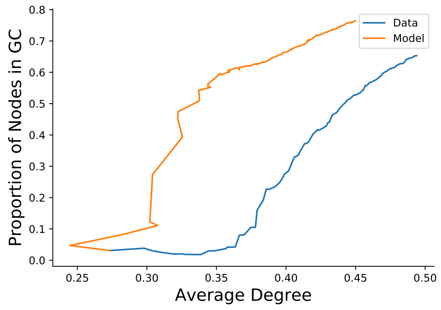


(f) Debate Network

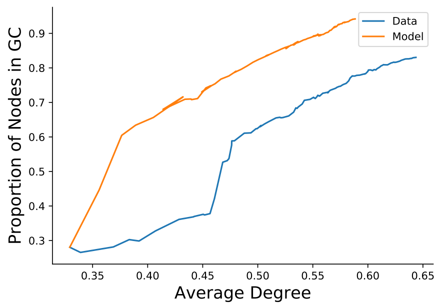
Figure 4.8: Proportion of nodes and links in the giant component over time.



(a) Soccer Network



(b) Earthquake Network



(c) Debate Network

Figure 4.9: Average degree vs proportion of nodes in the GC over time.

Prediction

In this section we present the proposed model with a different objetive. We use the Soccer network to implement a simple example of prediction. We use information from the data until the midle of the process in order to predict information at the end of the process. It means, we use information from G_1 and until G_{50} in order to predict information of G_{100}. Specifically, we are interested in the number of links, the maximun in-degree, the maximum out-degree and if the network modeled can be structurally similar to the real network.

We fit the 5 parameters in the model at $t = 50$ with the real data. The values $\alpha_{in}, \alpha_{out}, \beta_{in}$ were obtained with the modification of the DE algorithim explained in Chapter 3 but with times $t = 12, 25, 40, 50$.

Then, we simulate the model with these parameters and we generate a complete sequence of 100 networks. Finally, we repeat this idea 100 times and we collect the informarion in the last networks generated.

Table 4.9 shows the values of number of links, maximun in-degree, and maximum out-degree for the real data. Also it shows the rounded mean of these values in the 100 simulations, and their standard deviation (SD).

	Data	Mean	SD
Number of Links	7936	6815	3.2
Maximun In-degree	612	817	324.5
Maximun Out-degree	35	28	19.1

Table 4.9: Results of 100 simulations.

We can note that the mean of number of links is far from the reality 15%. The mean in the maximun in-degree and maximum out-degree are far from the reality 33% and 20%, respettively. In addition, they have a large SD like in the BA model with superlinear preferential attachment.

Besides, in order to get information of the structural similarity we compute the cut distance between the modeled networks and the real network. These distances have mean $\hat{\mu} = 8.85e\text{-}7$ and standard deviation $S = 6.85e\text{-}7$. In this case, the 100 simulations return a distance smaller than a distace with a BA model.

Chapter 5

Conclusions and Future Work

In this thesis we have worked with data downloaded from Twitter in real time. We generated networks of users based in their interaction using the tweets collected. We analyzed these networks and studied different models of random graphs in order to propose a model of users interaction in this social network.

The proposed model is a modification of the Barabási-Albert model. There are two main difference between them. First we use directed graphs. Second, we have the two types of non-linear preferential attachment, sublinear and superlinear. Also, in the proposed model, at every step we can add more than one vertex or edge to the graph. Moreover, we divided vertices and edges in current and new. Instead of add edges only when a vertex is added, we add edges in two differents updates in the model, from current vertices and from new vertices. As in the BA model, the edges are added with a preferential attachment. However, the proposed model have parameters to use non-linear preferential attachment.

The proposed model have 5 parameters. We explained how to choose them by a fixed data set. For example, we created a fitness function of error and we used differential evolution (DE) to minimize it. In fact, we took advantage of the model and we modified the DE method.

We studied the cut distance between graphs and how to analyze heavy tailed distributions to compare the proposed model with the real data.

The proposed model have similarities with the evolution of the real networks by construction. Even more, the proposed model shows a better performance structurally than the BA model by comparing with the cut distance. Besides, a power law fit is enough to explain the in-degree and out-degree distributions over time in the data and in the proposed model. The fact that we used two parameters that affect directly the in-degree distribution can be a reason for having the in-degree values closer to reality than the out-degree values. On the other hand, the proportion of vertices and edges in the giant component between the proposed model and the real data are very different. However, both have a similar increasing trend that the BA model clearly does not have. In general, we think that the proposed model explain better the evolution of the graphs than the BA model. Despite these results, we believe that the proposed model could be improved.

Additionally, we tried to use the proposed model in order to predict information in the future. We get good results in structural similarity with the cut distance. However, specific information like the number of edges, maximum in-degree and maximum out-degree appear to be still underpredicted.

As a future work, we will analyze other information in the graphs. For example, the spectrum in the laplacian matrix in order to get information about communities. Besides, we will try to develop a new model. We expect that in the future we can also explain the proportion of vertices an edges in the giant component. Furthermore, we belive that a new model can also fit the number of vertices added at every step. In addition, we can parallelize the DE method in order to analyze large graphs and to get faster results.

Bibliography

[1] Albert, R. and Barabási, A.; *Emergence of scaling in random networks*; Science; 286:509-512; 1999.

[2] Alon, N. and Naor, A.; *Approximating the Cut-Norm via Grothendieck's Inequality*; SIAM J. Comput. 35(4), pp. 787-803; 2006.

[3] Alstott, J., Bullmore, E. and Plenz, D.; *powerlaw: a Python Package for Analysis of Heavy-tailed Distributions*; PLoS ONE 9(1); 2014.

[4] Barabási, A.; *Network Science*; Cambridge University Press, U.K.; July 2016.

[5] Bollobás, B., Riordan, O., Spencer, J. and Tusnády, G; *The degree sequence of a scale-free random graph process*; Random Structures and Algorithms; 18:279-290; 2001.

[6] Borgs, C., Chayes, J.T., Lovász, L., Sós, V. T. and Vesztergombi, K.; *Convergent Sequences of Dense Graphs I: Subgraph Frequencies, Metric Properties and Testing*; Adv. Math 1801-1851 arXiv:math/0702004; 2008.

[7] Choi, J., Sethuraman, S. and Venkataramani, S.C; *A scaling limit for the degree distribution in sublinear proferential attachment schemes*; Rand. Struct. Alg. 48, 703-731; 2015.

[8] Clauset, A., Shalizi, C.R. and Newman, M.E.J.; *Power-law Distributions in Empirical Data*; SIAM Review 51(4), pp. 661-703; 2009.

[9] Diestel, R., *Graph Theory*, fifth edition, Springer, Hamburg, Germany, 2017.

[10] Erdös, P. and Rényi, A.; *On Random Graphs. I*; Publicationes Mathematicae; 6: 290-297; 1959.

[11] Erdös, P. and Rényi, A.; *On the Evolution of Random Graphs.*; Magyar Tur. Akad. Mat. Kutató Int, Közl, 5, 17-61; 1960.

[12] Estrada, E. and Knight, P.A.; *A First Course in Network Theory*; Oxford University Press, U.K.; May 2015.

[13] Frieze, A. and Kannan, R.; *Quick Approximation to Matrices and Applications*; Combinatorica, vol. 19, no. 2, pp. 175-220; 1999.

[14] Gillespie, C.S.; *Fitting Heavy-tailed Distributions: The poweRlaw Package*; Journal of Statistical Software, 64(2), pp. 1-16; 2015.

[15] Lovász, L.; *Large Networks and Graph Limits*; American Mathematical Society Colloquium Publications 60. Amer. Math. Soc. Providence, Ri.; 2012.

[16] Muniruzzaman, A.N.M.; *On Measures of Location and Dispersion and Test of Hypothesis on a Pareto Distribution*; Calcutta Statistical Association Bulletin 7,115-123; 1957.

[17] Newman, M.; *Networks: An Introduction*; Oxford University Press, U.K.; May 2010.

[18] Price, K. and Storn, R.; *Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces*; Journal of Global Optimization, vol. 11, no. 4, pp. 341 - 359; 1997.

[19] Price, K., Storn, R. and Lampinen, J; *Differential Evolution: A Practical Approach to Global Optimization*; Springer; Germany; 2005.

[20] Sethuraman, S. and Venkataramani, S.C.; *On the Growth of a Superlinear Preferential Attachment Scheme*; arXiv:1704.05568; Apr 2017.

[21] van der Hofstad, R.; *Random Graphs and Complex Networks*; Cambridge University Press; UK; May 2016.

[22] Wen, Z. and Yin, W.; *A feasible method for optimization with orthogonality constrains*; Math. Program., col. 142, no. 1-2, pp. 39-434; 2013.