FINAL MASTER PROJECT

TITLE: Overlay network topological properties on dynamic P2P

DEGREE: MASTEAM

AUTHOR: Daniel Quintas Rodríguez

DIRECTOR: Roc Meseguer

DATE: 31 – October – 2014

TITLE: Overlay network topological properties on dynamic P2P

AUTHOR: Daniel Quintas Rodríguez

DIRECTOR: Roc Meseguer

DATE: 31 – October – 2014

Overview

Actually one of popular networks to share information are usually peer-to-peer networks where the overlay topology is continuously reshaped. That can affect important properties that affects the efficiency of the nodes, that are part of the network.

So for P2P overlay, scale-free networks offer the possibility that as the network becomes very large it will control certain key properties for those type of networks.

The objective is provide a tool for testing in a controlled environment, a known network that don't fulfill scale-free characteristics and apply modifications to the network elements to approach to a network that meets it.

The result is an application for monitoring the properties and shape of a given overlay network while simple practical strategies is applied in order to modify it to meet certain properties of a scale-free network.

ÍNDEX

FIGURE INDEX

TABLE INDEX

INTRODUCTION

Peer-to-peer or P2P theoretically is a decentralized network model with neither element is client or server, but each node of the network can act like both, although in practice this characteristic is not completely meet in some P2P networks. P2P systems can be use for distributed storage or media sharing. Since are decentralized networks and others functionalities and usually are formed in virtual overlays.

The overlay of these networks are continuously reshaped by the entry or exit of users or nodes, connection problems in the physical network, important resources locations... and the network must maintain certain properties so his performance will remain the same. So P2P must be scalable and robust on changes on his topology.

One way to try to achieve this is the named, scale-free networks, that allows a rapid growth with a mix of highly connected and poorly connected nodes and relative stable diameter of the network even with a great number of nodes. There are complex algorithms to generate these types of networks, but what happens if with some techniques, simple in a practical way, are applied in order to meet certain patterns distribution.

The objective of the project is prove, in a controlled environment, if with simple and logical techniques is possible to obtain or modify a known network to make fulfill characteristics of a scale-free overlay at the same time information about his network properties and shape changes are being recorded.

The report is organized in the following way, first some concepts will be briefly explained then the initial objectives of the project will be presented. Afterwards the general architecture of the application will be shown and the modules will be introduced as well. Next the modules will be explained in detail and later on some practical data will be shown. And finally the conclusions will be exposed.

CHAPTER 1. CONCEPTS

In this chapter we introduce the concepts that will be discussed later in the report. It will start with a introduction about the networks the application is focused, the key parameters for this project and a brief discussion about power law in networks.

1.1 Networks

The network is the overall element for the application. A networks can be seen as computing devices where pass data to each other along connections. The type of data or the elements forming the network is not important for this project, so these computing or physical devices will be called nodes as a general terminology. As well connections between these nodes will be referred as edges.

Also depending on how the edges are connected between nodes a network or graph can be defined in different ways. For the application purpose will center the different types of networks in only two.

1.1.1 Directed Network

In graph theory, a directed network is a graph with a set of nodes connected by edges, where the edges have a direction associated with them. The next figure represents a simple directed graph.

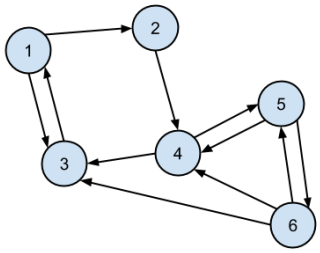

Figure 1 Directed graph

1.1.2 Undirected Network

An undirected graph the set of nodes are connected together, where all the edges are bidirectional. The next figure is a simple undirected graph.

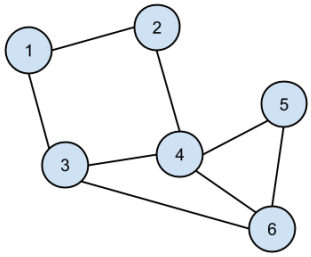

Figure 2 Undirected graph

1.2 Network parameters

Each of those elements, have properties which together define the network. Now we will go through some of them briefly.

1.2.1 Node degree

Basically the node degree is the number of edges have a node. Depending on whether is an incident edge or a not, the degree can be divided into indegree or outdegree. So for example with the previous figures:

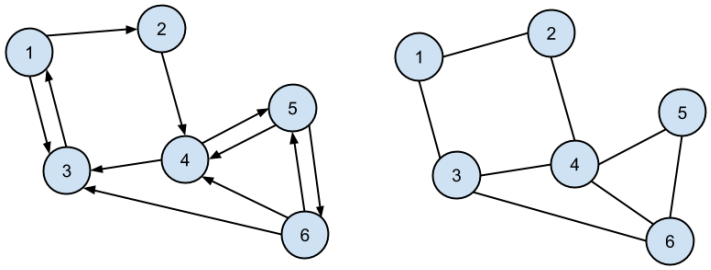

Figure 3 Graphs

For the undirected graph as the nodes are consider bidirectional, there is no indegree or outdegree, it can be determined:

Node	Degree
1	2
2	2
3	3
4	4
5	2
6	3

Table 1 Undirected degree example

And for the directed graph, the degree is split into an indegree and outdegree value, so:

Node	Indegree	Outdegree
1	1	2
2	1	1
3	3	1
4	3	2
5	2	2
6	1	3

Table 2 Directed degree example

1.2.2 Neighborhood

The neighborhood of a node in a graph can be a sub graph with all the nodes and edges where this node are connected with it. For example, in previous the undirected graph the neighborhood of the node 4, it will be the nodes 2,3,5 and 6 and the corresponding edges between them.

1.2.3 Diameter

The diameter of a network, is the longest shortest path. So is the largest number of nodes which must be traversed in order to travel form one node to another one. Continuing with the examples using the previous graphs, the diameter is 6 for the directed graph, form node 3 to node 6, and 4 for the undirected, from node 5 to node 1.

1.2.4 Average path length

Is the average shortest path for all possible pairs of nodes. It measure the efficiency of information transfer on network.

1.2.5 Centrality

Centrality are indicators which identify the most important nodes in a graph:

- Betweenness: quantifies the number of times a node acts as a bridge along the shortest between two other nodes.
- Closeness: is the inverse of the sum of a node distance to all other nodes.
- Eccentricity: is the maximum distance between a node and any other node in the graph.

1.2.6 Clustering coefficient

This coefficient measure the degree where the nodes in the graph tend to cluster together.

1.3 Power Law and Scale-free networks

Statistically, a power law is a power relationship between two quantities. What makes this interesting is that this property is a characteristic of scale-free networks where degree distribution follow a power law distribution.

Some of the characteristics of this type of networks are that the highest degree nodes act like hubs where others hubs with less degree are connected to them, and this smaller hubs, are connected to other nodes with less degree and so on. A network with that property is consider to have a strong robustness to failure due to the hierarchy explained before, this property can be very important to limit the hop count or diameter, while the network is growing. The next figure is a simple representation of how would look like more or less:

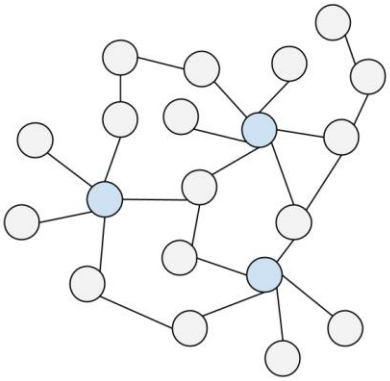

Figure 4 Simple example of Scale-free network representation

Another characteristic of these networks is the clustering coefficient distribution, which decrease as the node degree increase. That implies that low degree nodes are part of very dense community, and those communities are connected to each other through hubs, and this feature intensifies as the network becomes very large.

The degree distribution determine if a network fulfill the power law distribution. So first of all the numbers of nodes for each degree have to be known. After that using tools like Excel or some frameworks, like the one is used in this project and it will be presented more forward can be estimated the constant values x_{min} and the exponent, k. Then just have to calculate using:

$$y=X_{min}*(1/1-e(-x/k))$$

Where y is the predicted value for the distribution. Finally just need to do the difference between the predicted value for a degree minus the number of nodes for a degree and square the result. Now the sum square error is known, so if a change is made in the network and it needs to be determine if is an improvement towards a power law distribution the new sum square error should be minor than the previous one.

CHAPTER 2. REQUIREMENTS

The principal objective of this project is offer a tool to analyze a transformation of a random or existing network into one that fulfills a power law in his degree distribution.

To accomplish it, those are the objectives purposes:

- Develop an application to manage the process
- The application must be able to do a fit of a power law with the degree distribution of the network.
- The application must implement suitable strategies to modify the network into one that fulfill a power law in his degree distribution.
- The application must be compatible to principal network extensions for his input.
- Generate a text output with different properties from the new network generated at each step.
- Generate an output representation of the new network with a proper extension compatible with other tools at each step.

The project experiments with the manipulation of the node degree property from the whole network, increasing or decreasing, depending on the actual degree distribution of the network and the calculated prediction with the fitted power law.

Aside from the possibility of compare the properties and visual representation from the original network at each step while the changes are being made in order to see the evolution into the new network.

CHAPTER 3. ARCHITECTURE

In this chapter will present an overview of the elements involve in the project application using a block diagram and a brief description of the parts.

3.1 General architecture

It can be represented with the following figure:

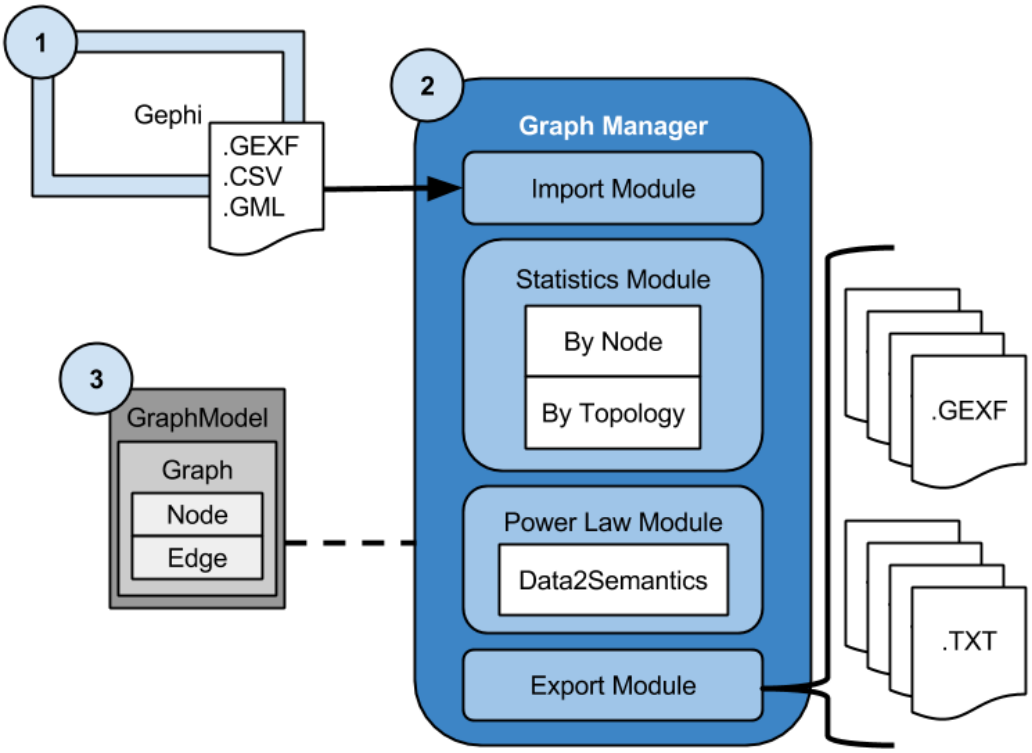

Figure 5 General architecture.

At Figure 5 there are the following blocks:

3.1.1 Gephi

Gephi is an Open tool that permits the visualization and exploration of different kinds of networks like directed or undirected graphs. That platforms can generate random data, useful to create graphs of different sizes for test our application, in addition to apply layout algorithms to give the shape to the graph and have a more friendly visualization. Or import an existing file in standard graph file formats like:

- GEXF
- GDF
- GML
- GraphML
- Pajek NET
- CSV
- Spreadsheet

The use of that platform in the project mostly is for visualization of the output of the application and check how the graph evolves with each step.

3.1.2 Graph Manager

Graph Manager involve all the application designed and developed in this project. Through a given graph it modify the different elements to the point that the degree distribution looks like as a power law distribution making use of the different strategies discussed above.

To achieve that, it use a framework for java named Gephi-Toolkit. Basically allow you to use some features of the visualization platform, Gephi [9], through command line or in a java project. As it is important to track some metrics and statistics from the network is going to be treated, that framework provide a way to get those common data and keep the track of the graph in each change.

This manager are split into several modules focused each one in one job, briefly:

- **Import module**, involve in get an existing standard file graph and parse it into a workable java object
- **Statistics Module** are in charge to deliver several metrics and statistics, discussed previously, of the current graph or a specific node.
- The **power law module**, manage the strategies to implement on networks and determine if the changes is an improvement into a power law distribution, and if it is not reverse the changes. For that purpose it use a small java library, named Data2Semantics, for the analysis of power law distributed data, useful for estimate parameters like the uncertainty of a given data set.
- **Export Module** will generate several standard graph file between the process. That will be useful for look the evolution of the graph through the changes

3.1.3 Data Model

It talked about keeping track of network changes. The way this is done, is using a persistent object model through all the process. The model is composed of several levels.

- GraphModel level contains the complete graph structure and determines some properties for the network like which type of graph (directed, undirected...).
- Graph, inside the GraphModel level, acts like an accessor to read or modify the structure of the network, like applying filters to sort nodes, for example. It is possible have multiple Graphs in one GraphModel so it allow merge different networks.
- Within the Graph there are a list of Nodes and Edges which contains the data for each element in the network and his particular data.

CHAPTER 4. DESIGN AND IMPLEMENTATION

This chapter present the design and implementation of the application developed identified in the previous chapter. It will be discussed each module and after, also it will be shown a normal flux of one change in a graph for different strategies.

The flow of the whole application can vary, that means that which strategies are used or when, can be defined before run the application through code.

4.1 Import module

As stated earlier, the project use the Gephi-Toolkit as a help to manage the files and elements of a graph. It works similar as if you where using Gephi program itself.

At the import module, it creates a workspace where the graph model will be linked but this workspace needs a referenced project. The following figure represents the flow of that module:

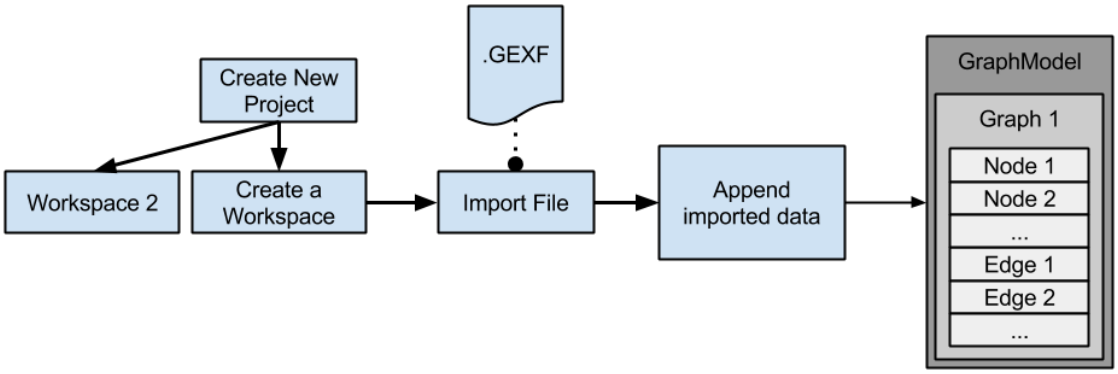

Figure 6 Import module flow

A project can handle several workspaces, as the same way a Graph Model can have several graphs associated. In this case, it doesn't need more than one workspace or one graph to do all the process of the application. At the end the graph described at the file, will be imported and parsed to the data model described previously.

Moreover the import module supports all the files than Gephi can. But in a practical way it will use those standard graph files:

4.1.1 GEXF

It is a language for describe complex networks structures and is developed for the Gephi project. The following figure shows a piece of an actual gexf file:

```
1 <?xml version="1.0" encoding="utf-8"?><gexf version="1.
2   <graph defaultedgetype="undirected" mode="static">
3     <attributes class="node" mode="static">
4       <attribute id="0" title="isDNS" type="boolean" />
5       <attribute id="1" title="isProxy" type="boolean"
6     </attributes>
7     <nodes>
8       <node id="0022" label="0022">
9         <attvalues>
10          <attvalue for="0" value="False" />
11          <attvalue for="1" value="False" />
12        </attvalues>
13      </node>
14      <node id="0015" label="0015">
15        <attvalues>
16          <attvalue for="0" value="False" />
17          <attvalue for="1" value="False" />
18        </attvalues>
19      </node>
        [...]
284   </nodes>
285   <edges>
286     <edge id="0" source="0004" target="0034" />
287     <edge id="1" source="0004" target="0036" />
288     <edge id="2" source="0004" target="0021" />
289     <edge id="3" source="0015" target="0014" />
290     <edge id="4" source="0015" target="0057" />
291     <edge id="5" source="0015" target="0056" />
292     <edge id="6" source="0015" target="0019" />
```

Figure 7 GEXF example

So it is in a XML format, there are tags for each element of a graph.

The first you see is the graph tag [line 2], where is set some of the attributes of it, like is an undirected graph.

After that comes the nodes tag [line 7] where compress all the nodes with his respective data, like his id or label,

And finally after the nodes tag is closed [line 284] there is the edges tag [line 285], where all the edges are defined, for example the edge with the id 4 [line 290] is the link between the nodes with the id "0015" and "0057".

4.1.2 GML

Graph Modeling Language describe graphs in hierarchical ASCII-based way. Here is an example of how it looks:

```
graph [
  directed 1
  node [
    id 1
    label "100monkeystyping.com"
    value 0
    source "Blogarama"
  ]
  node [
    id 2
    label "12thharmonic.com/wordpress"
    value 0
    source "BlogCatalog"
  ]
  node [
    id 1490
    label "zeph1z.tripod.com/blog"
    value 1
    source "Blogarama"
  ]
  edge [
    source 267
    target 1394
  ]
  edge [
    source 267
    target 483
```

Figure 8 GML example

4.2 Statistics Module

Here will perform calculations for metrics and statistics for the current graph, as well, apply filters to facilitate process to other modules.

The module are spitted into two parts, one for calculations and manipulations involving a single node, and the other when involves several elements of the network ergo the topology.

Each of one of this measures or filters, are separated also into modules. The next figure represents those modules:

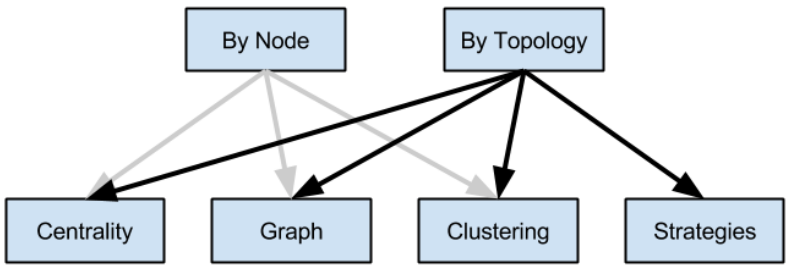

Figure 9 Statistics modules

4.2.1 Centrality module

All involved with centrality measures. Those are the measurements made in this module:

- By Node:
  - Betweenness, for one or all nodes.
  - Closeness, for one or all nodes.
  - Eccentricity, for one or all nodes.
- By Topology:
  - Diameter.
  - Path length.
  - Betweenness mean.
  - Closeness mean.
  - Eccentricity mean.

4.2.2 Graph module

This module is responsible for basics measurements and sort through elements to deliver a properly set of data for the processes that request it.

- By Node:
  - Node degree.
  - Node in-degree.
  - Node out-degree.
  - Node neighbors.
  - In coming edges for a node.
  - Out coming edges for a node.
  - Resolve if two nodes are connected between them.
- By Topology:
  - Nodes and edges recount
  - Average, maximum, minimum degree.
  - Deliver filtered lists by degrees, quantities of nodes according to degrees, nodes between a degree interval, nodes with zero degree...

4.2.3 Clustering module

A module for clustering measurements:

- By Node:
  - Clustering, for one or all nodes.
- By Topology:
  - Average clustering coefficient.

4.2.4 Strategies module

Some measurements necessary to carry out the strategies for approach a power law distribution

- By Topology:
  - Obtain the actual degree distribution
  - Deliver the error between the actual network and the expected power law distribution.

4.3 Power law module

This module has the job to use the strategies like increase or decrease a node degree, explained in previous chapters, as well as control those changes are the correct ones to approach more and more to a power law distribution.

First of all the distribution is calculated at the beginning, using the library named Data2Semantics, just need a data set, which is the number of nodes for each degree in the network. As result we got the distribution parameters of the best fit the library can obtain.

With those parameters is possible predict the number of nodes at each degree we need to simulate the power distribution fit. Also it is possible to know the mean square error (MSE) between the actual network and the distribution fit obtained, so after a modification in a node degree is easy to determine if that change get the network close to that fit.

In case the modification does not improve the distribution, it starts a revert process to the point before the modification was applied

Now is possible to apply one of the strategies and modify at least the degree of one node.

4.3.1 Strategy of decrease a degree

This strategy is applied in case of obtain a node with a degree that the actual number of nodes in that specific degree is higher than the number of expected nodes in the predicted degree distribution. Basically what is done is remove an edge of that node check if was an improvement for the degree distribution

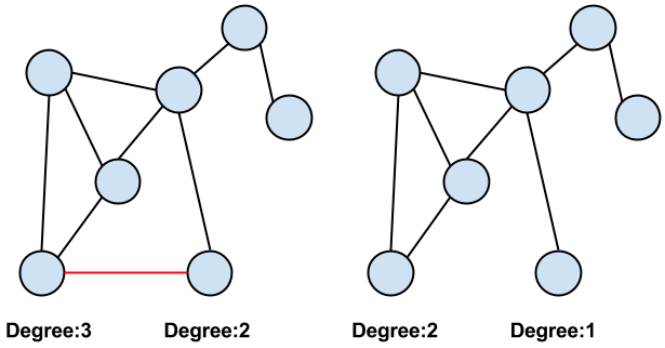

Figure 10 Simple degree decreasing

The next figure show the flow for the strategy of decrease a degree in the application context:

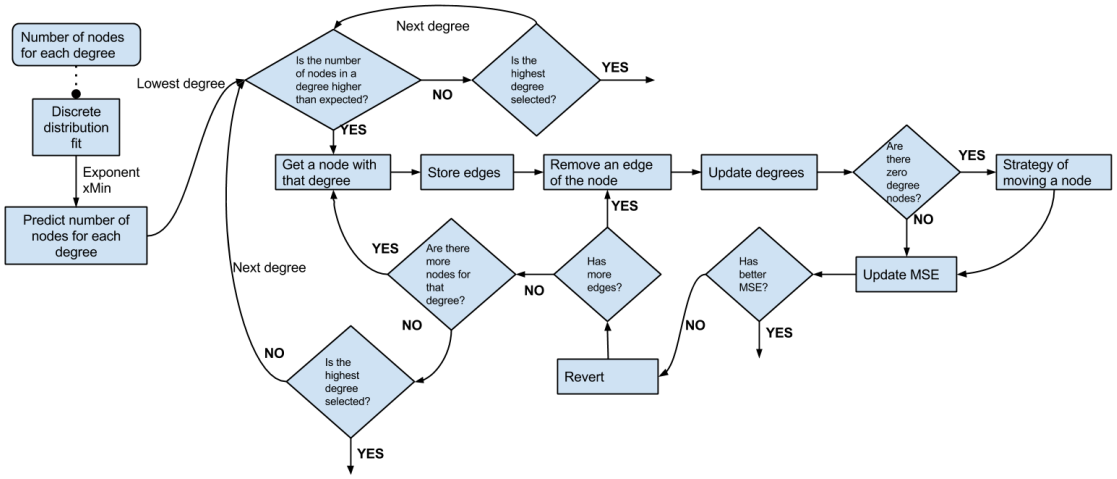

Figure 11 Decrease degree flow

The first 3 steps, are the explained previously, it allow to calculate a theoretical distribution with the experimental data and be able to obtain how many nodes should have in each degree.

After that the strategy for decrease the number of nodes for a degree starts. The modulo go over all the degrees of the network starting for the lowest one, checking each one if need to be decrease the number of nodes to get closer to the theoretical distribution.
Next step is get a node of that degree, no matter which, and store his edges, and remove each one, calculating the degree distribution of the actual network, and the MSE has now. If the process of remove the edge leaves a node with zero degree the strategy of moving a node into the network would be applied for that one.

If the MSE is better that before, the process is considered finish, and the application will stop the strategy. If not, the changes will be revert and will try with other edge, other node of the same degree or will skip to another degree and will do the flow again until it found improved change or there are no more nodes to try.

Each edge or node introduced in the flow are store in a list of excluded nodes or edges with the purpose of tracing which were selected and not select it again.

4.3.2 Strategy of increase a degree

When a node have a degree with the actual number of nodes in that specific degree is lower than the expected, this strategy is applied. An edge is added to that node connecting with a random node from the network.

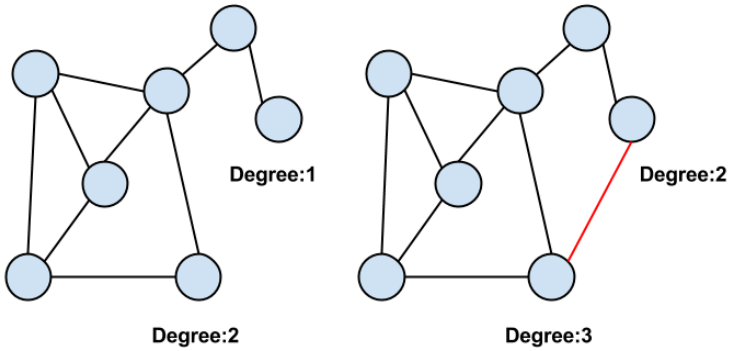

Figure 12 Simple degree increasing

The next figure show the flow for the strategy of increase a degree in the application context:

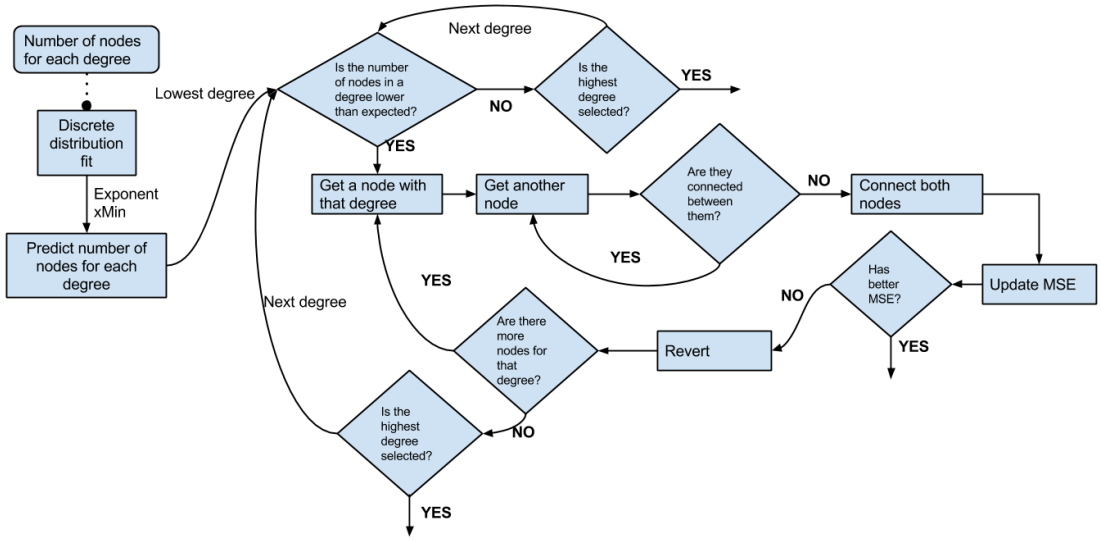

Figure 13 Increase degree flow

As usual, the three first steps it allow to calculate a theoretical distribution with the experimental data and be able to obtain how many nodes should have in each degree.

Once a degree has less nodes than expected, a node with that degree and a randomly node from the network with the same degree is selected. It checks if both nodes are connected between them, if is true, another node is randomly selected, if they are not connected it creates an edge between them.

With the new edge the MSE is calculated again, in the case that improve, the process is finished. If not, the changes in the network are revert and another node with a degree with less nodes than expected is selected.

Each node with the degree with less nodes than expected introduced in the flow are store in a list of excluded nodes with the purpose of tracing which were selected and not select it again.

4.3.3 Strategy of moving a node

This strategy useful either entry a new node to the network or re-entry a node that due to the other strategies became isolated from the network. The idea is to connect this new node to other random nodes from the network until it have the average degree of the network.

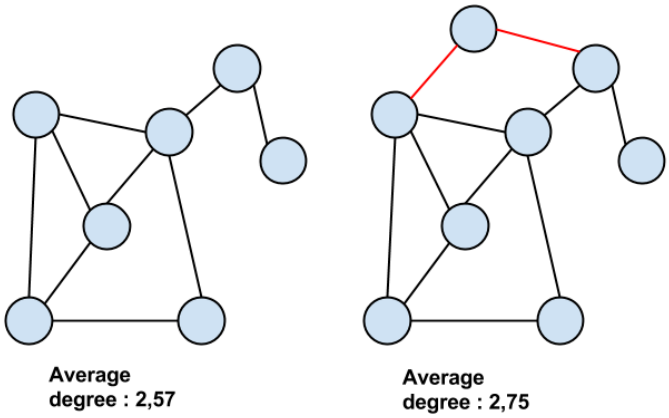

Figure 14 Simple node entry

The next figure show the flow for the strategy of moving a node in the application context:

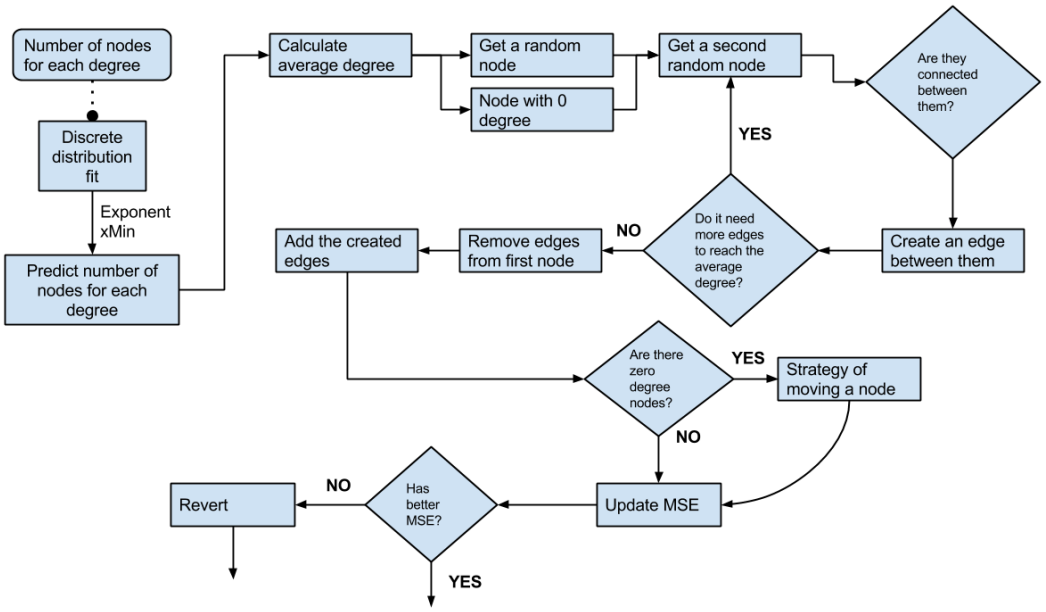

Figure 15 Moving a node flow

This flow is the one use to add those nodes that had zero degree due for some change made in the network. Also can be use like a normal one to approach the power law distribution.

It starts calculating the average degree in the network, to have some control on the modification in the degree distribution. After that, it choose 2 nodes in a randomly way, or just one and a zero node degree, check aren't connected, and create an edge between them. But those edges are not currently in the network yet.

Once it have enough edges for the first or the zero node degree have an average degree, the actual edges for that node are removed from network and added the edges were created before.

Finally, like others strategies, it checks if there are any new zero node degree, in case it find someone, the process repeats for that node or nodes. After that, the MSE is recalculated and evaluated, if improves, the change are permanent, if not, the changes are reverted, any case the process is finished.

4.4 Export module

The export module handle the output of the application, from a representative graph in a PDF or GEXF format to a text file with metrics of the network. And this process can be done at start, end or in between changes either each one of the changes or by intervals. This is set by code before run the application, but by default the application will create each output every time the MSE is at least 10% lower than before the change.

4.4.1 Graph output

This outputs is a representative picture of the network at one moment.

The network is treated to apply some filters in order to improve the visualization:

- Color rank by degree, a paler color means less degree.
- Size rank by centrality, a bigger size means more centrality. By size means both node and the label of it.
- An algorithm from Gephi is applied in order the network is more visually appealing

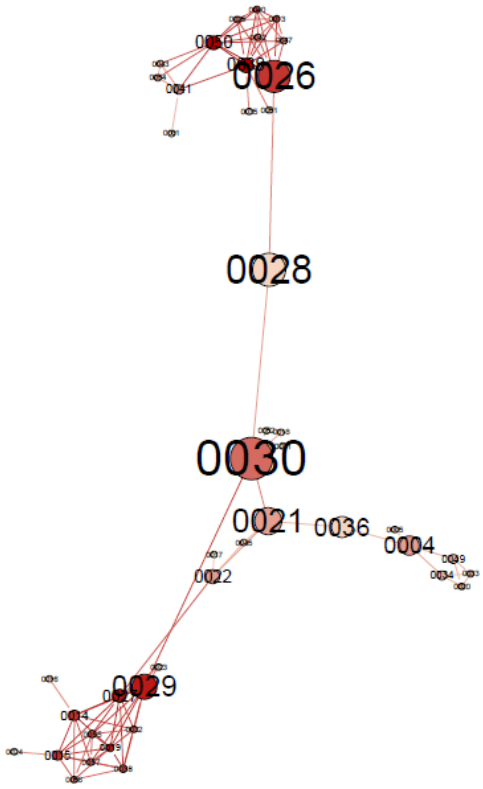

Figure 16 Output graph example

The figure above is an example of who visually look a network after the filters and algorithm are applied

4.4.2 Text output

The text output contains several metric of the network in that instant using the statistics module that can be useful:

- Diameter
- Average, maximum and minimum degree
- Average Clustering Coefficient
- Edge and nodes count
- Path length
- MSE and power law parameters
- Number of nodes for each degree
- Average closeness and eccentricity centrality.

```
Diameter: 15.0
Avrg Degree: 2.109452736318408
Avrg Clustering Coefficient: 0.020555555555555556
Edge count: 212
Node count: 201
Max Degree: 21
Min Degree: 1
Path Length: 6.537313432835821
Diff^2: 19709.948908154416
xMin: 1
Exponent: 1.7700000000000002
Number Nodes for each degree: [141, 16, 8, 11, 10, 6, 4, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1]
Centrality betweennes [avrg]: 553.7313432835821
Centrality clossness [avrg]: 6.537313432835819
Centrality Ecc [avrg]: 11.422885572139304
```

Figure 17 Text output example

The figure of above is an example of how it looks.

CHAPTER 5. RESULTS ANALYSIS

In this chapter will be analyzed a couple of networks that was used in the application. First the original graph will be presented, after the results obtained from the application will be represented.

5.1 Mesh base graph, small network

The overlay of the this network and this properties are the following:

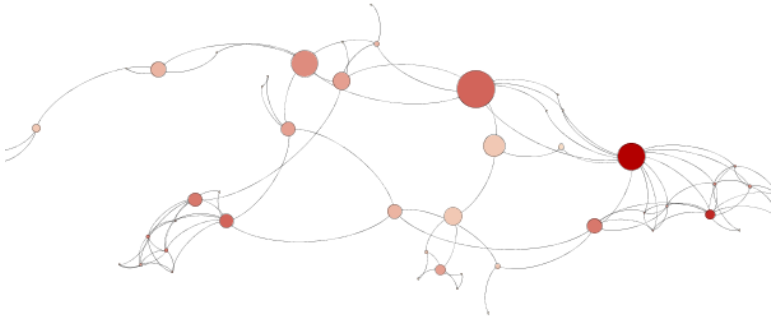

Figure 18 Original mesh base graph

Diameter	8
Average Degree	4.26
Average Clustering coefficient	0.69
Path Length	3.78

Table 3 Mesh original properties

The fit founded by the framework used for this stage are considering the values of $x_{min,}$ 4 and an exponent of 2,9399999999999804. The next table represent the degree distribution for these values.

Degree	# nodes	Theoretical # of nodes	$Diff^2$
1	4	13,8731607	97,4793016
2	14	8,10502748	34,750701
3	7	6,254376	0,55595515
4	6	5,38011259	0,3842604
5	7	4,89332126	4,43809529
6	4	4,59729219	0,35675796
7	4	4,40753125	0,16608172
8	2	4,28175256	5,20639475
9	0	4,19652458	17,6108185
10	0	4,13790776	17,1222806
11	1	4,09717927	9,59251944
12	0	4,06867883	16,5541474
13	1	4,04863613	9,29418224
		TOTAL:	213,511496

Table 4 Degree distribution

The first column represent a specific degree, the second column is the number of nodes of one degree that are in the network, the next column is the predicted number of nodes for the found values for the fit, and the last column is de square difference between the theoretical and empirical values. And the total is just the sum of all the values of the last column.

So before enter to the network at the end of the process, we can expect that the number of nodes with one degree will increase from only 4 number of nodes for this degree. The square difference will decrease, so the empirical and theoretical values will be similar. Possibly due to the nature of scale-free network, the diameter, the average clustering coefficient and path length will decrease as well.

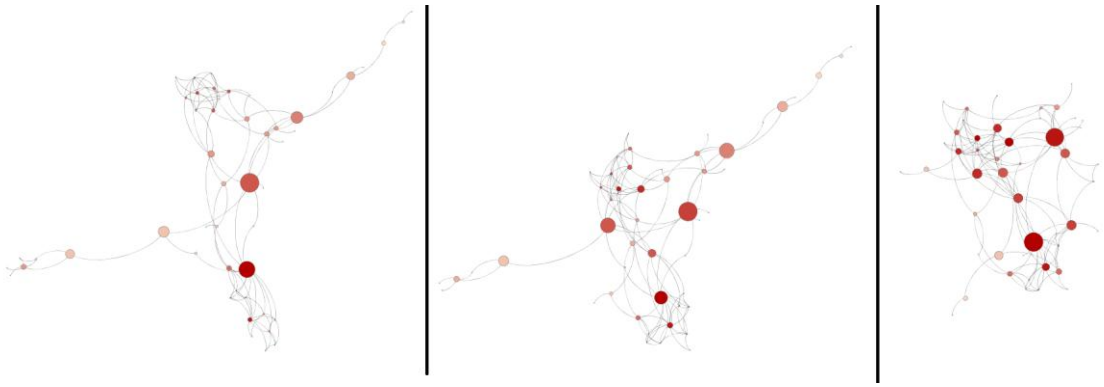

Figure 19 Evolution mesh core graph

The previous figure is a representation of the graph at the very early stages of the changes, a medium stage and the final stage. Is a way to see how the network become more and more clustered and the hubs nodes are defined.

Diameter	6
Average Degree	5.04
Average Clustering coefficient	0.508
Path Length	2.73

Table 5 Mesh properties at final stage

As it was explain the diameter, clustering coefficient, and path lenght decreased. Furthermore the average degree increased due probably to the high density hubs.

The next table shows the degree distribution at the final stage, the values for the fit at this point are of $x_{min,}$ 3 and an exponent of 2,83:

Degree	# nodes	Theoretical # of nodes	$Diff^2$
1	11	10,1041646	0,80252101
2	6	5,93283295	0,00451141
3	5	4,59816921	0,16146798
4	4	3,97020452	0,00088777
5	4	3,62238242	0,14259504
6	3	3,41220228	0,16991072

7	3	3,2784567	0,07753814
8	3	3,19053014	0,03630173
9	3	3,13148107	0,01728727
10	3	3,09125591	0,00832764
11	2	3,06358708	1,13121747
12	3	3,04442802	0,00197385
		TOTAL:	2,55454004

Table 6 Final Degree distribution

The data show a long tail with high degree hubs, a characteristic of scale-free networks, and the square difference decrease significantly.

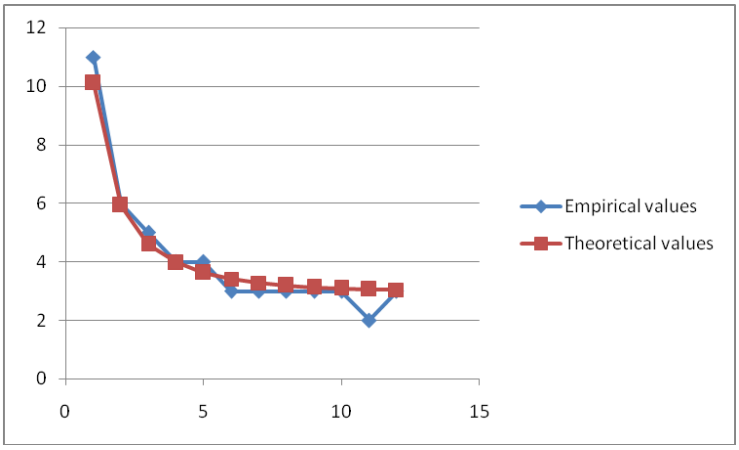

Figure 20 Exponential representation of the degree distributio

The previous figure is the graphical representation of the empirical and theoretical values. As there are still some errors, there are some values that not are exactly the same value as the theoretical number, also has to consider that in the practical scenario is a discrete environment, while the theoretical one is continues. Anyway the practical log-log should be close to a straight line.

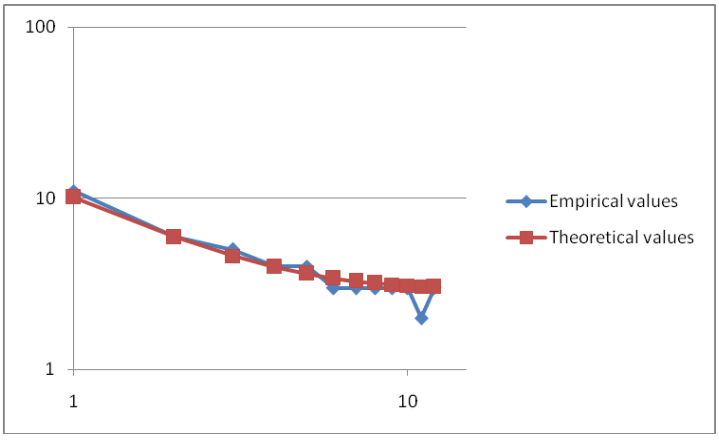

Figure 21 Log-log degree distribution

5.2 Cast core graph, big network

The overlay and properties of this network are the following:

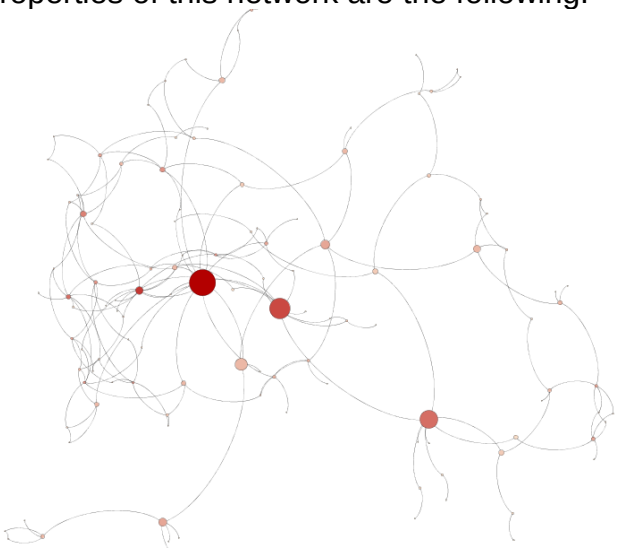

Figure 22 Cast graph original overlay

Diameter	10
Average Degree	2.9
Average Clustering coefficient	0.18926141014053102
Path Length	4.7747899159663865

Table 7 Cast original properties

The fit founded by the framework used for this stage are considering the values of $x_{min,}$ 1 and an exponent of 1.59. Becaouse the high number of nodes the degree distribution will be shown with a graphical representation.

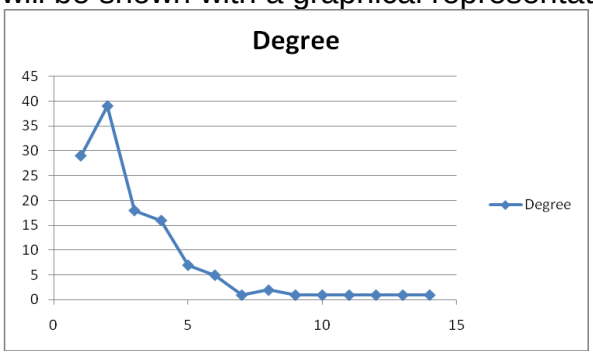

Figure 23 Original Cast graph degree distribution

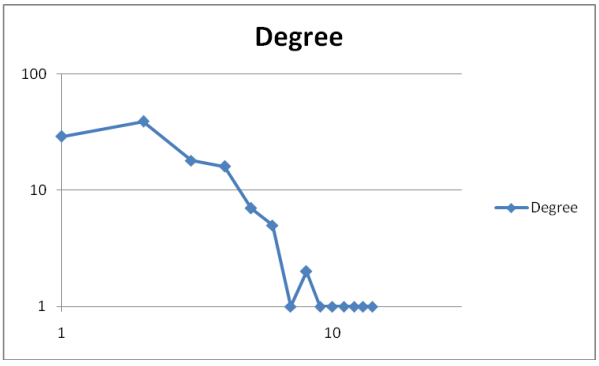

Figure 24 Log-log degree distribution of Cast graph

As it can be see, the degree distribution it is not a power law at low degrees where are less one degree nodes than two degrees nodes, for example, the final result should look more like a exponential function in his degree distribution.

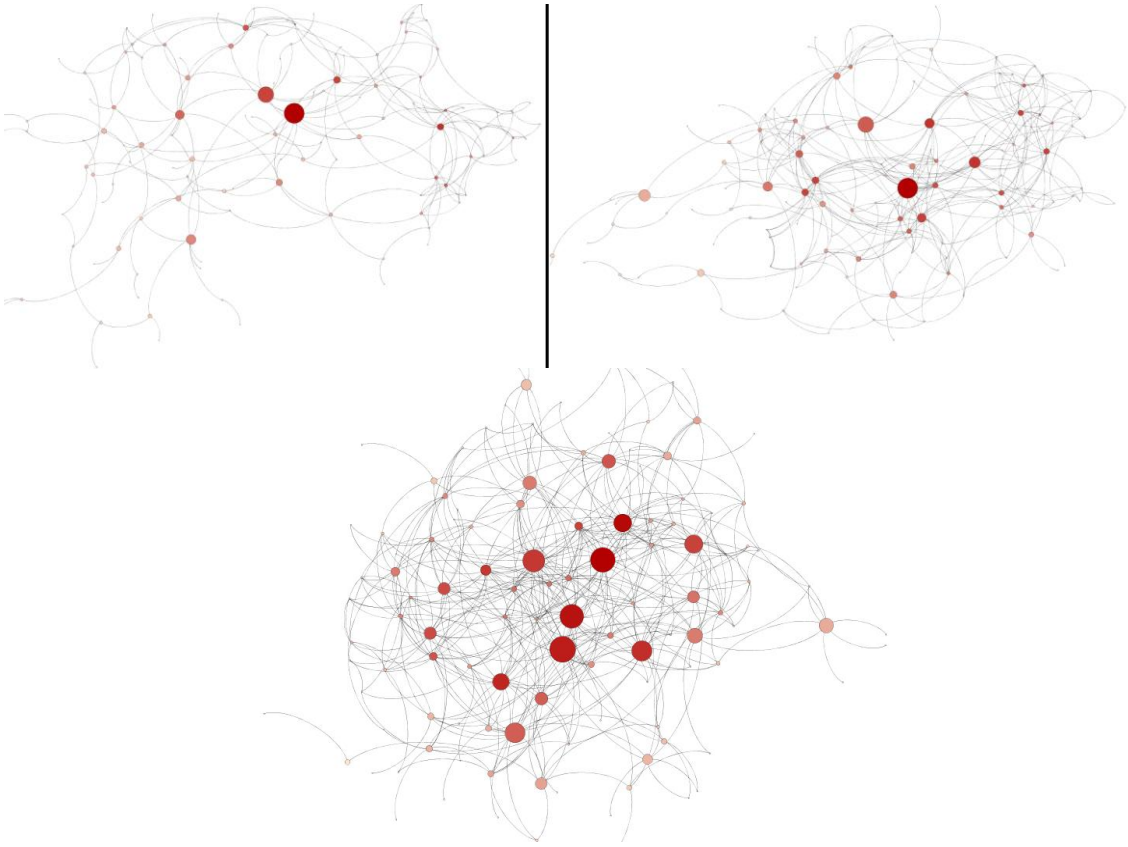

Figure 25 Evolution of Cast core graph

As before the previous figure, represent a early stage, a medium stage and the result of all the changes. The network again become more and more clustered and the hubs nodes are defined. The same effect occurs on his properties, diameter and path length are decreased while the average degree is increased, but clustering coefficient also increased a bit, that may be due to the framework to calculate the fit that added an offset at the values of the tail, so, in networks where the clustering coefficient were already low can be affected negatively:

Diameter	5
Average Degree	8.3
Average Clustering coefficient	0.23216677458240154
Path Length	2.6656862745098038

Table 8 Final network properties

At the end of the process the $x_{min,}$ is 5 and an exponent of 3.49, as before the degree distribution will be shown as a graphical representation

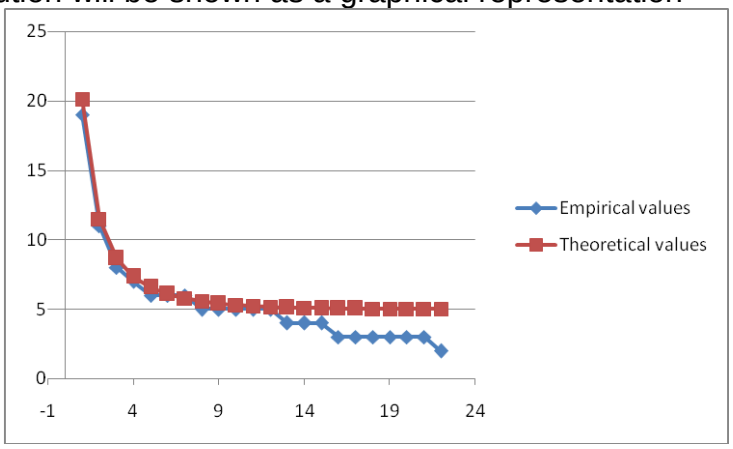

Figure 26 Degree distribution result

And the log-log of the degree distribution where should be like a straight line until the values start to stabilize.

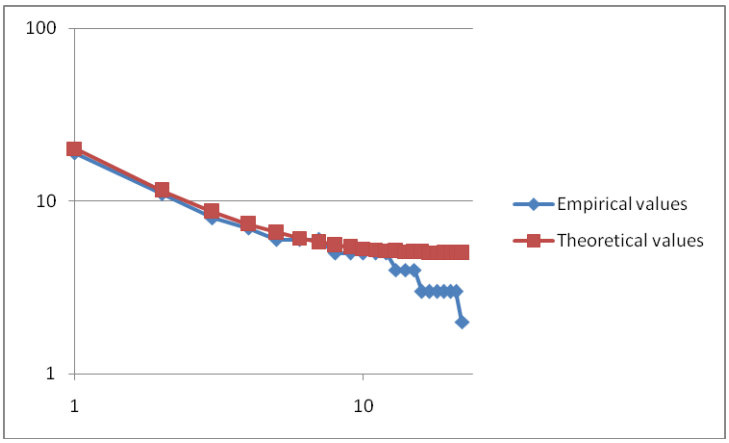

Figure 27 Log-log degree distribution result

CHAPTER 6. CONCLUSIONS

At the beginning the objectives were exposed which all of them were achieved in his majority:

- The developed application is able to calculate a fit of a power law with the degree distribution of a given network
- Have been developed several strategies which together make possible the modification of a existing network in order to meet a power law in his degree distribution
- The application accepts the principal extensions of networks as his input.
- A text output with several properties of the current status of a network is created every step of the process
- Also a graphical output of the overlay network is generated every defined step

The project have test that with simple techniques in a practical view is possible to make changes to a network and approach his degree distribution to a power law previously calculated.

6.1 Future improvements

Is possible that the framework used to do the power law fit can be improve, as it have been exposed this fit apply an unwanted offset that can affect negatively some properties like the clustering coefficient of a network, so a future improve will be substitute the current power law framework with another way to calculate a proper fit.

Also, exist the possibility in the process of the network changes that some group of nodes become isolated. This is solved for alone nodes but not for a group of nodes connected between them but not with the main network. So another future improve will be investigate a way to detect these situations a correct them, maybe identifying communities and check all of them are connected with each one, for example.

REFERENCES

Gephi: Oficial web of Gephi platform
URL: http://gephi.github.io/

Gephi-Toolkit: Oficial wiki for Gephi Toolkit
URL: https://wiki.gephi.org/index.php/Toolkit_portal

- Gephi-Toolkit's javadoc.

URL:https://gephi.org/docs/toolkit/

Data2Semantics:Github of data2semantics
URL: https://github.com/Data2Semantics/powerlaws

Network's properties: Web pages with information about network properties, power law distribution and scale-free networks
URL: http://mathinsight.org/
URL: http://www.richardclegg.org/previous/networks2/SpecialLecture_06.pdf
URL: https://www.wikipedia.org/
URL: http://p2pfoundation.net/
URL: http://tuvalu.santafe.edu/~aaronc/powerlaws/