Analysis of a Model of Social Networks

Weichun Xu

Harinath T Prabhakaran

**Contents**

[**1.** **Social Network** 5](#_Toc392146587)

[**2.** **Preface of our project** 5](#_Toc392146588)

[**3.** **Model of Social Network** 5](#_Toc392146589)

[**3.1.** **Algorithm of the Model** 5](#_Toc392146590)

[**3.2.** **Motivation of the Model** 5](#_Toc392146591)

[**3.2.1** **Implementation of Step 1 of the Model** 5](#_Toc392146592)

[**3.2.2** **MATLAB code** 6](#_Toc392146593)

[**3.2.3** **Implementation of Step 2 of the Model** 9](#_Toc392146594)

[**3.2.4** **Implementation of Step 3 of the Model** 10](#_Toc392146595)

[**3.2.5** **Implementation of Step 4 & 5 of the Model** 12](#_Toc392146596)

[**3.2.6** **Generate Four Networks** 17](#_Toc392146597)

[**4.** **Vertex degree distribution** 19](#_Toc392146598)

[**5.** **Clustering spectrum** 22](#_Toc392146599)

[**6.** **Average Nearest-Neighbors Degree** 25](#_Toc392146600)

[**7.** **Average Shortest Path Lengths** 28](#_Toc392146601)

[**8.** **Average Number of k-clique-communities** 32](#_Toc392146602)

[**9.** **Conclusion** 38](#_Toc392146603)

[**10.** **Contribution** 38](#_Toc392146604)

**List of figures**

[Figure 1 GUI interface for simulation 7](#_Toc392146605)

[Figure 2 generate random network with 10 vertices 8](#_Toc392146606)

[Figure 3 the connection among the 10 vertices 8](#_Toc392146607)

[Figure 4 the network grows to50 vertices (Pinit = (1, 0.95; 2, 0.05), Psec = uni(0~3)) 15](#_Toc392146608)

[Figure 5 the connection among 50 vertices (Pinit = (1, 0.95; 2, 0.05), Psec = uni(0~3)) 16](#_Toc392146609)

[Figure 6 the network grows to 500 vertices (Pinit = (1, 0.95; 2, 0.05), Psec = uni(0~3)) 16](#_Toc392146610)

[Figure 7 the connection among 500 vertices (Pinit = (1, 0.95; 2, 0.05), Psec = uni(0~3)) 17](#_Toc392146611)

[Figure 8 degree distribution with vertices No. 1000 19](#_Toc392146612)

[Figure 9 degree distribution with Vertices No. 1100 20](#_Toc392146613)

[Figure 10 degree distribution with Vertices No. 2000 20](#_Toc392146614)

[Figure 11 degree distribution with Vertices No. 5000 21](#_Toc392146615)

[Figure 12 clustering spectrum with Vertices No. 1000 23](#_Toc392146616)

[Figure 13 clustering spectrum with Vertices No. 1100 23](#_Toc392146617)

[Figure 14 clustering spectrum with Vertices No. 2000 24](#_Toc392146618)

[Figure 15 clustering spectrum with Vertices No. 5000 24](#_Toc392146619)

[Figure 16 Knn with Vertices No. 1000 26](#_Toc392146620)

[Figure 17 Knn with Vertices No. 1100 26](#_Toc392146621)

[Figure 18 Knn with Vertices No. 2000 27](#_Toc392146622)

[Figure 19 Knn with Vertices No. 5000 27](#_Toc392146623)

[*Figure 20 Shortest Path with Vertices No. 1000* 29](#_Toc392146624)

[*Figure 21 Shortest Path with Vertices No. 1100* 29](#_Toc392146625)

[*Figure 22 Random network with Vertices No. 10* 30](#_Toc392146626)

[*Figure 23 network grow to Vertices No. 10* 30](#_Toc392146627)

[*Figure 23 average shortest path of some network is less than 1* 31](#_Toc392146628)

[*Figure 25 cFinder is used to calculate community size* 32](#_Toc392146629)

[*Figure 26 Number of Communities with Vertices No. 1000* 35](#_Toc392146630)

[*Figure 27 Number of Communities with Vertices No. 1100* 35](#_Toc392146631)

[*Figure 28 Number of Communities with Vertices No. 2000* 36](#_Toc392146632)

[*Figure 29 Number of Communities with Vertices No. 5000* 37](#_Toc392146633)

[*Figure 30 Number of Communities with Vertices No. 9000* 37](#_Toc392146634)

1. **Social Network**

Social Networks is the Economic, political and social interaction between people forming relationships. The interactions could be in the form of relationships among relatives, friends, colleagues with whom they share their thoughts, risks and favors on regular basis and get influenced with the decision they take from doing business to politics.

1. **Preface of our project**

It is evident that theoretical studies of processes and collective behavior taking place on social networks would beneﬁt from realistic social network models. Essential characteristics for social networks are believed to include assorted mixing, high clustering, short average path lengths, broad degree distributions, and the existence of community structure. Here, we propose a new model that exhibits all the above characteristics. The motive of our project is to grow a random network and analyze it by generating certain algorithms, compare it to the ideal model and predict the algorithm’s accuracy.

1. **Model of Social Network**

Here we are about to construct a social networking model, which consists of vertices (nodes) and links formed between them. This network is restricted to some basic constraints which are explained below.

* 1. **Algorithm of the Model**

1. start with a seed network of N0 vertices
2. pick on average mr1 random vertices as initial contacts
3. pick on average ms0 neighbors of each initial contact as secondary contacts
4. connect the new vertex to the initial and secondary contacts
5. repeat steps 2~4 until the network has grown to desired size
   1. **Motivation of the Model**
      * 1. **Implementation of Step 1 of the Model**

To realize the step 1 of the model, use Matlab “rand” function to generate a N0 by N0 matrix and compare the generated matrix with “randThresh”. If the element of the matrix is greater than or equal to “randThresh”, set the element as 1, otherwise set it 0. Then set the diagonal element to 0 as the vertices cannot connect to itself. Finally go through all the elements and if element (i, j) is equal to 1, set element (j, i) to 1.

* + - 1. **MATLAB code**

To implement this model we used the MatLab coding, below is the code used which shows how to initialize a random network.







Here we have used the GUI tool of MatLab (Figure 1) to represent the network, the left half shows the plot representing network initialization and the right half shows all the commands available to control and analyze various attributes of the network.

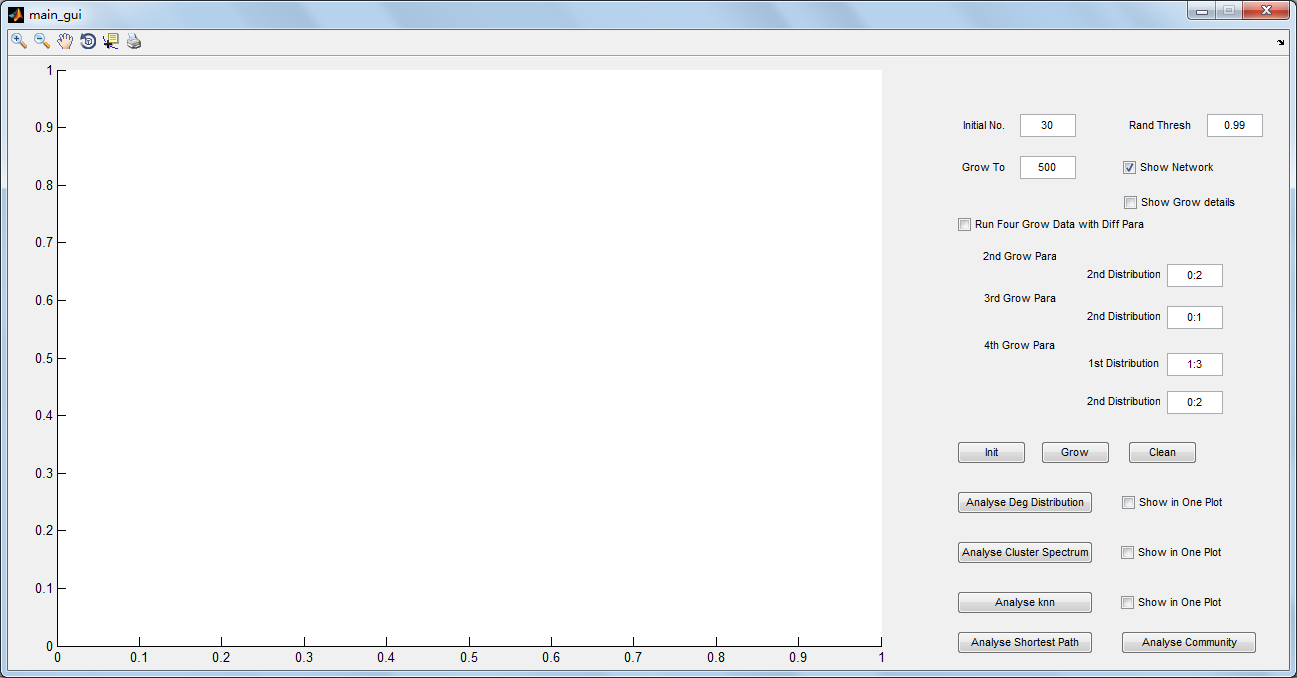


Figure 1 GUI interface for simulation

In Figure *2*, we have initialize the number of nodes to 10 in the beginning and we have specified the random threshold value of 0.92 (probability of forming network) and then we are specifying the network growth size to 50 (Figure 3), which is the new network formed with 50 vertices (nodes) in it.

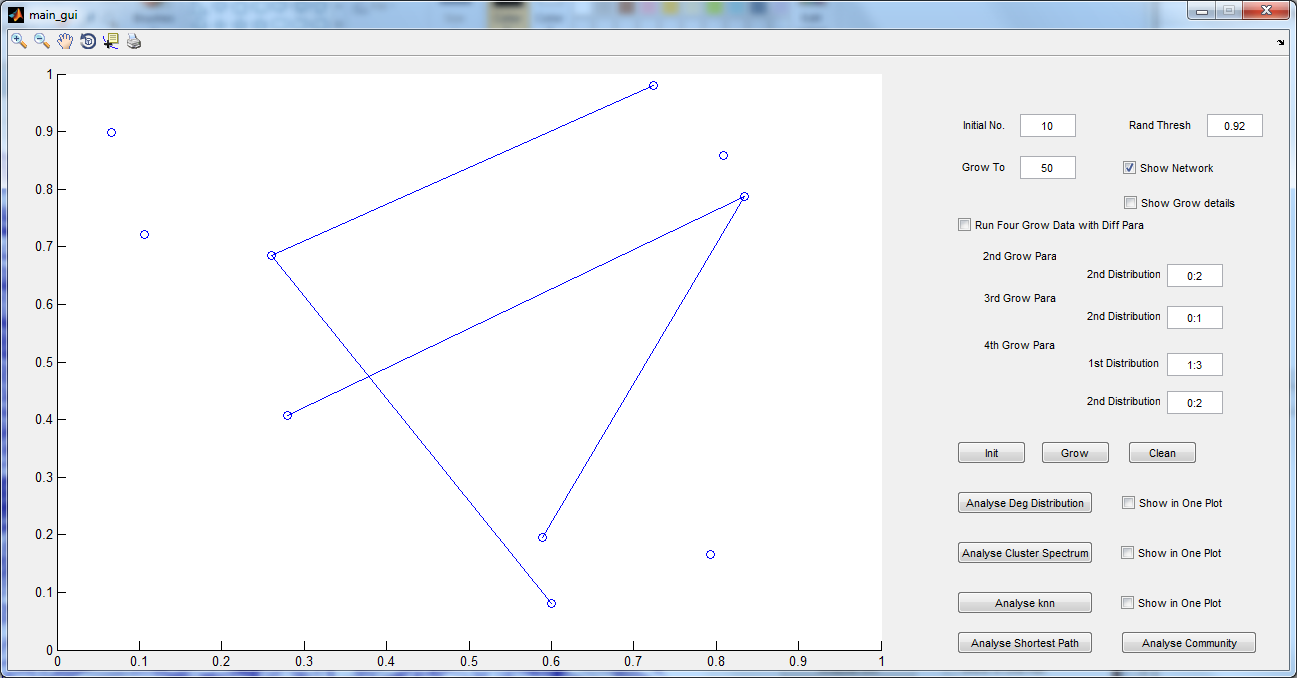


Figure 2 generate random network with 10 vertices

The network can also be represented in a more lucid manner using a tool called “Gephi” (Figure *3*), it functions just the same as a MatLab plot but more clear in network representation. Here we have to specify the number of nodes (vertices) and the number of edges (links) then it automatically constructs and displays a network with all its properties.

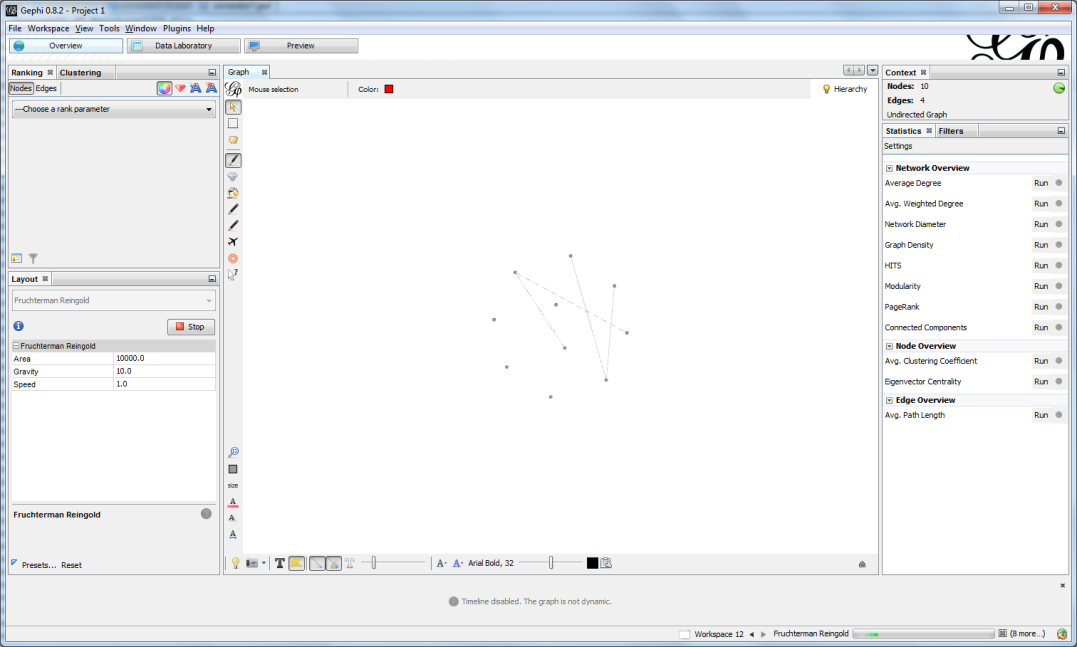


Figure 3 the connection among the 10 vertices

* + - 1. **Implementation of Step 2 of the Model**

**Initial contact:**

Imagine a network with ‘n’ number of nodes and ‘e’ number of edges, now a new node is introduced into this network, now this new node becomes a member of this network by forming linking with any of the already existing nodes, which is the initial contact.

* + - 1. **Motivation of step 2**

Here we are about to pick an average of mr1 random vertices as initial contact to grow our particular network. To realize this step in the model, we used the code “mr\_prob\_matrix\_index = 1 + sum(rand() >cumsum(mr\_prob\_matrix(:, 2)));” to generate the random number of initial contacts for a new vertex and the distribution of the number is based on the probability matrix mr.

* + - 1. **MatLab code**

For motivating the process of picking random number of vertices of initial contact we developed a function which generates some random number as the initial contacts. The MatLab code below represents that function.



Depending on number of initial contacts chosen, use the function ‘randi’ to randomly pick up a first contact from the existing vertices in the network and this process is carried out in MatLab to realize choosing random initial contacts.



* + - 1. **Implementation of Step 3 of the Model**

**Secondary contact**

Once a new node links to and existing network choosing an initial point of contact, it can develop a relationship with the neighbors of the initial point of contact and get some benefit out of it.

* + - 1. **Picking random number of secondary contact**

So here again we have to pick a random number of secondary contacts to link with. To realize this step in the model, for each first contact of the new vertex, use the code “no\_of\_second\_con\_vs\_first\_contact(2, i) = sum(rand() >cumsum(ms\_prob\_matrix(:, 2)));” to get the number of the second contacts. Then use ‘randi’ to randomly pick up the second contacts. However the duplicated secondary contact or the initial contact will be removed automatically when the picking up process is carried out.

* + - 1. **MatLab code to realize this function**







* + - 1. **Implementation of Step 4 & 5 of the Model**
      2. **Connect the new vertex with initial and secondary contacts**

Here we are generating edges (links) between the new node and the existing node in the network with some random number of initial and secondary contacts chosen.

* + - 1. **Motivation of step 4 and 5 of the model**

Connect the initial contacts and the secondary contacts with the new vertex to realize the step 4 of the model. Then check if the network has grown to a required size. If not, repeat the previous steps to pick up the first and second contacts for the next new vertex until the network grows to the required size, which is realization of the step 5 of the model.

* + - 1. **MatLab Code**

Below is the MatLab code to form edges with new vertex and repeat previous steps till the network is fully grown.







The plot below (Figure 4) shows the network consisting of 10 initial vertices with random threshold of 0.92 and it is then grown to a full network with 50 vertices. Here the probability of forming link with the first initial contact is 0.95 and the probability of having another initial contact is just 0.05. The probability of the number of secondary contacts the new vertex linking to is a uniform distribution between 0 and 3.

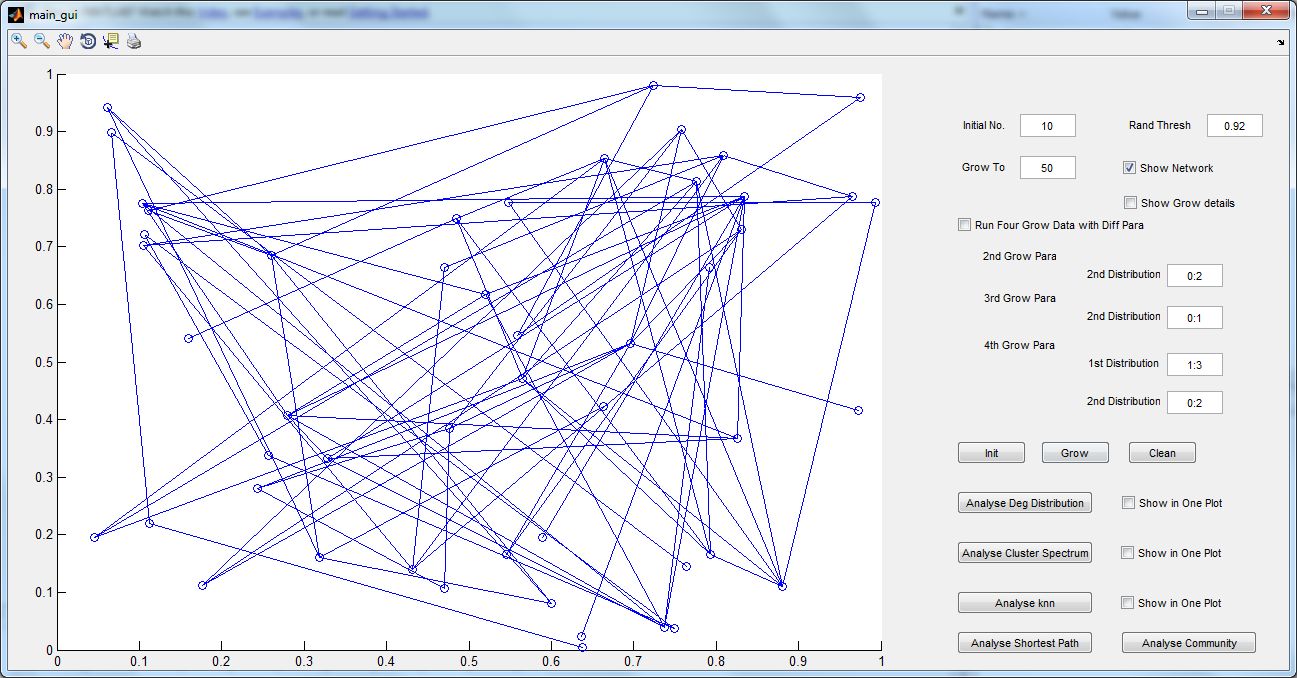


Figure 4 the network grows to50 vertices (Pinit = (1, 0.95; 2, 0.05), Psec = uni(0~3))

This figure below (Figure 5) is another way of representing the same network using the ‘Gephi’ tool, which has 50 nodes and 83 edges.

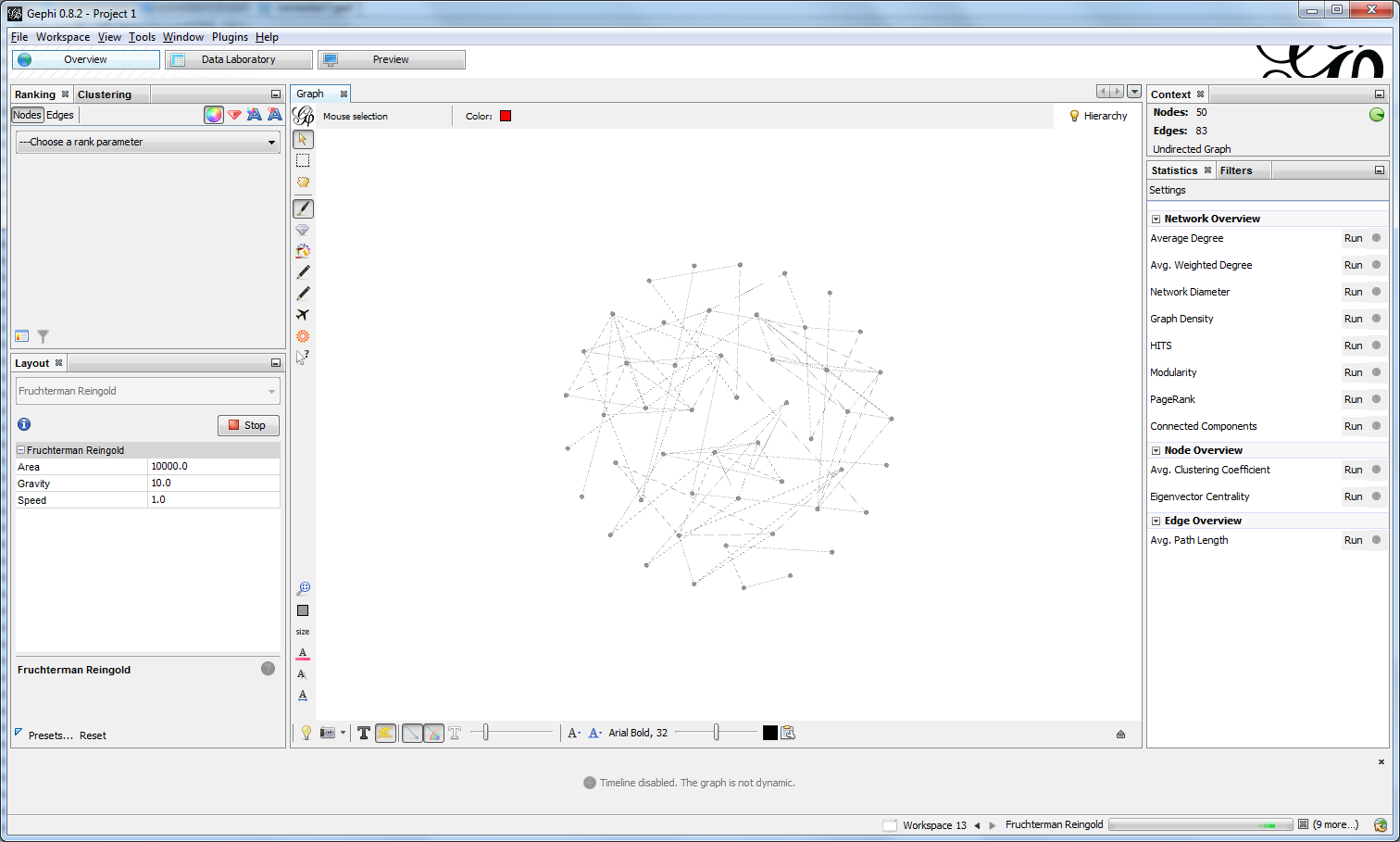


Figure 5 the connection among 50 vertices (Pinit = (1, 0.95; 2, 0.05), Psec = uni(0~3))

Once we have realized a network with 50 vertices, we experimented out algorithm to grow a network containing 500 vertices and checked the network growth. Since it is a GUI representation, we can directly specify the network size in the boxes given.

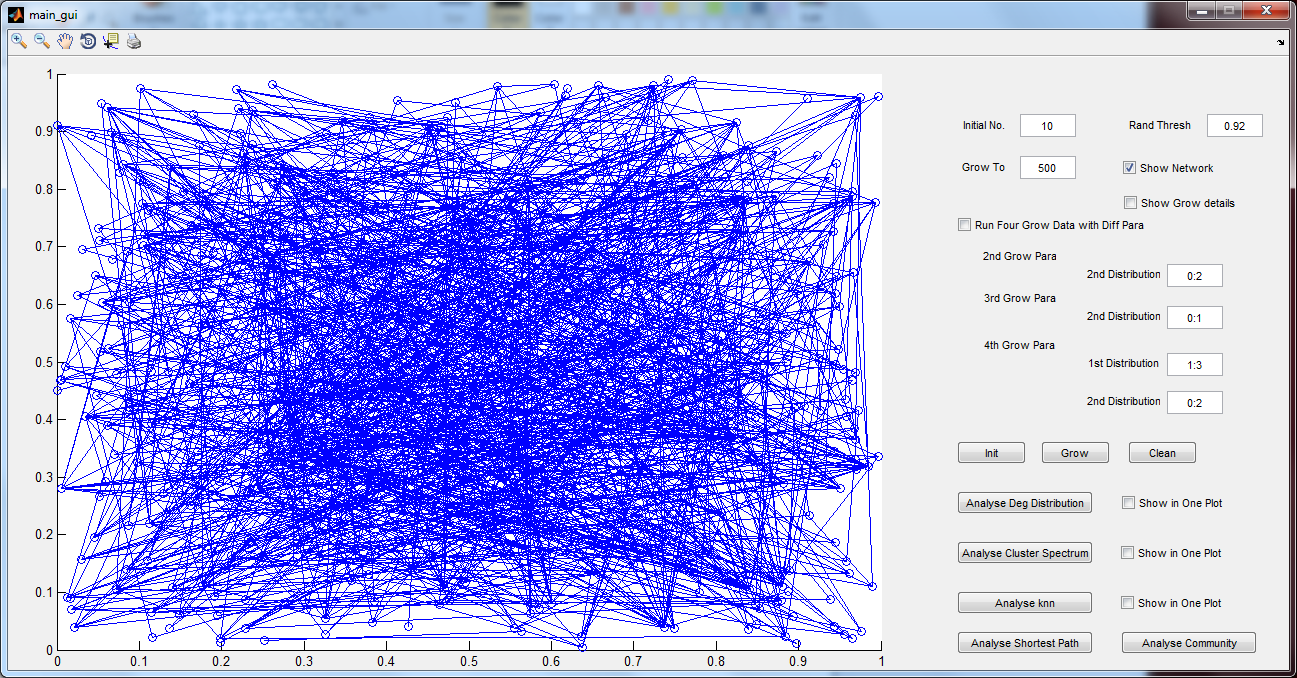


Figure 6 the network grows to 500 vertices (Pinit = (1, 0.95; 2, 0.05), Psec = uni(0~3))

The above plot (*Figure 6*) is the MatLab way of representing the network with 500 nodes and links formed between them, the network is an undirected network and it is not clearly plotted and looks like a mess.

In order to represent the network with 500 nodes in a more presentable manner we made use of the “Gephi” tool again (*Figure 7*). Here the network area is a bigger circle consisting of 500 nodes, in which the number of edges is 1180 to form a fully grown network. Here the connections between various nodes are spotted clearly unlike the MatLab plot.

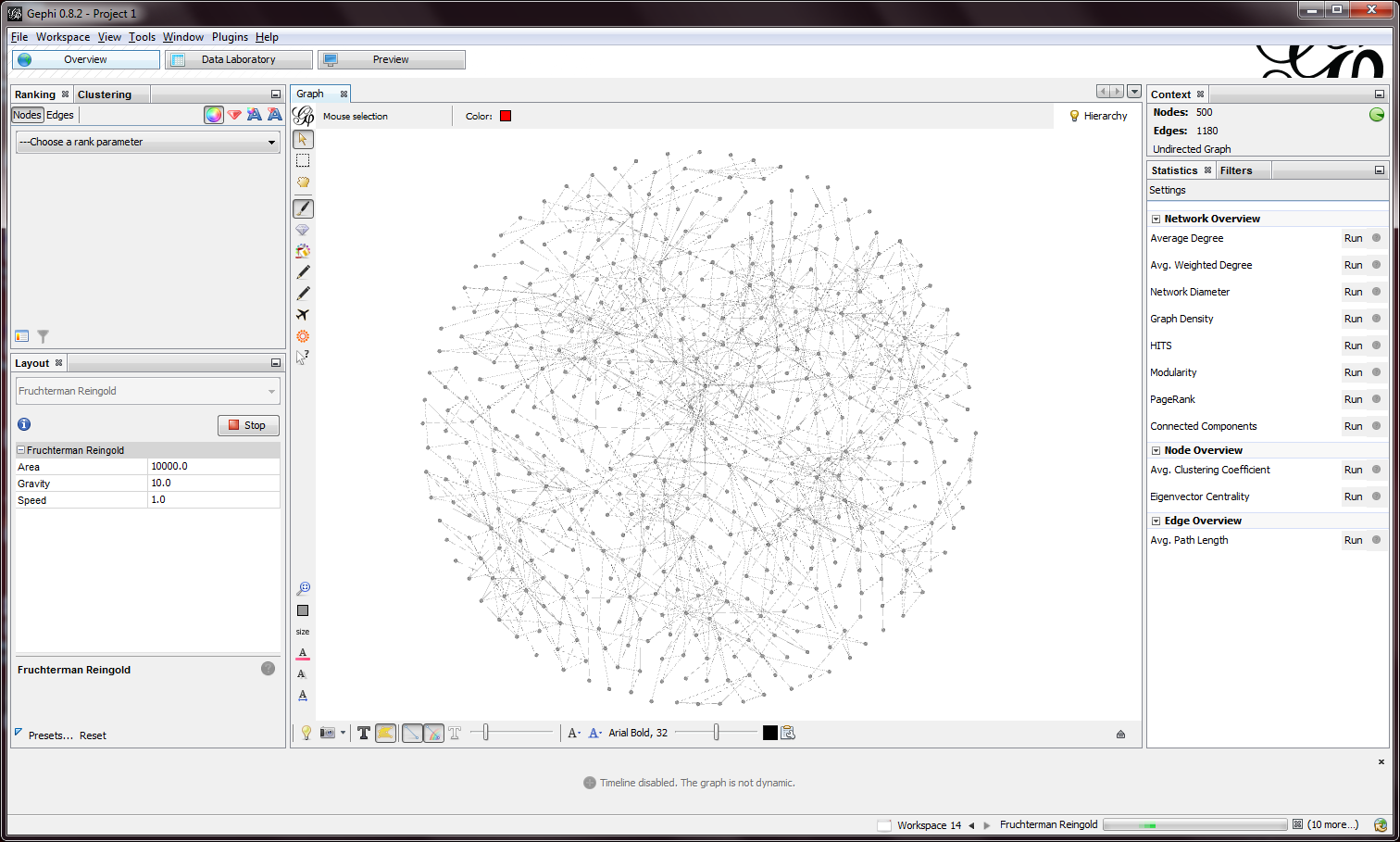


Figure 7 the connection among 500 vertices (Pinit = (1, 0.95; 2, 0.05), Psec = uni(0~3))

* + - 1. **Generate Four Networks**

In the simulation, we generate four different networks which are based on different probability distribution of mr and ms.

**Network1:**

mr = ms =

Here probability of forming links with first initial contact is 0.95 (one initial contact) and 0.05 (two initial contact). The probability to choose the number of second contacts is ms(0~3), which means the number can be 0~3 with each number having 0.25 probability.

**Network2:**

mr = ms =

Choosing initial contact is same as above, but here we left choosing only 0~2 secondary contacts with each number having probability of 0.33.

**Network3:**

mr = ms =

Choosing initial contact is same as above. Here we are left choosing only 0~1 secondary contact with each number having probability of 0.5.

**Network4:**

mr = ms =

In this case we can pick 1~3 initial contacts with each number having 0.33 probability and choose 0~2 secondary contacts with the same 0.33 probability.

* + - 1. **MatLab code to realize the above function**



1. **Vertex degree distribution**

**Degree**

A degree represents the number of links or neighbors each node (vertices) has. There tends to be a range of degree in a network. If any node has large number of degree then it is considered as one of the important members of the network. Degree characterizes importance and influence in that particular network. Other parameters like the centrality, closeness, betweeness, etc. are all the terms associated with the degree.

From the above network generated with certain number of nodes and vertices, we are finding the degree of nodes and compare it with the total degree associated with each node in a network. It is found that nodes having smaller degree are large in number and vice versa.

**Ideal model**

An ideal model, use the following equation to get the degree distribution: where, ,

**Simulation**

In simulation, get all the values of k and then for each k, count the number of vertices and divide the number by the total vertices to get the degree distribution.

From the figures below, the tendency of the simulation is the same as that of ideal model, but as we cannot simulate for 1,000,000 vertices, thus the result is not very matched with the result on paper.

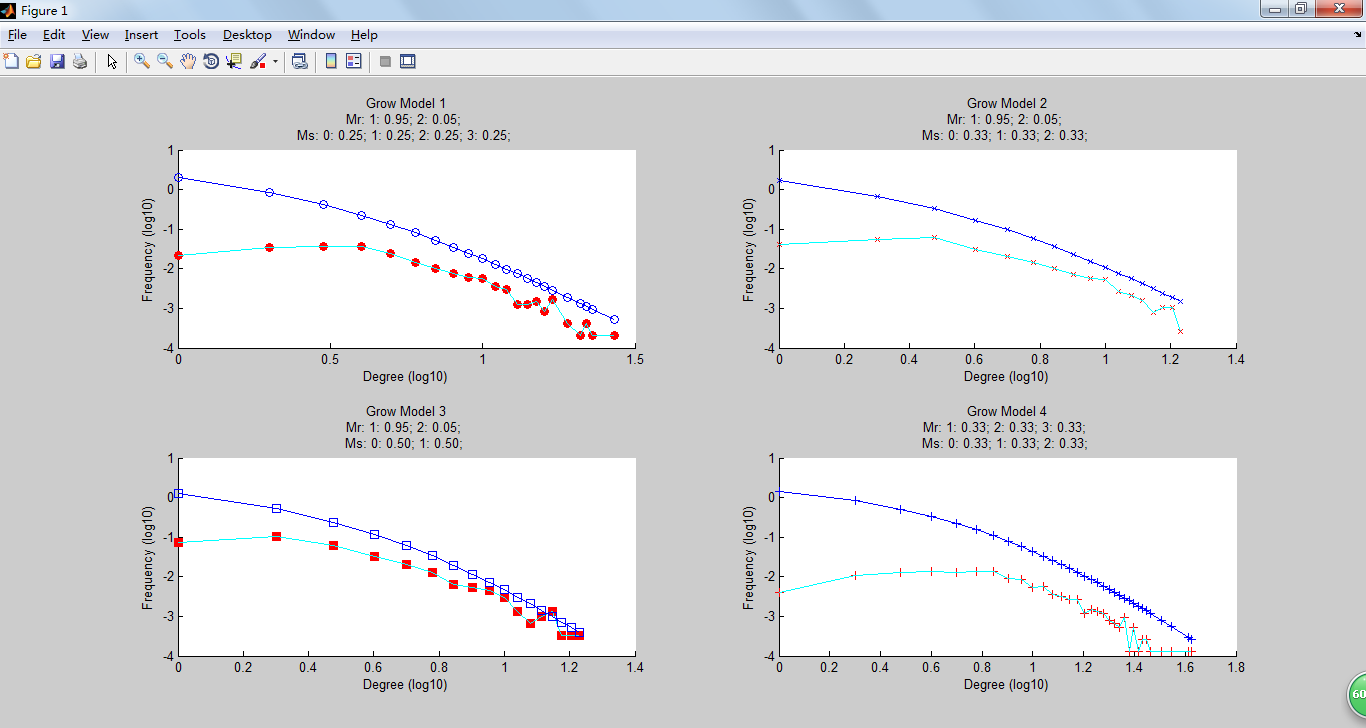


Figure 8 degree distribution with vertices No. 1000

This figure below (Figure 9) shows the number of vertices increased from 1000 to 1100. This increase in the number of vertices contributed the accuracy to increase quite a bit to match with the ideal model.

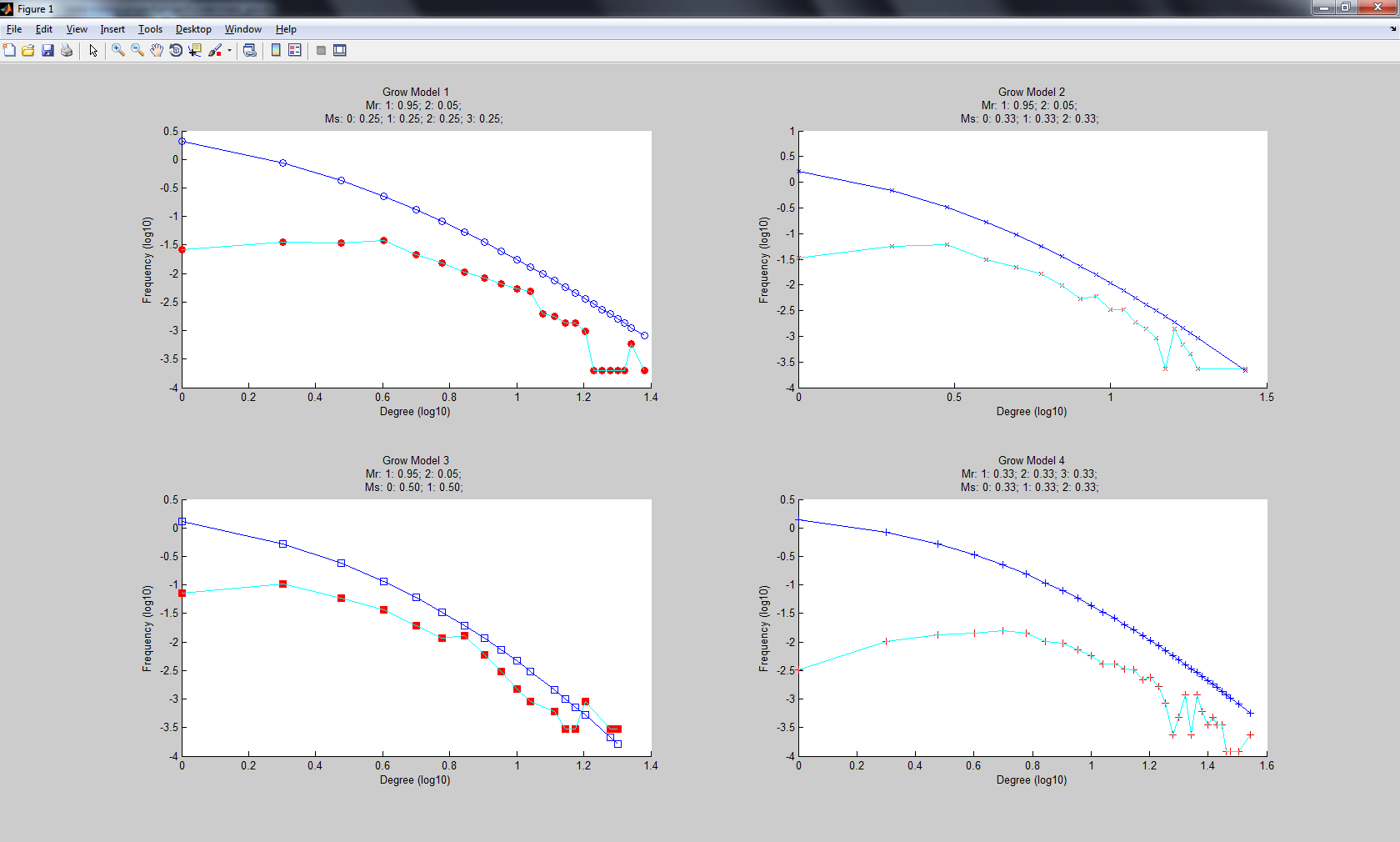


Figure 9 degree distribution with Vertices No. 1100

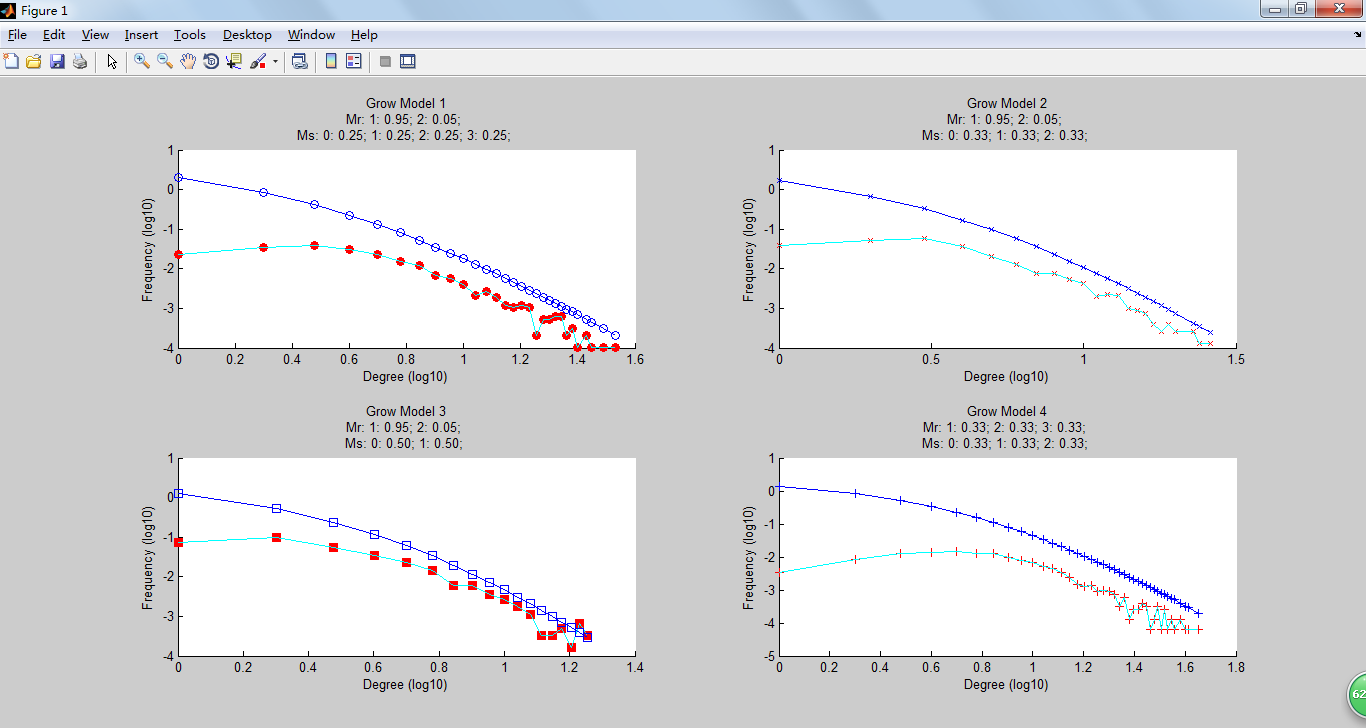


Figure 10 degree distribution with Vertices No. 2000

The above degree distribution (Figure 10) denotes another increase in the number of vertices used to construct the network from 1100 to 2000. This huge increase in the number of vertices contributed a significant increase in the model accuracy where in the growth model we can find that the ideal and the simulated distribution are almost similar with small variations.

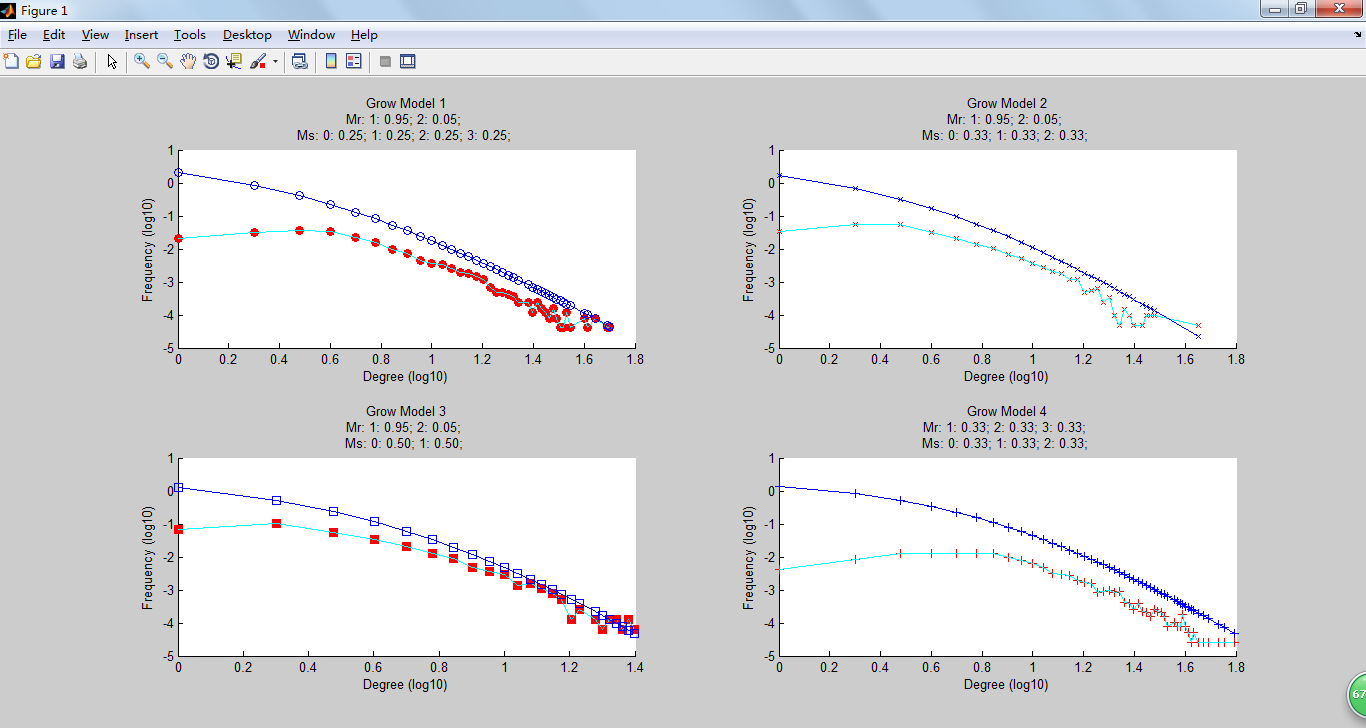


Figure 11 degree distribution with Vertices No. 5000

The above figure (Figure 11) represents the log10 degree distribution with 5000 vertices. Now the accuracy of the simulated model has increased and has less variation in comparison with the prior.

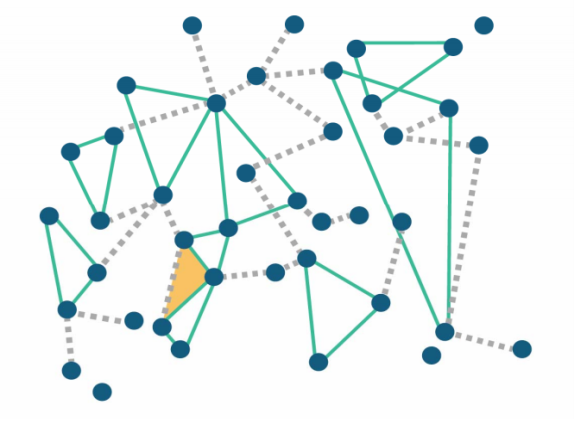
**Result**

Thus we have calculated the degree of each nodes, compared it with the total nodes in the network and jotted down its importance in that particular network. Then we made a comparison between our network of nodes with the ideal network and through this comparison it is clear that, as the number of vertices in a network increase, we could probably achieve close to 100% fit to the ideal model. But this takes a lot of time to simulate and run. For example if the vertices number increased to 1 million we can assume that it can have more than 99% fit but takes weeks to process.

1. **Clustering spectrum**

**Triangles**

Triangles are formed when a vertex has two neighbors and a relationship is formed between those two.



**Clustering spectrum with simulation**

* In the full grown network calculate the real number of triangles formed for each vertices (calculate now many of a node’s neighbor has relationship between themselves)
* Divide the real number of triangles with the potential maximum number of triangles that can possibly form

**Comparison is made between the ideal algorithm model with the algorithm we simulated and accuracy of the algorithm is calculated.**

**Ideal model**

In ideal model, use the following equation to get the clustering spectrum: where, , , ,

**Simulation model**

In simulation model we calculate the degree number of one vertex and based on that the number of maximum triangle can be obtained. Then go through the vertices to check the real number of triangle which is then divided by the maximum number to get the clustering spectrum.

Below figures gives the tendency of the simulation model, which is surprisingly the same as that of ideal model and this result of the simulation matches the result on paper too.

Figure 12, Figure 13 and Figure 14 shows the results of clustering spectrum from 1000 vertices to 2000 vertices. Here we can see that there is not much difference in the accuracy as we increase the number of vertices by a thousand. But on the other hand we found that there is more than 90% match between two models which is a good start.

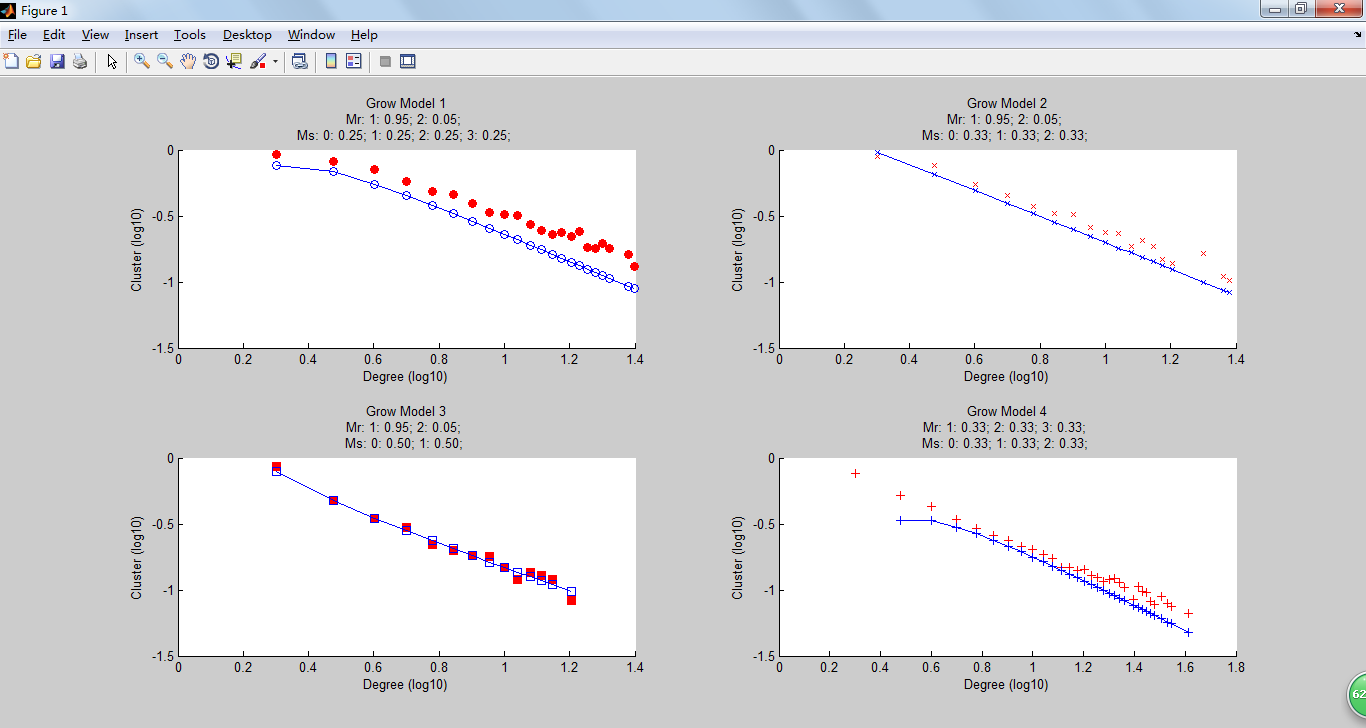


Figure 12 clustering spectrum with Vertices No. 1000

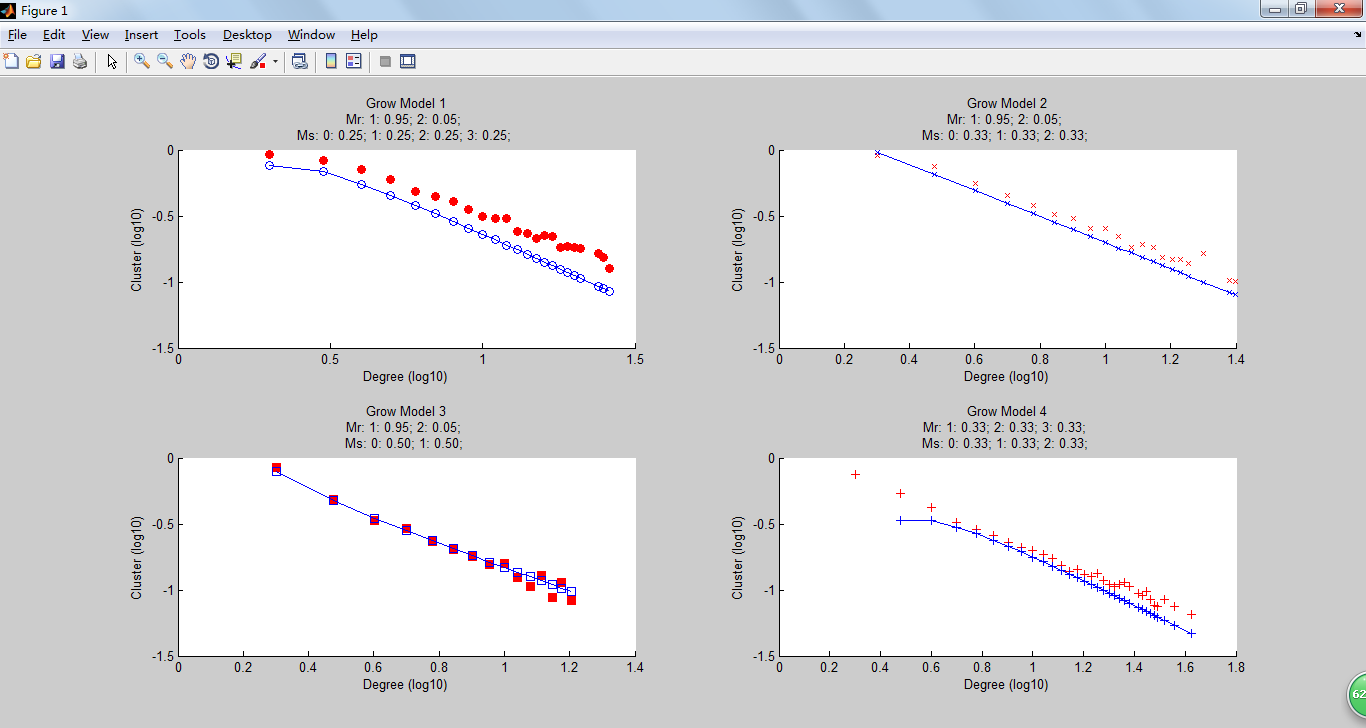


Figure 13 clustering spectrum with Vertices No. 1100

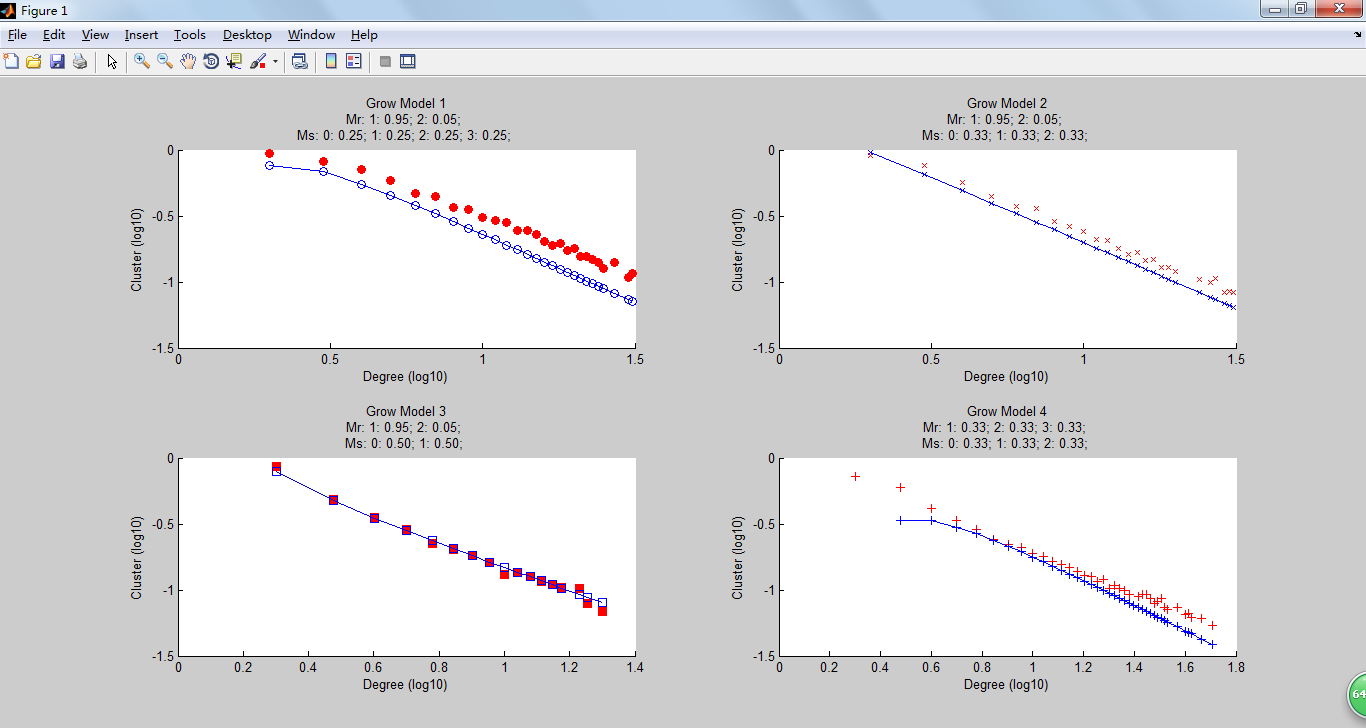


Figure 14 clustering spectrum with Vertices No. 2000

In Figure 15, we have alternated the algorithm a bit by increasing the number of vertices from 2000 to 5000. This is a significant increase in the number of vertices. This drastic change contributed the accuracy efficiency to increase a little bit projecting out more than 95% of fit between the simulated and the ideal model.

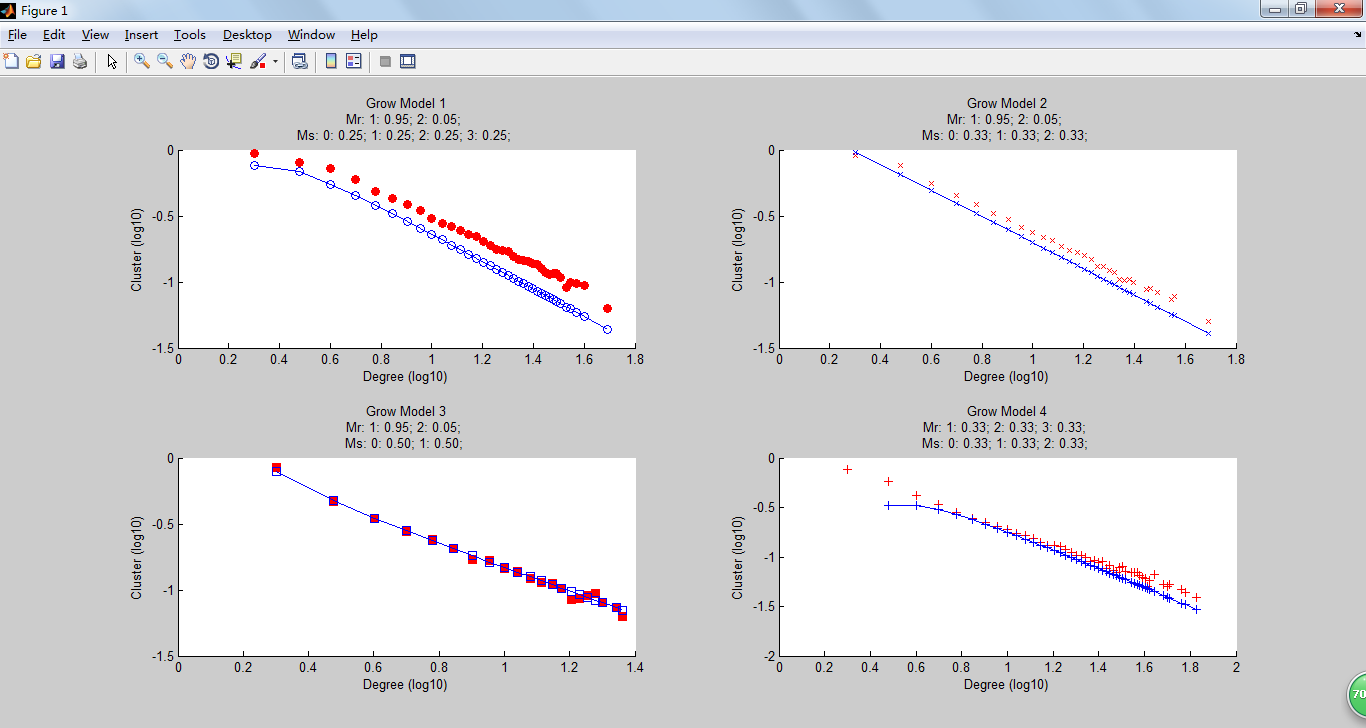


Figure 15 clustering spectrum with Vertices No. 5000

**Result**

In this section of the paper we calculated the number of triangles formed in our simulated network and analyzed its fit with an ideal network. Therefore a comparison is made between the real and the simulated algorithm and thus the efficiency of our algorithm is measured.

1. **Average Nearest-Neighbors Degree**

**Nearest neighbor**

Nearest neighbor degree can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

**Example:**

This is a simple network consisting of 17 nodes, where the red dot in the center is the first member in the network which has 4 degree (neighbor) and each neighbor has two or more nodes connected to them. According to the concept of nearest - neighbor degree the contribution made by the nearest neighbor is more than that of the contribution made through indirect relationship. In probability stand point this could be explained below

To get the average nearest-neighbor degree, the following equation is used: .

𝐹𝑜𝑟 𝑡ℎ𝑒 𝑟𝑒𝑑 𝑝𝑜𝑖𝑛𝑡, 𝑘 𝑜𝑓 𝑤ℎ𝑖𝑐ℎ 𝑖𝑠 4:

P(3|k) = 1/4

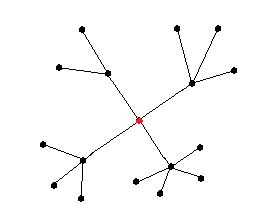
P(4|k) = 2/4

P(5|k) = 1/4

Knn(k) = 3 \* P(3|k) + 4 \* P(4|k) + 5 \* P(5|k)

= 3 \* 1/4 + 4 \* 2/4 + 5 \* 1/4

Knn(k) = 0.75 + 2 + 1.25 = 4



The figures below almost resembles to the one in the paper, to make it more accurate we are in need to simulate 1 million vertices, which will be time consuming. So we ran it for lesser number ranging from 1000 to 5000.

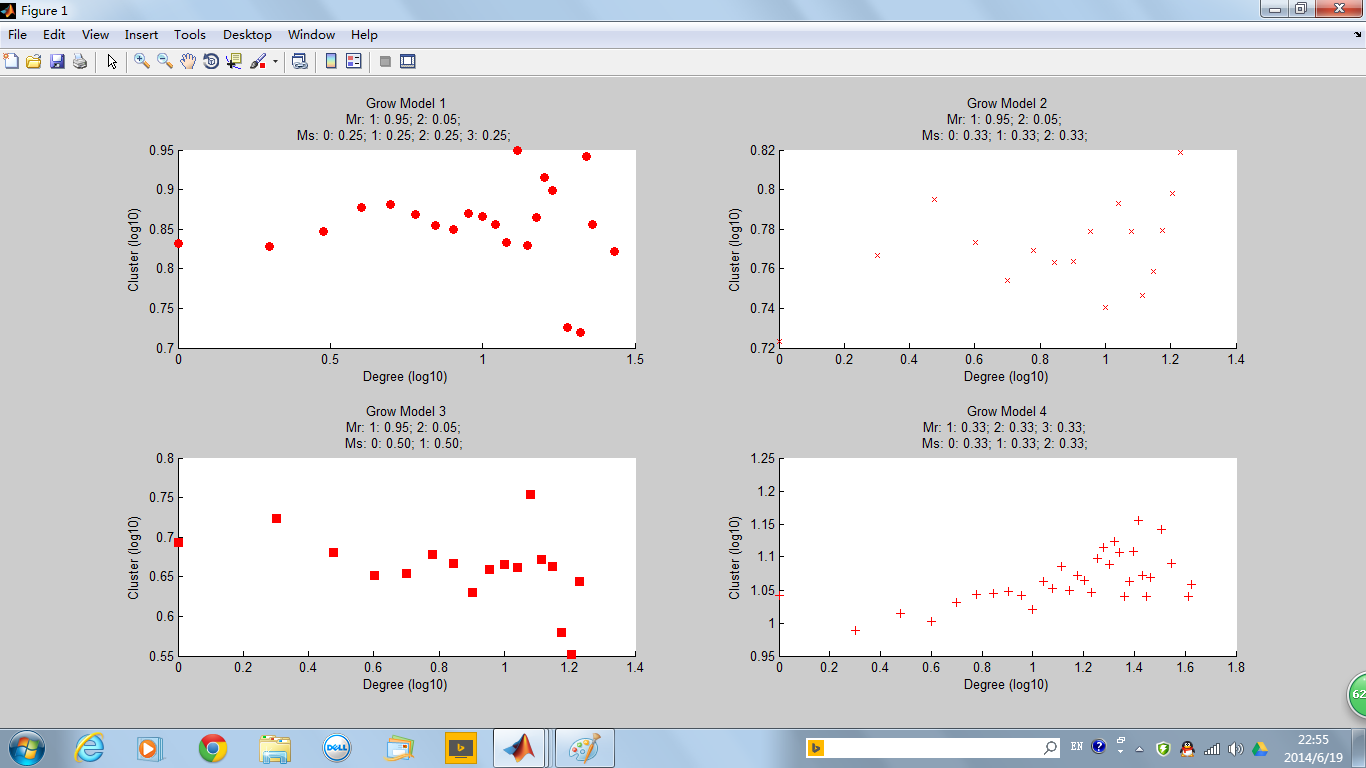


Figure 16 Knn with Vertices No. 1000

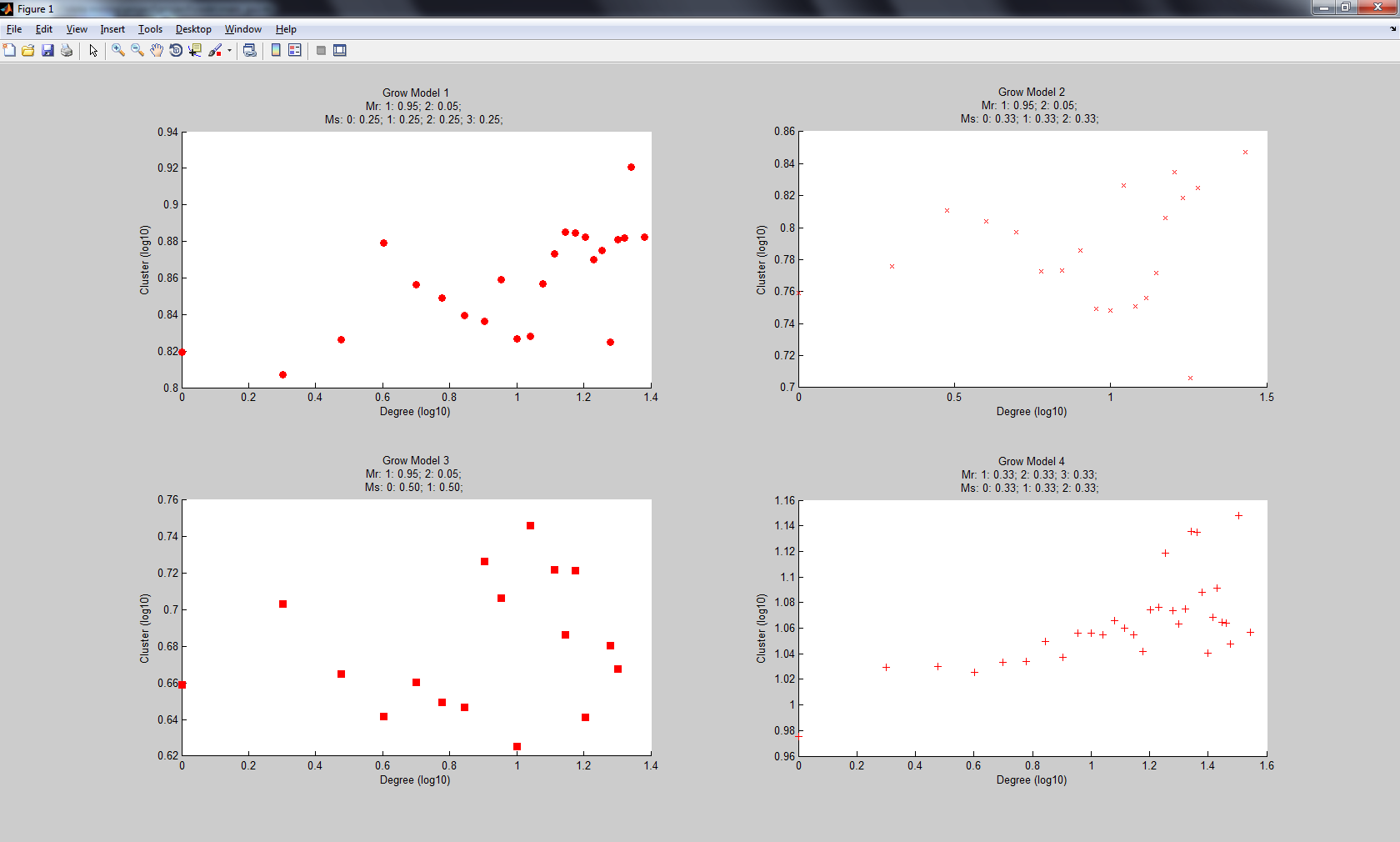


Figure 17 Knn with Vertices No. 1100

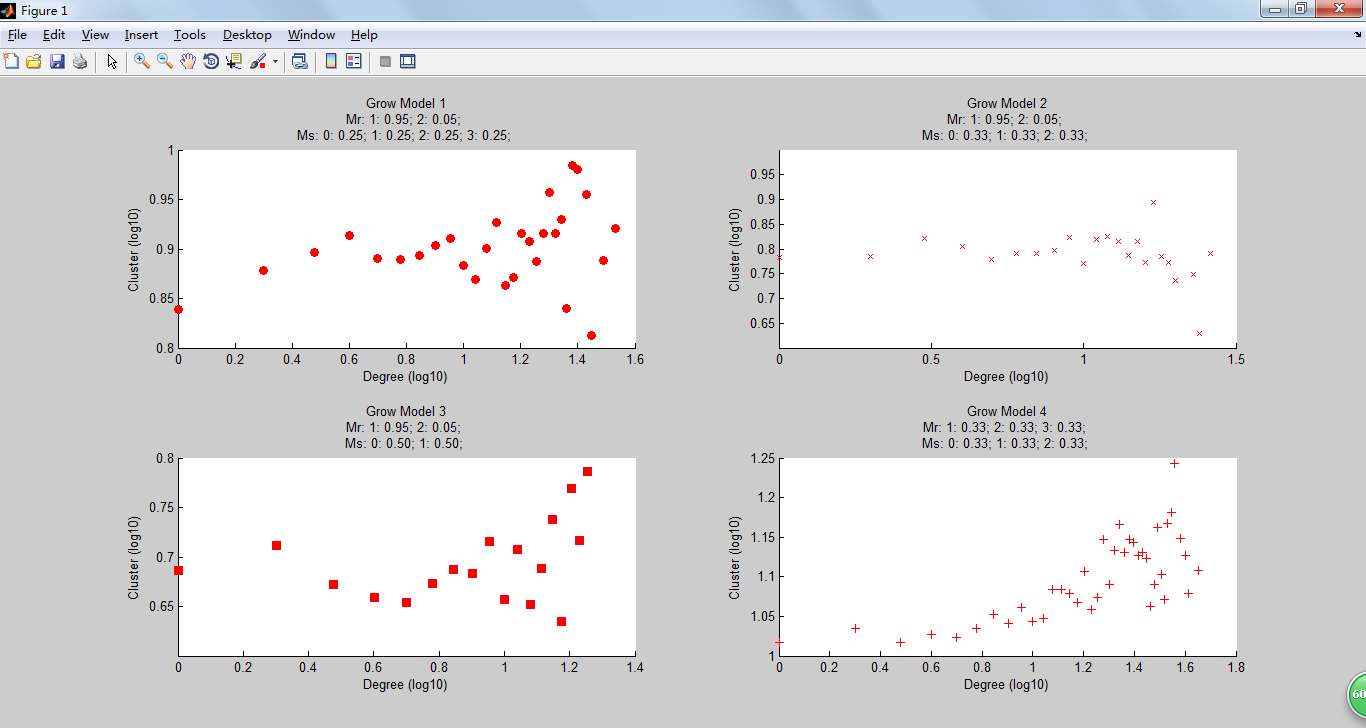


Figure 18 Knn with Vertices No. 2000

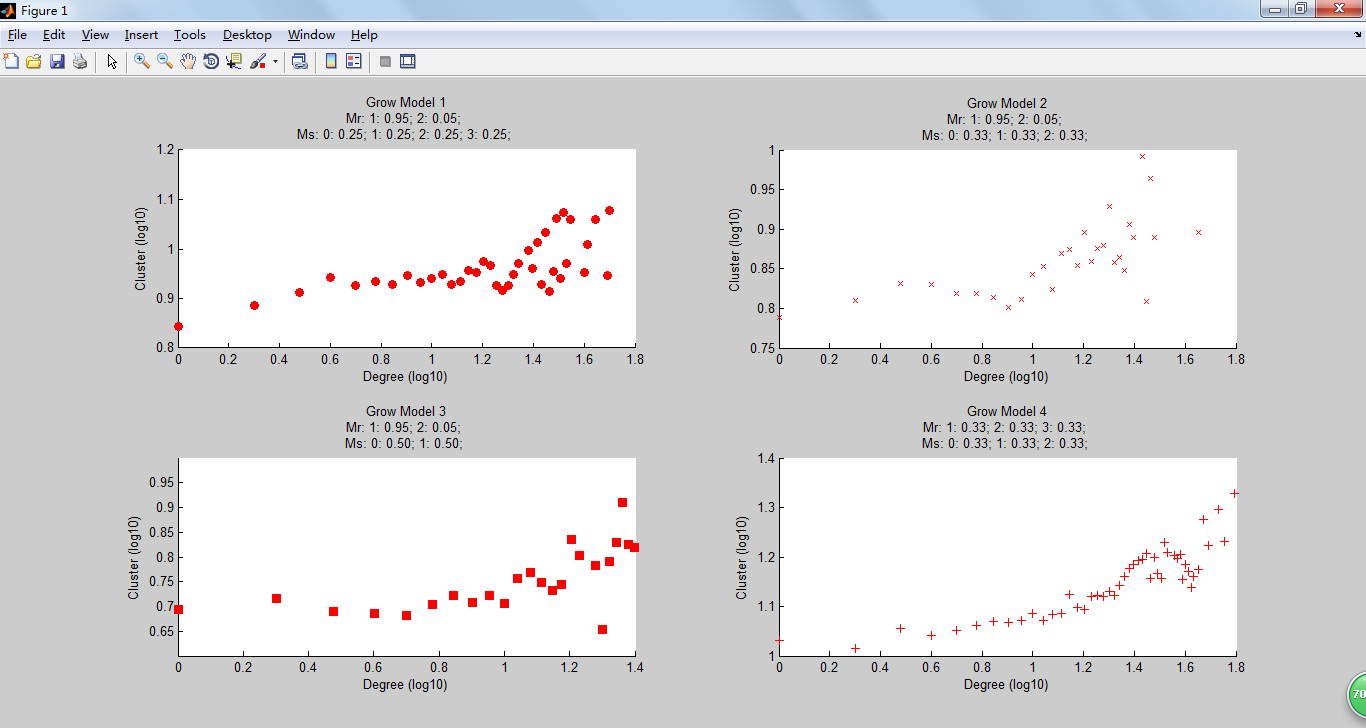


Figure 19 Knn with Vertices No. 5000

All the above figures represent the K nearest neighbors (Knn). The plot is drawn having the degree of the nodes in the x- axis and clusters (triangles) formed in the y axis. There are totally 4 different network models used with different probabilities of choosing contacts. The network in terms of Knn almost seems similar in all the four different models. The more the number of vertices present the clearer is the plot.

**Result**

Thus in any network the benefit we get from the nearest neighbor is greater than the farther. It applies to all fields of science, economics, etc. Here we have found the number of incidental triangle formed due to relationship between the neighbors of a same node. Here we have analyzed the data accuracy by running with different number of vertices ranging from 1000 – 5000 and we figured out that as the number of vertices increase, the spread and variation is less.

**Problem**

In the paper, totally 1 million vertices were run and the plot was drawn, but as we are limited with resources we could run a maximum of 5000 vertices and we have displayed its result above.

1. **Average Shortest Path Lengths**

**Path**

A path in a network is nothing but a walk from with each node being distinct.

The length of a path will be K-1 the links involved

**Shortest path**

Shortest path is path to take that has the shortest the number of vertices to reach the target node

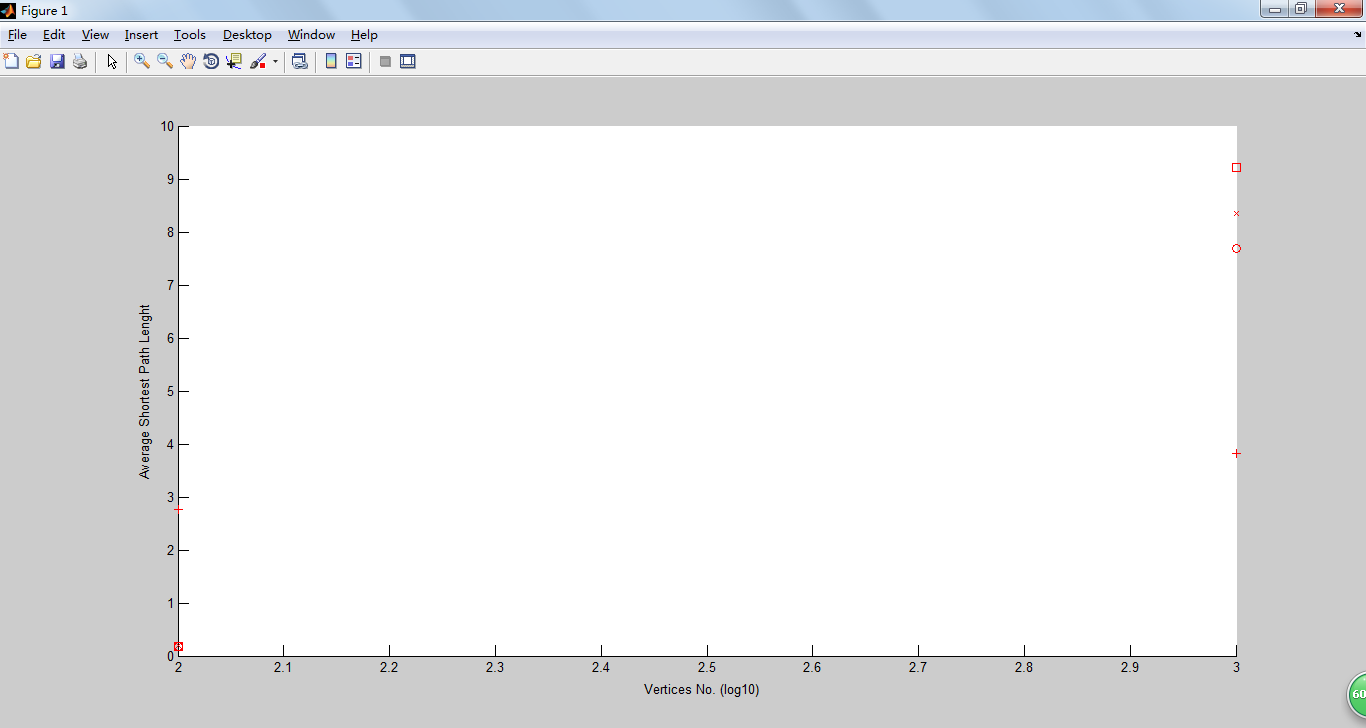
**Finding average shortest path lengths**

Here we are about to calculate the average shortest path lengths, and for calculating that, function ‘sparse’ is used on the vertices connection matrix to get a sparse vertices connection and function ‘graphshortestpath’ is used to calculate the shortest path between 2 points and sum all the values of the shortest path, which will be divided by the number of the combination of the vertices.

**Average shortest path in our network**

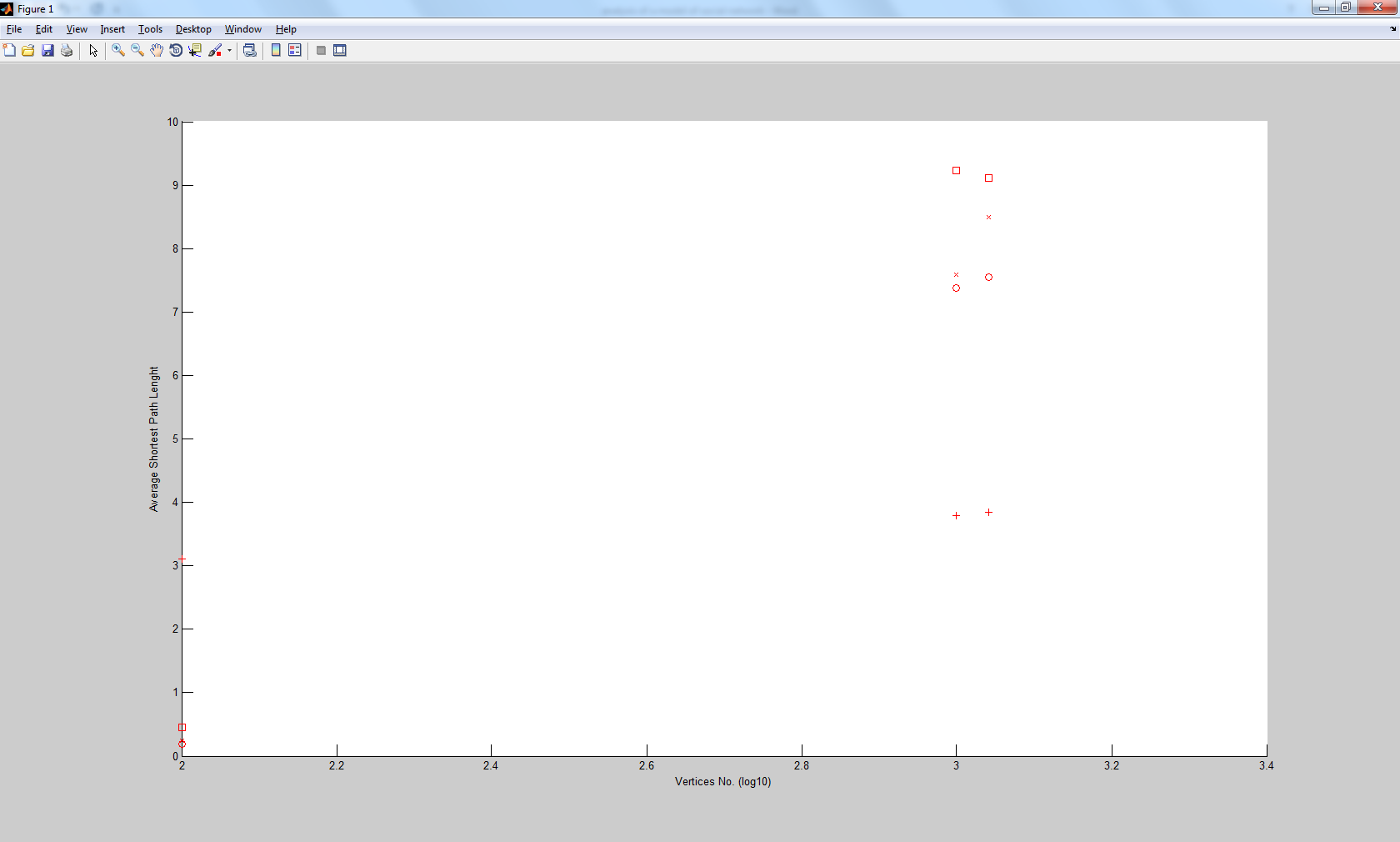
From the figures below, the tendency of the simulation is the same to that on the paper. With the increase of the network, the average shortest path length becomes longer.

In the simulation, as it will take a long time to calculate the shortest path for a large network, we only run this on some smaller network.

**

*Figure 20 Shortest Path with Vertices No. 1000*

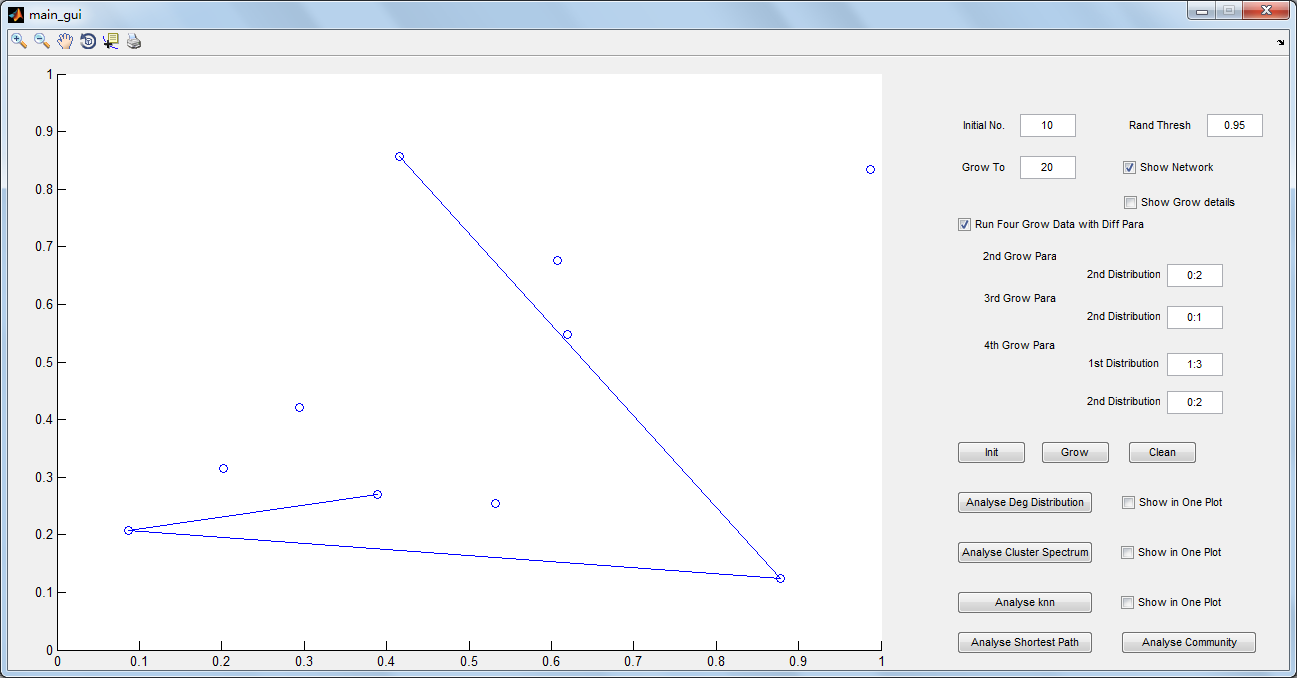
Here the network consists of 1000 vertices, for all the 4 models we run the average shortest path was ranging between 3 and 10.



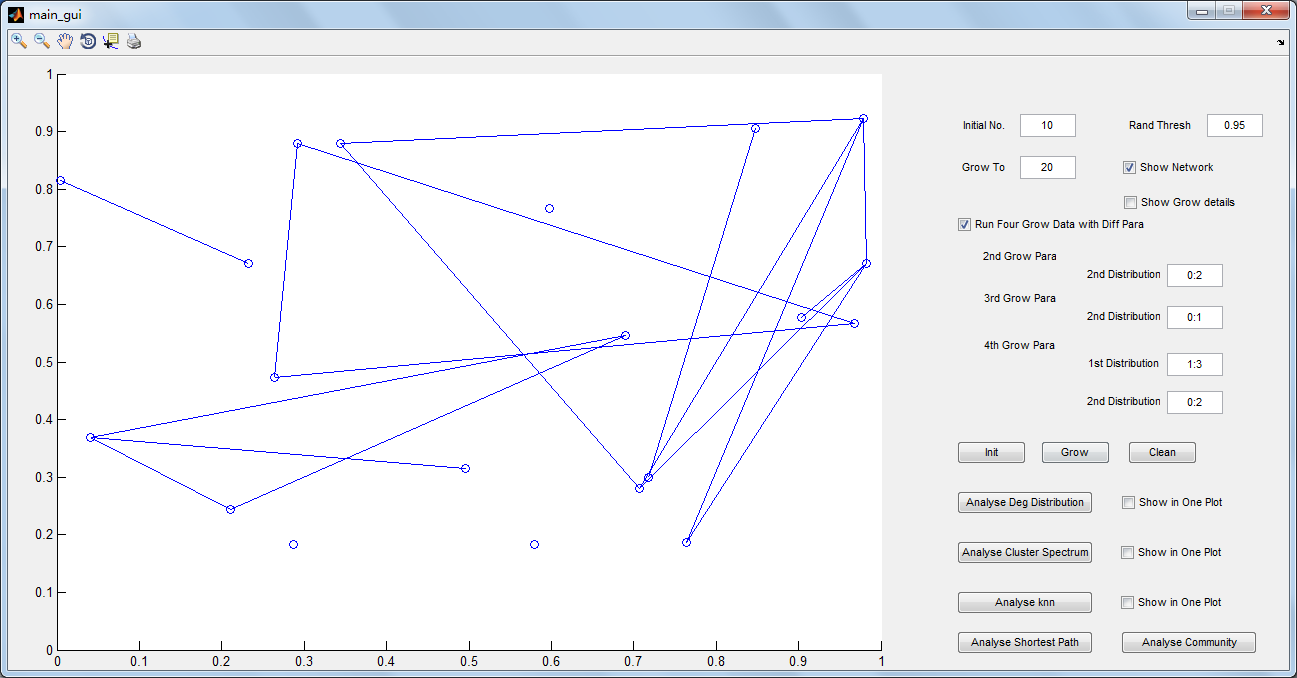
*Figure 21 Shortest Path with Vertices No. 1100*

As we increase the number of vertices by 100 we find that the degree remains the constant with slight variation in the average shortest path length. Based on the common knowledge, the shortest path should increase with the growth of the network, which matches the growth of three of the four networks in the Figure 21, however in one of the four networks the shortest path is decreased, which happens only when the newly added vertices connect different communities.

However if the network is a small network with only a few vertices, the result of average shortest path lengths may be less than 1 as for some of the 2 points there is no path to link them and there is no path connecting the isolated nodes.

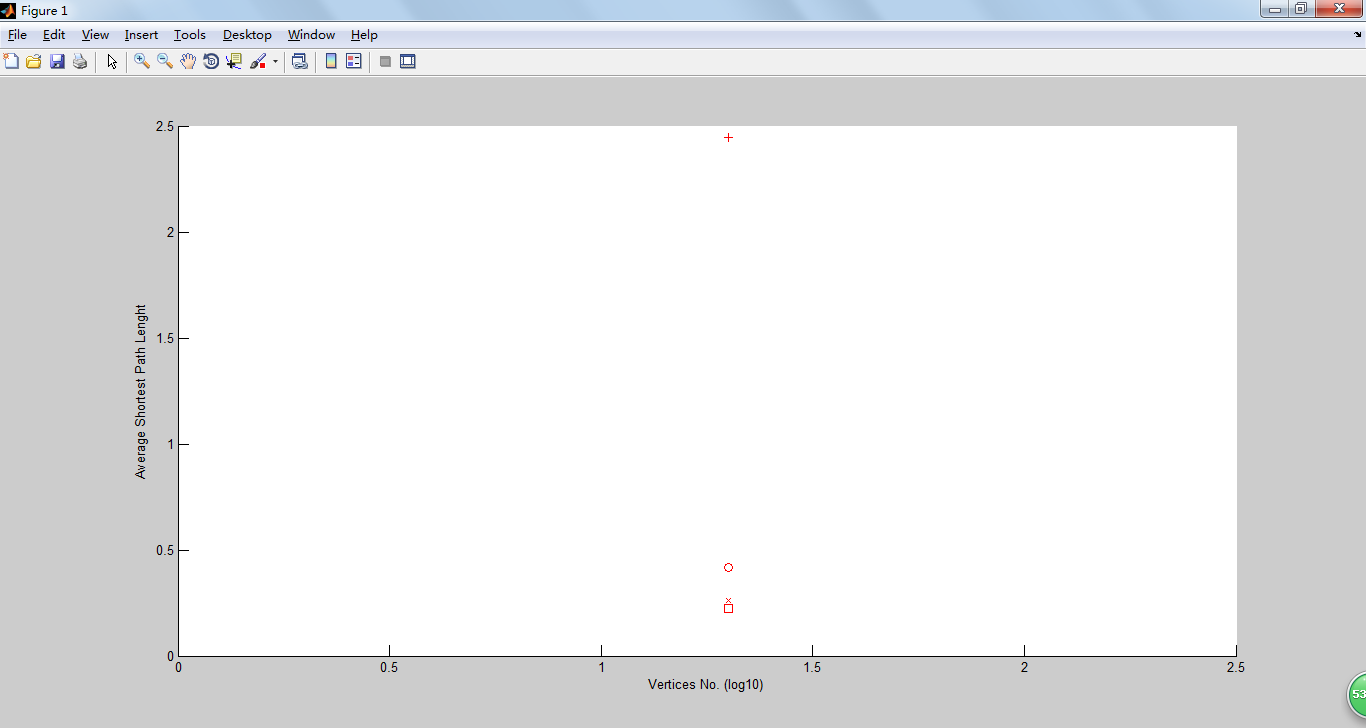


*Figure 22 Random network with Vertices No. 10*



*Figure 23 network grow to Vertices No. 10*

This plot (Figure 24) shows the average shortest path of the above network with 20 nodes. It is found that except one model of network all others have the average shortest path lesser than 1. This means that, when a network is having some isolated nodes or if there exists no path between two nodes, then the average shortest path falls. The overall network has resulted average shortest path less than 0.5, with one network model having 2.5.



*Figure 24 average shortest path of some network is less than 1*

**Result**

We can infer two things from the above discussion

1. The average shortest path is high for network with large number of vertices
2. It can even be lesser than 1 for small networks as it has some isolated nodes and no path is existing between two nodes.
3. **Average Number of k-clique-communities**

**Communities**

The term *network community* is defined as a group of nodes that are more densely connected to each other than to other nodes in the network.

**K- Clique communities**

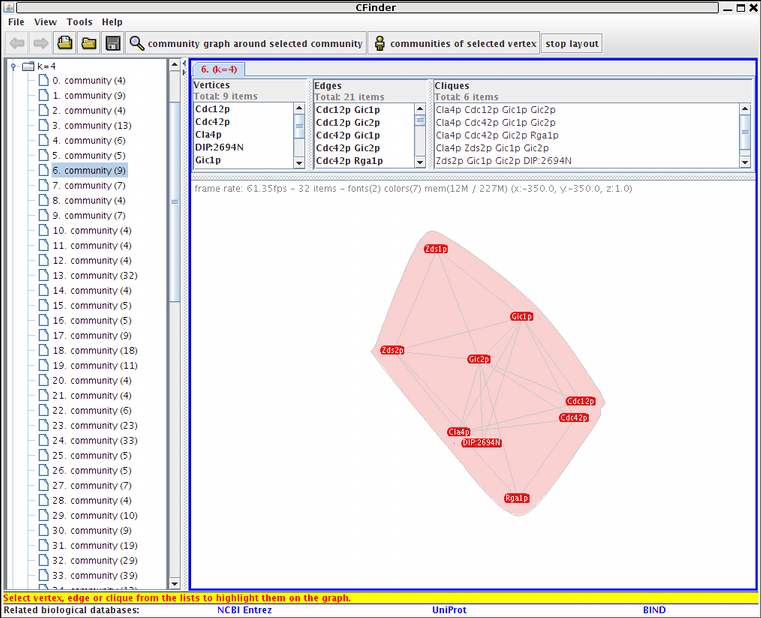
Uncovering the overlapping community structure of complex networks in nature and society

**Average Number of k-clique-communities**

In our simulation, we use Matlab to generate the link between two points in the network and write it to a text file. Then use ‘CFinder’ tool to calculate the community size and number. Finally we use python script to collect the result of CFinder and plot the result on the figure.

With the growth of the network, when coefficient k is getting bigger, the number of community for each k is also getting bigger at the same time.

Below is the CFinder tool, which we made use of to bring out the communities present in our network



*Figure 25 cFinder is used to calculate community size*

Once the communities are derived out, we made use of the Python language to collect the result.

**Python code**

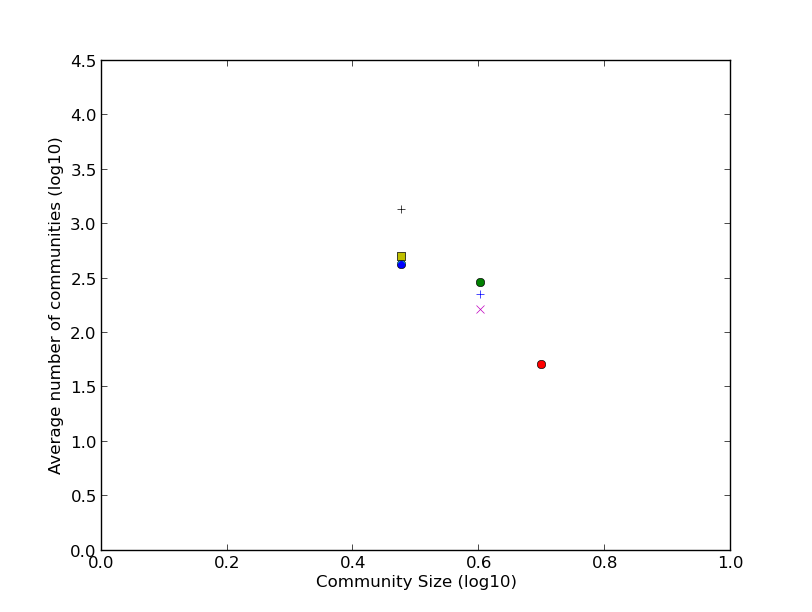








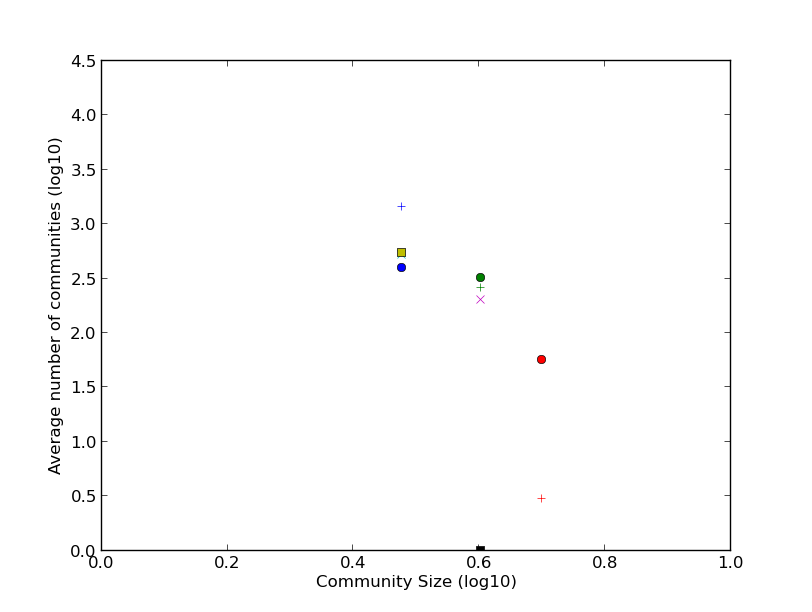
Using the above code we have extracted the result from the CFinder and represented it in the form of plots for analyzing the k-clique communities.



*Figure 26 Number of Communities with Vertices No. 1000*

Figure 26 represents the plot drawn between the community size and average number of communities present with the same community size. The points indicate the 4 network models we used. This full grown network consists of 1000 vertices.

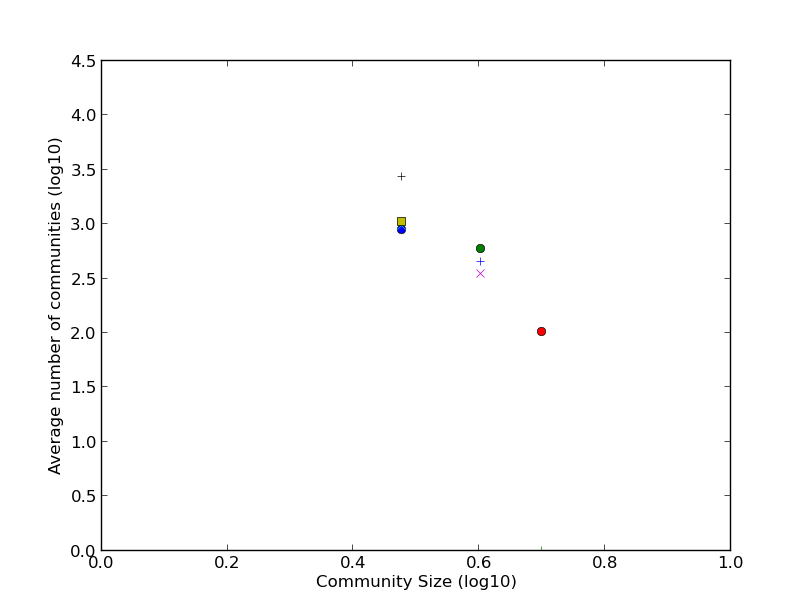
The network with 1100 vertices (Figure 27) doesn’t differ much, but some small change in the average number of communities spotted.

**

*Figure 27 Number of Communities with Vertices No. 1100*

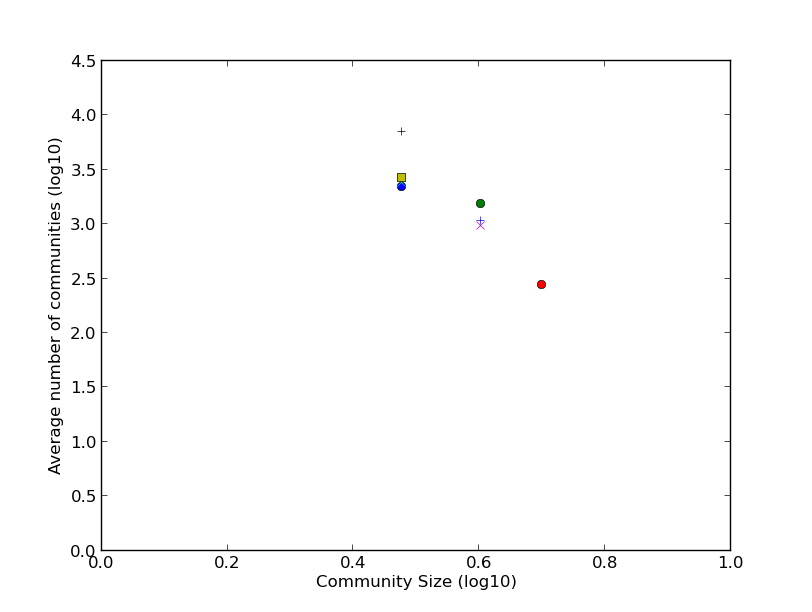
The figure below represents a plot having 2000 vertices. Here we can see some superficial increase in the community size. So it is evident that when we increase the number of vertices then the number of communities will also increase. Considering this we ran some trials of networks of large vertices to test if there is any change the community size.

**Trial 1 – 2000 vertices**

**

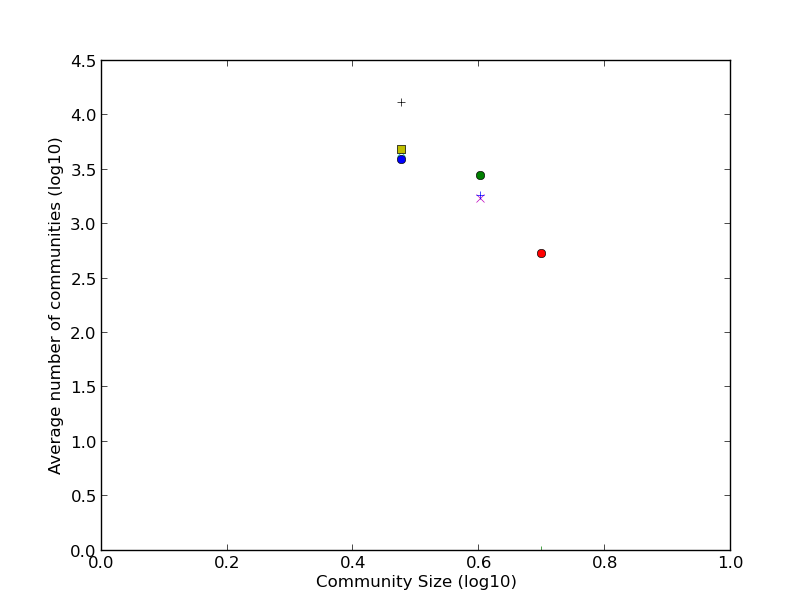
*Figure 28 Number of Communities with Vertices No. 2000*

**Trail 2 – Network with 5000 vertices**

**

*Figure 29 Number of Communities with Vertices No. 5000*

**Trial 3 – Network with 9000 vertices.**

**

*Figure 30 Number of Communities with Vertices No. 9000*

**Result**

By running different trials on network size it is evident that as we increase the numbers of vertices present in a network, the greater the formation of communities with community size being constant to the specified value of k.

1. **Conclusion**

The whole idea of project is to grow a random network up to a specified size and then analyze this network and bring out certain parameters such as the degree, clusters, triangle, average shortest path, k-clique communities found, etc. Compare and contrast between different network model we had and then predict our algorithm accuracy to that of the ideal ones. The concept for our project was derived from (Toivonen, Jukka, Saramaki, Hyvonen, & Kaski, 2006) article. We were given with step by step instruction of how to develop a model and analyze it. We tried maximum to imitate this paper on our network and found the tendency that most of the situations was matched perfectly, with some limitations on the network size because of limited resources.

1. **Contribution**

Weichun Xu: 60%

Harinath T Prabhakaran: 40%