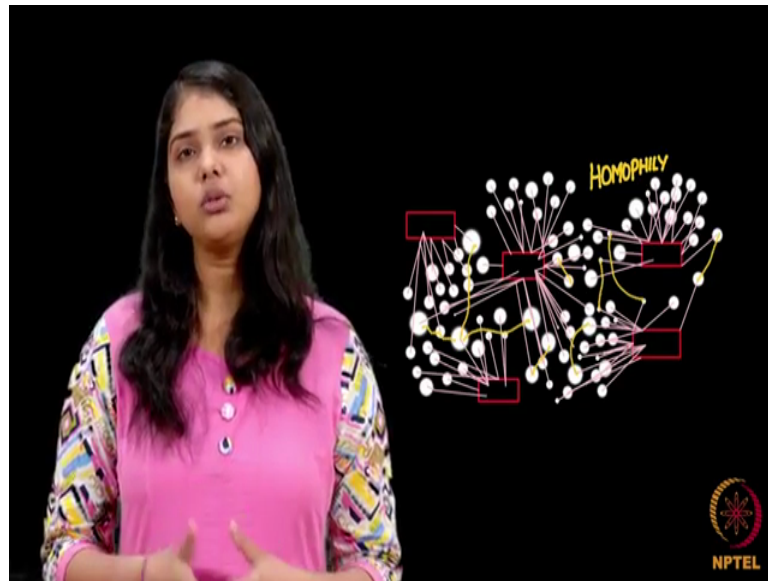


Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Lecture – 49
Strong and Weak Relationships (Continued) and Homophily
Fatman Evolutionary Model - Implementing Homophily

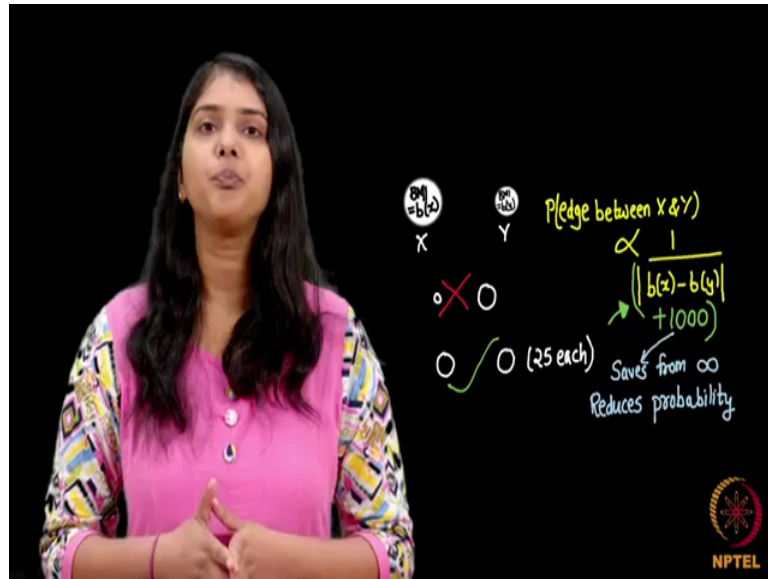
(Refer Slide Time: 00:10)



Till now we have made a network which has 100 people of different different body weights and then we have 5 social foci including a gym and an and eat out place and we have seen that initially everybody is a part of exactly one social foci and with time people will start becoming the parts of more social foci because of membership closer.

Next what we want to implement in this coding what we want to happen in this network is homophily that is the people who are having singular BMI should become friends with each other. So, we are going to use a very simple method to do it. So, we see that the probability of two people becoming friends with each other is inversely proportional to the difference between their BMIs.

(Refer Slide Time: 00:53)



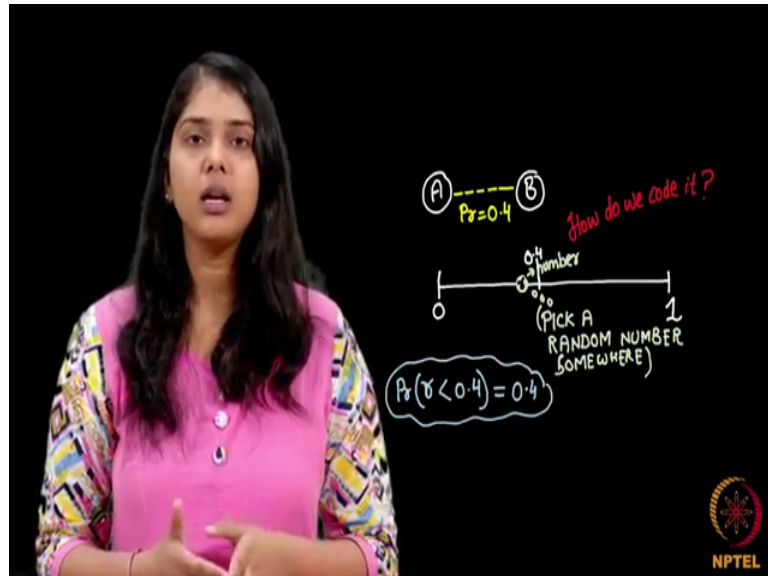
So, more is the difference between your BMIs less probable two people are to become friends with each other. So, if let say a person is underweight having a BMI of 15 and a person is overweight having a BMI of 40. So, it is very unlikely that they will become friends. And if two people they have almost the same BMI. So, let us say 25 each then they are likely to become friends. So, we can directly use a formula that probability of two people becoming friends with each other is inversely proportional to the difference between their BMIs.

So, probability of an edge between u and v is proportional to one divided by the absolute difference between them. So, we take absolute to avoid a negative there. So, we have absolute there and we add an extra factor of plus 1000 here. Why do we add an extra factor of plus thousand here is if you see by this formula if we do not have a 1000 here and let us say the BMIs of two people are equal, let us say 25 each then the probability of them becoming friends becomes 1 by 0 which is infinity, so we have a 1000 here which not only help us deal with this infinity issue what it does is it reduces all the probabilities.

So, although the probability of one person becoming connected to another is remains inversely proportional to the difference between their BMIs, but we want this probabilities to be less because since we want to see the network to be evolving we do not want a lot of edges to enter at the same time. So, adjust should come slowly and

slowly. So, this plus 1000 helps us reduced all the probability value, it scales all the probability values down. So, we use this formula for implementing homophily.

(Refer Slide Time: 03:16)



And one interesting concept here which I would like to tell you, you see here what we are going to do is the here are two people. Let us say A and B and we are saying that assume the probability with which A and B become friends is 0.4. So, there is a 40 percent chance that A and B become friends and 60 percent chance that they do not become friends. How do we implement this in our coding? So, this random package helps us do this. So, what I do is assume, I pick a random number random real number from a number line from 0 to 1 as shown here.

I want some event to happen with a probability of 0.4. So, now, I have a random number here between 0 and 1. So, I pick a random number let us say it is here, what is the probability that this random number is less than 0.4. So, since this random number can be anywhere on this number line if you see the probability that this random number is less than 0.4 is actually 0.4 itself. So, 0 to 0.4 is a smaller slot, 0.4 to 1 is a bigger slot. So, there is a lesser probability of this number falling below 0.4 and there is a higher probability it lies in the range 0.4 to 1. So, we will be using this method to implement what we are saying. So, assume there are two nodes u and v and we want them to become friends with probability 40 percent 0.4. What will do is generate this random

number are from 0 to 1 and if its value is less than 0.4 we make an edge between these two people otherwise we do not make an edge.

(Refer Slide Time: 04:55)



So, now before implementing homophily I want to do one extra thing. So, we have added colour to a nodes. So, we have added blue colour for the person node and red colour for the foci nodes.

(Refer Slide Time: 05:05)

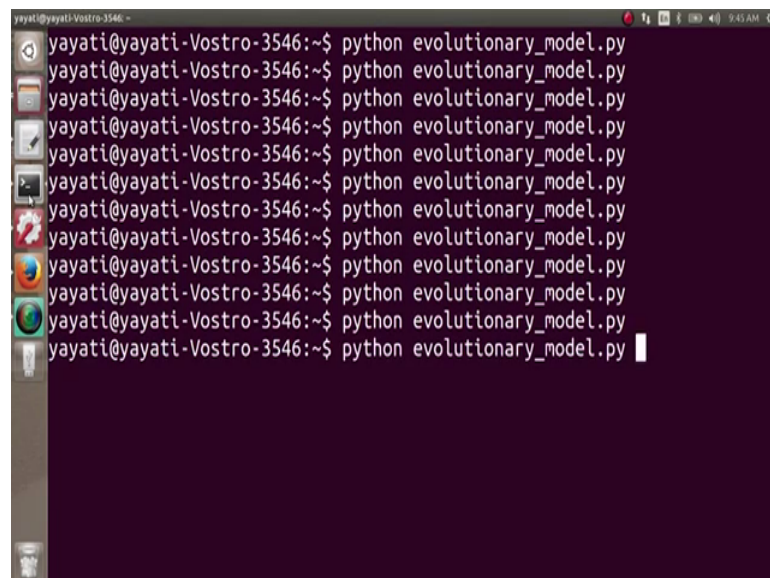
```
revolutionary_model.py (-) - gedit
revolutionary_model.py x
i=i+1
def get_colors(G):
    c=[]
    for each in G.nodes():
        if G.node[each]['type']=='person':
            if G.node[each]['name']==15:
                c.append('green')
            elif G.node[each]['name']==40:
                c.append('yellow')
            else:
                c.append('blue')
        else:
            c.append('red')
    return c
def get_foci_nodes():
    f=[]
```

What I want to see I want to see underweight and overweight people separately, I want the underweight people to appear in let say green colour and overweight people to appear

in yellow colour, so that when we look at this graph across different different timestamps we can see that how the number of people are changing and how the clustering between them is happening.

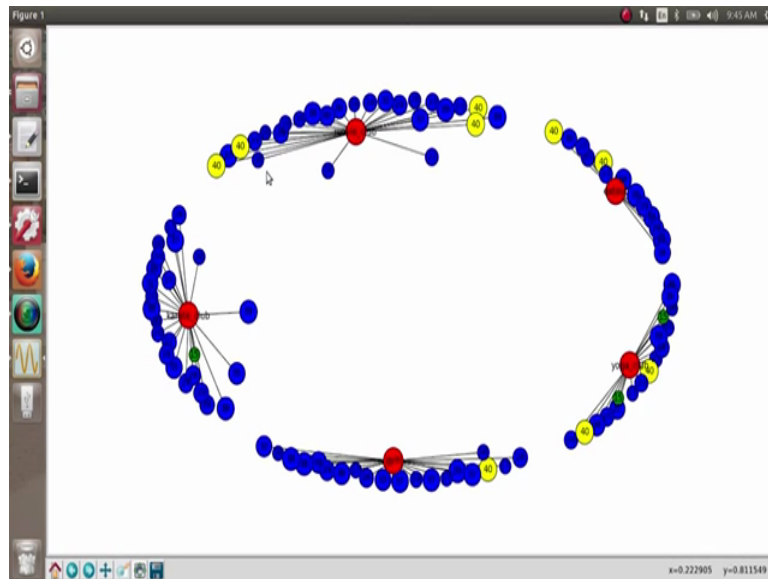
So, what we do here is if G node each type equals to equals to person and then further we have if G dot node each name equals to equals to 15. So, what we do is in this case c dot append what do we append here is a green and then we have else if which is written as elif in python. So, elif G dot node each name equals to equals to 40, if it is 40 c dot append and what do we append here is a yellow colour. And still if some node is left; obviously, many nodes are left having the BMI in between these two then we do c dot append blue. Let us save it and next let us look at the output.

(Refer Slide Time: 06:55)

A terminal window with a dark purple background. The window title is 'yayati@yayati-Vostro-3546'. The terminal shows a series of 12 identical lines, each consisting of a prompt 'yayati@yayati-Vostro-3546:~\$' followed by the command 'python evolutionary_model.py'. The cursor is at the end of the last line. The window includes a standard Linux desktop taskbar on the left with icons for home, applications, and system status, and a top status bar showing the time as 9:45 AM.

So, you see here the graph looks I think all the more beautiful now. So, we have these node. So, blue nodes are mostly the normal nodes red nodes are the foci nodes. So, here you see all the nodes having BMI of 40 comes in the yellow colour and all the nodes having this BMI of 15 comes in the green colour.

(Refer Slide Time: 06:59)



Next if we go head and implement homophily in this network, let us see how do we implement homophily. So, after adding foci edges what we want to do if we want to implement homophily.

(Refer Slide Time: 07:30)

```

evolutionary_model.py (-) - gedit
def add_foci_edges():
    foci_nodes=get_foci_nodes()
    person_nodes=get_persons_nodes()
    for each in person_nodes:
        r=random.choice(foci_nodes)
        G.add_edge(each,r)

def homophily(G):
    pnodes=get_persons_noded()

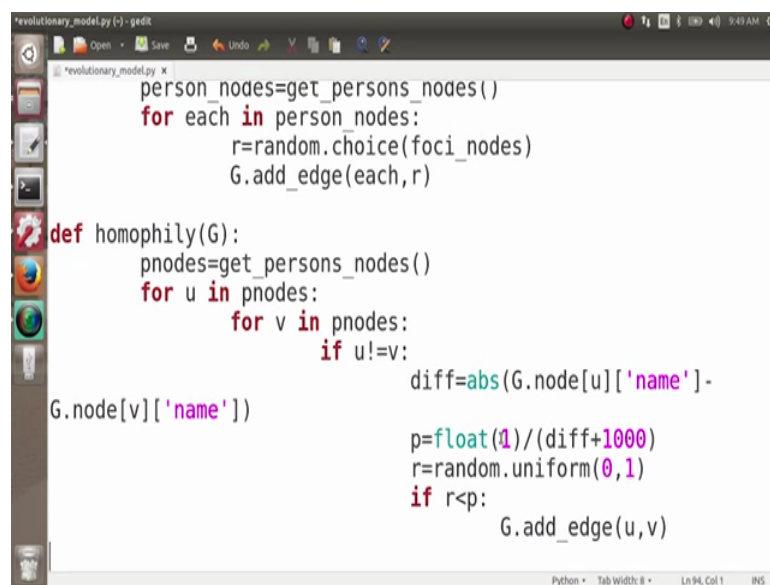
G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)

```

Now what does homophily do? So, what will be implementing here will be one iteration of homophily what does homophily says nodes which are alike similar to each other they tend to become friends with each other and we have already looked at the formula for this.

So, we define homophily here, define homophily G and what does it do first of all it gets all the person nodes. So, person node equals to get persons nodes because you see homophilies only going to happen between the people not the social foci. So, this function is explicitly defined for the person nodes. So, we get all the person nodes and for simplicity, let us simply name this array as pnodes, pnodes refer to person nodes pnodes equals to get persons nodes. You can also notice here that I am nowhere pass the parameter G here because G here is being considered as a global parameter - pnodes equals to get persons nodes.

(Refer Slide Time: 08:57)



```

revolutionary_model.py (-) - gedit
revolutionary_model.py x
person_nodes=get_persons_nodes()
for each in person_nodes:
    r=random.choice(foci_nodes)
    G.add_edge(each,r)

def homophily(G):
    pnodes=get_persons_nodes()
    for u in pnodes:
        for v in pnodes:
            if u!=v:
                diff=abs(G.node[u]['name']-
G.node[v]['name'])
                p=float(1)/(diff+1000)
                r=random.uniform(0,1)
                if r<p:
                    G.add_edge(u,v)

```

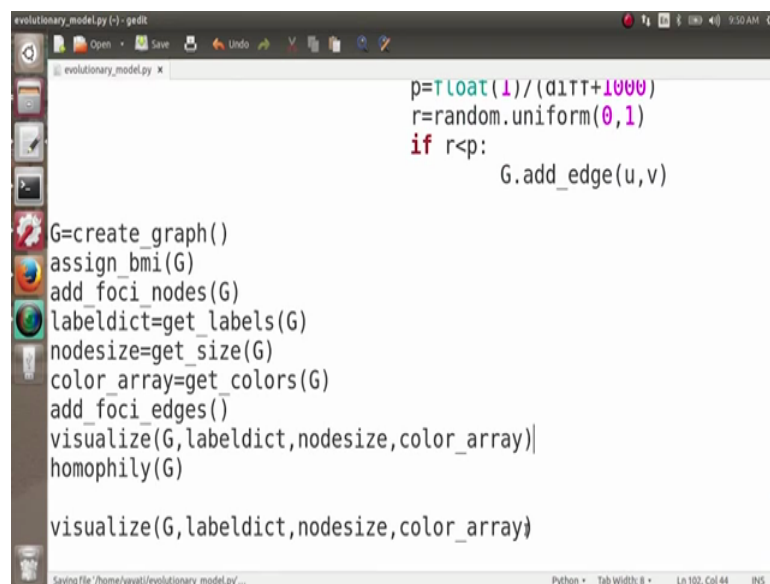
So, I have all the person nodes, now I want to see whether there BMI similar to each other and if it is similar to each other I will add edges between them what I do for u in pnodes. So, u is my first node for u in pnodes and then I have for v in pnodes, so it picks each of the person nodes every possible combination of two person nodes one by one and obviously, my u should not be equal to v because the person is not going to make tie with himself. So, if u is not equals to v what do we do is we calculate this difference in their BMI.

So, what is the difference in their BMI? It is the absolute value of, absolute value of G dot node each very sorry G dot node u name which carries the BMI of this person minus G dot node v name. So, I have this difference here, after having this difference what is the probability that these two nodes will connect to each other as we have seen is

inversely proportional to this difference - greater the difference lesser the probability, lesser the difference higher the probability. So, probability of these people connecting is one divided by this difference plus 1000. So, we have this 1000 here to keep this probabilities small.

And next as discussed before we have a random number r , random real number r which takes its value from 0 to 1 and if this r is less than p what do we do; if G do not add edge between u and v . I think it is clear. So, what we do is we calculate the difference between two nodes, probability of these nodes connecting is inversely proportional to this difference.

(Refer Slide Time: 11:28)



```
evolutionary_model.py (-) - gedit
evolutionary_model.py x
p=float(1)/(d1+1000)
r=random.uniform(0,1)
if r<p:
    G.add_edge(u,v)

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
add_foci_edges()
visualize(G,labeldict,nodesize,color_array)
homophily(G)

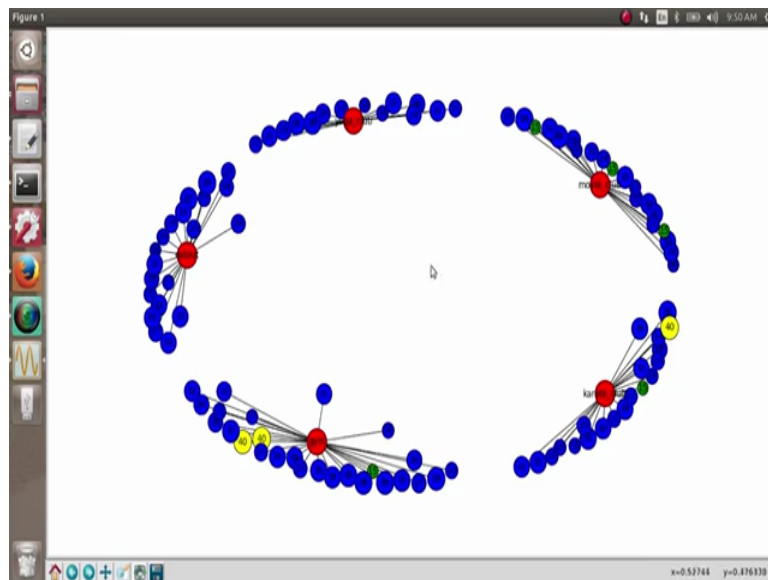
visualize(G,labeldict,nodesize,color_array)
Saving file /home/yayati/evolutionary_model.py ... Python • Tab Width: 8 • Ln 102, Col 44 100%
```


(Refer Slide Time: 11:33)

```
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

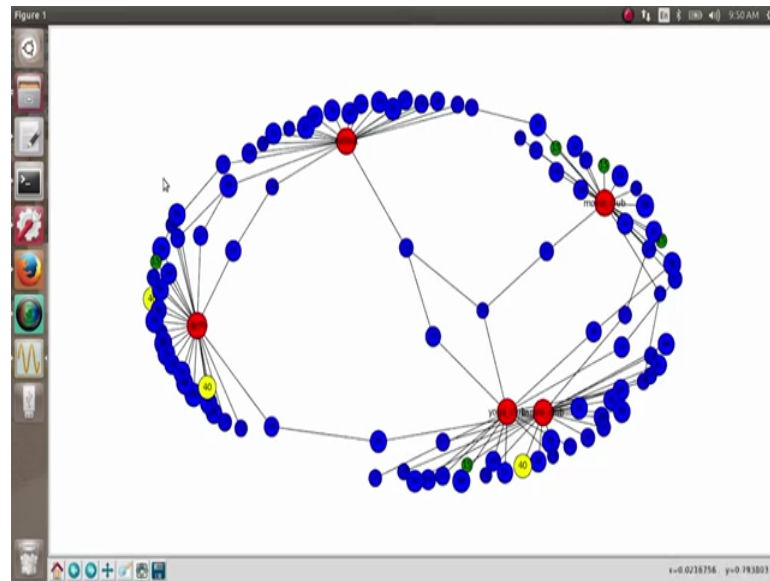
And with this probability these two nodes connect in every iteration. Let see what we want to do is let us visualize the graph once before homophily and once after homophily.

(Refer Slide Time: 11:39)



So, this is our graph before homophily, this is our graph before homophily and this is our graph after homophily.

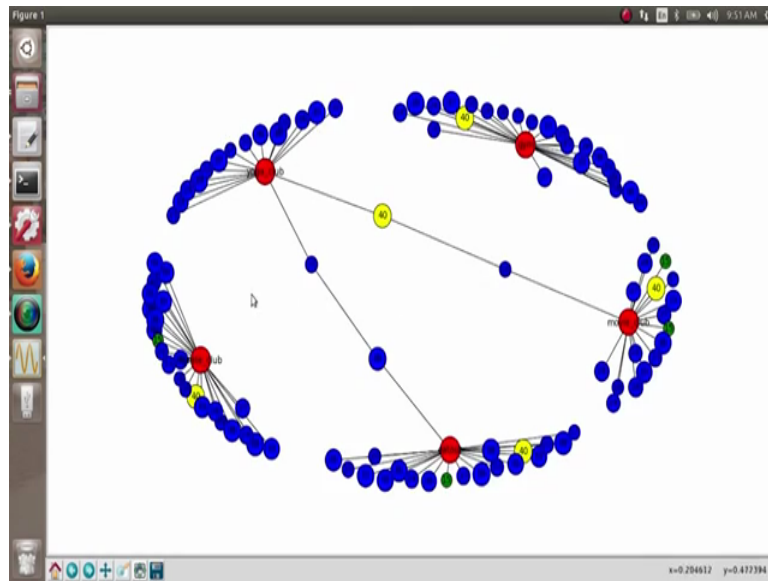
(Refer Slide Time: 11:45)



So, you can see here now we have many many relationships across different clusters. So, these nodes previously they existed in different clusters, but you can see now our graph has become connected and more and more people has connected with each other. So, here we have an edge between 35 25, 25 39, 39 27 and so on and these edges are in accordance with their body weights BMIs.

So, we can make this probability actually a little lesser if you want less number of edges to be added at every step that is totally up to us. So, we can increase this probability of connection, we can decrease this probability of connection, but the probability should be inversely proportional. So, if you want to decrease the number of edges being added at every step what we can do is we can make this value all the more higher and this probability gets less here. So, you see here that this was our initial network.

(Refer Slide Time: 12:51)



And this is our new network. So, you see here less number of edges have been added, less number of edges have been added it here.