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

Rich Get Richer Phenomenon
Lecture - 125
Forced Versus Random Removal of Nodes (Attack Survivability)

In this video we are going to implement Forced Versus Random Failure of Nodes which is also called Attack Survivability. We are going to take a network and we will be removing nodes from this network and we will observe the effect of removal of nodes from the network. We will be using two strategies to remove the nodes. In the first strategy we will remove the nodes randomly, and in the second strategy we will remove the nodes which are highly connected.

We will observe that when we remove the nodes which are highly connected, it requires very less iterations for the graph to become disconnected. This sort of information can be used by some agent for example, if he has the information of who is more connected and who is not. He, if in case he wants to damage the network for example, by spreading some sort of virus, he will not be attacking a random nodes rather, he will attack the nodes that are more connected, so that the network becomes infected to a greater extent.
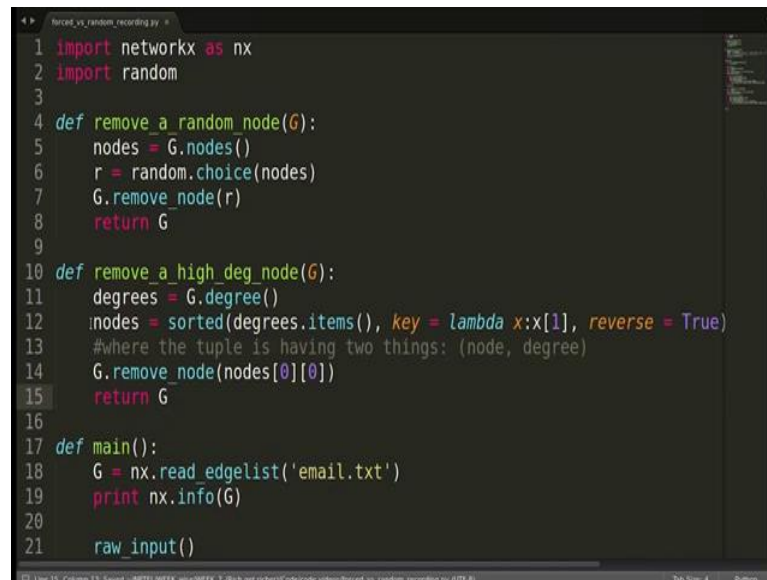
So, there is one thing we will observe and implement, and we will also see one more thing in the context of the kinds of graphs. For example, the first thing that we checked on real world networks, real world networks sort of follow preferential attachment where there are there is existence of hubs in the network which are responsible for connection of most part of the network. So, when you remove then the graph becomes, becomes disconnected.

We are going to observe the same thing that is the removal of nodes on random networks as well. And we will observe that it does not make much difference on a random network when we remove the nodes randomly as compared to when we remove the nodes selectively; that is we remove the nodes which are having high degree.

That is because there is there are no hubs in random networks which are responsible for the connection of most part of the network; so the number of iterations that it will require

in the first case that is random removal and the second case that is selective removal. In the case of random networks will not make much difference. We are going to implement that and we will observe the number iterations that are required. Let us get started with implementation.

(Refer Slide Time: 02:23)


```python
import networkx as nx
import random

def remove_a_random_node(G):
    nodes = G.nodes()
    r = random.choice(nodes)
    G.remove_node(r)
    return G

def remove_a_high_deg_node(G):
    degrees = G.degree()
    nodes = sorted(degrees.items(), key = lambda x:x[1], reverse = True)
    #where the tuple is having two things: (node, degree)
    G.remove_node(nodes[0][0])
    return G

def main():
    G = nx.read_edgelist('email.txt')
    print nx.info(G)

    raw_input()
```

Let me import the packages first, we might also need random package, let us define our main. I already have downloaded a real-world network that is email network and that is an edge list format, we will be making use of that. This network is basically undirected. So, there is link from a node a to node b, if they exchanged any email with each other. So, that sort of network is what we will be using.

Let me read that network n x.read_edgelist email.txt. Ok I have kept this in the same folder where I have get this file and then let me print some details about this network. So, I will write n x.info, so that I get the basic statistics about the graph. Let me call this function let us now check whether the graph has been read or not it has been converted into graph object or not.

(Refer Slide Time: 03:46)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Cod
e/code videos$ python forced_vs_random_recording.py
Name:
Type: Graph
Number of nodes: 1133
Number of edges: 5451
Average degree:   9.6222
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Cod
e/code videos$
```

So, this is the basic statistics about the graph. It has more than 1000 nodes and more than 5000 edges and the average degree is there. So, let us get back here. So, we are going to remove the nodes from this network in 2 ways: let me print the first page random removal ok. We are going to randomly remove the nodes. Let me create a copy of the given graph because, we are going to remove nodes in 2 ways.

So, we will work on both the copies. So, this is the first copy G1 from which we will remove the nodes randomly. First let us check whether the graph is connected or not, is G1 connected ok, I will write nx.is_connected(G1) ok. We have to keep a track of how many nodes have to be removed to make the graph disconnected when we randomly remove.

So, let us create variable nodes removed when we randomly removed initialise to 0. Now while the network becomes disconnected, we have to keep removing the nodes. So, we will write nx.is_connected(G1), we have to keep removing. So, we should better create a function for removing a node I create that function in a while. I will write G1 is equal to remove a random node.

So, this function I will just create, I am going to pass the graph here ok. And as you keep removing we will increment the counter for nodes remove random plus is equal to one and in every iteration we will keep checking whether graph is connected or not we will actually print that or let me just copy paste this sorry ok.

So, we at every iteration we are going to check whether graph is disconnected or not and in the end when we are done we will come out this loop and the graph becomes disconnected and at that point of time will print the number of nodes that needed to be removed to make the graph disconnected. Randomly chosen we will print nodes remove random ok. We will lastly used to press an enter here, as well as here ok. We are going to remove the in the second strategy, we are going to remove the nodes selectively based on the degree.

(Refer Slide Time: 07:06)


```
29          nodes_removed_random += 1
30          print 'Is G1 connected?', nx.is_connected(G1)
31      print 'nodes removed when randomly chosen', nodes_removed_random
32
33      raw_input()
34
35      print '###Selective removal###'
36      G2 = G.copy()
37      print 'Is G2 connected?', nx.is_connected(G2)
38      nodes_removed_selective = 0
39
40      while(nx.is_connected(G2)):
41          G2 = remove_a_high_deg_node(G2)
42          nodes_removed_selective += 1
43          print 'Is G2 connected?', nx.is_connected(G2)
44      print 'nodes removed when selectively chosen', nodes_removed_selecti
45
46
47
48  main()
49
```

So, let me just copy paste this entire thing and I will just made changes into that. So, in the second strategy, we are removing selectively, selective removal we will create another copy G2 of the graph. We will see whether G2 is connected or not. So, I am just making changes there. Nodes removed when selective removal was there. Let me write selectively, selective and again G2 and here also G2 and nodes remove the selective is G2 connected.

Actually, should I have written G only here, anyway nodes removed when random when selectively chosen. Nodes removed selectively right; I think there is a mistake here. Ok the only thing left is now to create this function that is 2, these 2 functions we have very minute, we have to remove the node selectively ok. So, remove node remove high degree node may be ok. So, we are yet to create these 2 functions, the first function is remove a random node and the second function is remove a high degree node.
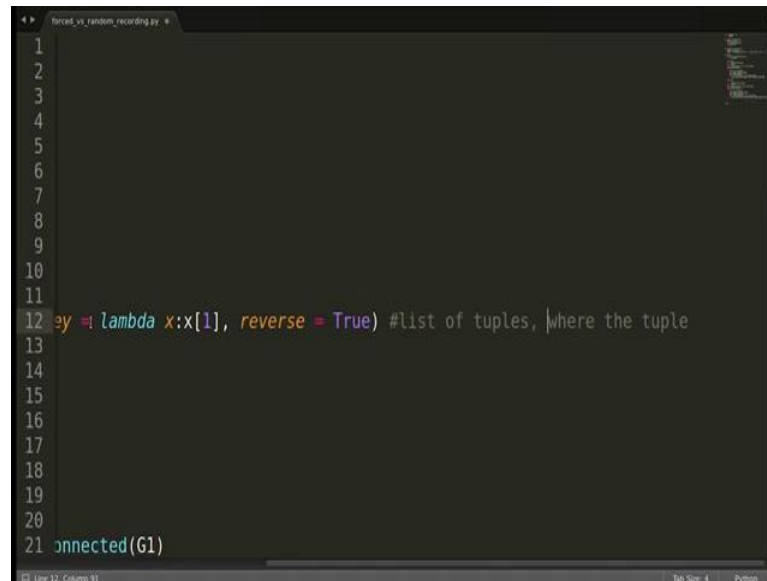
Let us create these functions ok. This function we have to create. So, out of all the nodes we have to choose 1 node randomly and we have to remove that. Let me get a list of all the nodes. nodes is equal to G.nodes, I have to randomly remove a node out of this while use random or choice out of nodes and we have to remove it. So, I will write G. remove node, which node to be removed or has to be removed and at this point we have remove the nodes from the graph and we will return it ok. Let us make a generalized just keep G here.

So, we pass the G and we return the G. So, whichever graph was input 1 more was removed, and it was returned. So, that function is being used here, everything seems fine, this syntax and return type, everything is fine. Next thing is to create this function remove a high degree node. Let us get this function now ok. What we have to do here? We have to see the degrees of the nodes and we have to sort them accordingly and the node which has highest degree will be removed.

How do we do this? Firstly, let us get dictionary having the degrees of all the nodes. So, G.degree ok, now, now out of this dictionary we have to get a sorted list of the key value pairs right. So, what we can do is a these are basically not nodes these are edges because, when we sort a dictionary we get list of tuples where every tuple is every tuple is actually key and value where key is node and the value is degree, so it make sense to call it nodes.

So, what we can do is we can use sorted function and we will pass degree degrees degree might be key or because, degree is the name of the function, so let us call it degrees. deg.items_key = lambda the same old method that we used to get a sorted dictionary. And then X, we want to sort based on the value and we want to sort in descending order reverse = true, this should work I am sorry.

(Refer Slide Time: 11:32)

So, what is nodes? Nodes basically is list of tuples where the tuple (Refer Time: 11:41) percent is here where the tuple is having 2 things node and the yes node and the degree right and it is sorted. So, the first node will be having the highest degree and that is what we should remove.

So, I will write G.remove_node, which node we have to remove out of the nodes list we have to take the first entry that be a tuple and out of that tuple we have to take the first value that is going to be the node that has to be removed right. So now, let us return G ok. This should work it is it no mistake. So, we are calling that function here G2 is equal to remove a high degree node and we are incrementing the counter and we are connecting, that is all. It should work let us check let us check.

(Refer Slide Time: 12:43)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Cod
e/code videos$ python forced_vs_random_recording.py
Name:
Type: Graph
Number of nodes: 1133
Number of edges: 5451
Average degree:   9.6222

###random removal###
Is G1 connected? True
Is G1 connected? False
nodes removed when randomly chosen 1

###Selective removal###
Is G2 connected? True
Is G2 connected? False
nodes removed when selectively chosen 1
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Cod
e/code videos$ python forced_vs_random_recording.py █
```

So, this is a basic statistics as we saw earlier too, I am pressing enter ok. So, in this case we are randomly removing, and we have to remove only 1 node to make the graph disconnected. Now I am pressing enter and we are now selectively removing and again we need here only one node to make a graph disconnected.

So, by chance in this case as you see whatever node we removed randomly that was by chance a high degree node that is that is how the graph becomes disconnected. Let me run this function again and let me see again it is taking one it is surprising.

(Refer Slide Time: 13:26)

```
e/code videos$ python forced_vs_random_recording.py
Name:
Type: Graph
Number of nodes: 1133
Number of edges: 5451
Average degree:   9.6222

###random removal###
Is G1 connected? True
Is G1 connected? False
nodes removed when randomly chosen 1

###Selective removal###
Is G2 connected? True
Is G2 connected? False
nodes removed when selectively chosen 1
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Cod
e/code videos$
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Cod
e/code videos$ python forced_vs_random_recording.py
█
```

And again let me run this again.

(Refer Slide Time: 13:27)

```
###random removal###
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? False
nodes removed when randomly chosen 16
```

So, in this case you see randomly removal took 16 iterations and the selective removal took only 1.

(Refer Slide Time: 13:37)

```
Number of nodes: 1133
Number of edges: 5451
Average degree:   9.6222

###random removal###
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? True
Is G1 connected? False
nodes removed when randomly chosen 13
```

Let me run this again in random removal, it took 13 and in then I in the selective one it took 1, I am just going to run it last time. So, again in randomly it took 17 and in selective it took 1. You can just keep checking that ok. So, that is one thing. In case of

real-world networks this sort of thing happens. Now, we are going to check the same thing on a random network ok.

(Refer Slide Time: 14:10)


```python
 9
10 def remove_a_high_deg_node(G):
11     degrees = G.degree()
12     nodes = sorted(degrees.items(), key = lambda x:x[1], reverse = True)
13     #where the tuple is having two things: (node, degree)
14     G.remove_node(nodes[0][0])
15     return G
16
17 def main():
18     print '###########Real-world networks#############'
19     G = nx.read_edgelist('email.txt')
20     print nx.info(G)
21
22     raw_input()
23     print '###random removal###'
24     G1 = G.copy()
25     print 'Is G1 connected?', nx.is_connected(G1)
26     nodes_removed_random = 0
27
28     while(nx.is_connected(G1)):
29         G1 = remove_a_random_node(G1)
```

So, let me let me write here that the all this was for all this was for real world network or a network which basically follows power law distribution and preferential attachment, real world networks right. This does not look good, let me write this does not, let me write this. So, this was for real world networks. This looks very bad anyway, let us go ahead.

(Refer Slide Time: 14:49)


```python
47     print '###########Random Network#############'
48     H = nx.erdos_renyi_graph(1000, 0.2)
49     print nx.info(H)
50
51     raw_input()
52     print '###random removal###'
53     H1 = H.copy()
54     print 'Is H1 connected?', nx.is_connected(H1)
55     nodes_removed_random = 0
56
57     while(nx.is_connected(H1)):
58         H1 = remove_a_random_node(H1)
59         nodes_removed_random += 1
60         print 'Is H1 connected?', nx.is_connected(H1)
61     print 'nodes removed when randomly chosen', nodes_removed_random
62
63     raw_input()
64
65     print '###Selective removal###'
66     H2 = H`.copy()
67     print 'Is G2 connected?', nx.is_connected(G2)
```

So, next we are going to do the same things for a random for a random network. Now how can we create a random network? Ok in the previous videos we created an Erdos Renyi network. We can use that code to create a network and we can also use a function from network x where we have a function where we pass the values of n and p and we get a random network.

So, in order to tell you the function, I will use that function only over here. Let me call the graph H. I wrote nx.erdos_renyi network. So, this is a I am sorry, graph. So, this is the name of the function and let me pass some n, the total number of nodes and the value of t the probability. This should give us random network. Let me print the basic information of H.

After that we want to remove the nodes as the same way we did earlier. Let me copy paste the entire code here and we will make changes into that to this right. So, I am just pasting the entire thing here, random network starting here only. Raw input and then random removal l the graph name is H, what to write to ok. We are making copy of H and then we are checking whether H1 is connected or not and nodes removed is find and then nx is this connected H, (Refer Time: 16:47) I could have actually use here replace like replace function here, it is ok. So, everything else remains the same, I am just changing the graph here ok.

(Refer Slide Time: 17:00)


```
57      while(nx.is_connected(H1)):
58          H1 = remove_a_random_node(H1)
59          nodes_removed_random += 1
60          print 'Is H1 connected?', nx.is_connected(H1)
61      print 'nodes removed when randomly chosen', nodes_removed_random
62
63      raw_input()
64
65      print '###Selective removal###'
66      H2 = H.copy()
67      print 'Is H2 connected?', nx.is_connected(H2)
68      nodes_removed_selective = 0
69
70      while(nx.is_connected(H2)):
71          H2 = remove_a_high_deg_node(H2)
72          nodes_removed_selective += 1
73          print 'Is H2 connected?', nx.is_connected(H2)
74      print 'nodes removed when selectively chosen', nodes_removed_selecti
75
76
77  main()
```

I actually should have use replace method, but I think we are done we will almost done now, check it last one alright. So, I think we have good to go. We have just taken random network now and we are applying random removal as well as selective remove in this case. And we are continuing the number of iterations that are required in both the cases ok.

(Refer Slide Time: 17:39)


```python
 7          G.remove_node(r)
 8          return G
 9
10 def remove_a_high_deg_node(G):
11          degrees = G.degree()
12          nodes = sorted(degrees.items(), key = lambda x:x[1], reverse = True)
13          #where the tuple is having two things: (node, degree)
14          G.remove_node(nodes[0][0])
15          return G
16
17 def main():
18                                    I
19          '''
20          print '###########Real-world networks#############'
21          G = nx.read_edgelist('email.txt')
22          print nx.info(G)
23
24          raw_input()
25          print '###random removal###'
26          G1 = G.copy()
27          print 'Is G1 connected?', nx.is_connected(G1)
```

Since, we have already observed the behaviour on a real-world network, let me comment that part. So, that we are able to concentrate on only real-world network.

(Refer Slide Time: 17:45)


```python
40          print 'Is G2 connected?', nx.is_connected(G2)
41          nodes_removed_selective = 0
42
43          while(nx.is_connected(G2)):
44              G2 = remove_a_high_deg_node(G2)
45              nodes_removed_selective += 1
46              print 'Is G2 connected?', nx.is_connected(G2)
47          print 'nodes removed when selectively chosen', nodes_removed_select)
48
49          '''
50          print '###########Random Network#############'
51          H = nx.erdos_renyi_graph(1000, 0.2)
52          print nx.info(H)
53
54          raw_input()
55          print '###random removal###'
56          H1 = H.copy()
57          print 'Is H1 connected?', nx.is_connected(H1)
58          nodes_removed_random = 0
59
60          while(nx.is_connected(H1)):
```

So, I am commenting this part up to which point this point ok.

(Refer Slide Time: 17:50)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Cod
e/code videos$ python forced_vs_random_recording.py
###########Random Network#############
Name: gnp_random_graph(1000,0.2)
Type: Graph
Number of nodes: 1000
Number of edges: 100063
Average degree: 200.1260

###random removal###
```

So, let us check our code. So, this is for random network we have 1000 nodes and these many edges. I am pressing enter, random removal is taking place.

(Refer Slide Time: 18:01)

```
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? True
Is H2 connected? False
nodes removed when selectively chosen 928
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Cod
e/code videos$
```

So, you see this is taking a lot of time it is taking a (Refer Time: 18:07) number of time. It is actually going on and on as you can see it is going on and on still going on ok. So, the number of nodes we had to remove from a random network when we randomly choose them was 985. So, (Refer Time: 18:28) 985.

I am pressing enter here. Now we are selectively removing that is we are choosing the nodes with high degree and then we are removing them. It is again going on and on surprisingly it is going on and on; let us go on and on 928 as you can see. So, in the case of random network when we randomly it remove the nodes, it took a some 938 nodes to make the graph disconnected, and in case when we remove the nodes based on their degrees that is we remove the high degree nodes and every iteration, it again took as 928 nodes to make the graph disconnected.

Now that is happening because, there are no hubs in random networks which are connecting a lot of nodes. So, all the nodes are sort of equally likely because, equally likely in the sense of the number of nodes that they are connected to. So, there is no preference because, the edges were added randomly and there is no preference towards addition of nodes to some particular nodes in this.

So, that is a sort of difference that we observe in case of real-world networks as well as random networks, because real world networks are not random and they actually exhibit some sort of preference towards the nodes.