1. Objectives

In this project, we will study a real network and specifically, we are going to study the properties of this network such as connectivity, degree distribution, and different ways to compute weighted network's community, largest community and sub-community and overlapped community detection. This project serves as a good review of all previous projects and gives a good view of the real stuffs in our life.

2. Problems

2.1. Problem 1

As we construct the graph from given file as a directed graph, we found this graph is not connected. However, the giant connected component is very large, which comprises 10487 vertices out of 10501 of the whole graph.

2.2 Problem 2

Since the graph is directed, the in-degree and out-degree distribution of the graph are calculated separately, which are shown below as figure 2.1 and figure 2.2.

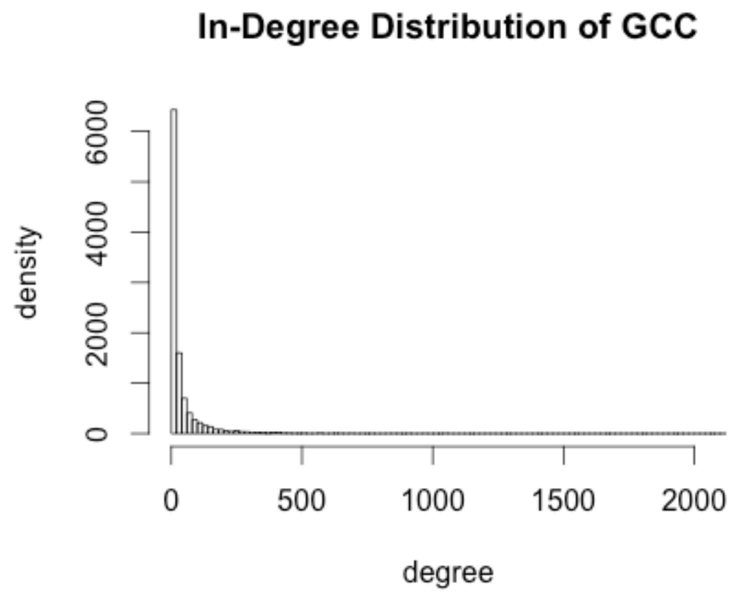

figure 2.1 in-degree distribution

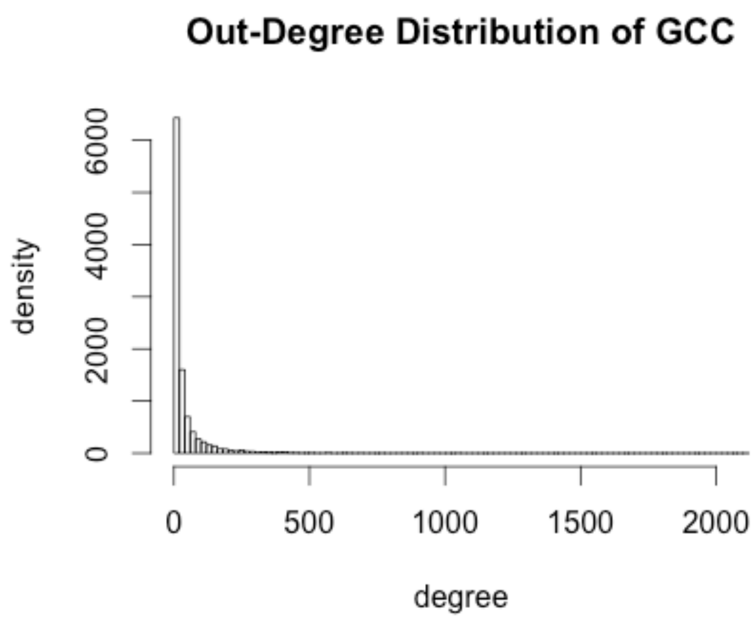

figure 2.1 out-degree distribution

2.3 Problem 3

We tested both option 1 and option 2 to get the community structure of the graph.

For option 1, we firstly converted the graph into an undirected graph and then used label propagation method to compute the community structure, which is shown as below in figure 3.1. The sizes of community are displayed in table 3.1 and the modularity of this model is 0.000102.

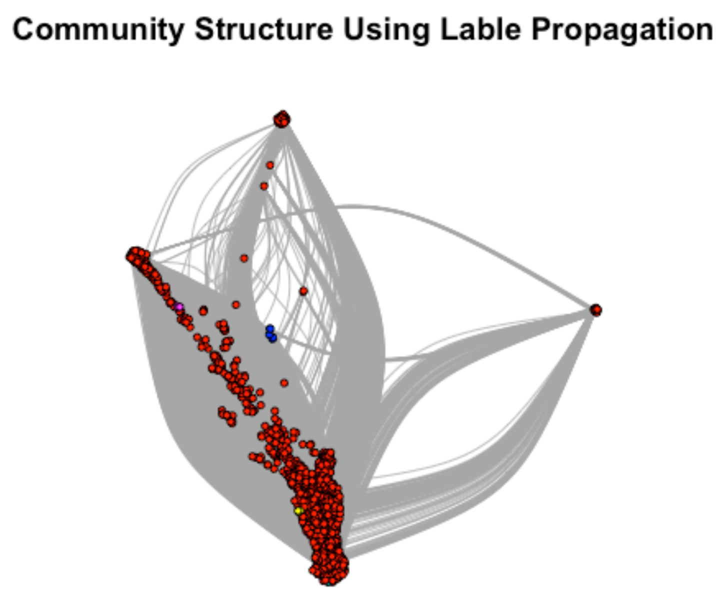

Figure 3.1 Community Structure using fast-greedy algorithm

community	1	2	3	4	5
size	10474	4	3	3	3

Table 3.1 sizes of communities of option 1

For option 2, we calculated the weight of two edges between two nodes as the square root of their product, which converted the directed graph into an undirected one. Then apply fast greedy algorithm and label propagation algorithm separately, which yields the community structure as shown in figure 3.2 and figure 3.3. The sizes of top communities of each model are shown in table 3.2 and table 3.3.

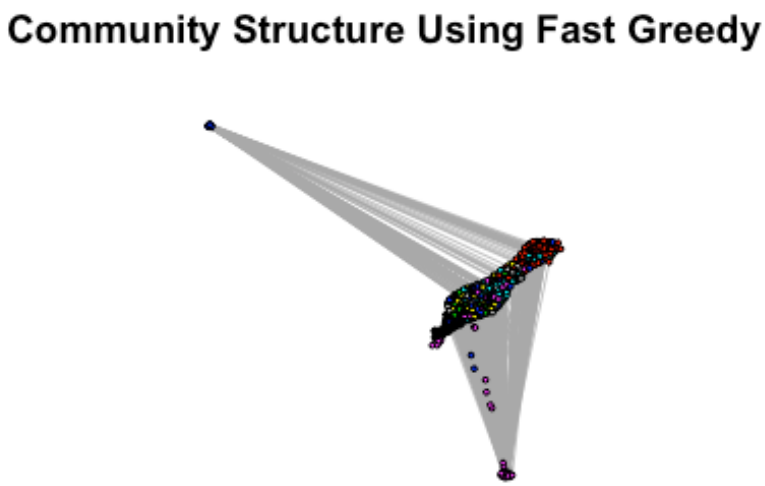

Figure 3.2 Community Structure using fast greedy algorithm

community	1	2	3	4	5	6	7	8
size	1856	1666	1022	2266	731	1236	633	1077

Table 3.2 sizes of communities of option 2 using fast greedy method

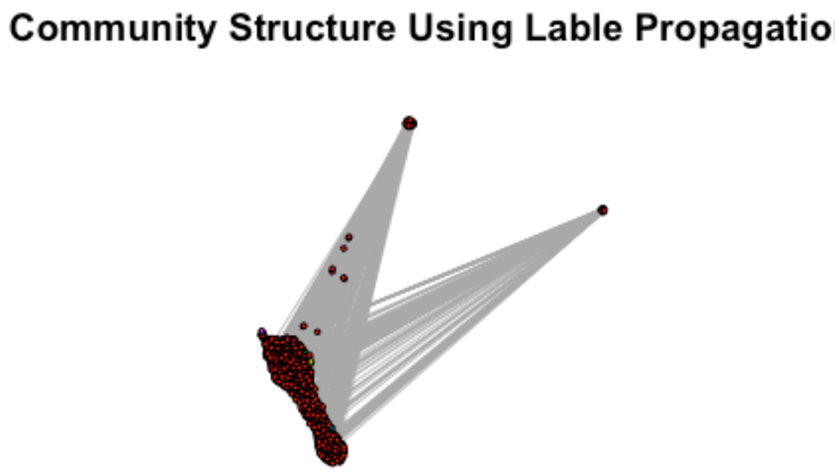

Figure 3.3 Community Structure using label propagation algorithms

community	1	2	3	4	5	6
size	10467	2	4	5	6	3

Table 3.3 sizes of communities of option 2 using fast greedy method

We found the results are not similar between two models since fast greedy algorithm breaks the graph into more communities while the label propagation tends to extract a giant sub community. The modularity of label propagation model is 0.000177 while the top modularity of communities in fast greedy are all 0.329.

2.4 Problem 4

As we deleted those nodes not belonging to the max sub-community and reconstruct the giant sub-community, which has 2266 vertices, we could compute the community structure of this sub-community by using both fast greedy algorithm and label propagation algorithm. The modularity of the top communities of fast greedy method are all 0.359 while the modularity of label propagation is 0. This is consistent with what we get in problem 3. The structure determined by two algorithms are shown as following in figure 4.1 and figure 4.2.

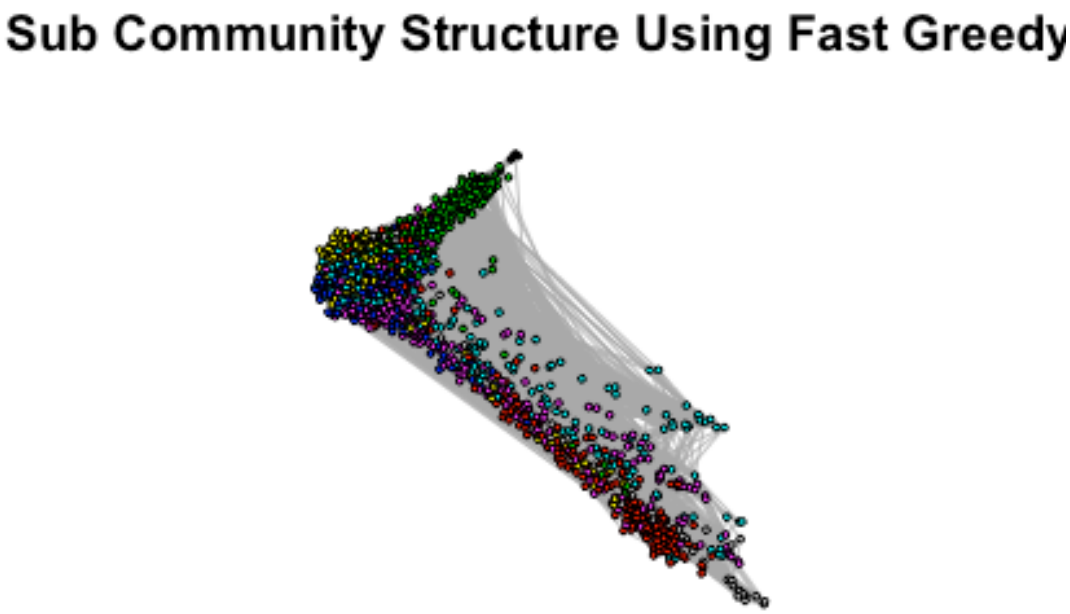

Figure 4.1 Sub Community Structure using fast-greedy algorithm

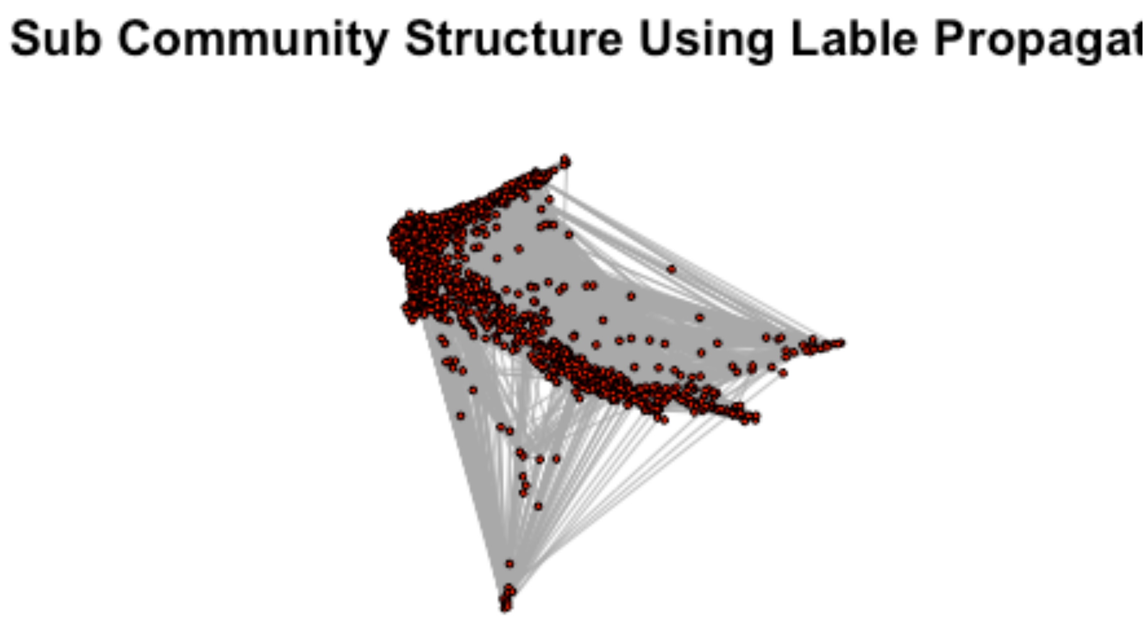

Figure 4.2 Sub Community Structure using label propagation algorithm

The result is consistent with the conclusion of problem 3, which is fast greedy algorithm tends to divide the graph into more small communities while label propagation algorithm tends to form less, in this sub-graph only one, communities.

2.5 Problem 5

We deleted the nodes not belonging to the community with size over 100 and calculate the modularity and sizes of the sub-communities. There are 8 such sub-communities we've found and their properties are shown through table 5.1 to table 5.8.

Sub-Community No.1							
Sub-community structure using fast greedy algorithm							
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Size	244	452	413	490	84	136	37
Modularity	0.2249632						
Sub-community structure using label propagation algorithm							
Comm No.	**1**				**2**		
Size	1853				3		
Modularity	0.001301893						

Table 5.1 sub-community 1 properties using two algorithms

Sub-Community No.2									
Sub-community structure using fast greedy algorithm									
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Size	357	491	347	294	128	28	13	5	3
Modularity	0.3711271								
Sub-community structure using label propagation algorithm									
Comm No.	**1**				**2**				
Size	1663				3				
Modularity	0.0002311537								

Table 5.2 sub-community 2 properties using two algorithms

Sub-Community No.3										
Sub-community structure using fast greedy algorithm										
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Size	318	209	30	193	118	44	41	15	10	6

Comm No.	11	12	13	14	15	16	17	18	19	20
Size	6	4	3	3	3	11	4	4		
Modularity	0.5128581									
Sub-community structure using label propagation algorithm										
Comm No.	1	2	3	4	5	6	7	8	9	10
Size	677	223	13	9	13	13	14	4	6	8
Comm No.	11	12	13	14	15	16	17	18	19	
Size	4	8	4	5	4	3	5	4	5	
Modularity	0.3979645									

Table 5.3 sub-community 3 properties using two algorithms

Sub-Community No.4								
Sub-community structure using fast greedy algorithm								
Comm No.	1	2	3	4	5	6	7	8
Size	306	457	313	365	426	347	47	5
Modularity	0.3595153							
Sub-community structure using label propagation algorithm								
Comm No.	1							
Size	2266							
Modularity	0							

Table 5.4 sub-community 4 properties using two algorithms

Sub-Community No.5										
Sub-community structure using fast greedy algorithm										
Comm No.	1	2	3	4	5	6	7	8	9	10
Size	60	247	138	56	55	50	74	16	11	4

Comm No.	**11**	**12**	**13**	**14**	**15**				
Size	4	2	7	3	4				
Modularity	0.3980826								
Sub-community structure using label propagation algorithm									
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Size	573	120	6	4	10	5	4	4	5
Modularity	0.2980612								

Table 5.5 sub-community 5 properties using two algorithms

Sub-Community No.6									
Sub-community structure using fast greedy algorithm									
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Size	275	294	184	47	167	66	100	93	10
Modularity	0.3986121								
Sub-community structure using label propagation algorithm									
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**		
Size	1209	9	5	3	3	3	4		
Modularity	0．006987121								

Table 5.6 sub-community 6 properties using two algorithms

Sub-Community No.7										
Sub-community structure using fast greedy algorithm										
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Size	162	64	153	65	60	50	38	12	7	3
Comm No.	**11**	**12**	**13**	**14**	**15**	**16**				
Size	3	4	3	3	3	3				

Modularity	0.4796647									
Sub-community structure using label propagation algorithm										
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Size	421	150	5	16	5	3	4	4	4	3
Comm No.	**11**	**12**	**13**	**14**	**15**	**16**				
Size	3	3	3	3	3	3				
Modularity	0.3452324									

Table 5.7 sub-community 7 properties using two algorithms

Sub-Community No.8										
Sub-community structure using fast greedy algorithm										
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Size	190	253	145	153	75	49	88	80	16	6
Comm No.	**11**	**12**	**13**							
Size	11	4	7							
Modularity	0.5036454									
Sub-community structure using label propagation algorithm										
Comm No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Size	49	874	42	43	22	4	6	7	3	4
Comm No.	**11**	**12**	**13**	**14**	**15**	**16**				
Size	5	4	3	4	4	3				
Modularity	0.1914544									

Table 5.8 sub-community 8 properties using two algorithms

We could see that in a sub-community of a large graph, we could still find community structures of it. In addition, FG algorithm will get communities with relatively similar size, while LP tends to give us a giant one.

2.6 Problem 6

We used the community information we calculated in problem 3, plus additional functions in netrw to solve this problem. By generating the personalized page rank, we found the visiting probability of each node in the giant connected component. Then we picked those top 30 nodes with largest visiting probability and calculated M associated with them. Then the threshold was set to determine whether a node belonged to multiple communities. The threshold is very important, namely one may find many qualified nodes if the threshold is set as a small value while the other may find few nodes with a large threshold. The code is shown as below:

```
threshold = 0.1
#random walk
walkernum=1
multi_com = numeric(0)
for(i in 1:vcount(g))
{
  teleprob = rep(0,vcount(g))
  teleprob[i]=1
  rw = netrw(g,walker.num = walkernum,
             start.node = i,damping = 0.85,
             output.visit.prob=T,
             teleport.prob=teleprob)
  prob = rw$ave.visit.prob
  sorted_prob = sort(prob,decreasing=T,index.return=T)
  M=rep(0,length(com_lp))
  #sum largest 30 vj
  for(j in 1:30)
  {
    mj=rep(0,length(com_lp))
    mj[com_lp$membership[which(V(gcc)==V(g)[sorted_prob$ix[j]])]]=1
    M=M+sorted_prob$x[j]*mj
  }
  #if M has 2 or more elements >threshold then printout
  if(length(which(M>threshold))>=2)
  {
    node_M=c(i,M)
    #save communtiny info
    multi_com=rbind(multi_com,node_M)
    cat("node:",i,"has multi-community\n")
  }
}
```

For fast greedy algorithm, after testing we set the threshold to be 0.2, which yielded 128 nodes belonging to multiple communities. And for label propagation algorithm, the threshold is relatively small, a threshold of 0.1 could lead to 24 qualified nodes. Those nodes are described in table 6.1 and table 6.2.

Node belonging to multiple communities of fast greedy algorithm									
68	149	151	726	868	1406	1507	1518	2106	2266
2356	2997	3768	3769	3879	3907	3969	4040	4161	4250
4297	4312	4356	4365	4407	4583	4586	4642	4665	4993
5095	5471	5648	5850	5902	5963	5997	6005	6315	6797
6803	6818	6825	6897	6914	6919	7007	7051	7082	7158

Table 6.1 some example nodes id using FG algorithm with threshold = 0.2

Node belonging to multiple communities of label propagation algorithm									
4687	7979	8915	10176	10177	10178	10179	10348	10349	10350
10351	10352	10401	10402	10403	10464	10465	10466	10468	10475
10476	10477	10486	10496						

Table 6.2 nodes id using label propagation algorithm with threshold = 0.1

From the tables we see at least two interesting features. One is that the fast greedy algorithm has larger threshold and tends to have more nodes, which means it has more nodes belonging to multi-communities. This makes sense because fast greedy algorithm tends to make more sub-communities, which has been shown in previous problems. The second fact is that, those nodes sometimes appears in a contiguous manner, for example, nodes with id equals to 10177-10179, 10348-10352, 10401-10403 all appear in the set of label propagation algorithm. This indicates a spatial continuity of vertices.

3. Difficulty encountered

This graph is a good review of previous exercise and many of the operation has been seen before. However, it is not easy to do this task if given no indication, e.g. how to compute the community structure of a directed nodes. Also, the idea of using personalized PageRank to study to overlapped communities structures is awesome. And it’s hard to be thought out by one himself. Since the calculation of random walker needs the package of netrw, we have to switch system and environment to support this operation, which caused a little tricky problem. But generally this is acceptable. And we gained good view of real network structure by studying this project.