EE 232E

Graphs and Network Flows

Homework 3 Report

Pingyuan Yue: 504737715

Ke Xu: 604761427

Yuanyi Ding: 404773978

4/28/2017

Table of Contents

1. Giant Connected Component

In this problem, a directed graph is created on a basis of a real network dataset. The graph could be easily created using the graph.data.frame API in igraph library.

```
myData = read.table("sorted_directed_net.txt")
g = graph.data.frame(myData, directed = TRUE)
```

The graph has 10501 vertices and 427486 edges which is very large. The graph is not connected. To get the giant connected component, we can first decompose the graph into a set of subgraphs and find the subgraph which has the most number of vertices.

2. In-degree & Out-degree Distribution

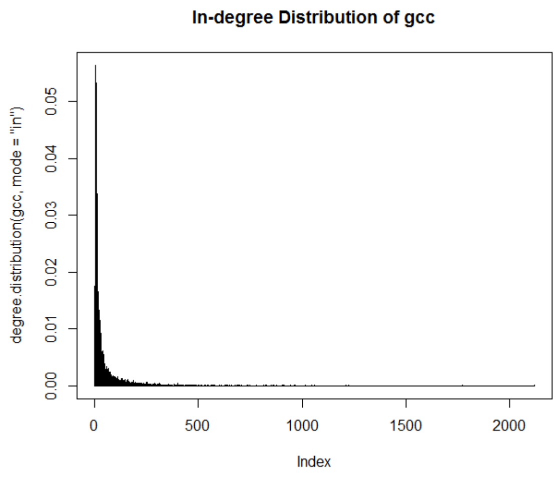


Fig2.1 In-degree Distribution of GCC

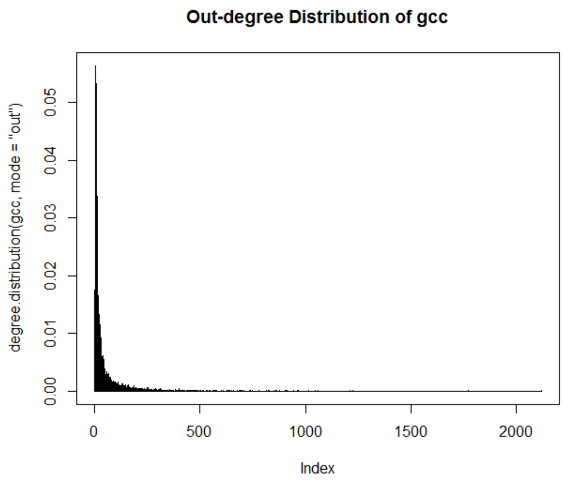


Fig2.2 Out-degree Distribution of GCC

The in-degree and out-degree distribution of the graph is plotted in the above two figures. From the figures above, it could be concluded that, in-degree and out-degree share similar pattern of distribution and the degree of most vertices in GCC is small.

3. Community of Undirected Graph

The first option to convert a directed graph into undirected one is to remain the number of edges unchanged, and just remove the directions. The second option is to merge the two directed edges (if exists) between two vertices. In this problem, we discuss these two options separately and use both fast greedy algorithm and label.propagation algorithm to calculate the community structure.

First option:

The community structure is calculated using label.propagation method after converting the graph to undirected graph. Analyzing the community structure, the property of the result is shown below:

Method	Size	Modularity
label.propagation	3	0.0001496995

Second option:

For the second option, the community structure is calculated using both fast greedy and label.propagation method after converting the graph to undirected graph. Analyzing the community structure, the property of the result is shown below:

Method	Size	Modularity
label.propagation	7	0.0002338966
fastgreedy	76	0.2200379

From the results above, it could be concluded that the results of fast greedy method and label propagation method are not similar. It turns out that fast greedy algorithm tends to divide the whole graph into more smaller communities while label propagation method tends to remain larger communities. In terms of modularity, fast greedy method turns out to generate better result.

4. Community of Largest Community

In this problem, we isolate and extract the largest community computed from fast greedy method in the above part. And then we calculate the community structure of the largest community using both fast greedy method and label propagation method. The result is shown below:

Method	Size	Modularity
label.propagation	1	0
fastgreedy	35	0.2678167

The result above is consistent with our conclusion in problem 3. Since label.propagation method tends to divide the graph into smaller part, the graph could not be further divided in this situation, while the fast greedy algorithm could work just fine.

5. Sub-Community Structures with Sizes Larger than 100

In this problem, we need to find all the sub-community structures of the communities whose sizes are larger than 100. We extract the nodes in the community with size larger than 100 and calculate the modularity and sizes of the sub-communities. Below are the results we have found:

Sub-community index: 1

Modularity of subcommunity using fastgreedy algorithm is: 0.216763

Community Sizes Table

Number	1	2	3	4	5	6	7	8	9	10	11
Sizes	1413	641	1363	47	22	10	5	6	4	3	5
Number	12	13	14	15	16	17	18	19	20	21	22
Sizes	3	3	5	4	3	4	4	3	3	2	2

Modularity of subcommunity using label propagation algorithm is: 9.397061e-05

Community Sizes Table

Number	1	2
Sizes	3552	3

Sub-community index: 2

Modularity of subcommunity using fastgreedy algorithm is: 0.2678167

Community Sizes Table

Number	1	2	3	4	5	6	7	8	9	10	11
Sizes	1800	1114	1401	112	33	33	15	14	6	8	7
Number	12	13	14	15	16	17	18	19	20	21	22
Sizes	4	4	8	4	6	3	4	2	6	7	3
Number	23	24	25	26	27	28	29	30	31	32	33
Sizes	3	3	3	3	2	3	2	2	3	3	2
Number	34	35									
Sizes	2	3									

Modularity of subcommunity using label propagation algorithm is: 0

Community Sizes Table

Number	1
Sizes	4628

Sub-community index: 3

Modularity of subcommunity using fastgreedy algorithm is: 0.5351793

Community Sizes Table

Number	1	2	3	4	5	6	7	8	9	10	11
Sizes	37	328	351	72	103	373	117	99	86	65	16
Number	12	13	14	15	16	17	18	19	20	21	
Sizes	13	11	20	9	8	7	9	5	8	3	

Modularity of subcommunity using label propagation algorithm is: 0.2411865

Community Sizes Table

Number	1	2	3	4	5	6	7	8	9	10	11
Sizes	1398	101	8	19	11	8	25	8	10	9	9
Number	12	13	14	15	16	17	18	19	20	21	22
Sizes	10	5	12	9	7	9	10	5	7	5	8
Number	23	24	25	26	27	28	29	30	31	32	33
Sizes	7	4	6	7	3	3	3	4	4	3	3

Sub-community index: 4

Modularity of subcommunity using fastgreedy algorithm is: 0.7217026

Community Sizes Table

Number	1	2	3	4	5	6	7	8	9	10	11
Sizes	53	13	21	12	10	12	6	11	7	6	4

Number	12	13	14
Sizes	3	4	4

Modularity of subcommunity using label propagation algorithm is: 0.675296

Community Sizes Table

Number	1	2	3	4	5	6	7	8	9	10	11
Sizes	67	3	12	15	13	6	7	4	7	4	5

Number	12	13	14	15	16	17	18
Sizes	2	5	4	3	3	3	3

6. PageRank for Overlapped Communities Structures

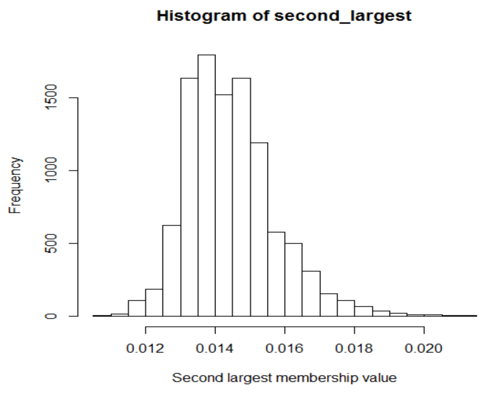
Firstly we used the community information we computed in problem 3, as well as functions in package ‘netrw’ to solve this problem.

By generating the personalized page rank, we then find 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, in which M and m are both matrix of n_nodes*n_communities. Here n_nodes is the number of nodes and n_communities is the number of communities generated. Then the threshold was set to determine whether a node belonged to multiple communities. Here we just implemented it by finding the second largest community characteristic in each node vector and set the threshold so that when the value is larger than the threshold it means the node belongs to at least 2 communities.

For fast greedy algorithm, after testing we set the threshold to be 0.02, which yielded 12 nodes belonging to at least 2 communities.

Then we lower the threshold to 0.01 and it turns out to be 10487 nodes belonging to at least 2 communities.

The following figure shows the distribution of number of nodes that belong to at least 2 communities versus the threshold.



Figrue6.1: the distribution of number of nodes that belong to at least 2 communities versus the threshold

We can see from the above figure that when the threshold becomes larger, the number of nodes belonging to 2 communities becomes smaller.