# EE 232 E Homework 3 Report

Chandan Dhal:- 904588105

Sidharth Gulati:-104588717

Tushar Sudhakar Jee:-004589213

## ***Introduction***

In this homework we analyzed the properties of a real network and briefly compared two non-overlapping community detection algorithms, namely, fast greedy and label propagation. Later, we also defined a personalized PageRank algorithm to detect overlapping communities.

### ***Exercise #1***

***Network***

We generated a weighted directed network using the edge list and the weights of the edge from the dataset. The network parameters are tabulated as below:-

| Parameters | Values |
|---|---|
| Connected | False |
| Nodes | 10501 |
| Diameter | 6 |

### ***Exercise #2***

***Degree Distribution***

We analyzed the in-degree distribution and out-degree distribution of the nodes in the network. For a node, the number of head ends adjacent to a vertex is called the ***in-degree*** of the node and the number of tail ends adjacent to a vertex is its ***out-degree***. The plots for the degree distribution of in-degree and out-degree of the nodes are depicted in figures below.

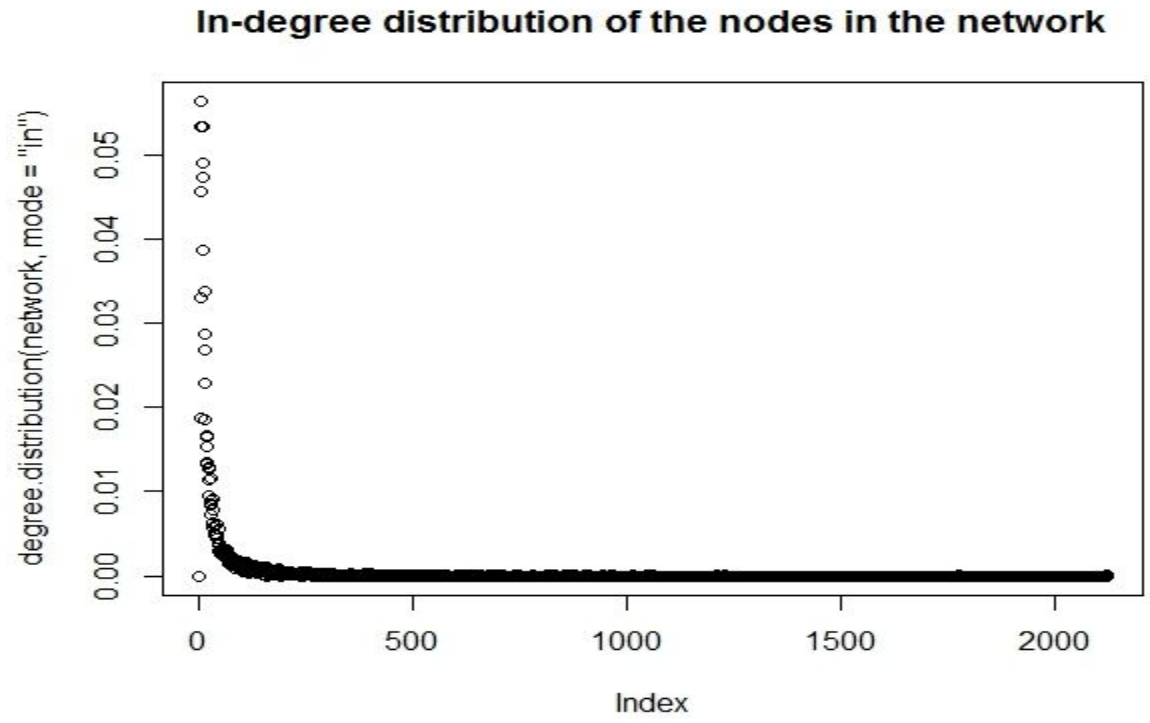


Fig. 1 In-degree Distribution of the nodes of the network

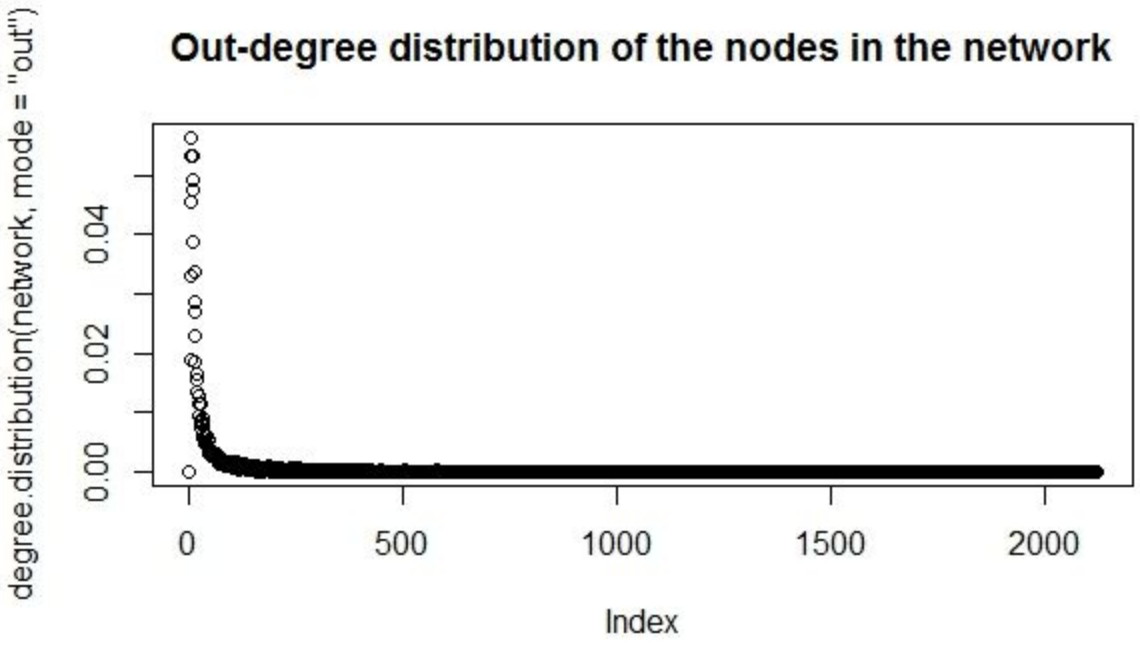


Fig. 2 Out-degree Distribution of the nodes of the network

From Fig 1 and Fig 2 we can observe that the in-degree distribution and the out-degree distribution are similar. Hence we can infer that in-degree and out-degree are correspond each other.

***Exercise #3***

***Giant Connected Component***

The network is disconnected and has a giant connected component in the middle of the network and has 10487 nodes. We used this giant connected component [GCC] to analyze the non-overlapping community detection algorithms. We decomposed the network to extract the GCC and plotted the in-degree and out-degree distribution of the GCC nodes. We also inferred that the GCC nodes also have similar in-degree and out-degree distribution, i.e. they correspond each other.

We can also infer that the in-degree and out-degree distribution of the network nodes and GCC nodes are somewhat similar to ***power-law distribution***.

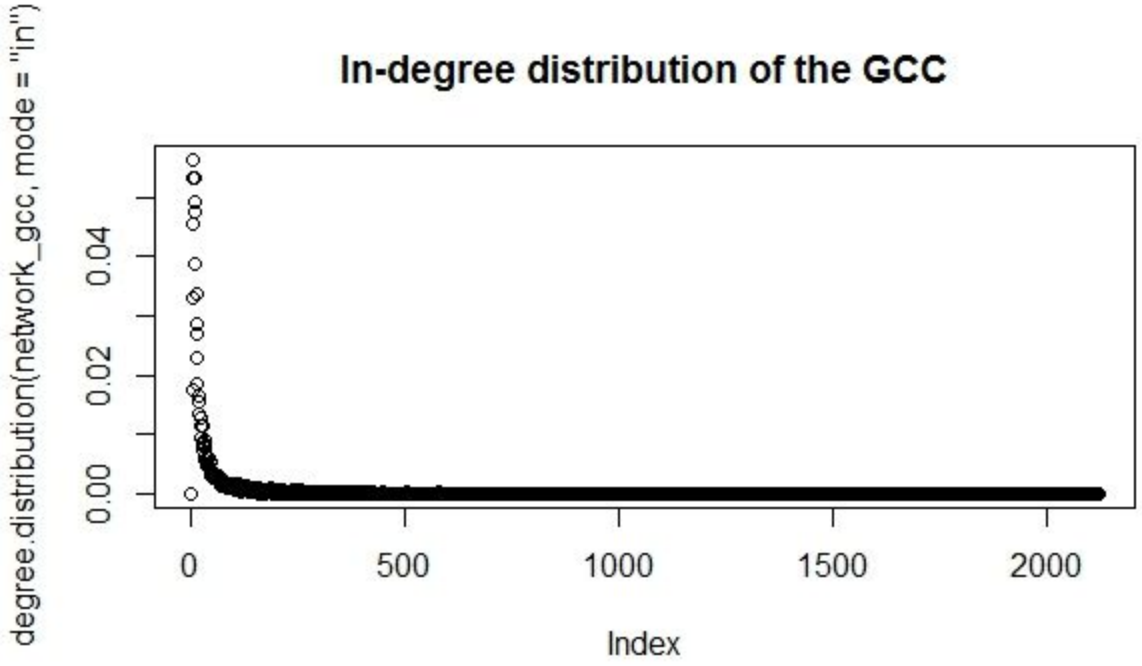


Fig 3. In-degree Distribution of the nodes in GCC

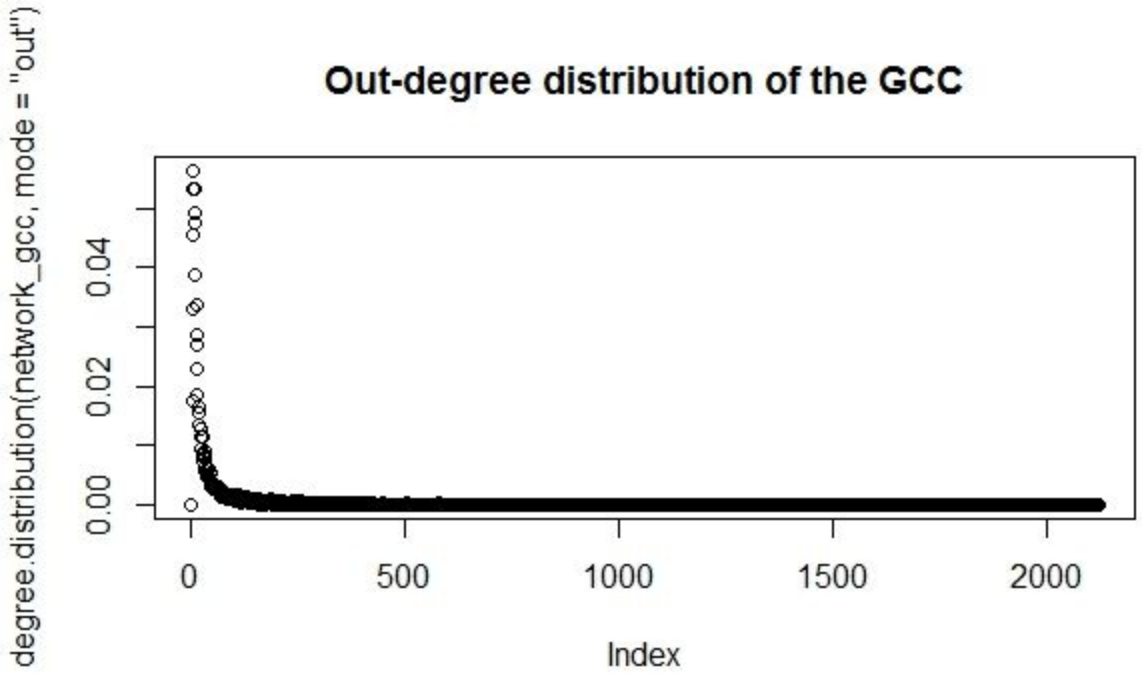


Fig. 4 Out-degree distribution of the network.

***Community Detection Algorithms***

In this exercise we briefly analyze two community detection algorithms.

- Label propagation community detection
- Fast greedy community detection

***Label propagation community detection***

It is a fast, nearly linear time algorithm for detecting community structure in network. Initially, every node is assigned with a unique label. At every step, each node update it's label to a new one which most of its neighbors shares. The stop criterion is until every node has a neighbor that is the maximum label of its neighbors. In this fashion, densely connected group of nodes can reach a consensus on a unique label and can form a community quickly. This community detection algorithm does not involve modularity optimization. Also, this community detection algorithm can find communities in both simple and non-simple undirected networks.

***Fast greedy community detection***

It tries to find the communities in the network by optimizing a modularity score. Starting from a set of isolated nodes, the links of the original graph are iteratively added such to produce the largest possible increase of the modularity. In this fashion, it results in a partitioning with maximum modularity score

possible on the given network. This community detection algorithm can find communities in simple undirected networks.

These algorithms require undirected network. Hence we require to convert our detected weighted directed GCC to weighted undirected GCC. In this homework we discuss two options to convert the directed weighted network to undirected weighted network.

*Option 1 (Label Propagation algorithm):*

In this method we removed the direction from edges while keeping the number of edges and used label propagation algorithm to determine network community structure. The resulting network is not a simple undirected network as there are possibility of multiple edges between nodes.

The various community parameters are tabulated below:-

| Community ID | Number of nodes |
|---|---|
| 1 | 10476 |
| 2 | 3 |
| 3 | 2 |
| 4 | 3 |
| 5 | 3 |

The modularity score of the partition is 6.792434e-05. The modularity score of this partitioning is **very low** because this algorithm does not find the communities by optimizing the modularity score. Instead it finds the communities by a majority voting in the neighborhood. Hence it returns communities of uneven sizes as result the modularity score of the partition is very low.

The figure below depicts the resultant community structure of label propagation algorithm.

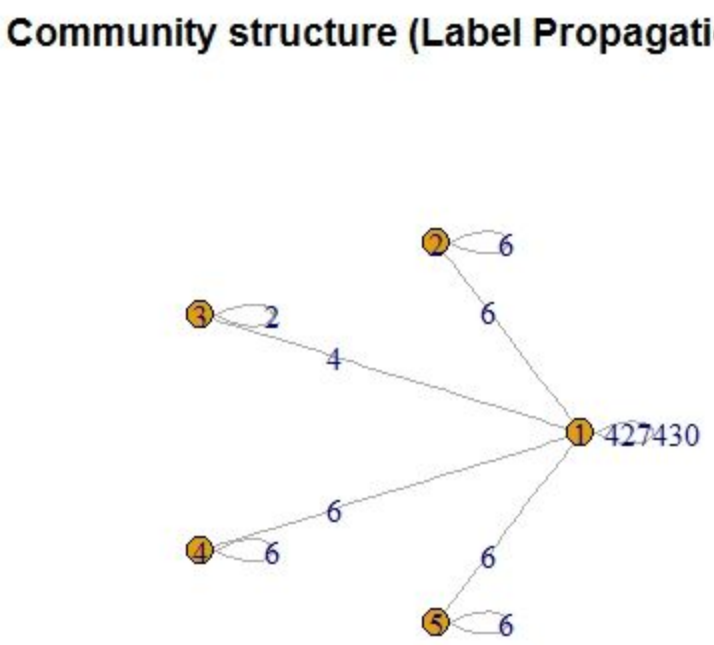


Fig 5. Community structure, the circles represents the 5 communities. The loops around the circles represents the number of edges between the nodes in that community and the lines between two communities represents the number of edges between two communities. For example, there are 427430 edges between the nodes in community 1 and 4 edges between community 1 and 3.

*Option 2 (Label Propagation and fast greedy algorithms comparision):*

In this option, we merged the two directed edges to one undirected edge defined the weight of the undirected edge by the geometric mean of the weights of the two directed edges.This resulting network is a simple weighted undirected network because there can only be one edge between two nodes in the network. We used both label propagation and fast greedy algorithm to define the community structure.

*Label Propagation algorithm results*

The modularity score of the partition is 0.001, which quite evident because of the uneven sizes of the communities. The various community parameters are tabulated below:-

| Community ID (label Propagation) | Number of nodes |
|---|---|
| 1 | 10472 |
| 2 | 4 |
| 3 | 3 |
| 4 | 3 |
| 5 | 5 |

*Fast Greedy algorithm results*

The modularity score of the partition is 0.328. The modularity score of this partitioning is **high** [compared to level propagation] because this algorithm finds the communities by optimizing the modularity score. Hence it returns communities of somewhat even sizes **unlike label propagation** as result the modularity score of the partition is very low. The various community parameters are tabulated below:-

| Community ID (Fast Greedy) | Number of nodes |
|---|---|
| 1 | 1836 |
| 2 | 791 |
| 3 | 1701 |
| 4 | 1213 |
| 5 | 2316 |
| 6 | 634 |
| 7 | 963 |
| 8 | 1033 |

The figure below depicts the resultant community structure of fast greedy algorithm.

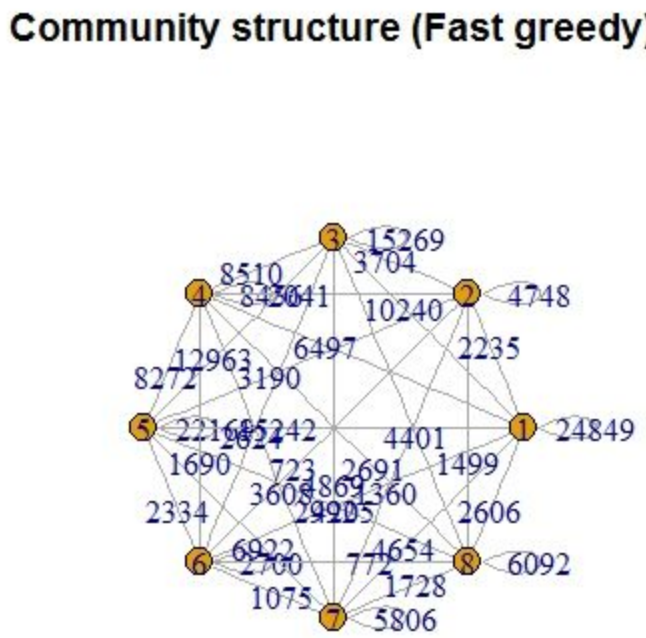


Fig 6. Community structure, the circles represents the 8 communities. The loops around the circles represents the number of edges between the nodes in that community and the lines between two communities represents the number of edges between two communities. For example, there are 24849 edges between the nodes in community 1 and 3704 edges between community 1 and 3.

***Exercise #4 and #5***

In this exercises we analyze the sub community structures of the communities found using the fast greedy community detection algorithm. Sub community structures are the communities within a community. The algorithm to detect subcommunity is summarized as below:

1. Separate the community detected by the fast greedy algorithm from the network and generate a new network of the detected community.
2. Use fast greedy algorithm to find the community structure of the new network.

3. If community size is more than 100, then repeat step 1 and 2, to determine its community structure.

***Subcommunity Structure of largest community***

The above algorithm was used to determine the subcommunity structure of the largest community detected by fast greedy algorithm. In the exercise 3 we are concerned for only the largest community, hence only steps 1 and 2 were implemented, where the step 2 is only implemented on the largest community found in step 1. The various subcommunity parameters of the largest community are tabulated below:

| Community ID | Number of nodes |
|---|---|
| 1 | 39 |
| 2 | 378 |
| 3 | 417 |
| 4 | 370 |
| 5 | 32 |
| 6 | 301 |
| 7 | 341 |
| 8 | 438 |

The figure below depicts the resultant subcommunity structure of the largest community detected by fast greedy algorithm.

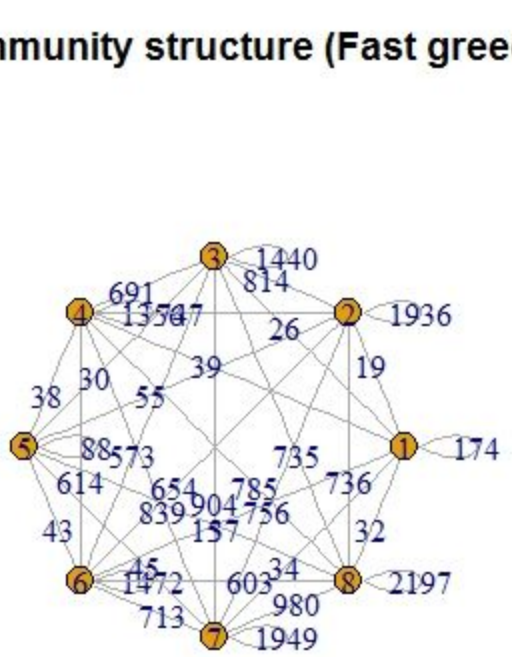


Fig 7. Subcommunity structure of largest community, the circles represents the 8 communities. The loops around the circles represents the number of edges between the nodes in that community and the lines between two communities represents the number of edges between two communities. For example, there are 174 edges between the nodes in community 1 and 26 edges between community 1 and 3.

***Subcommunity Structure of community size greater than 100***

The above algorithm was used to determine the subcommunity structure of the largest community detected by fast greedy algorithm. The algorithm resulted in all 8 communities to have size greater than 100, which is quite evident from the results of exercise 3.
The various subcommunity parameters of each resulted community are tabulated below:

Community 1:

| Community ID | Number of nodes |
|---|---|
| 1 | 262 |
| 2 | 454 |
| 3 | 492 |
| 4 | 398 |
| 5 | 88 |
| 6 | 126 |
| 7 | 16 |

Community 2:

| Community ID | Number of nodes |
|---|---|
| 1 | 134 |
| 2 | 67 |
| 3 | 262 |
| 4 | 113 |
| 5 | 65 |
| 6 | 59 |
| 7 | 31 |
| 8 | 15 |
| 9 | 13 |
| 10 | 4 |
| 11 | 7 |
| 12 | 4 |

| | |
|---|---|
| 13 | 7 |
| 14 | 6 |
| 15 | 4 |

Community 3:

| Community ID | Number of nodes |
|---|---|
| 1 | 502 |
| 2 | 358 |
| 3 | 346 |
| 4 | 142 |
| 5 | 303 |
| 6 | 32 |
| 7 | 10 |
| 8 | 5 |
| 9 | 3 |

Community 4:

| Community ID | Number of nodes |
|---|---|
| 1 | 279 |
| 2 | 189 |
| 3 | 281 |
| 4 | 88 |
| 5 | 53 |

| | |
|---|---|
| 6 | 159 |
| 7 | 69 |
| 8 | 98 |
| 9 | 4 |

Community 5

Community is the largest community in the network. The subcommunity structure and parameters are discussed in the above section of this exercise.

Community 6:

| **Community ID** | **Number of nodes** |
|---|---|
| 1 | 170 |
| 2 | 68 |
| 3 | 78 |
| 4 | 156 |
| 5 | 40 |
| 6 | 43 |
| 7 | 33 |
| 8 | 19 |
| 9 | 8 |
| 10 | 3 |
| 11 | 3 |
| 12 | 4 |
| 13 | 3 |

| | |
|---|---|
| 14 | 3 |
| 15 | 3 |

Community 7:

| Community ID | Number of nodes |
|---|---|
| 1 | 296 |
| 2 | 198 |
| 3 | 88 |
| 4 | 169 |
| 5 | 77 |
| 6 | 29 |
| 7 | 65 |
| 8 | 10 |
| 9 | 6 |
| 10 | 3 |
| 11 | 3 |
| 12 | 4 |
| 13 | 8 |
| 14 | 7 |

Community 8:

| Community ID | Number of nodes |
|---|---|
| 1 | 190 |
| 2 | 57 |

| | |
|---|---|
| 3 | 248 |
| 4 | 124 |
| 5 | 90 |
| 6 | 72 |
| 7 | 25 |
| 8 | 112 |
| 9 | 83 |
| 10 | 6 |
| 11 | 9 |
| 12 | 6 |
| 13 | 4 |
| 14 | 7 |

***Inference from subcommunity structure exercise***

The table below briefly summarizes the community structures and its subcommunity parameters (in ascending order of the community size tabulated in exercise 3 fast greedy section).

| **Community ID** | **Number of nodes** | **subcommunity** | **Modularity score of subcommunity structure** |
|---|---|---|---|
| 6 | 634 | 15 | 0.478 |
| 2 | 791 | 15 | 0.419 |
| 7 | 963 | 14 | 0.5 |
| 8 | 1033 | 14 | 0.505 |
| 4 | 1213 | 9 | 0.398 |
| 3 | 1701 | 9 | 0.372 |

| | | | |
|---|---|---|---|
| 1 | 1836 | 7 | 0.223 |
| 5 | 2316 | 8 | 0.362 |

We infer that as the community size increases, the number of sub communities decreases and, as the number of nodes in the community increases the modularity score decreases.

***Exercise #6***

As previously seen that fast greedy and label propagation method find communities within networks by assigning different nodes to different communities. But the fact a node may belong to multiple communities is not taken into account by these algorithms. A case in point would be for a node to be equally connected with two sets of nodes but these algorithms only assign the node to one of the communities. However we can use this initial one community only data and come up with techniques to check whether nodes belong to multiple communities.
As shown earlier,random walks across networks gives the average probability of visiting a node, given we started from a certain node. Nodes strongly connected within a community are more likely to be visited if we start from a node within this community. Multiple community membership of a node can also be defined by this method. The algorithm for obtaining this multi-community membership is defined as follows:

1. Perform random walk starting from each node in the network to obtain the average probability of visiting a node 'j' ($v_j$) assuming the walker started from node 'i'. It is higher for nodes that are within the same communities.
2. The random walk was performed with the damping probability as 0.85. However, to ensure that the starting point would always be the same node the local.pagerank parameter of the netrw function was set to TRUE. This meant that if the walker decided to stop crawling and teleport, it would always teleport to the node that it started at ensuring that visit probabilities calculated represent correct information.
3. Subsequently, we use the one community membership information(from the fast greedy algorithm) to create a membership vector $m_j$ for each node. The number of elements in this vector is equal to the number of communities found by the fast greedy algorithm. All the elements are zero except the element corresponding to the one community which it belongs which is one.

4. A soft measure of how strongly or weakly a node 'i' belongs to one of the 'K' communities the vector $M_i$ is calculated as follows:

$$\vec{M}_i = \sum_j v_j \vec{m}_j \ ,$$

5. This vector will contain values at all elements indicating how strongly a node belongs to a certain community with the highest value corresponding to the community found by the fast greedy algorithm.
6. Finally, to make the algorithm more robust we threshold values of the vector based on if they are larger than half the highest membership score in the vector.

The algorithm above proves that different nodes could in fact quite strongly belong to two or even three communities. The illustrations are as shown below:

| Node ID | FG membership | Mi vector values | Multi community membership |
|---|---|---|---|
| 3404 | 4 | [0.1250, 0.0475, 0.2050, 0.2900, 0.0150, 0.2150, 0.0525, 0.0525] | 3,4,6 |
| 7056 | 3 | [0.2300, 0.0175, 0.4525, 0.0700, 0.1300, 0.0225, 0.0450, 0.0325] | 1,3 |
| 201 | 5 | [0.14,0.17,0.135,0.1,0.2775, 0.0350,0.0450,0.0975] | 1,2,5 |

Using the above method, the multi community membership was calculated for all the nodes of the network. The plot below shows a distribution of the number of communities the nodes belong to. While majority of the nodes belong to a single community we can see that a significant number belong to two or more communities. Another observation is that this distribution also follows a sort of power law distribution i.e. very few nodes belong to a large number of communities. This indicates that there are few nodes that are very highly connected to a lot of nodes.

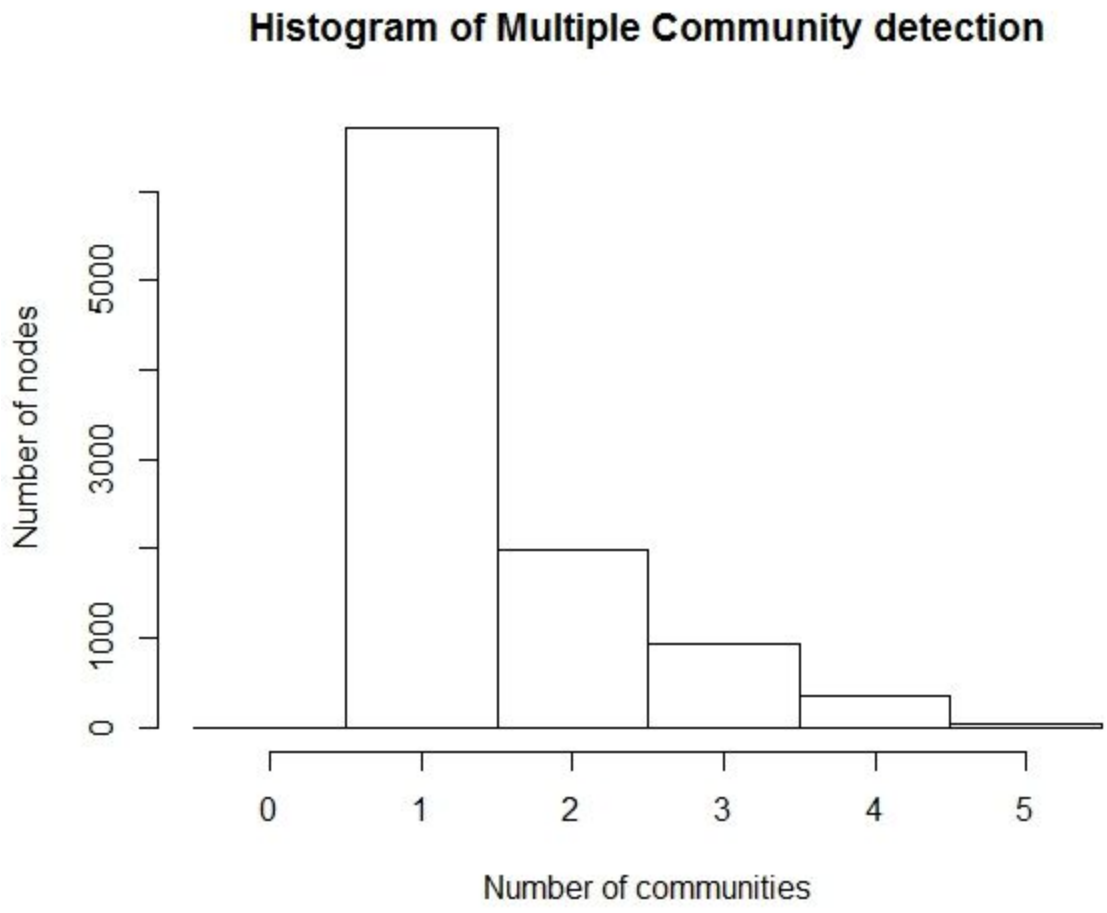


Fig 8: Multiple community detection Histogram

***Conclusion***

In this homework we used an actual network and analyzed two community structure detection algorithms, namely, label propagation and fast greedy algorithms. Initially we discussed two options on converting a directed network to undirected network and analyzed the community and subcommunity structures. In exercises, we observed that level propagation is a fast algorithm and has very low modularity. The algorithm has low modularity as it does not optimize the modularity score and results in communities of very different sizes. Fast greedy algorithm, has higher modularity score compared to label propagation algorithm and results in communities of somewhat similar sizes. Hence fast greedy algorithm resulted to be somewhat slower compared to label propagation. We used the fast greedy algorithm to measure subcommunity structure of the largest community and communities greater than size 100.

Finally, we studied overlapping community structure, i.e. a node can belong to multiple communities at the same time. The community detection algorithms in igraph package fails to find overlapping

communities. Hence, we used the personalized PageRank algorithm from HW2 along with a scoring vector to find the overlapping communities. We found many nodes belonging to multiple communities, however the number of nodes belonging to multiple communities decreased fairly quickly.

***Citations:***

[1]Community Detection in Networks with Node Attributes
Jaewon Yang ,Julian McAuley and Jure Leskovec