Homework 3

Professor - Vwani Roychowdhury

Class - EE232E Graphs and Network Flows

**1. Is this network connected? If not, find out the giant connected component. And in the following, we will deal with this giant connected component.**

|  |
| --- |
| **[Code]** |
| # Problem 1  g <- read.graph("c:/down/sorted\_directed\_net.txt", directed=TRUE, format="ncol")  is.connected(g)  # Clustering  clusterlist <- clusters(g)  size <- clusterlist$csize  numcluster <- clusterlist$no  # Find the GCC  GCCindex <- which.max(size)  nonGCCnodes<-(1:vcount(g))[clusterlist$membership!=GCCindex] #compares the membership vector with the index , then multiply TRUE with node list  GCCfinal <- delete.vertices(g,nonGCCnodes)  cat(sprintf("Original Graph Vertex:%d Edge:%d\nGiant Connected Component Vertex:%d Edge:%d",vcount(g),ecount(g),vcount(GCCfinal),ecount(GCCfinal))) |



Figure 1-Network Connection

As you see in Figure 1, the network is not connected. So we found a giant connected component.



Figure 2-Vertex and Edge numbers of Original network and GCC

In Figure 2, we could observe that the original network consists of 10501 vertices and 427486 edges, and the giant connected component consists of 10487 vertices and 427472 edges.

**2. Measure the degree distribution of in-degree and out-degree of the nodes.**

|  |
| --- |
| **[Code]** |
| # Problem 2  plot(degree.distribution(GCCfinal,mode = "in"), main="In Degree Distribution")  plot(degree.distribution(GCCfinal,mode = "out"), main="Out Degree Distribution")  identical(indegree,outdegree) |

|  |
| --- |
|  |
|  |

Figure 3-Degree Distribution (In/Out)

The in degree distribution and the out degree distribution seemed almost same.



Figure 4-Maximum probability of In/Out degree distribution

The maximum probabilities of in and out degree distribution were same to 0.05635549.



Figure 5-Comparing In/Out degree distribution

In Figure 5, we confirmed that the in degree distribution and the out degree distribution were exactly same.

**3. We would like to measure the community structure of the network. First, we need to convert it into an undirected network. Then we use fastgreedy.community and label.propagation.community to measure the community structure. Are the results of these two methods similar or not?**

|  |
| --- |
| **[Code]** |
| # Problem 3  undirected\_g\_option1 <- as.undirected(GCCfinal, mode = "each")  undirected\_g\_option2 <- simplify(undirected\_g\_option1, remove.multiple = TRUE, edge.attr.comb=list(weight="prod"))  E(undirected\_g\_option2)$weight <- sqrt(E(undirected\_g\_option2)$weight)  is.directed(undirected\_g\_option1)  is.directed(undirected\_g\_option2)  cat(sprintf("Undirected Network Option1 Edge : %d\nUndirected Network Option2 Edge : %d",ecount(undirected\_g\_option1),ecount(undirected\_g\_option2)))  # For Option 1 - Label Propagation  # Label Propagation Community  lp\_op1 <- label.propagation.community(undirected\_g\_option1)  # For Option 2 - Fast greedy & Label Propagation  # Fast greedy Community  fg\_op2 <- fastgreedy.community(undirected\_g\_option2)  # Label Propagation Community  lp\_op2 <- label.propagation.community(undirected\_g\_option2)  sizes(lp\_op1)  sizes(fg\_op2)  sizes(lp\_op2) |



Figure 6-Number of Edges of Undirected Network Option 1 & 2

We converted the original network which is directed to the undirected network. There are two options; first option is keeping the number of edges unchanged and just removing the directions, and second option is merging multiple edges between two nodes and setting the weight be a square root of multiplication of two weights. In Figure 6, we could observe that the number of edges of the option 1 was 427472 and the number of edges of the option 2 was 213736.

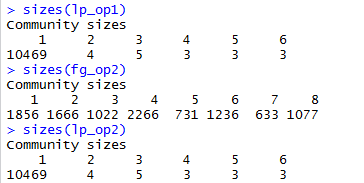


Figure 7-Community sizes

In Figure 7, we could observe that community sizes of each community detection algorithm. The first one was by label propagation algorithm and option 1 undirected network. The number of detected communities was 6 and the number of community 1 was 10469. Most of nodes were classified as community 1. The second one was by fast greedy algorithm and option 2 undirected network. The number of communities was 8 and the maximum number of size was 2266 which is community 4. The last one was by label propagation algorithm and option 2 undirected network. And the result was same as the first label propagation algorithm with option 1 network.

**4. Find the largest community computed from fastgreedy.community. Isolate the community from other parts of the network to form a new network, and then find the community structure of this new network. This is the sub-community structure of the largest community.**

|  |
| --- |
| **[Code]** |
| # Problem 4  largest\_community <- which.max(sizes(fg\_op2))  sub\_community\_network <- induced.subgraph(undirected\_g\_option2, which(membership(fg\_op2) == largest\_community))  fg\_sub\_community <- fastgreedy.community(sub\_community\_network)  dendPlot(fg\_sub\_community)  plot(fg\_sub\_community,sub\_community\_network,  layout=layout.fruchterman.reingold,  vertex.size=3,  vertex.label=NA,  edge.arrow.size=.1  ) |



Figure 8-Largest community size and index

The largest community computed from fast greedy was 4. And the size of the community 4 was 2266. And then we created new sub network of community 4.



Figure 9-Community sizes of sub-community

The sub community was created by fast greedy algorithm. In Figure 9, we could observe each sub community size. The number of sub community was 8.

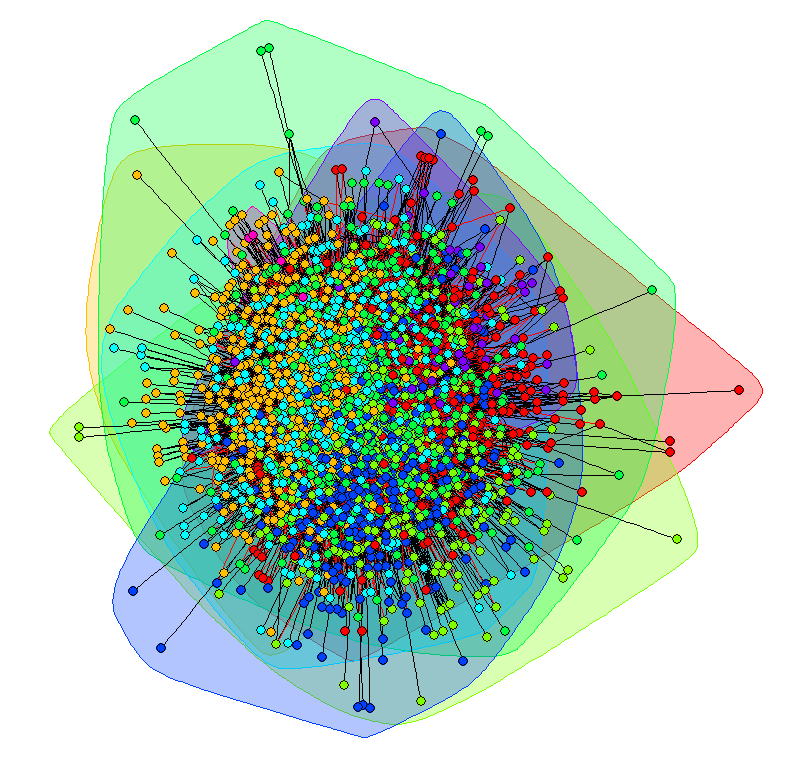


Figure 10-Community Structure Plot

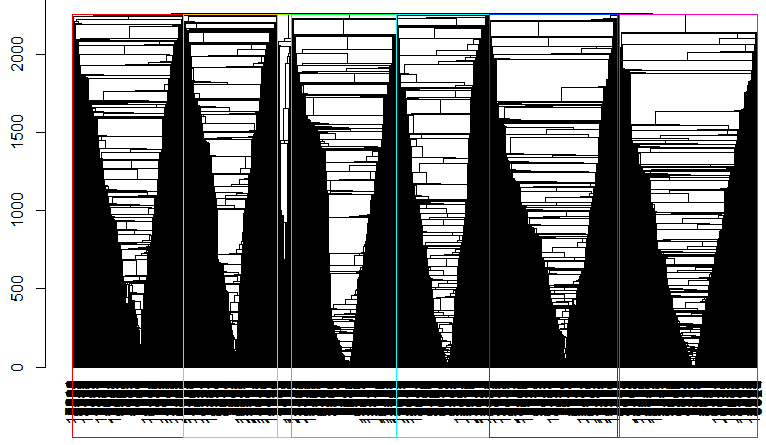


Figure 11-Community Structure DendPlot

In Figure 10 and 11, we observed the sub-community structure of the largest community.

**5. Find all the sub-community structures of the communities with sizes larger than 100.**

|  |
| --- |
| **[Code]** |
| # Problem 5  index\_large\_sub <- which(sizes(fg\_op2)>100)  for (i in index\_large\_sub) {  sub\_community\_network <- induced.subgraph(undirected\_g\_option2, which(membership(fg\_op2) == i))  fg\_sub\_community <- fastgreedy.community(sub\_community\_network)  title <- paste("Community ",i)  dendPlot(fg\_sub\_community)  plot(fg\_sub\_community,sub\_community\_network,  layout=layout.fruchterman.reingold,  vertex.size=3,  vertex.label=NA,  edge.arrow.size=.1,  main=title  )  } |

|  |  |
| --- | --- |
| C:\down\com 1 _plot.png | C:\down\com 1 _dend.png |

Figure 12-Community 1 Structure Plot & DendPlot

|  |  |
| --- | --- |
| C:\down\com 2 _plot.png | C:\down\com 2 _dend.png |

Figure 13-Community 2 Structure Plot & DendPlot

|  |  |
| --- | --- |
| C:\down\com 3 _plot.png | C:\down\com 3 _dend.png |

Figure 14-Community 3 Structure Plot & DendPlot

|  |  |
| --- | --- |
| C:\down\com 5 _plot.png | C:\down\com 5 _dend.png |

Figure 15-Community 5 Structure Plot & DendPlot

|  |  |
| --- | --- |
| C:\down\com 6 _plot.png | C:\down\com 6 _dend.png |

Figure 16-Community 6 Structure Plot & DendPlot

|  |  |
| --- | --- |
| C:\down\com 7 _plot.png | C:\down\com 7 _dend.png |

Figure 17-Community 7 Structure Plot & DendPlot

|  |  |
| --- | --- |
| C:\down\com 8 _plot.png | C:\down\com 8 _dend.png |

Figure 18-Community 8 Structure Plot & DendPlot

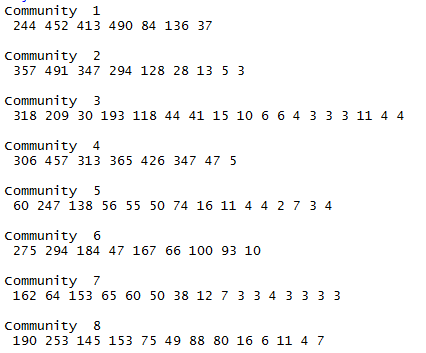


Figure 19-Sub-community sizes

**6. Both fastgreedy.community and label.propagation.community assume that each node belongs to only one community. In practice, a node can belong to two or more communities at the same time. There is no command in igraph that can detect overlapped communities. Here we are going to use personalized PageRank to study the overlapped communities structures.**

**Suppose that we start the random walk from a node i with the damping parameter 0:85 in the original directed network. We will obtain the visiting probabilities of all nodes in the network. These visiting probabilities reflect the relation between the other nodes and node i. On the other hand, we already have some knowledge of the community structure obtained from the above question. Therefore, we can compute**

**where vj is the visiting probability of node j and ~mj is its community membership computed from 3. Here, the community membership is expressed as a vector so that it can be adapted to multiple memberships. Suppose you have found n communities in 3, and ~mj is a n dimensional vector with only one element being 1. The resulting Mi is a measurement of the multiple memberships. To include all nodes in the network in the sum might make the computation amount too large. So you can take the largest 30 vj to include in the sum.**

**A threshold is needed to remove the memberships that have very small values in Mi. Choose a proper threshold.**

**Note that visit probabilities of the random walk started from a node can be obtained by a personalized pagerank with a teleportation probability distribution that is 1 to the node under examination and 0 to every other node.**

**Try this method to see whether you can find examples of nodes that belong to multiple communities. List at least 3 examples if you find such nodes.**

|  |
| --- |
| **[Code]** |
| # Problem 6  M\_prob <- matrix(ncol=2,nrow=vcount(GCCfinal))  for (v\_iter in 1:vcount(GCCfinal)) {  tele\_prob <- rep(0,vcount(GCCfinal))  tele\_prob[[v\_iter]] <- 1  ppr <- page.rank (GCCfinal, algo = "prpack", vids = V(GCCfinal), directed = TRUE, damping = 0.85, personalized = tele\_prob, weights = NULL, options = NULL)  sorted\_prob <- sort(ppr$vector,decreasing = TRUE)      M\_vector <- rep(0,max(fg\_op2$membership))    for (i in 1:30) {  m\_vector <- rep(0,max(fg\_op2$membership))  member <- fg\_op2$membership [[which(V(GCCfinal)$name == attributes(sorted\_prob)[[1]][i])]]  m\_vector[[member]] <- 1  v\_prob <- sorted\_prob[[i]]  M\_vector <- M\_vector + v\_prob\*m\_vector  }  M\_prob[v\_iter,1] <- sort(M\_vector, TRUE)[1]  M\_prob[v\_iter,2] <- sort(M\_vector, TRUE)[2]    print(v\_iter)  }  difference\_prob <- M\_prob[,1]-M\_prob[,2]  plot(difference\_prob)  for (low\_diff in 1:5) {  node\_id <- which(difference\_prob == sort(difference\_prob, FALSE)[low\_diff])    tele\_prob <- rep(0,vcount(GCCfinal))  tele\_prob[[node\_id]] <- 1  ppr <- page.rank (GCCfinal, algo = "prpack", vids = V(GCCfinal), directed = TRUE, damping = 0.85, personalized = tele\_prob, weights = NULL, options = NULL)  sorted\_prob <- sort(ppr$vector,decreasing = TRUE)      M\_vector <- rep(0,max(fg\_op2$membership))    for (i in 1:30) {  m\_vector <- rep(0,max(fg\_op2$membership))  member <- fg\_op2$membership [[which(V(GCCfinal)$name == attributes(sorted\_prob)[[1]][i])]]  m\_vector[[member]] <- 1  v\_prob <- sorted\_prob[[i]]  M\_vector <- M\_vector + v\_prob\*m\_vector  }    title <- paste("Node : ",node\_id," M-vector")  plot(M\_vector, main=title)    } |

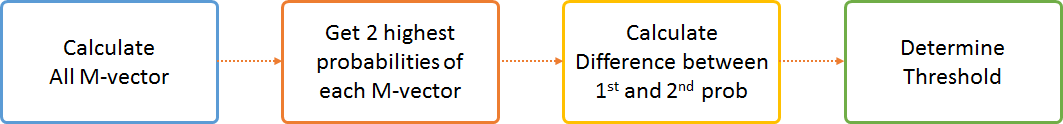


Figure 20-Process of determining threshold

For determining threshold, we created the process in Figure 20. First, we calculate M-vectors of all vertices. And get 2 highest probabilities of each M-vector. And then, we calculated difference between 1st and 2nd probabilities. And plot the graph of difference and determine which threshold is appropriate.

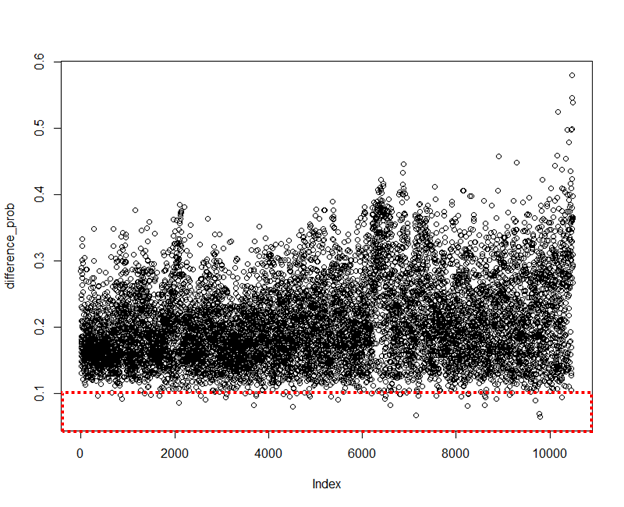


Figure 21-Difference of 1st and 2nd highest probabilities distribution

In Figure 21, we observed the distribution of probabilities differences. The reason why we plot the difference value is low difference means that the node can be belonging multiple communities with higher probability. So our interest of finding nodes which may be belonging multiple communities is a low probability area, red squared area in Figure 21. And we recalculate M-vector of 5 nodes which difference is the lowest.



Figure 22-Lowest 5 difference of probabilities

When we observed some M-vectors, we found that most of case which belonging to one community seem to have around 20-30% probability to one dominant community. So we think that if the node belongs to 2 or multiple communities, the probability will be divided. Therefore, we think that 0.1 is the appropriate threshold. We will check that our assumption is correct.

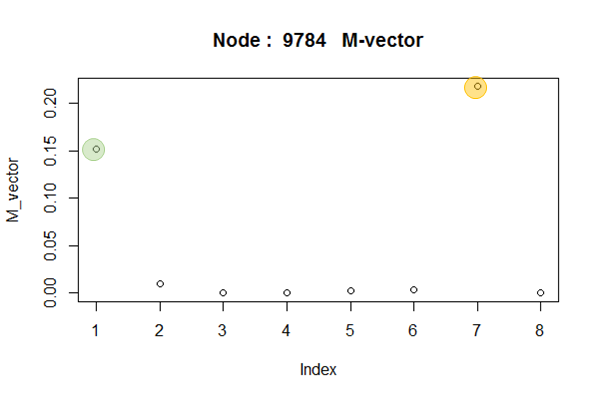


Figure 23-M-vector Node 9784

The first lowest difference case was node 9784. The highest probability was about 0.22 and the next highest probability was about 0.15. Rest of probabilities were almost close to 0. So we could determine this 9784 node may belongs to community 1 and 7.

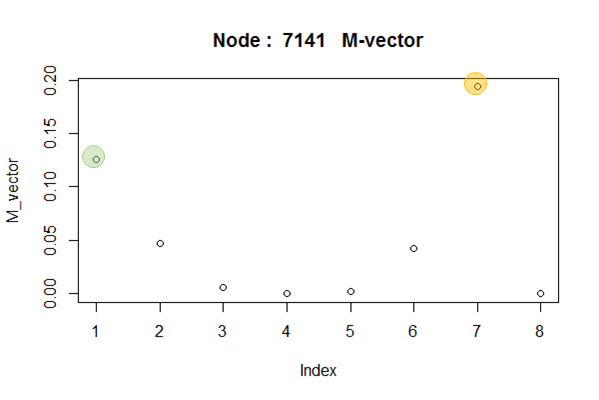


Figure 24-M-vector Node 7141

The second lowest difference case was node 7141. The highest probability was about 0.19 and the next highest probability was about 0.13. The probabilites of community 2 and 6 were about 0.04 and rest of probabilities were close to 0. So we could determine this 7141 node may belongs to community 1 and 7.

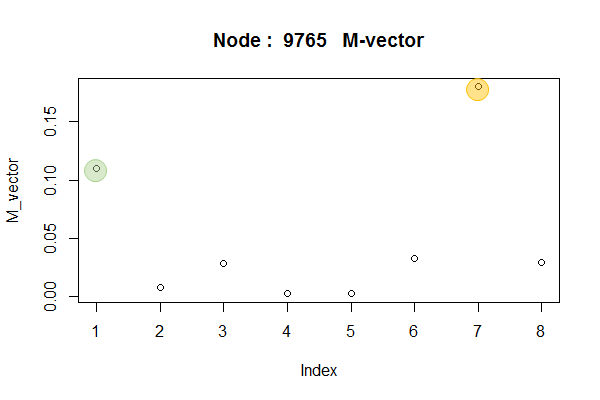


Figure 25-M-vector Node 9765

The third lowest difference case was node 9765. The highest probability was about 0.19 and the next highest probability was about 0.11. The probabilites of community 3, 6, and 8 were about 0.025 and rest of probabilities were close to 0. So we could determine this 9765 node may belongs to community 1 and 7.

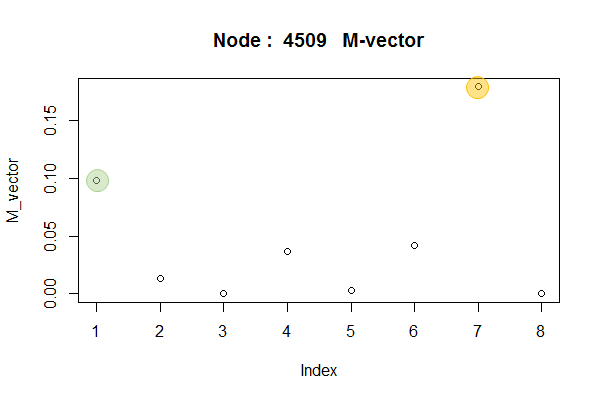


Figure 26-M-vector Node 4509

The fourth lowest difference case was node 4509. The highest probability was about 0.19 and the next highest probability was about 0.099. The probabilites of community 4 and 6 were about 0.04 and rest of probabilities were close to 0. So we could determine this 4509 node may belongs to community 1 and 7.

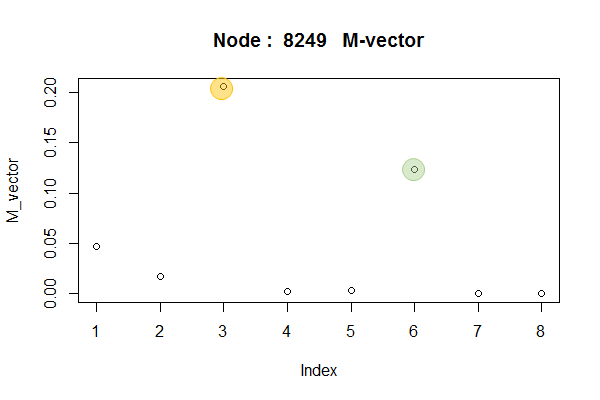


Figure 27-M-vector Node 8249

The fifth lowest difference case was node 8249. The highest probability was about 0.2 and the next highest probability was about 0.12. The probabilites of community 1 and 2 were about 0.049, 0.01 and rest of probabilities were close to 0. So we could determine this 8249 node may belongs to community 3 and 6.

From previous five examples, we observed that 4 cases except 4509 fourth case were satisfied the threshold 0.1 (10%) of our assumption. However, in the 4509 node case, the second highest probability was close to 0.1. So we think that we can find multiple communities with threshold 0.1. And the other interesting thing is that all examples seem to belong to two communities. We could not find any node which belongs to three or more communities.