Big Data Community Detection using Girvan-Newman Algorithm

By Group Five (Dex Hunter, Jing Wu, Siqi Yang)

I. Introduction

Modern computer science has become the backbone for other subjects since machines are capable of arduous calculations beyond human's scope. Another overwhelming exposed trend is so called "big data", which can be interpreted in various understandings. With the curiosity of exploring the connections between datasets, three students from Xi'an Jiaotong-Liverpool University and University of Electronics and Electrical Engineering determined to apply Girvan-Newman Algorithm to several given datasets and explore and visualize the communities underneath the hood.

II. Problem Statement

This project intends to explore the hidden connections in social network.

Explore the communities inside each datum. Visualize the clusters and analysis the graph.

III. Brief Description of the CD Algorithm used

The algorithm used to detect communities in this project is Girvan-Newman Algorithm. The idea behind this algorithm is first to calculate all the edge betweenness of all edges and then remove the edge with highest edge betweenness and repeat these two steps until desired amounts of clusters is formed[1].

IV. Complexity Analysis

The time complexity of Girvan-Newman algorithm is O(m^2n) where m indicates edges and n indicates vertices or O(n^3) on a sparse graph[2]. Thus, when the edge amount exceeds, the time to calculation will exponentially increase.

V. Experiment Setup and Datasets Used

There are four datasets given by RI2001A course, so we analysis the following files:

edges.csv small test graph, (6 nodes and 7 edges),

G1.txt network of FB account K, (?? nodes and 178 edges),

G2.txt network of FB account U, (?? nodes and 2390 edges),

G3.txt tweet graph from Assignment, (?? nodes and 260 edges).

The tool used in this project is a programming language called R, which is capable of processing large dataset.

VI. Result, Visualization and Analysis

Since the numbers of nodes for each file are missing, Hunter decided to first calculate the missing values. The results for each file are

50(G1.txt), 271(G2.txt), 503(G3.txt) nodes. The next step is to visualize each graphs as upcoming figures.

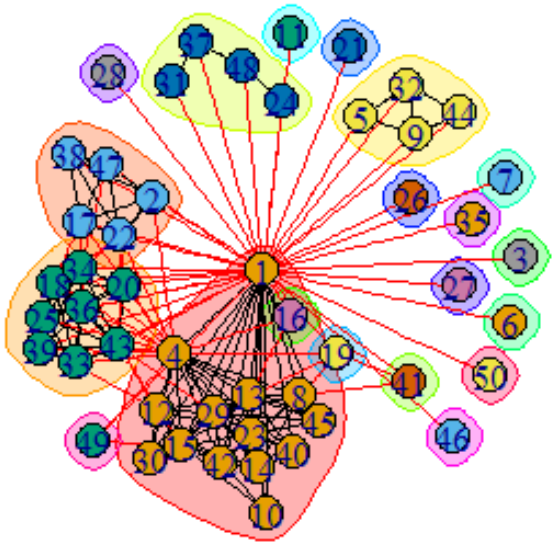

Figure 1

The visulisation for G1 file is shown above. From Figure 1, it is clear that there are 4 major communities while rest nodes are in their own separate clusters in this graph. Since there are only 50 nodes, the graph visualisation is also quite clear.

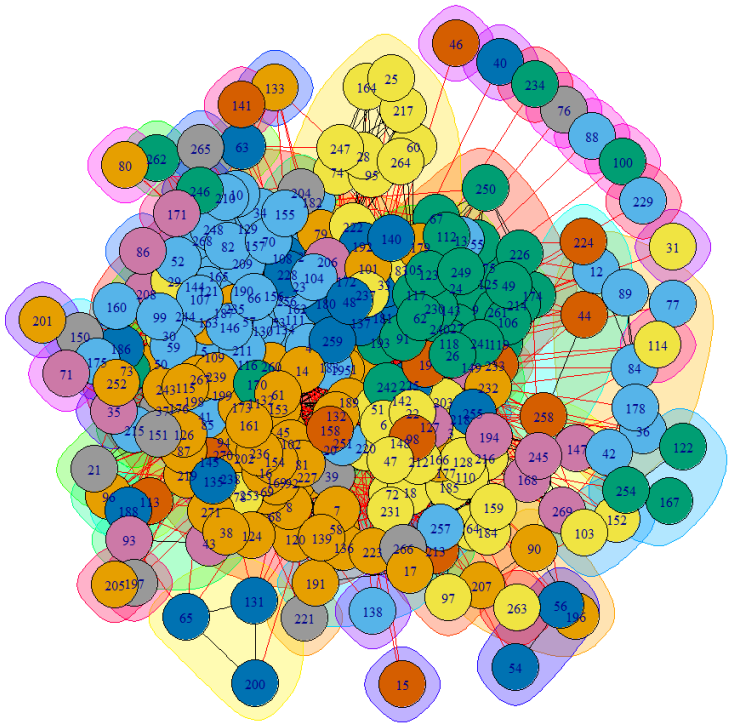

Figure 2

For the G2 file, since there are 271 nodes and 2390 edges, thus the genereated graph become a mess when we use default node size, but from Figure 3 we can see if we shrink the vertex size, then the pattern become more clearly. As with the G1 file case, there are several major communities in the middle while a lot more separate nodes in their own clusters.

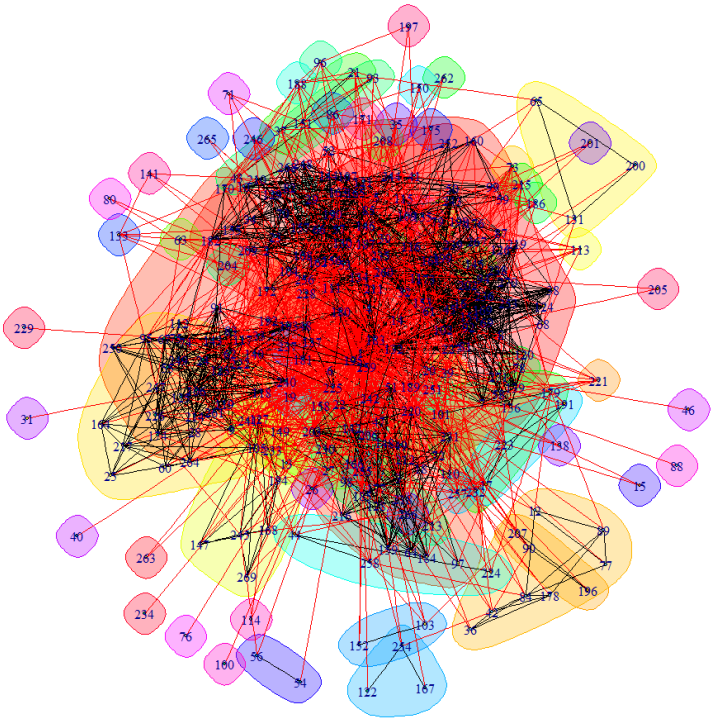

Figure 3

Visualisation for G1 and G2 are pretty straight forward while the upcoming graph showed us another possibility of community pattern.

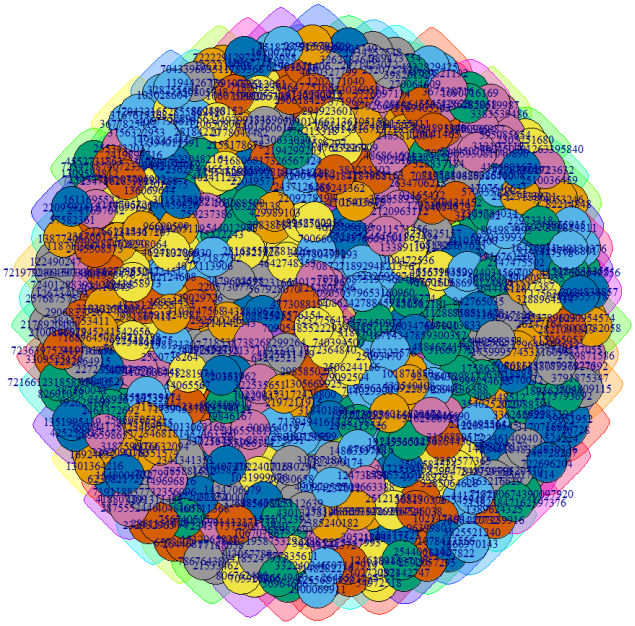

Figure 4

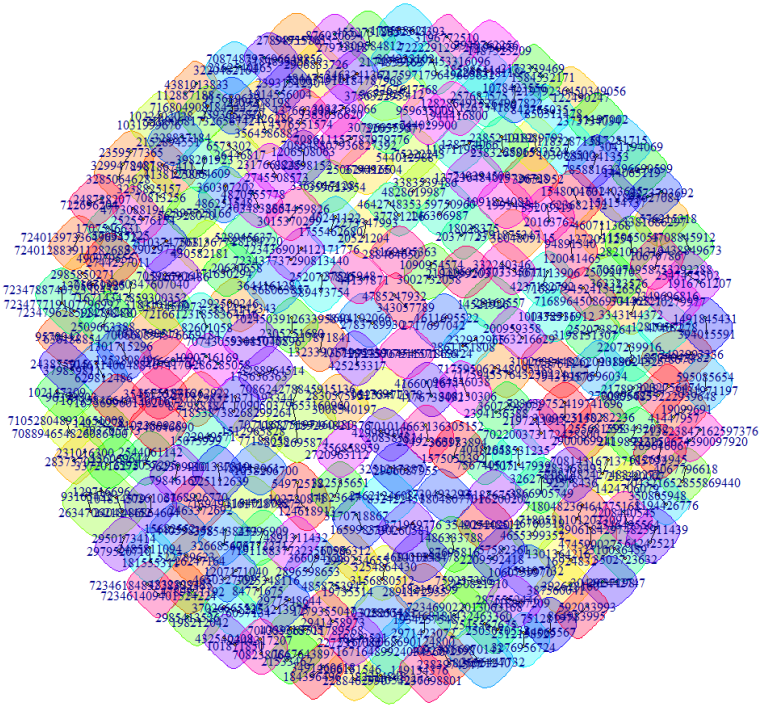

Figure 5

In the case of G3 file with 503 nodes and 260 edges, it actually can be inducted as a decentralized graph since the amount of nodes is almost the

twice size of its edges. Comparing to the previous two graphs where the amount of nodes is smaller than the amount of edges, the G3 file provides another pattern of communities where there is no central major community, instead scattered communities. Demonstrated by Figure 5, visualization of dataset confirmed the team's induction.

Since each dataset is small, Girvan-Newman edge betweenness algorithm is sufficient enough to solve the problem. However, as the dataset grows, the algorithm our team used would fail to process or it would take a tremendous amount of time to finish the graph. Therefore, we need to find alternative solutions when dataset is larger. There are several algorithms that have been implemented inside R are capable of finding communities in large network such as leading eigenvector, fast-greedy, multi-level, walktrap and label propagation.

VII. Conclusion

The dataset given by the module is not "big" enough to exceed the capacities of single laptop. The nodes and edges are few enough for graph tools to generate, while in some cases (e.g. CD1 assignment) the data given is about millions of rows which require a more efficient algorithm to calculate the communities otherwise the amount of time needed to finish calculation could be extremely large or exceed a single computer's capacities. The project gave us insight on how communities could possibly form in a large dataset

and how to use tools like R to manipulate and generate desired output data and visualization. Future studies that involve larger and more real-life-oriented dataset and various community detection algorithms could be implemented to examine the difference efficiency between data size and algorithms.

VIII. Lessons Learnt

The ideas of Kruskal's and Dijkstra's algorithms of finding Minimum Spanning Trees(MST) are taught and learned in the RI2001A module. Besides, community detection methods are applied through the implementation of R.

Teamwork in this project

Dex (leader of group) did most of R programming while Jing and Siqi wrote down the notes learned from the module and searched for better algorithms to solve the upcoming problems.

Acknowledgement

Many thanks to Prof Leong Hon-Wai and Prof Kal Ng for supervising this project and giving necessary support, and teaching assistants in this module. Without the help from them, our team could not have finished this project.

Reference

[1] T. Girvan, M. Girvan, M. Newman, T. Girvan, and T. Girvan, "Girvan – Newman algorithm," pp. 3–5, 2015.

[2] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 69, no. 6 2, pp. 279–307, 2004.

Appendix

R code used in this project:

1. Igraph visualization

```
ig <- function(df){
  a <- graph_from_data_frame(df, vertices=unique(c(df$V1, df$V2)), directed=F)
  b <- edge.betweenness.community (a, weights = E(a)$weight, directed = FALSE, edge.betweenness = TRUE, merges = TRUE, bridges = TRUE, modularity = TRUE, membership = TRUE)
  plot(b,a,vertex.size=3)
```

```
}
```

2. Pal system visualization

```
pg <- function(df){
  a <- graph_from_data_frame(df, vertices=unique(c(df$V1, df$V2)), directed=F)
  b <- edge.betweenness.community (a, weights = E(a)$weight, directed = FALSE, edge.betweenness = TRUE, merges = TRUE, bridges = TRUE, modularity = TRUE, membership = TRUE)
  edgs <- get.edgelist(a)
  clus <- data.frame(1:length(V(a)),b$membership)
  clus <- clus[unique(c(edgs[,1], edgs[,2])),]
  write.table(edgs, "D:/G.dat", sep="\t", row.names=F, col.names=F)
  write.table(clus, "D:/C.dat", sep="\t", row.names=F, col.names=F)
}
```

3. Counting the total number of nodes

```
tot <- function(data)
{
   return(length(unique(unlist(data[1:2]))))
```

```
}
```