ECE 232E Spring 2023

Social Network Mining

Facebook Network

Question 1

In this experiment the goal was to visualise the Facebook Social network and extract the following properties such as the number of nodes, edges, whether the graph is connected and the size of its Giant Connected Component (GCC). The visualisation of the graph can be seen in Fig. 1 and the properties of the graph are tabulated in Table.1

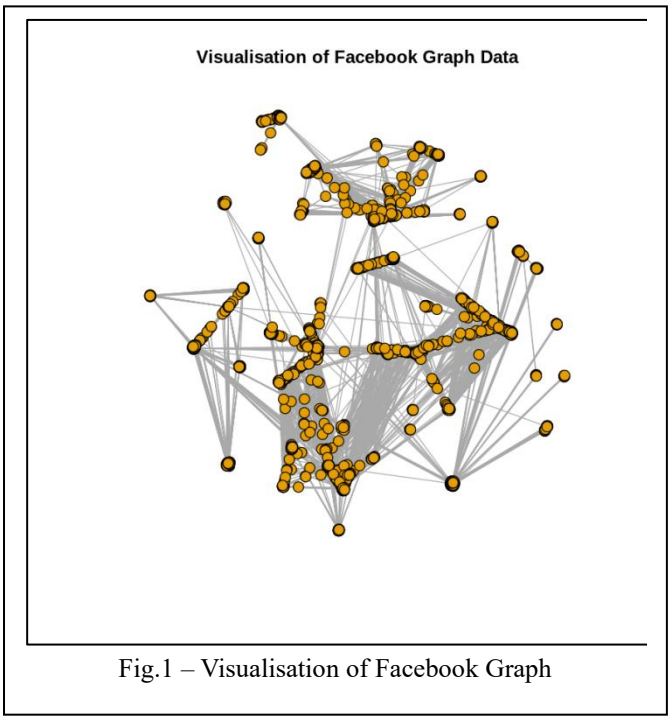

Fig.1 – Visualisation of Facebook Graph

Table 1: Properties of the Facebook Social Network Graph

Property	Value
Nodes	4039
Edges	88234
Is the Graph Connected	Yes
GCC Size	-

The number of nodes and edges accurately reflect the values shown on the website, and social networks are preferential attachment networks and as seen in the previous project, preferential attachment networks are always connected. Therefore, the existences of the graph's GCC and its size are not applicable since the entire network is connected.

Question 2

In this experiment, the goal is to find the diameter of the network and if the network is not fully connected, find the diameter of its GCC. From the previous question, we've seen that the network is fully connected and thus only the diameter of the Facebook network is to be found. After processing the graph, the diameter of the network was found to be 8, which signifies that any vertex is at max 8 hops aways from any other vertex within the graph.

Question 3

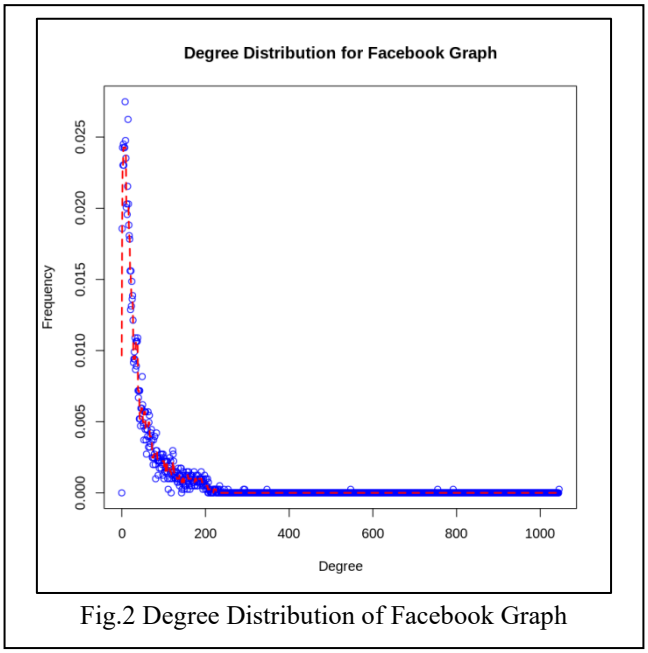

Fig.2 Degree Distribution of Facebook Graph

In this experiment, the goal is to plot the degree distribution of the graph and report the average degree of the graph. The visualisation of the degree distribution can be seen in Fig.2

As seen from the graph and from the previous project, the degree distribution of the Facebook graph follows that of the preferential attachment network and follows the power-law exponent. The equations describing the theoretical degree distribution are shown in equations (1) and equations (2). The equation describing the mean degree-distribution are shown in equation (3).

$$P(degree(v_i == k)) = \frac{c(k_{max},\gamma)}{k^{\gamma}} \quad (1)$$

$$c(k_{max},\gamma) = \frac{1}{\sum_{k=1}^{k_{max}} k^{\gamma}} \quad (2)$$

$$E(degree) = \frac{1}{m}\sum_{k=1}^{k_{max}} k \cdot N_k \quad (3)$$

From equation (1) and (2), we see that the degree-distribution follows a power-law exponent rule and $c(k_{max},\gamma)$ can be determined by summing over all possible values of k^{γ} , where k, represents the degree of a vertex and we then take the inverse of the sum.

From equation (3), we see that we are just summing over the product of k which is our desired degree of a vertex and N_k which is the number of nodes that have degree k. As the size of our graph m progresses towards infinity, the values of $\frac{N_k}{m} \rightarrow P_k$. The mean degree of the Facebook graph came out to 43.691

Question 4

In this experiment, the goal is to plot the degree distribution on a log-log scale and extract the parameters of the line-of best fit. The results for this are shown in Fig. 3. The parameters of the line of best fit are tabulated in Table 3.

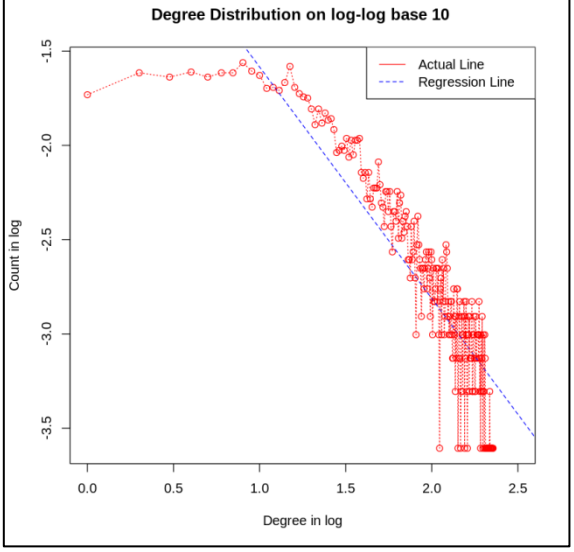

Fig.3 Degree Distribution on a Log-Log Scale

Table 3 Parameters of the Line of Best Fit For the Degree Distribution when plotted on log-log scale

Slope	-1.2279
Intercept	-0.3563

From equation (1) we can see that the probability of a node v_i having degree k is inversely proportional to k^{γ}, the relationship is shown in equation (4).

$$P(degree(v_i == k)) \sim \frac{1}{k^{\gamma}} \tag{4}$$

When plotted on a log-log scale the power-law exponent came out to be 1.2279 which means that $1<\gamma<2$. Because of this, the mean and variance of the graph are unbounded and instead of having a strong preferential attachment, the graph displays a weak-preferential attachment which means that instead of having a select-few nodes which are high-degree nodes, there are comparatively a larger amount of nodes that are high-degree nodes and this makes intuitive sense, since in a social network, nodes(people) tend to be friends with known nodes(people) and don't become friends with nodes that are unknown but of higher degree.

Question 5

In this experiment, the goal is develop a personalized network for ID “1” and report the number of nodes and edges present in the personalised network. The visualisation of the personalised network can be seen in Fig. 4 and the personalised graph has 348 vertices and 2866 edges.

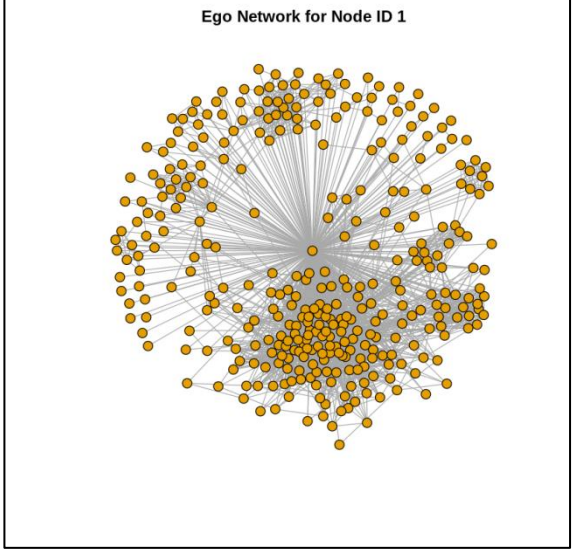

Fig.3 Personalised Network for Node ID 1

Question 6

Since the graph is a personalised network for Node ID 1, the graph would be connected via Node 1 in the worst case. The diameter of the graph came out to be 2. A trivial lower bound on the diameter would be 1 and a trivial upper bound on the diameter would be 2.

Question 7

In the graph, if the diameter is the lower bound it signifies that the graph is fully connected and each node is connected to every other node in the graph. The physical significance is that everyone knows each other in the friendship network.

If the diameter of the graph is the upper bound, it signifies that its not a fully-connected graph and while there may be some nodes that are connected to each other (Friends who know each other besides the central friend whose graph we developed), and there are nodes whose only common neighbour is the central node (Friends who only know one person in common).

Question 8

In this experiment, the goal is to find the number of core nodes and calculate the average degree of the core nodes. A core node is defined as a node which has more than 200 neighbours. In the Facebook graph, there are a total of 40 core nodes with an average degree of 279.375.

Question 9

In this experiment, the goal was to find the community structure for 5 different core nodes using 3 different clustering algorithms and to report their modularity as well. *Modularity* measures the strength of division into network modules and can range from -0.5 to 1, where the higher the value, the more "modular" the graph is or in simpler words, the easier it is to define communities inside the graph which have dense connections within the community but sparse connectivity to vertices outside of the community. Furthermore, for this question, the community structure was found using Newman's algorithm which works on maximising the Newman's modularity score and is shown in equation (5).

$$modularity(Cluster_i) = \sum_{i,j \epsilon Cluster_i} A_{ij} - \frac{k_i k_j}{2m} \quad (10)$$

A_{ij} is an element of the node-node incidence matrix and k_i/k_j are the degree of nodes inside $Cluster_i$. In the community-finding algorithm, the goal is to maximize the expected value of modularity over all clusters by iteratively updating the clusters by merging them if and only if it increases the total expected modularity of the graph. At the beginning the algorithm initialises all nodes to be their own cluster which are then iteratively merged to maximise the expected modularity of the graph. The modularity scores for the 5 different core nodes for each clustering algorithm are shown in Table 4. The community structures for all of the core nodes across the three different algorithms can be seen in Fig.4 – Fig. 118

Core Node	Fast-Greedy Algorithm	Edge-Betweeness Algorithm	Infomap Algorithm
1	0.413	0.353	0.389
108	0.436	0.507	0.508
349	0.250	0.134	0.095
484	0507	0.489	0.515
1087	0.146	0.028	0.027

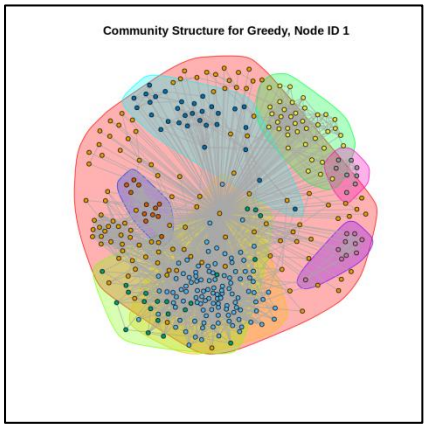

Fig.4 Community Structure for Node ID 1; Fast Greedy Algorithm

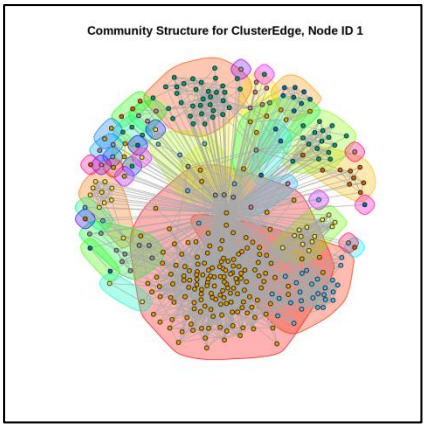

Fig.5 Community Structure for Node ID 1; EdgeBetweeness Algorithm

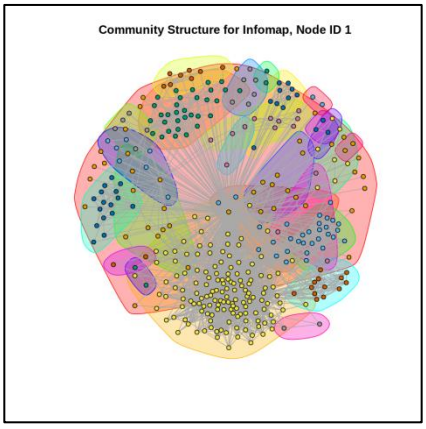

Fig.6 Community Structure for Node ID 1; Informap Algorithm

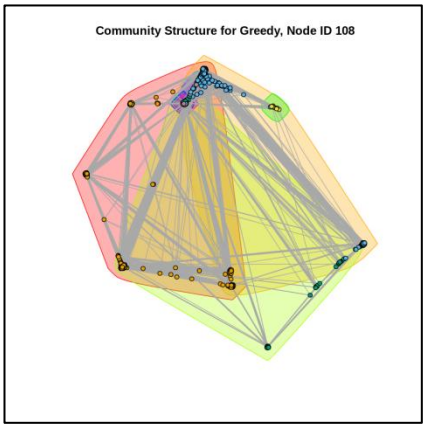

Fig.7 Community Structure for Node ID 108; Fast Greedy Algorithm

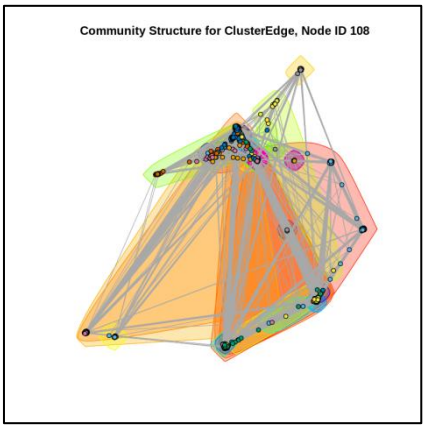

Fig.8 Community Structure for Node ID 108; EdgeBetweeness Algorithm

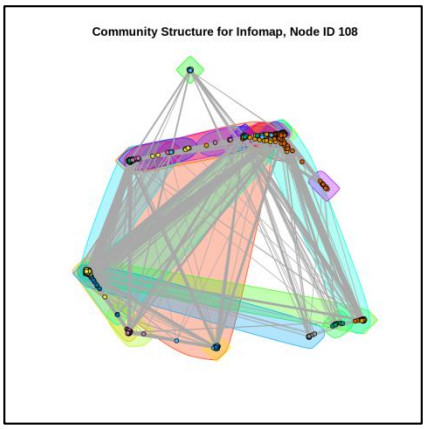

Fig.9 Community Structure for Node ID 108; Informap Algorithm

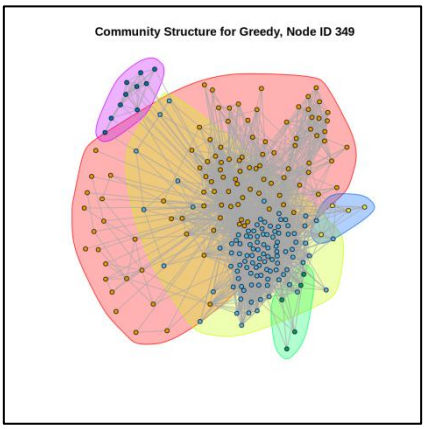

Fig.10 Community Structure for Node ID 349; Fast Greedy Algorithm

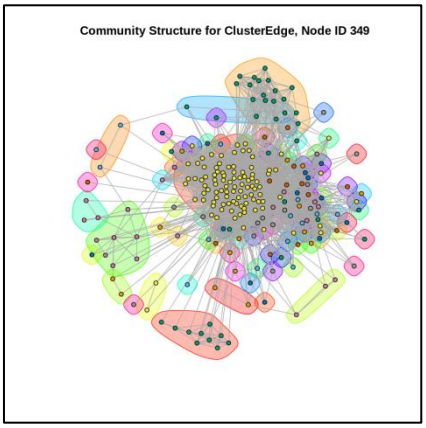

Fig.11 Community Structure for Node ID 349; EdgeBetweeness Algorithm

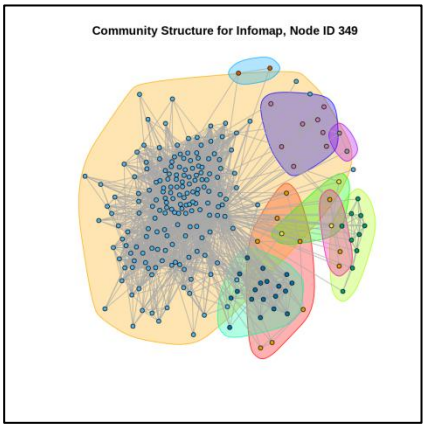

Fig.12 Community Structure for Node ID 349; Informap Algorithm

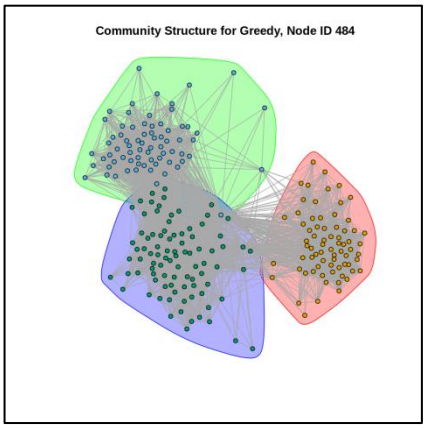

Fig.13 Community Structure for Node ID 484; Fast Greedy Algorithm

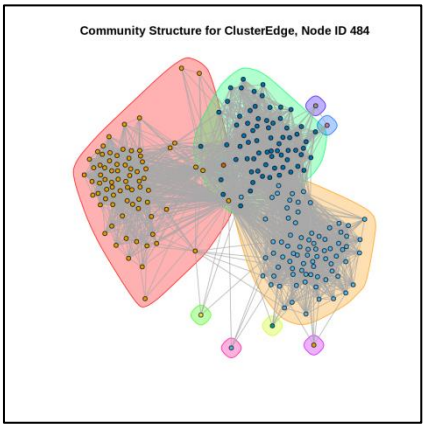

Fig.14 Community Structure for Node ID 484; EdgeBetweeness Algorithm

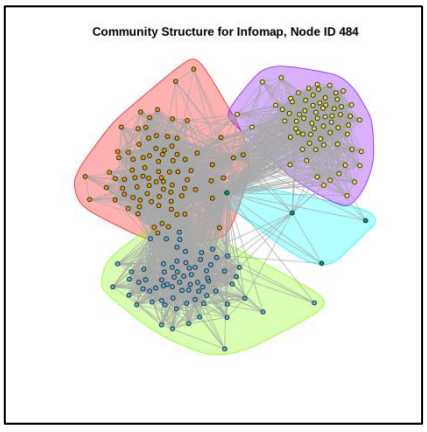

Fig.15 Community Structure for Node ID 484; Informap Algorithm

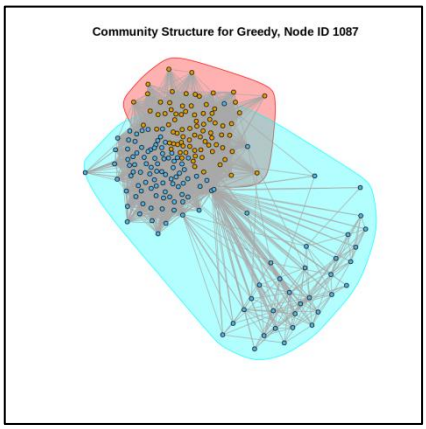

Fig.16 Community Structure for Node ID 1087; Fast Greedy Algorithm

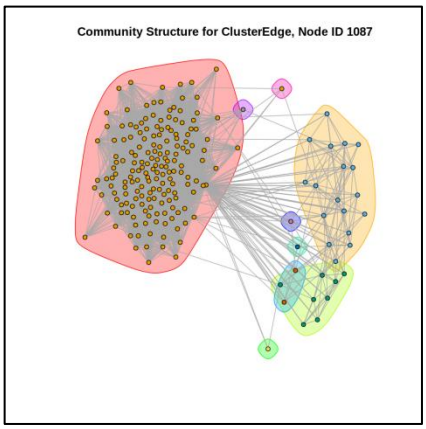

Fig.17 Community Structure for Node ID 1087; EdgeBetweeness Algorithm

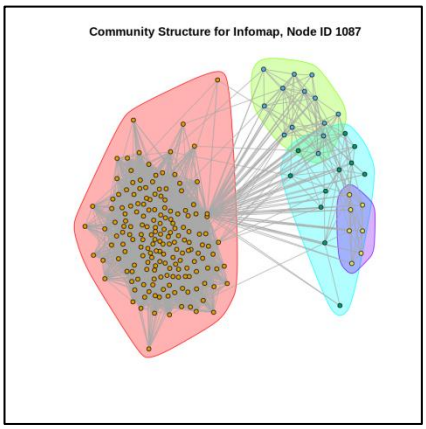

Fig.18 Community Structure for Node ID 1087; Informap Algorithm

The following conclusions can be made from Table 4 and from Fig.4 – Fig.18

1. Node 484 on average has the highest modularity scores when compared to the other core nodes. Node 108 comes in with a close second. When you look at both the nodes, they are highly modular which means that they have strong intra-community connectedness and poor inter-community connectedness.
2. Node 1087 has the lowest modularity score of all, while the pictorial representations show large modularity; which should intuitively mean a high modularity score, however the graph also displays strong inter-community connectivity which lowers the modularity score as seen in equation (10) if the product between $k_i\, k_j$ is high, the total modularity score will decrease, which is why the modularity for Node 1087 will decrease.
3. The performance of the edge-betweenness and infomap algorithm is comparable
4. The fast-greedy clustering algorithm tends to merge many clusters as seen in Fig.4-5, 10-11. In large networks and this is because of how the algorithm is implemented in the back-end with communities fewer than n nodes are merged into others and this is profound in the images cited above, especially large disconnected graphs where there are 'many' communities.
5. The fast greedy algorithm on average has the highest modularity score which is because it works by first creating individual communities and subsequently merging one into another iteratively till there is no increase in the modularity and since this is done in a greedy-manner, the modularity scores are higher than the other algorithms.
6. The edge-betweenness algorithm on average performs the worst, especially since the algorithm works by removing edges between vertices again and again until removing an edge no-longer makes a difference in the modularity score. This is a dual of the fast-greedy algorithm since it starts with a single community and iteratively removes one edge at time until the modularity score doesn't increase. The under-performance can arise from the fact that in a personal graph, there could be disconnected nodes or nodes with low degree which end up forming its own community (Fig.5-6).

Question 10

In this experiment, the goal was to repeat the previous experiment after removing the core node. The modularity scores for the 5 core nodes on the 3 different algorithms can be seen in Table 5 and can be visually inspected in Fig.19 – 33.

Table 5 – Modularity Scores of five different core nodes across three algorithmns

Core Node	Fast Greedy Algorithm		Edge Betweenness Algorithm		Infomap Algorithm	
	With core node	Without Core Node	With core node	Without Core Node	With core node	Without Core Node
1	0.413	0.442	0.353	0.416	0.389	0.418
108	0.436	0.458	0.507	0.521	0.508	0.521
349	0.250	0.246	0.134	0.151	0.095	0.247
484	0.507	0.534	0.489	0.515	0.515	0.543
1087	0.146	0.148	0.028	0.032	0.027	0.027

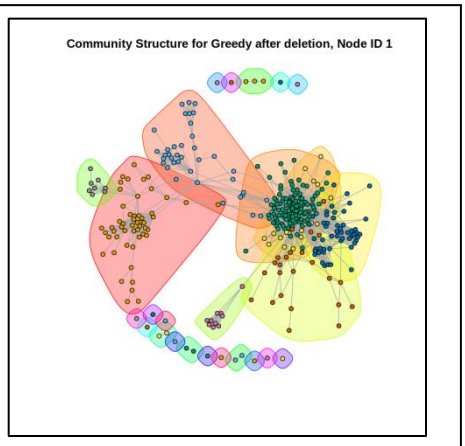

Fig.19 Community Structure for Node ID 1; Fast Greedy Algorithm

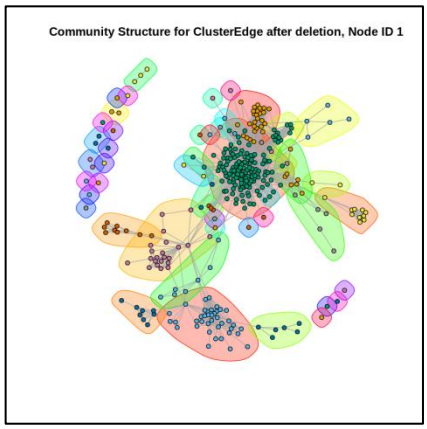

Fig.20 Community Structure for Node ID 1; EdgeBetweeness Algorithm

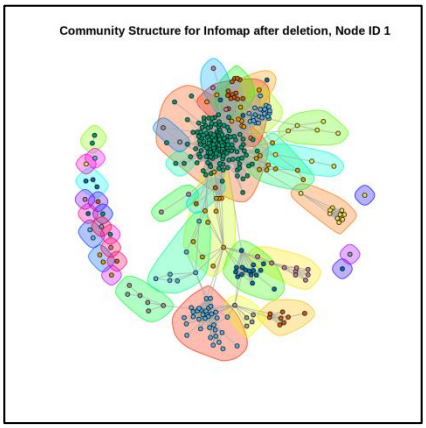

Fig.21 Community Structure for Node ID 1; Informap Algorithm

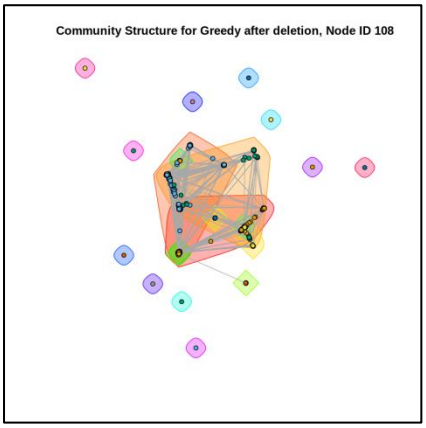

Fig.22 Community Structure for Node ID 108; Fast Greedy Algorithm

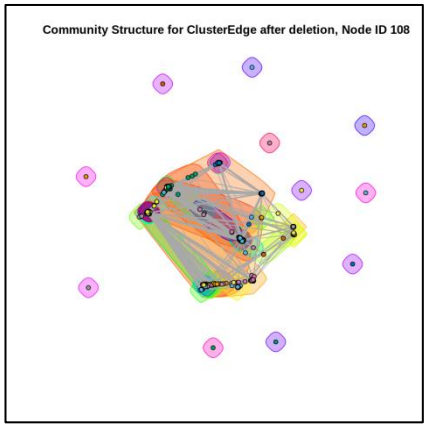

Fig.23 Community Structure for Node ID 108; EdgeBetweeness Algorithm

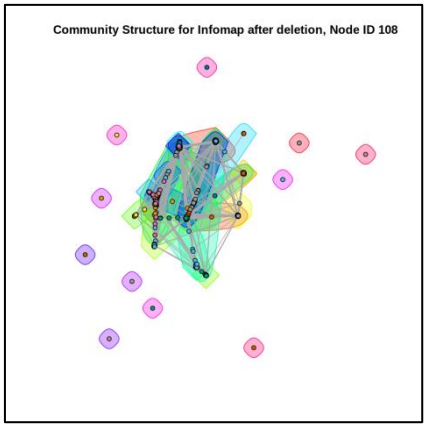

Fig.24 Community Structure for Node ID 108; Informap Algorithm

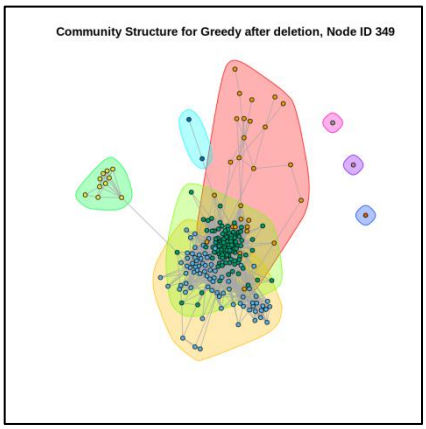

Fig.25 Community Structure for Node ID 349; Fast Greedy Algorithm

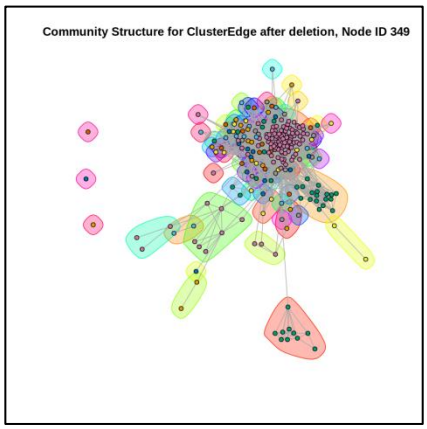

Fig.26 Community Structure for Node ID 349; EdgeBetweeness Algorithm

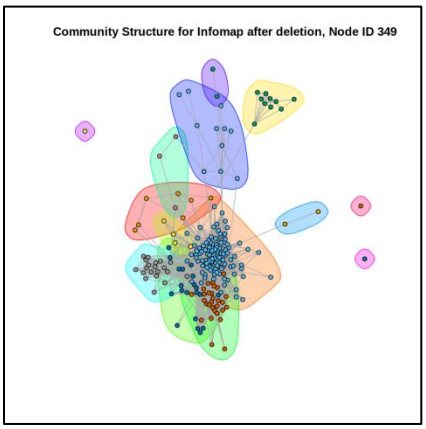

Fig.27 Community Structure for Node ID 349; Informap Algorithm

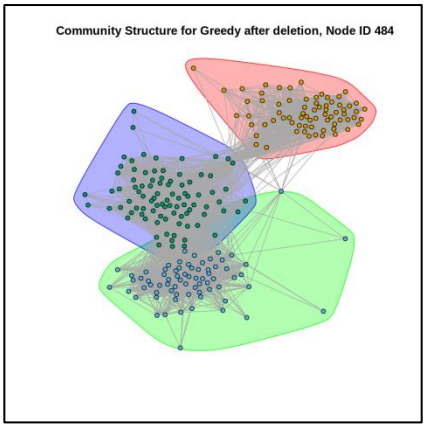

Fig.28 Community Structure for Node ID 484; Fast Greedy Algorithm

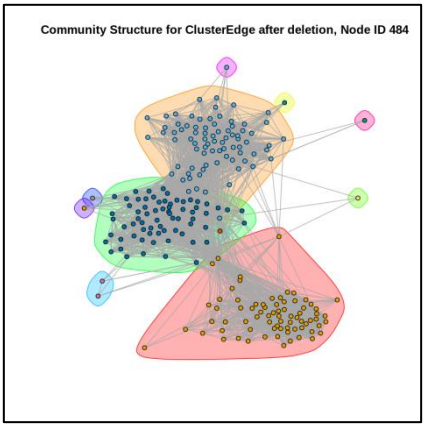

Fig.29 Community Structure for Node ID 484; EdgeBetweeness Algorithm

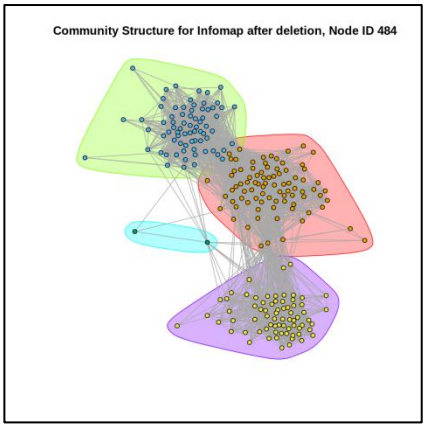

Fig.30 Community Structure for Node ID 484; Informap Algorithm

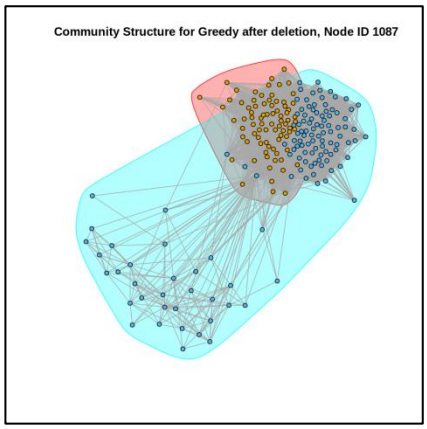

Fig.31 Community Structure for Node ID 1087; Fast Greedy Algorithm

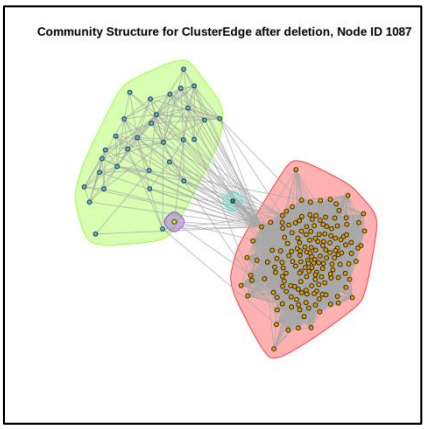

Fig.32 Community Structure for Node ID 1087; EdgeBetweeness Algorithm

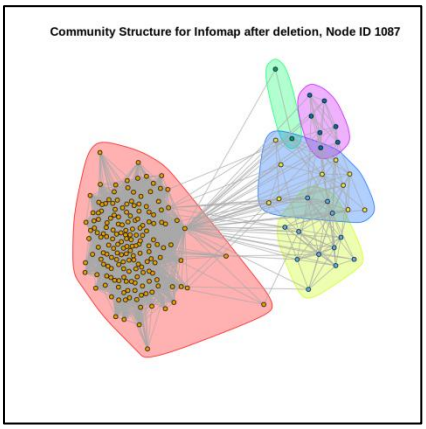

Fig.33 Community Structure for Node ID 1087; Informap Algorithm

After removing the core node, we can see from Table 5 the modularity scores have increased and that is visually seen in the graphs as there are now many more individual communities and this makes logical sense, since once the core node has been removed there are many disconnected communities with almost no existing connections to other graphs (Friends whose common friend was the core node). This

highlights the importance of the core node in the personalised graph. Once the core-node is remove, the clustering algorithms can create more communities since there are many disconnected vertices or vertices with weak connectivity when compared to the core node's connectivity. The fast-greedy algorithm still suffers from merging many individual nodes since communities with fewer members are generally merged into larger communities. Other algorithms such as edge-betweenness and infomap perform similarly.

Question 11

The relationship between the embeddedness of the core node and a non-core node is given by the following two equations. Inside the personalised network it is given by equation (11) and in the original graph it is given by equation (12). Embeddedness in a graph is a measure of how many neighbours do two pair of vertices share.

$$Embeddedness(v_i, v_c) = degree(v_i) - 1 \quad (11)$$

$$Embeddedness(v_i, v_c) = |deg(v_c) \cap deg(v_i)| \quad (12)$$

Question 12

In this experiment, the goal was to plot the histograms of dispersion and embeddedness for the five core nodes present. For ease of computing, equation (11) was used for computation, since the personalised network were created for earlier questions. *Dispersion* is defined as the sum of distances between every pair of mutual nodes, a non-core node shares with a core node. *Dispersion* is a metric used to measure how strong two nodes are connected by measuring the similarity in their neighbours, the more neighbours they connect to. The higher the dispersion score is, the more connected two nodes are. The equation used for calculating the dispersion in a personalised graph is shown in equation (13).

$$disp(u, v) = \sum_{a,b \in Intersection\ of\ (u,v)} distance(a, b) \quad (13)$$

Where u and v are any two nodes in the graph and the absolute dispersion is calculated by summing over all distances between the mutual neighbours of u and v. The visualisations of the embeddedness and dispersion are shown in Fig. 34 – 43.

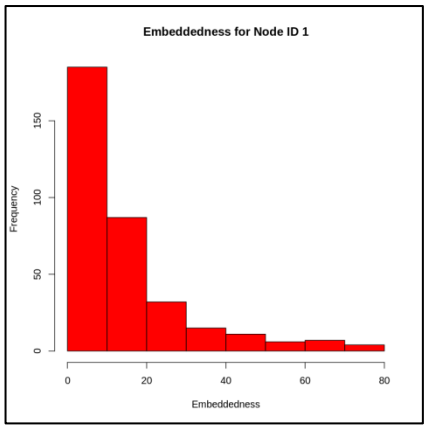

Fig.34 Embeddedness for Node ID 1

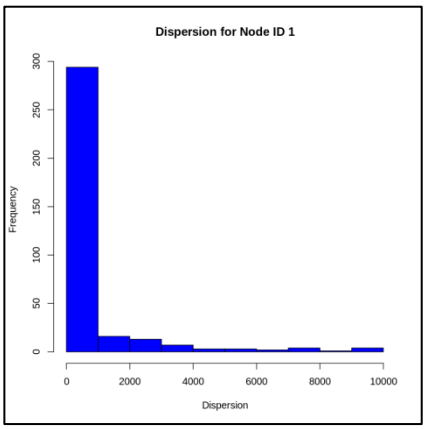

Fig.35 Dispersion for Node ID 1

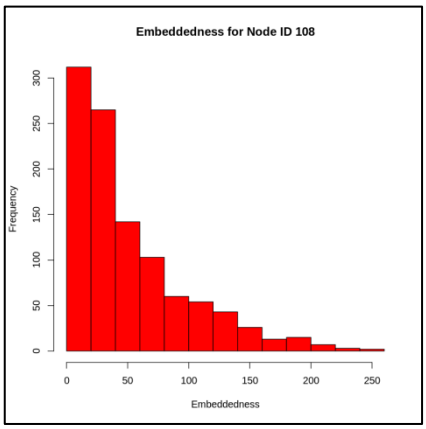

Fig.36 Embeddedness for Node ID 108

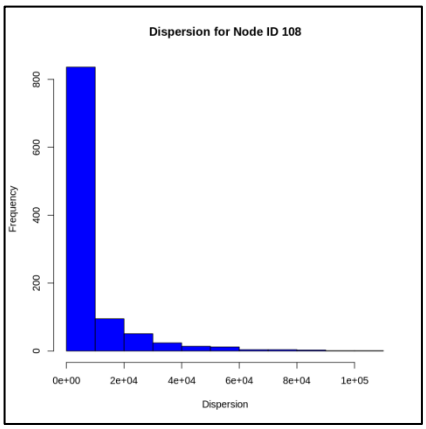

Fig.37 Dispersion for Node ID 108

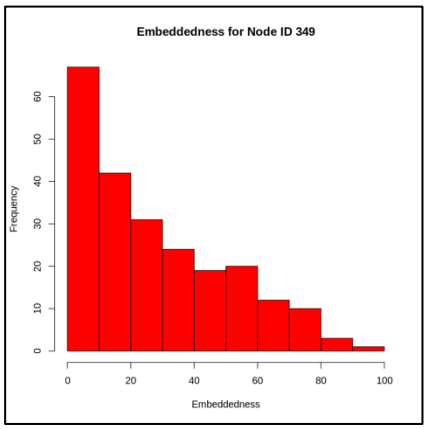

Fig.38 Embeddedness for Node ID 349

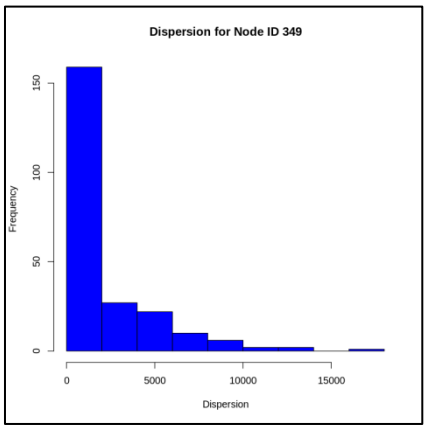

Fig.39 Dispersion for Node ID 349

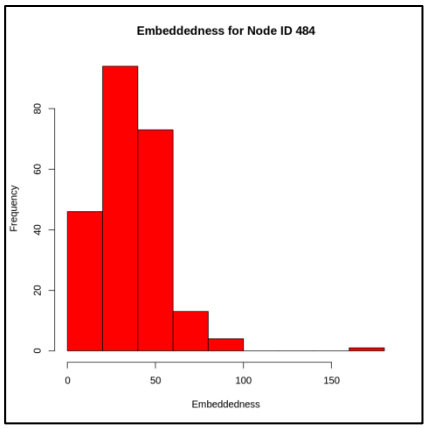

Fig.40 Embeddedness for Node ID 484

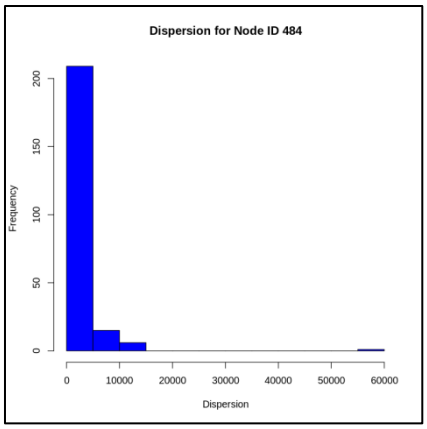

Fig.41 Dispersion for Node ID 484

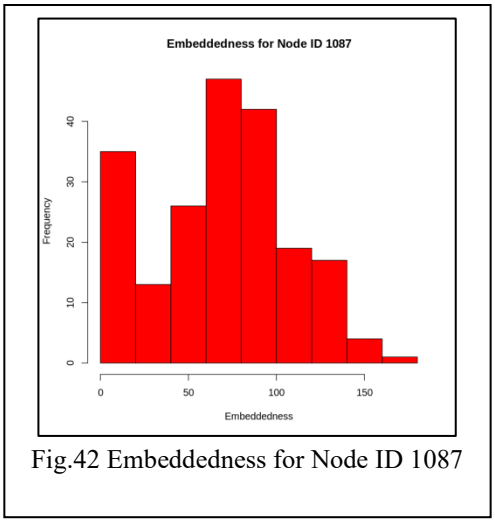

Fig.42 Embeddedness for Node ID 1087

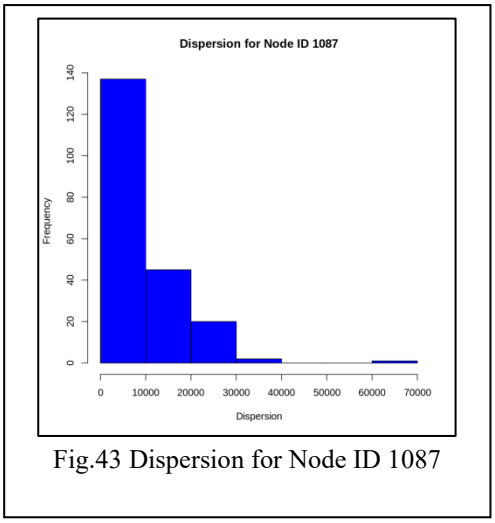

Fig.43 Dispersion for Node ID 1087

From the following graphs, we can make the following inferences

- Dispersion and Embeddedness for the most part interestingly follow a power-law distribution and the intuitive sense in both is that in both metrics, the underlying math in equations (11-13) rely on the degree of the node which is more obvious in the embeddedness. In dispersion, the larger the degree of a node is, the greater number of neighbours a node can have and potentially have a larger dispersion score. Thus, as we can see, both metrics seem to follow a power-law distribution as well.
- The range of both dispersion and embeddedness vary from each graph and this can be attributed to the size of the personalised network. Since a larger network can have a more diverse set of nodes with different degrees and as previously mentioned can thus lead to a dynamic range of dispersion and embeddedness scores.

Question 13

In this experiment, the goal is to plot the community structure for the five different core nodes and highlight the node with the maximum dispersion as well as highlight the incident edges to this node. This can be seen in Fig. 44-48.

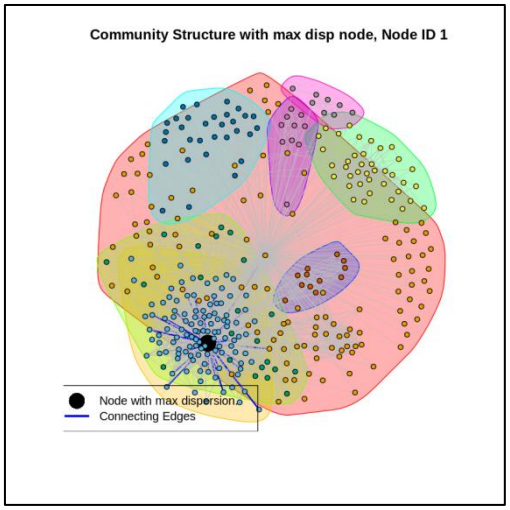

Fig. 44 – Community Structure for Node ID 1

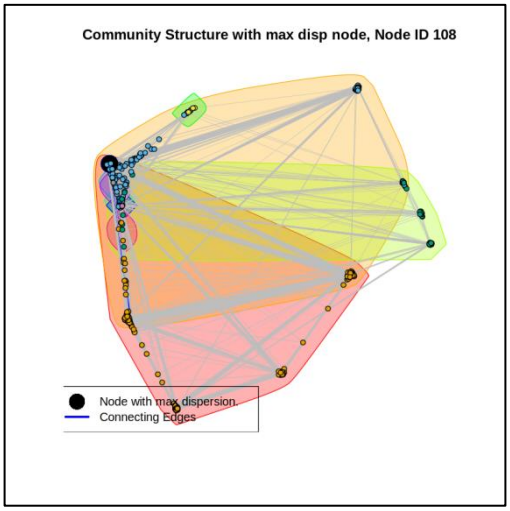

Fig. 45 – Community Structure for Node ID 108

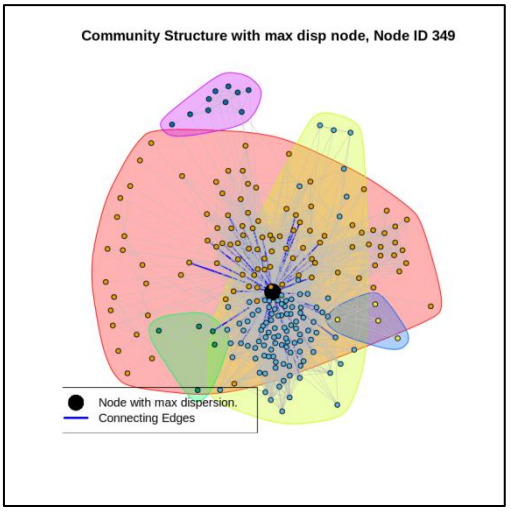

Fig. 46 – Community Structure for Node ID 349

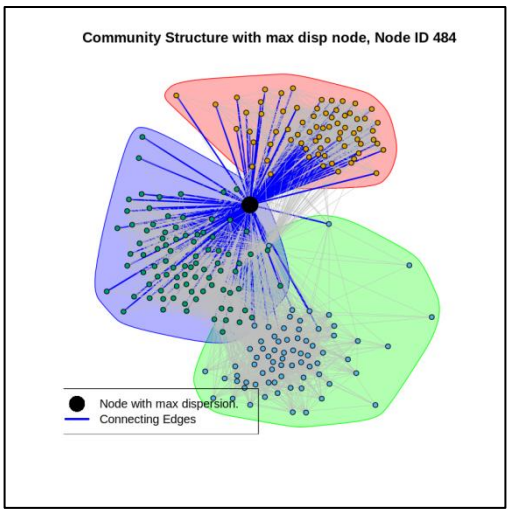

Fig. 47 – Community Structure for Node ID 484

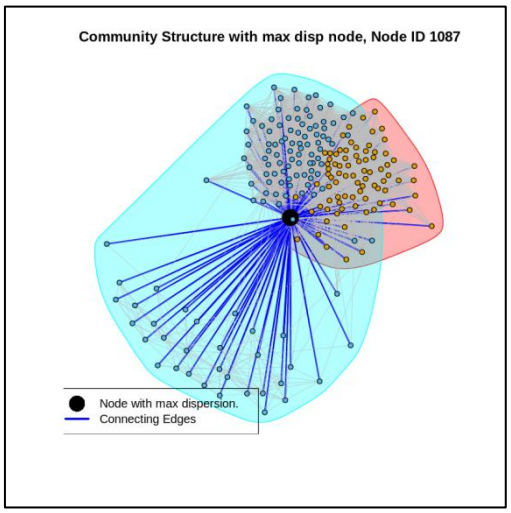

Fig. 48 – Community Structure for Node ID 1087

Question 14

In this experiment, the goal was to find the community structure for the personalised network created for each of the core nodes and similar to the previous question, highlight the node with the maximum embeddedness and maximum ratio of dispersion to embeddedness as well. To simplify visual inspection, both tasks were done separately and can be seen in Fig.49-58, furthermore the nodes that achieved the maximum embeddedness, dispersion and the ratio between the two are shown in Table 6.

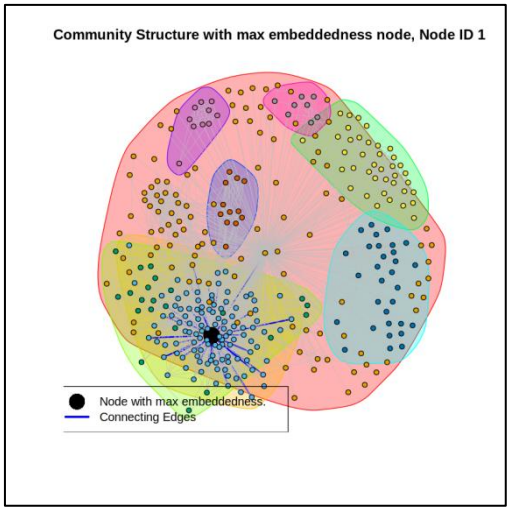

Fig. 49 – Community Structure for Node ID 1

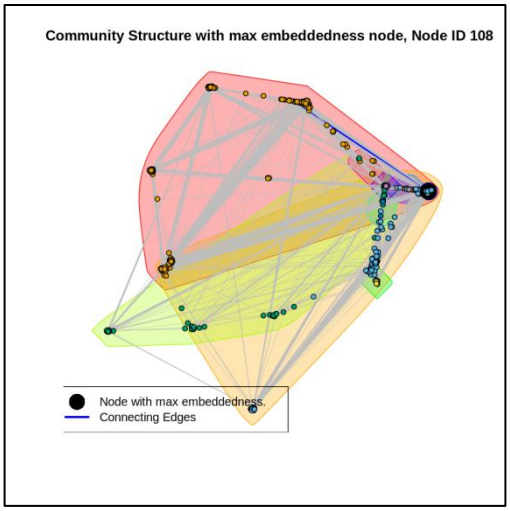

Fig. 50 – Community Structure for Node ID 108

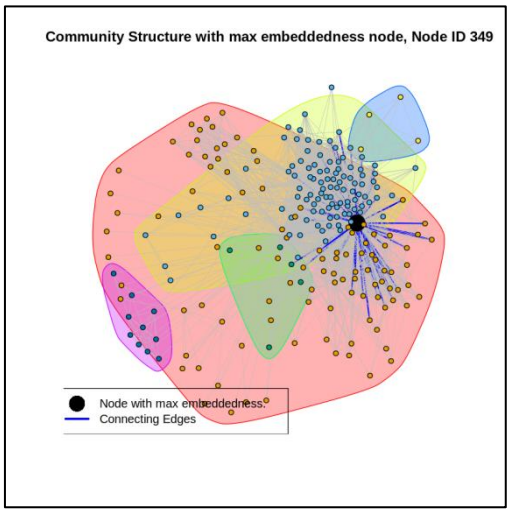

Fig. 51 – Community Structure for Node ID 349

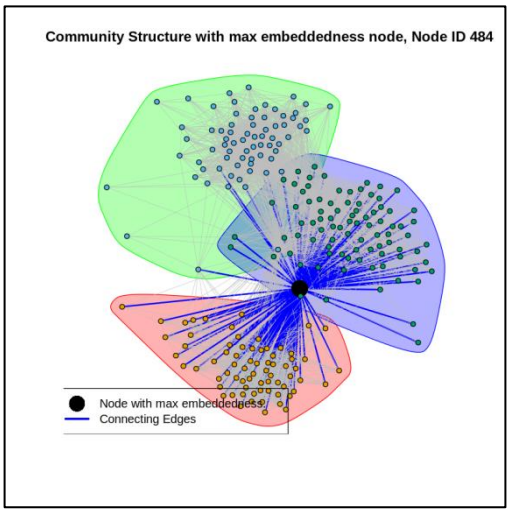

Fig. 52 – Community Structure for Node ID 484

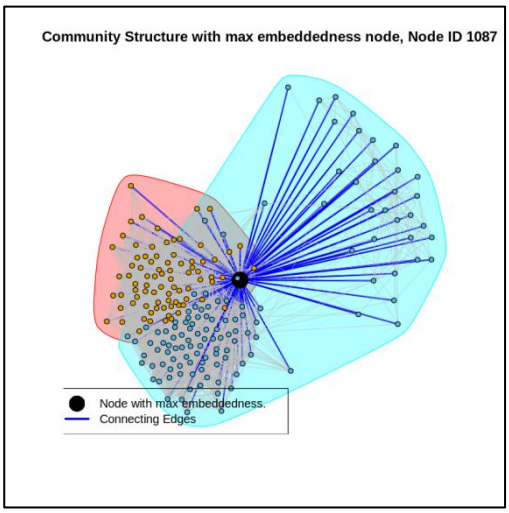

Fig. 53 – Community Structure for Node ID 1087

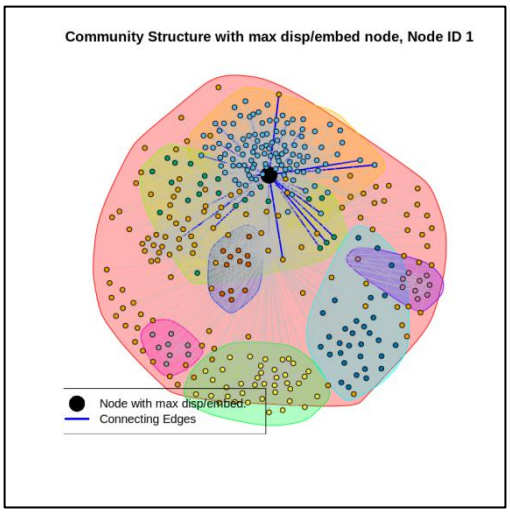

Fig. 54 – Community Structure for Node ID 1

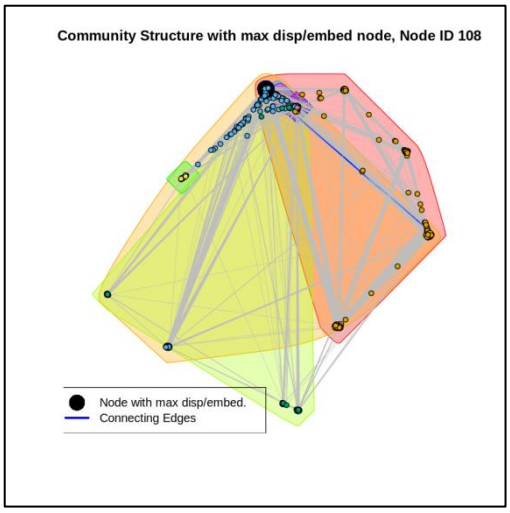

Fig. 55 – Community Structure for Node ID 108

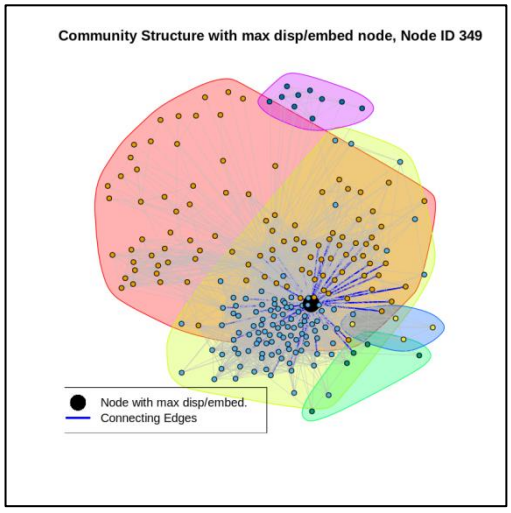

Fig. 56 – Community Structure for Node ID 349

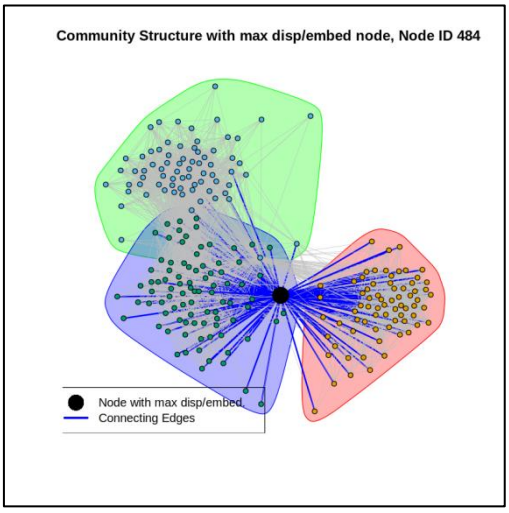

Fig. 57 – Community Structure for Node ID 484

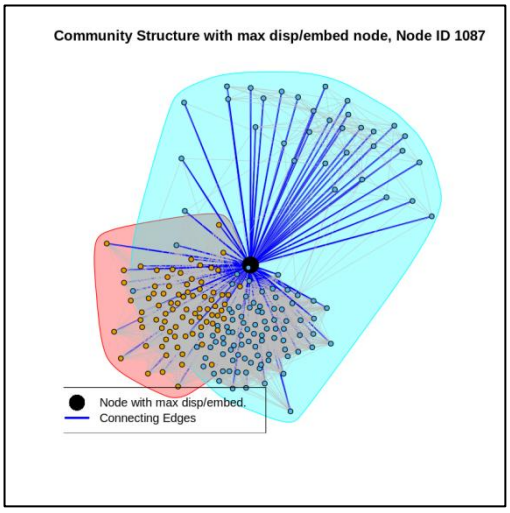

Fig. 58 – Community Structure for Node ID 1087

Table 6 – Vertices that acquire the maximum specified metric

Core Node	Maximum Dispersion	Maximum Embeddedness	Maximum Dispersion/Embeddedness
1	57	57	25
108	1889	1889	1889
349	377	377	377
484	108	108	108
1087	108	108	108

Question 15

From the last two questions we can make the following deductions:

1. Dispersion was implemented using equation(13) and what can be seen from Fig. 44-48 is that the node with the highest dispersion seems to be a node with many neighbours, and the intuitive sense is that between such a node and a core-node in the personalised network, their intersection would be large enough that when summing the distances over many neighbours can lead to large value. Dispersion also requires that the node has many neighbours interspersed through different communities in the personalised network to thereby increase the distances between the intersection. Dispersion measures the strength between two nodes by using the fact that two nodes have connections across many communities. The higher a dispersion score is, it means that two friends have more than intersection community in which they exist.
2. Embeddedness was implemented using equation(11) and what can be seen from Fig. 49- 53 and Table 6 is that the same set of nodes are the nodes with the maximum embeddedness as well and intuitively this does make sense since in embeddedness especially in a personalised network the node with the largest degree is the node with the maximum embeddedness. Embeddedness purely measures the degree of a node and it requires that a node have many neighbours or in terms of friendship network. The friend with the most friends besides the core-node is the node with the largest embeddedness.
3. Ratio of Dispersion to Embeddedness measures how dispersed is a node's neighbours in relation to how many neighbours does the node have. A node with a high ratio means that it has few friends spread across many different communities, since dispersion measures the distance between friends and embeddedness measures the neighbours. For a large value, the embeddedness needs to be low and dispersion needs to be high. For large networks this usually ends up reverting back to one of max dispersion/embeddedness node. For smaller networks, it finds a node whose dispersed friends are few but strongly connected.

Question 16

In this experiment, the goal was to find the number of neighbours Node 415 with exactly degree of 24. The length of N_r in this case came to 11.

Question 17

In this experiment, the goal is to compare three friendship recommendation algorithms by randomly deleting edges between vertices and seeing the algorithm recommends these edges back again. The accuracy of the algorithm was measured as the ratio between the intersection of recommended friends and deleted friends divided by the length of deleted friends. To make sure that the algorithm does not recommend more than it deleted, only t friends are recommended which is calculated by taking the length of the number of friends who were deleted in the first place. The results of all three algorithms are shown in Table 7.

Table 7 – Scores of all 3 friendship algorithms

Algorithm	Score
Common Neighbour	87.9%
Jaccard	85.1%
Adamic-Adar	88.2%

From Table 7, the Adamic-Adar got the highest score followed by the Common-Neighbour algorithm and the Jaccard algorithm scored the lowest of all three. However, with that being said, all three algorithms scored quite well, since the base accuracy came to be 85.1%, which means that in 100 predictions, the algorithm made 15 errors.

The Adamic-Adar algorithm given by equation (14). The Adamic-Adar algorithm works by accounting for the fact that a mutual friend with many other friends would be less likely to introduce each other while a mutual friend with fewer friends would be more likely to introduce each other.

$$Adamic\ Adar(i,j) = \sum_{k \in S_i \cap S_j} \frac{1}{\log(|S_k|)} \tag{14}$$

The common neighbour algorithm is given by equation (15). The common-neighbour algorithm is quite intuitive in the sense that if two nodes have many common friends, they are more likely to become friends as well.

$$Common\ Neighbour(i,j) = \left|S_i \cap S_j\right| \tag{15}$$

The Jaccard Algorithm works in between the two and is a statistical tool whose coefficient ranges from 0 to 1 and is given by equation (16). The Jaccard algorithm coefficient is high if the intersection and union are close to each other, that is the number of mutual friends is almost as big as the number of friends both together have, or in simpler words, it measures if both friends already run in similar social circles.

$$Jaccard(i,j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \tag{16}$$

The following conclusions can be made from the scores.

1. All three scores are close to each because the personalised graphs are quite small enough that there isn't much disparity between the scores, especially since node 415 isn't a core node, that means that the node's friends are weakly connected and don't exhibit strong connectedness as we did see in a core node.
2. Adamic-Adar and Common-Neighbour perform similar to each other, especially since the graph developed from Node 415 is a small graph, it is easy for both scores to follow a similar order.
3. Adamic-Adar and Jaccard, have close scores, but Jaccard underperforms since in cases like a disconnected node or when $\left|S_i \cup S_j\right| = 1$. The Adamic-Adar algorithm would still return 1 while the Jaccard core would come to 0, which means that the former still recommends it as a friend to add while the latter doesn't, and especially in a small graph where cutting a few edges can make a node disconnected from the latter, thus leading to Jaccard to perform worse. In large graphs, such a problem shouldn't occur especially since it would be rare to have disconnected nodes in the graph.

Google+ Network

In this section, we focus on the Google+ network and explore its properties using *Gplus* dataset.

Question 18

- In this question, we try to find out the number of personal networks, where personal networks are defined as the users (nodes in the graph) that have more than 2 circles. Where the circle is defined as a serial of nodes connected by directed edges. For example, the yellow node in Fig.59 has 2 circles (blue one and green one), so the yellow node represents a personal network.

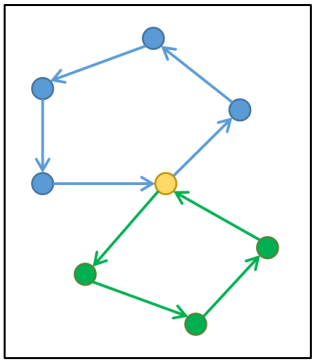

Fig.59 – A Example of Personal Network

- In *Gplus* dataset, there are **57** personal networks.

Question 19

- In this question, we investigate 3 specific networks (with node ID given below), and plot their in-degree and out-degree distribution in Fig. 60-65.
- Node ID 109327480479767108490
- Node ID 115625564993990145546
- Node ID 101373961279443806744

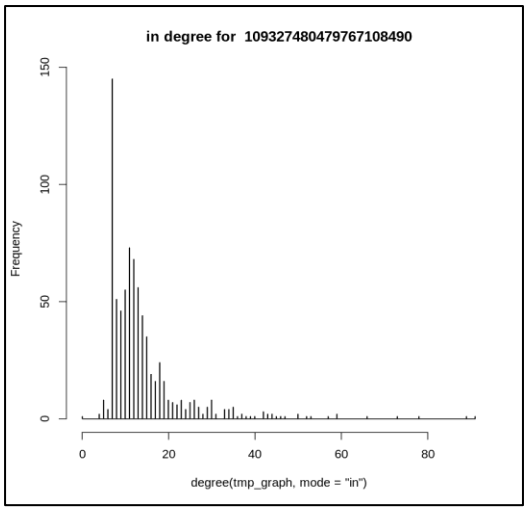

Fig.60 – In-Degree Distribution for Node ID 109327480479767108490

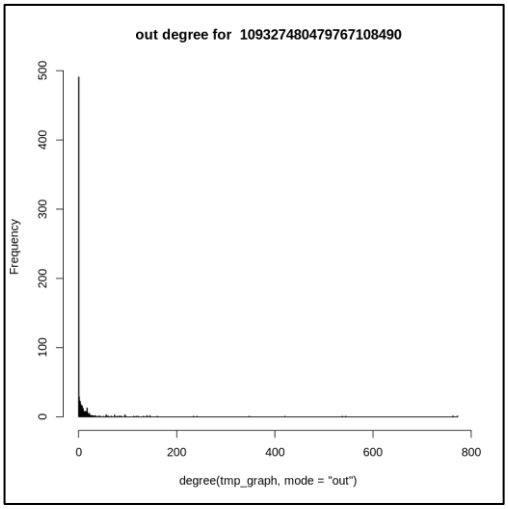

Fig.61 – Out-Degree Distribution for Node ID 109327480479767108490

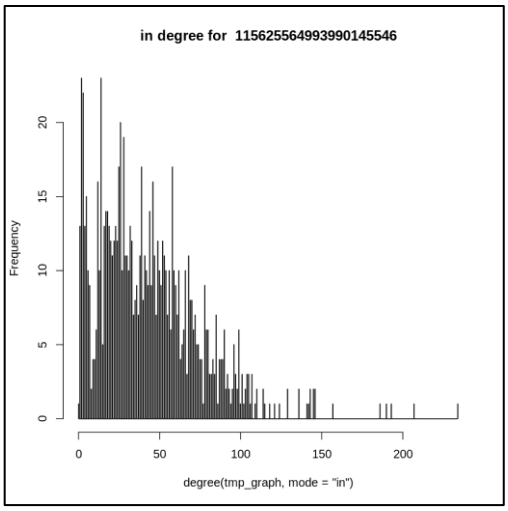

Fig.62 – In-Degree Distribution for Node ID 115625564993990145546

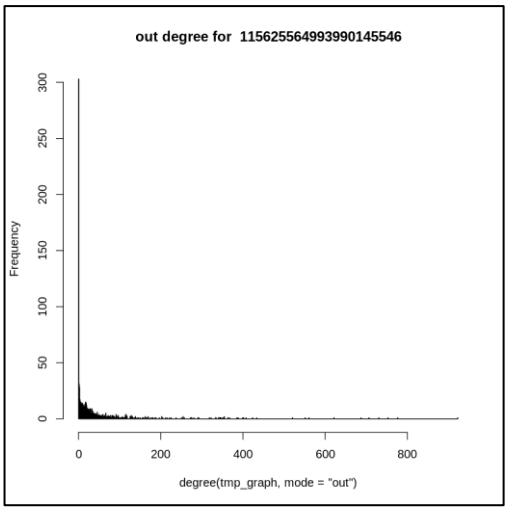

Fig.63 – Out-Degree Distribution for Node ID 115625564993990145546

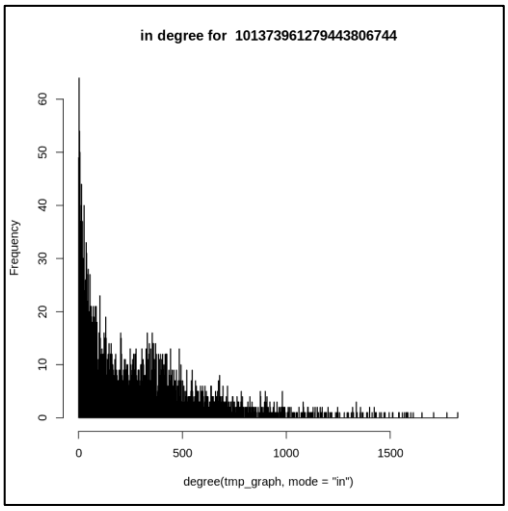

Fig.64 – In-Degree Distribution for Node ID 101373961279443806744

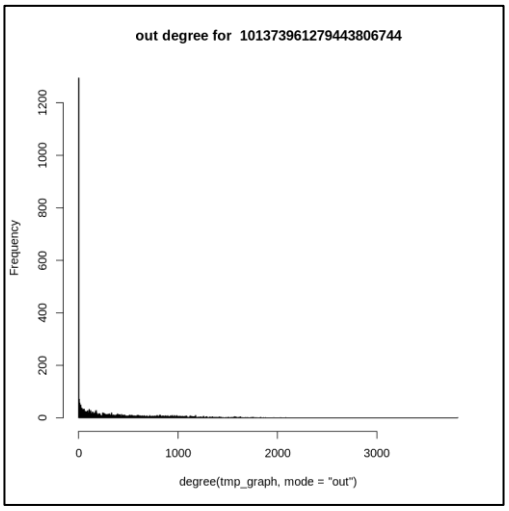

Fig.65 – Out-Degree Distribution for Node ID 101373961279443806744

- Based on the simulation results of the in-degree and out-degree distribution of these personal networks, we found that for in-degree distribution, these networks have similar (slightly different) distribution, which roughly follows power law distribution, and for out-degree distribution, these networks also have similar power law distribution.
- For each network, its in-degree and out-degree distribution is different, while both follow the power-law distribution.

Question 20

- In this question, we extract the community structures of these 3 personal networks using Walktrap community detection algorithm, which can be seen in Fig. 66-68.

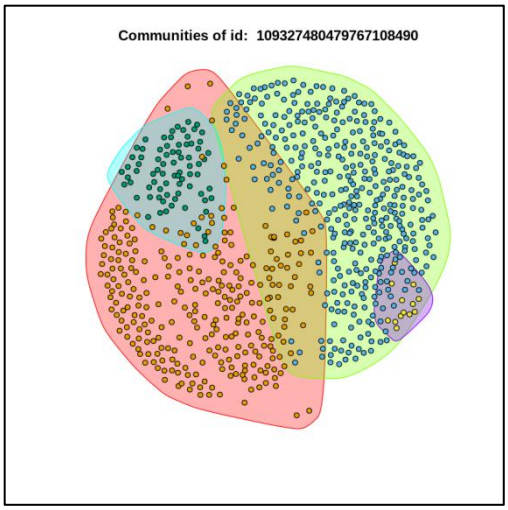

Fig.66 – Community Structures for Node ID 10932748047976710849

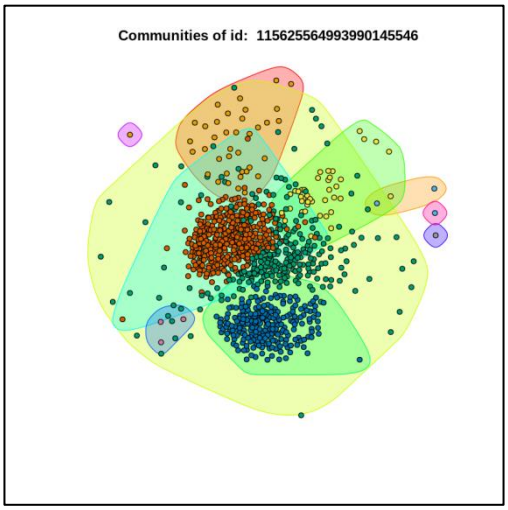

Fig.67 – Community Structures for Node ID 115625564993990145546

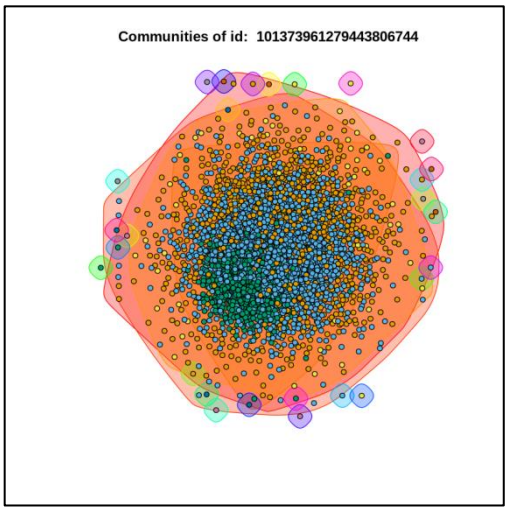

Fig.68 – In-Degree Distribution for Node ID 101373961279443806744

- Their modularity scores are summarized as below.

Table 8 – Modulariy Scores of Community Structures of 3 Personal Networks

Node ID	10932748047976710849	115625564993990145546	101373961279443806744
Modularity Score	0.2527	0.3194	0.1910

- We observed different modularity scores for these networks.
- A high modularity score indicates a clustering result that the nodes in a community are densely connected, while each community is sparsely connected to other communities. In other words, the

number of edges inside the communities are much more than that connecting the communities. 115625564993990145546 network has the highest modularity, and we can see in Fig.67 that there are many dense communities.

- A low modularity score indicates a clustering result that the nodes in a community are sparsely connected and the connections between communities are strong. In other words, the graph is has less characteristic of community structures, where there are many edges between nodes from a community and nodes from other communities. 101373961279443806744 network has the lowest modularity, and we can see the communities are strongly connected to each other in Fig.68.
- 115625564993990145546 network has intermediate modularity, indicating a moderate characteristic community structures.

Question 21

- In this question and the next question, we will explore the concept of homogeneity h and completeness c. Here we define some notations as follows.

- C is the set of circles, $C = \{C_1, C_2, C_3, ...\}$
- K is the set of communities, $K = \{K_1, K_2, K_3, ...\}$
- a_i is the number of people in circle C_i
- b_i is the number of people in community K_i with circle information
- N is the total number of people with circle information
- A_{ji} is the number of people belonging to community j and circle i

$$H(C) = -\sum_{i=1}^{|C|} \frac{a_i}{N} \log(\frac{a_i}{N}) \tag{17}$$

$$H(K) = -\sum_{i=1}^{|K|} \frac{b_i}{N} \log(\frac{b_i}{N}) \tag{18}$$

$$H(C \mid K) = -\sum_{j=1}^{|K|} \sum_{i=1}^{|C|} \frac{A_{ji}}{N} \log(\frac{A_{ji}}{b_j}) \tag{19}$$

$$H(K \mid C) = -\sum_{i=1}^{|C|} \sum_{j=1}^{|K|} \frac{A_{ji}}{N} \log(\frac{A_{ji}}{a_i}) \tag{20}$$

- And finally we state the expression for homogeneity, h as

$$h = 1 - \frac{H(C \mid K)}{H(C)} \tag{21}$$

- and the expression for completeness, c as

$$c = 1 - \frac{H(K \mid C)}{H(K)} \tag{22}$$

- Homogeneity h is defined to measure the degree of a clustering method that all data belonging to a cluster should have the same class.
- Completeness c is defined to measure the degree of a clustering method that all data with the same class should be contained in a same cluster.

Question 22

- In this question, we compute the h and c values for the community structures of the 3 personal networks, which are summarized in Table 9.

Table 9 – Homogeneity h and Completeness c of Community Structures of 3 Personal Networks

Node ID	10932748047976710849 0	11562556499399014554 6	10137396127944380674 4
Homogeneity *h*	0.8849	0.7970	-0.6792
Completeness *c*	0.5088	-0.1283	-1.2595

- For node ID 109327480479767108490, we observed the highest and positive h and c, indicating that the clustering method behaves better on this network than the other two. Still, the h and c are smaller than one, indicating that there are some nodes of different circles are clustered into a single community ($h < 0$), and there are some nodes of same circles that are clustered into a different community ($c < 1$), which can be seen in Fig. 66.
- For node ID 115625564993990145546, we observed a smaller but positive h and a negative c. Smaller h means that the homogeneity of this node is worse than that of 109327480479767108490. A negative c means that $H(K|C) > H(K)$, which may occur when there are some nodes that do not belong to any communities or some communities have few nodes. In Fig.67, there are some communities that only has one node, which indicates a circumstance that $c < 0$.
- For node ID 101373961279443806744, we observed negative h and c. For h, it is negative if $H(C|K) > H(C)$, which may happen when the graph is difficult or almost unable to do clustering, where many nodes of different circles are clustered into one community. For negative c, it is similar to network 115625564993990145546, where there are communities containing few nodes.

Cora Dataset

- In this section, we explore the classification problem of Cora dataset using different learning algorithms. Cora dataset contains 2708 papers related to Machine Learning with 7 different classes (Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning,Rule Learning, Theory). Each paper contains its own feature of 1433 words in the vocabulary. And the papers are related to each other by citation, forming a graph where the nodes represent the papers and the edges represent the citations.

Question 23

- In this question, we use Graph Convolutional Networks (GCN, [1]) to classify the 2708 documents to 7 classes. The features for learning of each document is a vector (length equals 1433) containing the occurrences of 1433 words in the vocabulary, as well as the adjacent matrix for the overall graph.
- We use a GCN with two *GCNConv* layers, each of them has 16 channels and a dropout rate of 0.5, and the L2 regularizer is set to be 2.5×10^{-4}. For training, we set the learning rate to be 0.01, and we train the model with 200 epochs. We have 140 nodes for training, and 2568 for testing. The hyperparameters are summarized in Table 10.

Table 10 – Hyperparameters for Classification

No. Layers	2	**Learning Rate**	0.01
No. Channels	16	**No. Epochs**	200
Drop-out Rate	0.5	**No. Training Set**	140
L2 Regularizer	2.5×10^{-4}	**No. Testing Set**	2568

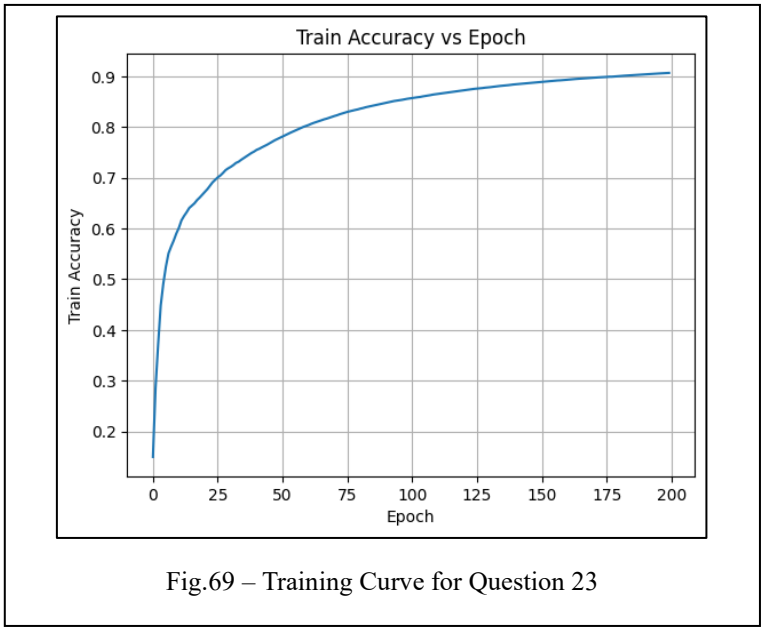

Fig.69 – Training Curve for Question 23

- The final test accuracy is **0.72**.

Question 24

- In this question, we use the same classifier (neural network) but 3 different features to do the classification, which are:

- Node2Vec Features
- Text Features
- Combined Features of the above two

- In order to get Node2Vec features, first we perform random walks on the graph to get sentences information, where each sentence contains the list of visited nodes of random walks. Then we convert the sentences to embedding vectors for each node using Word2Vec method.
- The Text features are 1433-dimensional text features.
  The Combined features are concatenated data of Node2Vec features and Text features.

- We choose neural network (MLP) as our classifier, and the training hyperparameters are summarized in Table 11.

Table 11 – Hyperparameters of Classifier

No. Layers	3	**L2 Regularizer**	2.5×10^{-4}
No. Channels @1	1024	**Learning Rate**	0.01
No. Channels @2	256	**No. Epochs**	100
No. Channels @3	64	**No. Training Set**	140
No. Labels	7	**No. Testing Set**	2568
Drop-out Rate	0.5		

- The final test accuracy of the three methods are summarized in Table 12.

Table 12 – Test Accuracy of the Three Methods

Features Used	Node2Vec Features	Text Features	Combined Features
Test Accuracy	0.5803	0.5944	0.6059

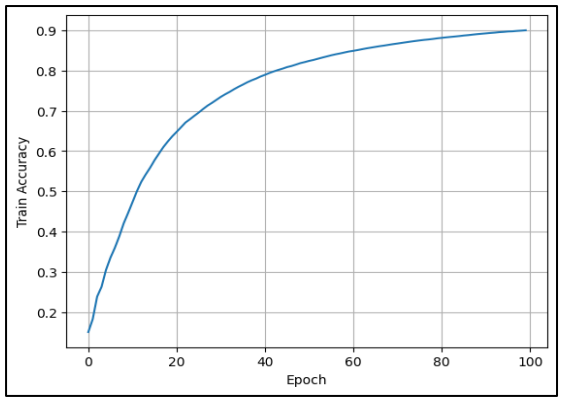

Fig.70 – Training Curve for Method with Node2Vec Features

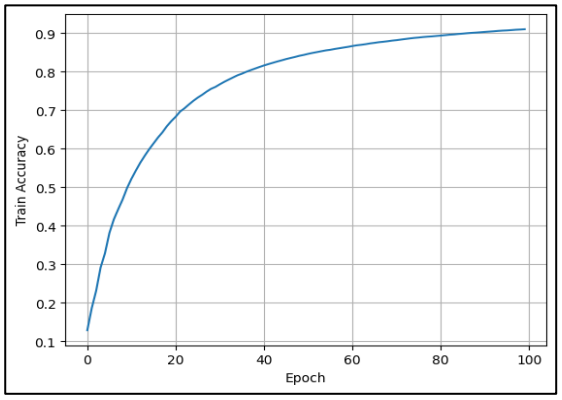

Fig.71 – Training Curve for Method with Text Features

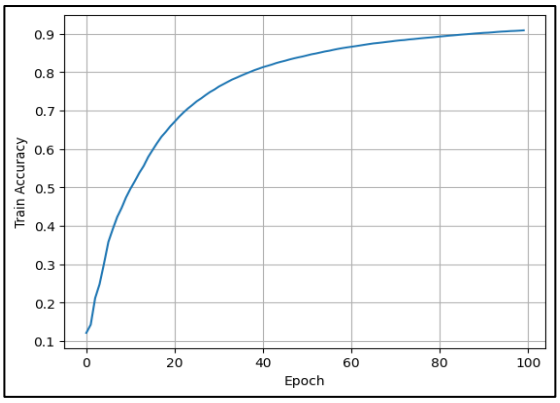

Fig.72 – Training Curve for Method with Combined Features

- Comparing method using Node2Vec features and method using Text features, we found the latter one outperforms by a small amount. We thought this is because there are more information in Text features. Comparing the three methods, the combined one achieved highest accuracy also because it contains more features data.

Question 25

- In this question, we run the PageRank to classify the each nodes. In order to do that, we do random walks on the GCC to get the PageRank of each node, then the predicted class for each node is the class that leads to maximum visits to that node.
- In the experiment, we set different strategies for random walks:

- Strategy 1: no teleportation, uniform probability of transitioning to each neighbor.
- Strategy 2: teleportation probability is 0.1, uniform probability of transitioning to each neighbor.
- Strategy 3: teleportation probability is 0.2, uniform probability of transitioning to each neighbor.
- Strategy 4: no teleportation, probability of transitioning to neighbors given by (23).
- Strategy 5: teleportation probability is 0.1, probability of transitioning to neighbors given by (23).
- Strategy 6: teleportation probability is 0.2, probability of transitioning to neighbors given by (23).

- In strategies 4,5,6, the probability of transitioning to neighbors is not uniform among the neighbors. Rather, it is proportional to the cosine similarity between the text features of the current node and the next neighboring node. Particularly, assume we are currently visiting a document x0 which has neighbors x_1, x_2, x_3. Then the probability of transitioning to each neighbor is:

$$p_i = \frac{\exp(x_0 \cdot x_i)}{\exp(x_0 \cdot x_1) + \exp(x_0 \cdot x_2) + \exp(x_0 \cdot x_3)}; \quad for \quad i = 1,2,3 \tag{23}$$

- The simulation results of the above methods are summarized as follows:

Table 13 – Test Accuracy of the Three Methods

	Teleportation Probability *p*	**Use TFIDF?**	**Accuracy**	**F1 Score**
Strategy 1	0	NO	0.71	0.66
Strategy 2	0.1	NO	0.75	0.71
Strategy 3	0.2	NO	0.74	0.69
Strategy 4	0	YES	0.59	0.55
Strategy 5	0.1	YES	0.73	0.69
Strategy 6	0.2	YES	0.72	0.66

- Based on the results, we found that in terms of teleportation probability p, better accuracy and f1 score are achieved when $p = 1$. Comparing Strategy 1-3 and Strategy 4-6, we found that better accuracy and f1 score are achieved when the probability of transitioning to neighbors is uniform.

Reference

[1] Kipf, Thomas N., and Max Welling. “Semi-supervised classification with graph convolutional networks.” arXiv preprint arXiv:1609.02907 (2016).