# 1. Objectives

The project is mainly about graph feature analysis on both facebook and google-plus social networks. Using igraph to extract graph information like community structures, core personal networks. Furthermore, by extracting features in communities and personal networks, we could analyze romantic relationship or to determine the type of communities. In general, the project is pretty tricky and meaningful for graph and network study.

# 2. Problems

## 2.1. Problem 1

We construct the graph from the edge list file using function read.graph(), the graph is connected and the diameter is 8. The average degree is 43.69. Figure 1.1 shows the distribution of degree of this graph.

As for the fitting curve, we use the stat\_smooth() function to generate the statistics model. After studying the shape of the curve, we decide to try two models, which are y ~ I(1/x\*a) + b\*x and y ~ I(exp(1)^(a + b \* x). And as we can see from figure 1.2, the curve of y ~ I(exp(1)^(a + b \* x) fits better.

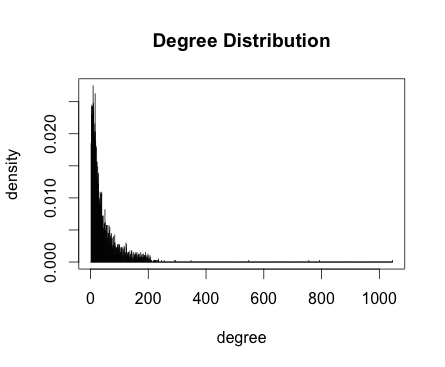


Figure1.1 Degree Distribution of generated graph



Figure1.2 Fitting curves with different models

The parameters and related statistics of this model is shown in table 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | estimated | Std Err | t value | Pr(>|t|) |
| a | -3.5940045 | 0.0078770 | -456.26 | <2e-16 |
| b | -0.0291488 | 0.003247 | -89.77 | <2e-16 |
| Residual standard error | | | 0.0006339 | |
| Number of iterations to convergence | | | 15 | |
| Mean Square Error | | | 1.4460696678752e-06 | |

Table 1.1 The summary of fitting model

## 2.2 Problem 2

We generate a sub-graph with node 1 and its neighbors, we can see from figure 2.1 that all the nodes (except for node 1) share a mutual friend node 1. The total edges of this personal network are 2866 and the total nodes are 348.

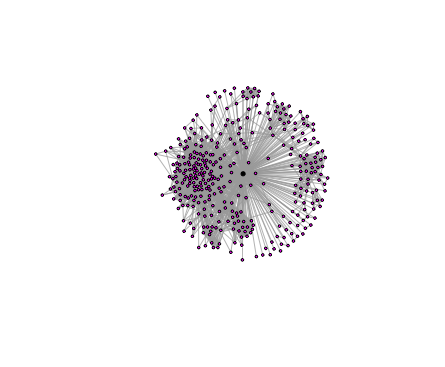


figure 2.1 personal network of node 1

## 2.3 Problem 3

We found 40 such core nodes in the graph and the average degree of such nodes are 379.375. We explored node 1, which is a core node, in details.

Figure 3.1 shows the community structure of personal network 1 using fast-greedy algorithm; figure 3.2 shows the community structure using edge-betweenness algorithm and figure 3.3 shows the community structure using infomap algorithms.

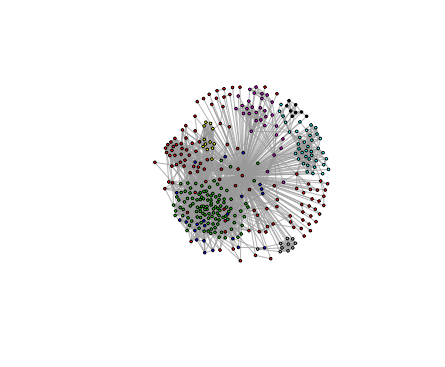


Figure 3.1 Community Structure using fast-greedy algorithm

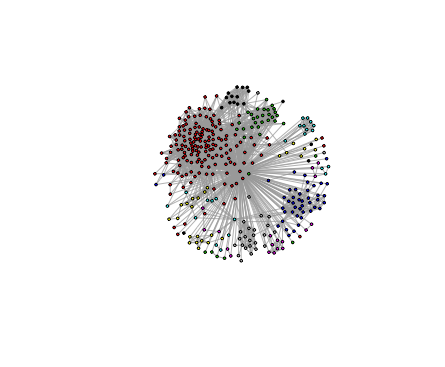


Figure 3.2 Community Structure using edge-betweenness algorithm

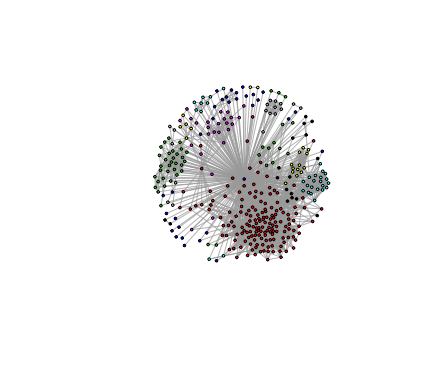


Figure 3.3 Community Structure using infomap algorithm

We see that, though determined using different algorithms, the communities in different graph have some apparent overlap, which mean these community structures have distinguish features. And it can be seen that edge-betweenness algorithm tends to break the graph into more partitions than other two algorithms.

## 2.4 Problem 4

We first generated a graph without the core node 1 and them found the community structure again as we did above using three different algorithms. The results are shown in figure 4.1-4.3.

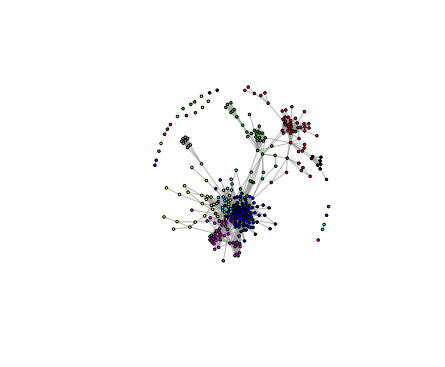


Figure 4.1 Community Structure using fast-greedy algorithm

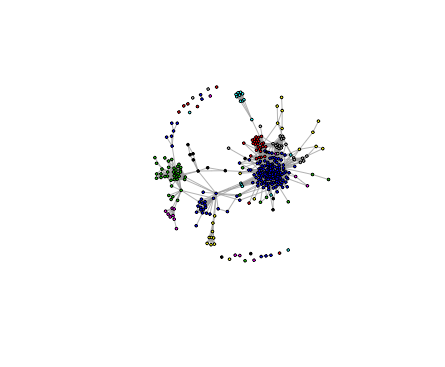


Figure 4.2 Community Structure using edge-betweenness algorithm

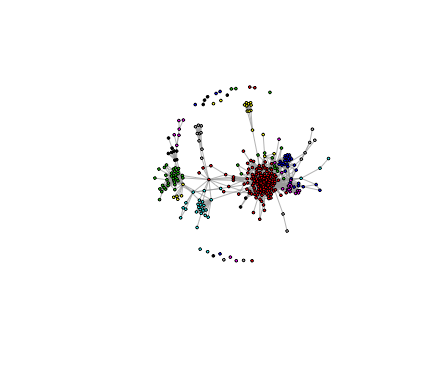


Figure 4.3 Community Structure using infomap algorithm

If we look further into those community structures, we may found that though they are structured without the core node, the partitions are actually similar. And this can be testified by examine the modularity of structures of problem 3 and problem 4. The difference of modularity is about 10% between two parts.

2.5 Problem 5

**Analysis:**

For this problem, we need to calculate two features about the nodes in a personal network. Embeddedness is calculating the number of mutual friends, which means the larger the embeddedness is, the more mutual friends you have. Dispersion is calculating the relationship among all your mutual friends, which means the larger the dispersion is, the more likely your mutual friends don’t know well about each other. We mainly use function intersect() to calculate embeddedness, and function shortest.path() to calculate dispersion.

**Results:**

From part 3, we find out 40 core nodes, for each node, we create a personal network, and plot the embeddedness and dispersion distribution over all the core personal networks.

Figure 5.1 and 5.2 shows the embeddedness and dispersion distribution respectively.



Figure 5.1 Embeddedness Distribution over all personal network

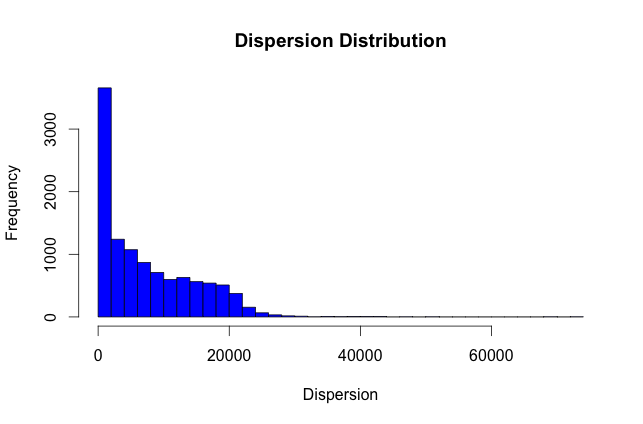


Figure 5.1 Dispersion Distribution over all personal network

We pick up 3 core nodes 1,3,10. For each core node, highlight the nodes with maximum dispersion, embeddedness and disp/ embed. The following 9 figures 5.3.1-5.3.3, 5.4.1-5.4.3, 5.5.1-5.5.3 show the features for code node 1,3,10 respectively. The core node is marked as the largest white node, while the max-feature node is marked as the second largest white node, and also the edge between them is highlighted.

**Node 1:**

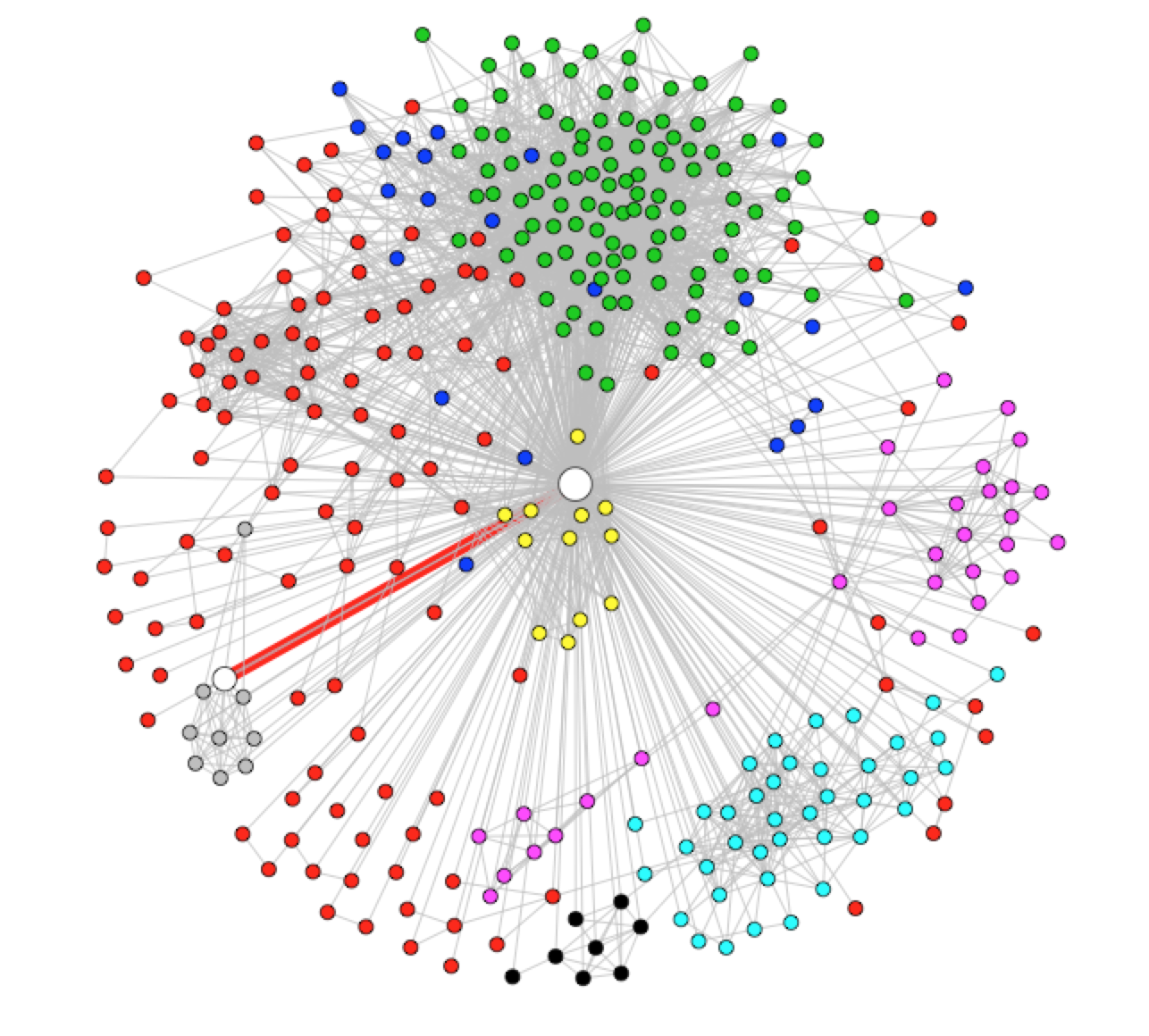


Figure 5.31 highlighted graph for node 1 with max-dispersion

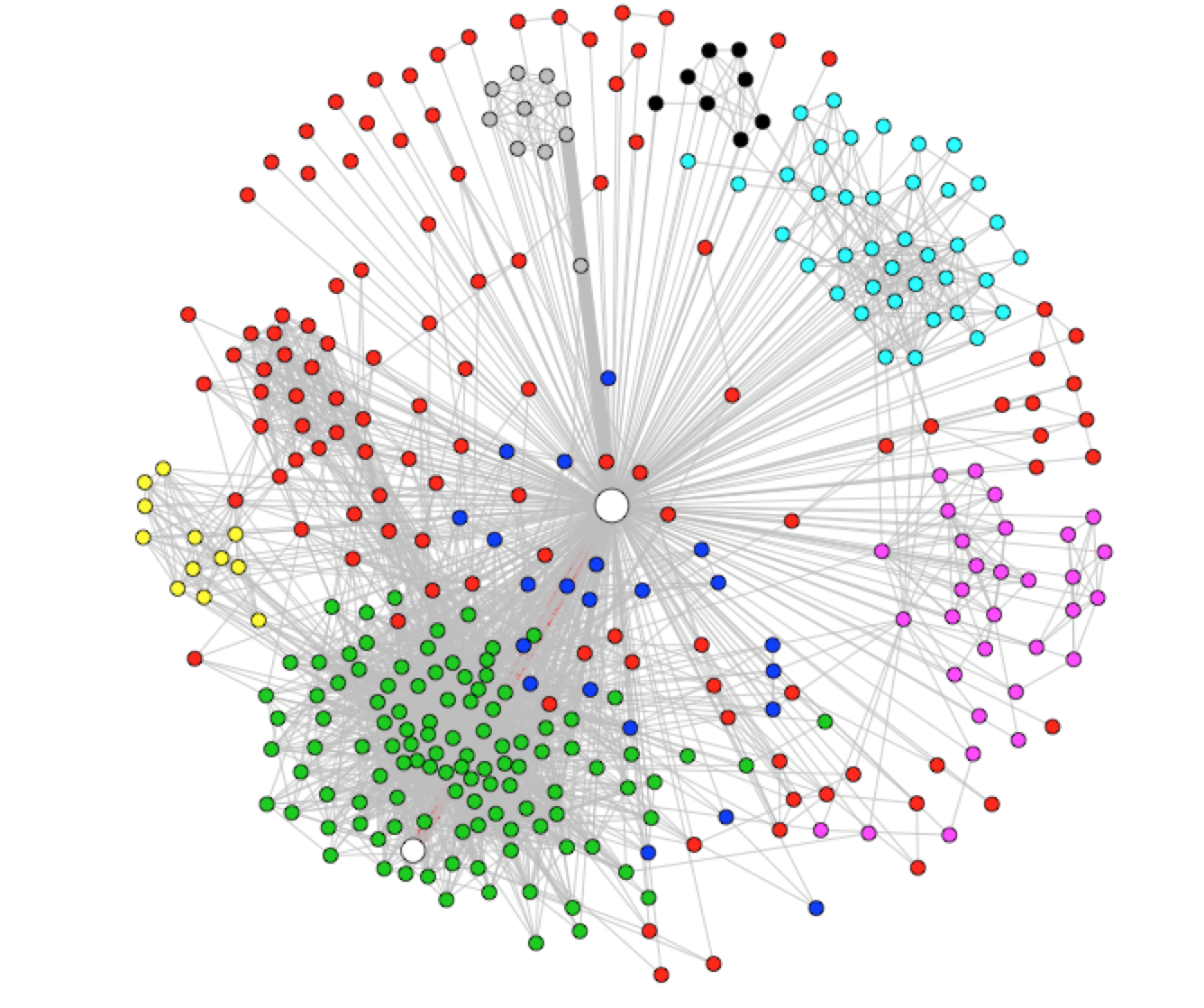


Figure 5.32 highlighted graph for node 1 with max-embeddedness

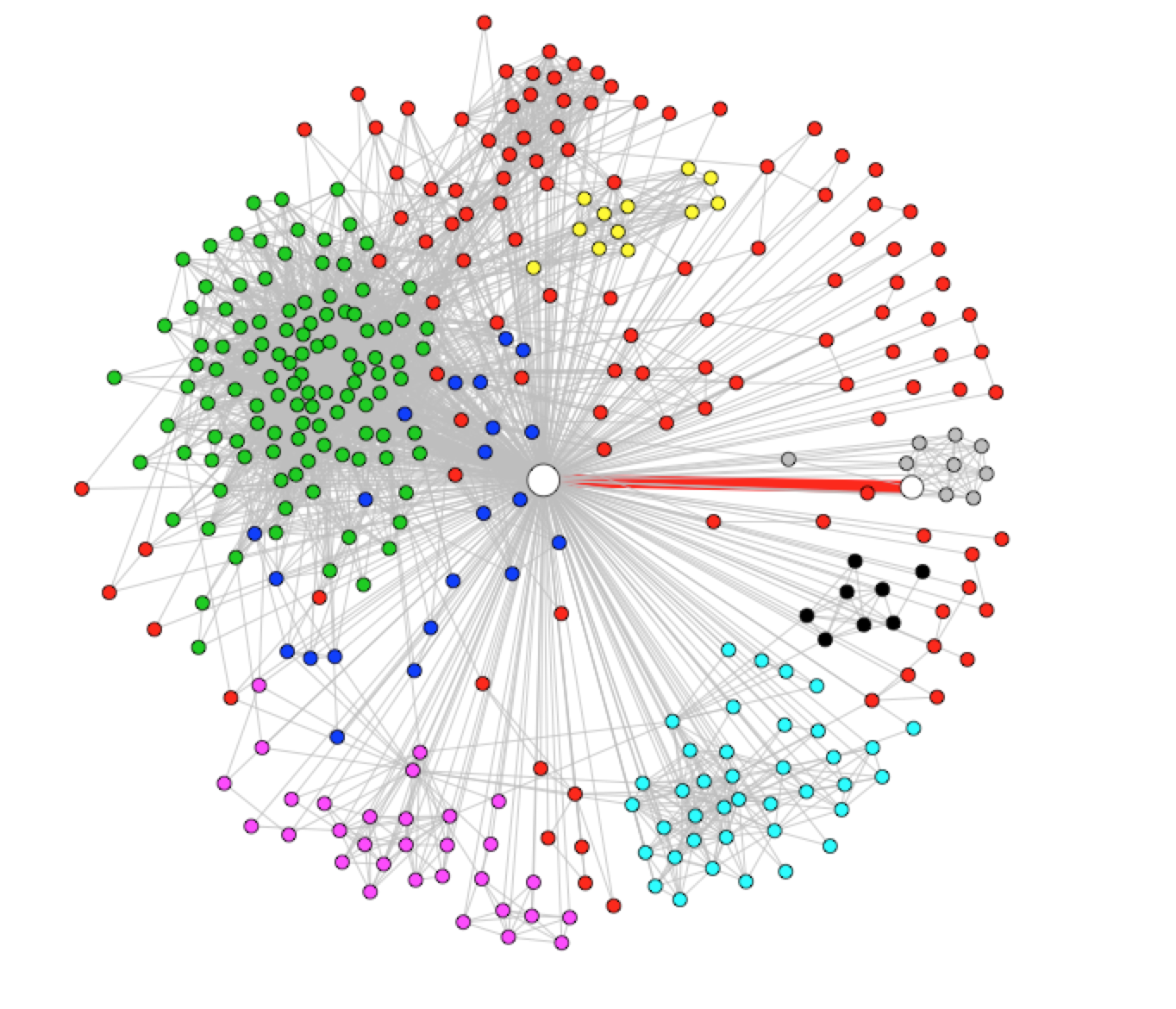


Figure 5.33 highlighted graph for node 1 with max-disp/embed

**Node 3:**

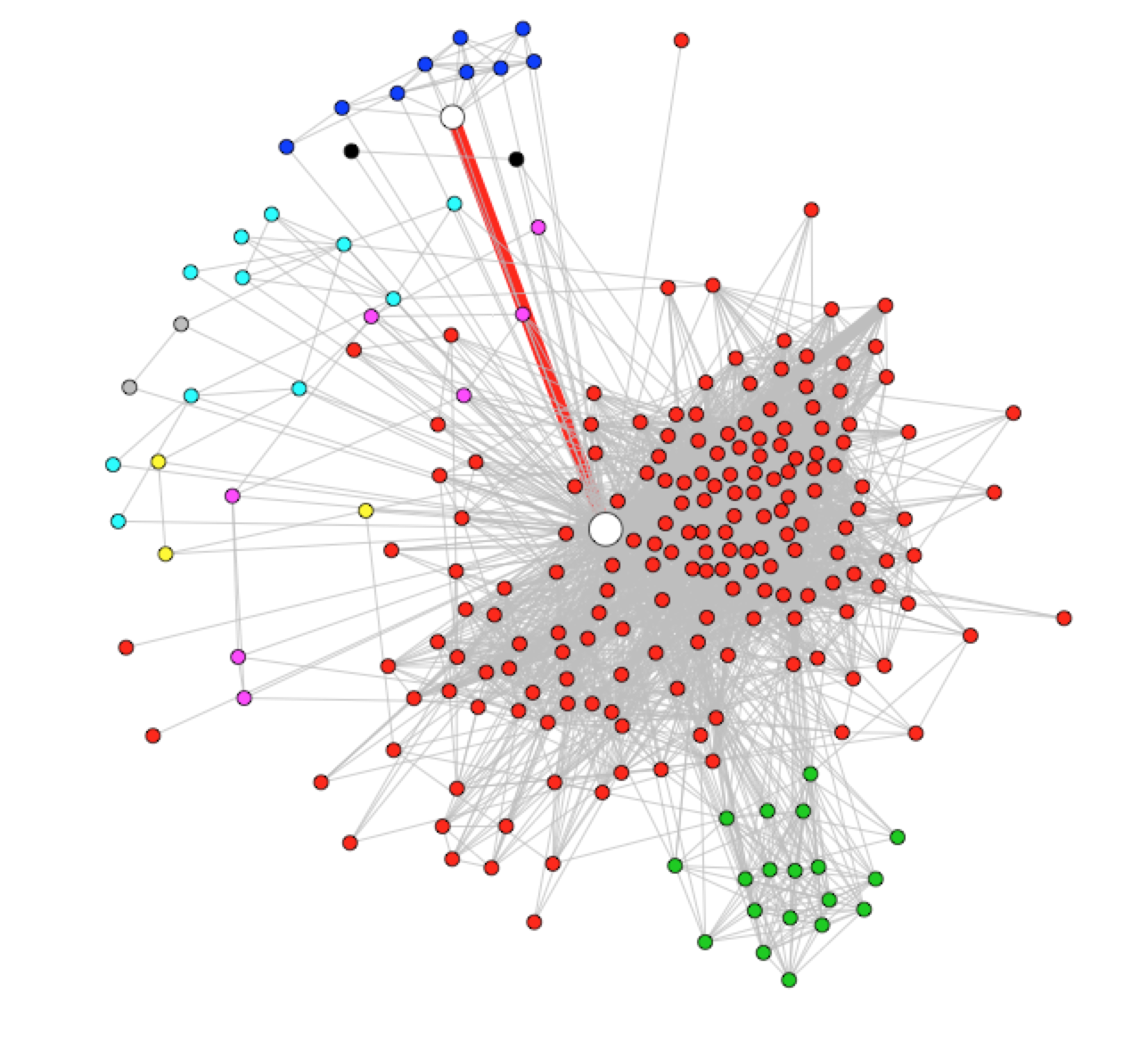


Figure 5.41 highlighted graph for node 3 with max-dispersion

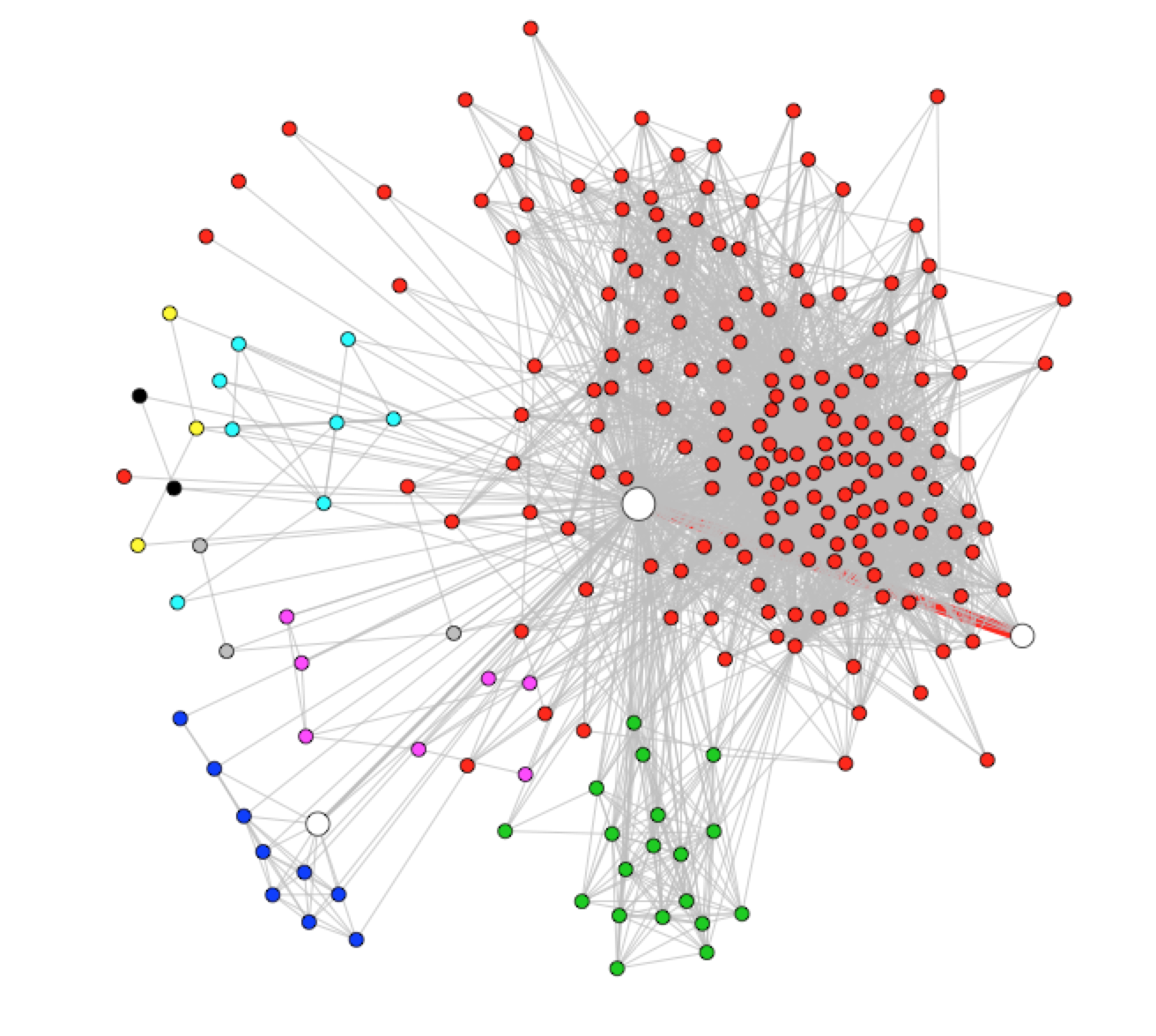


Figure 5.42 highlighted graph for node 3 with max-embeddedness



Figure 5.43 highlighted graph for node 3 with max-disp/embed

**Node 10:**

**Conclusion:**

Embeddedness: Embeddedness is the number of mutual friends. It could indicate the closeness between two people, which is easy-understanding. But it doesn’t necessarily means the two people are couple, because for people in the same class, they could have many mutual friends which are their classmates. So Embeddedness is only one factor.

Dispersion: Large dispersion means your mutual friends are not likely to know each other. This is pretty common in the real life. For example, a couple A and B, A’s friends knows B because of A and B’s relationship, but not necessarily know B’s friends. So dispersion could be one of the features, which could reflect romantic relationship.

When calculating dispersion using shortest.path, if the two nodes doesn’t have shortest path, the value will be Inf, but it seems that calculating in this way is not very reasonable. Maybe replacing Inf with the number of node in this particular personal network (the longest path could be in this personal network) is more reasonable. For example, in personal network A, only 2 nodes doesn’t have shortest path, but other nodes are strongly connected (even shortest path is only 1). And in personal network B, all nodes are not tightly connected. Which one should have higher dispersion? Still need more investigation.(Personal opinion)

Dispersion/Embeddedness: The ratio of dispersion and Embeddedness could be seen as an average dispersion for each mutual friend. It become larger when the dispersion become larger and the embeddedness become smaller. We could see that dispersion could be relative large when the embeddedness is large, even most their mutual friends knows each other. So the ratio is a kind of normalization. The combination of these two features could be more precise and reliable.

2.6 Problem 6

**Analysis:**

The problem is finding out features to determine the communities belong to certain kind of types. The types could be “classmates”, “families”. We try to use several features to find out the community which members have highest closeness and lowest closeness. The community with the highest closeness could probably be a “classmates” community, and the one with lowest closeness could probably be a “friends made on the internet” community. The closeness here we mean is the closeness of all members in the community, which means the better the members know each other, the higher the closeness.

We choose 3 features to represent the closeness of the community, which are average degree, clustering coefficient and density.

**Results:**

The following Table 6.1 shows the communities with maximum and minimum average degree, clustering coefficient and density respectively over all core personal networks.

Table 6.1 measure community type by average degree, clustering coefficient and density

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **core no** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| community\_index | 16 | 14 | 4 | 2 | 8 | 2 | 3 | 2 | 6 | 5 | 5 | 13 | 3 | 3 | 2 | 3 | 1 | 1 | 6 | 2 |
| max\_aver\_degree | 0.80 | 0.72 | 0.64 | 0.49 | 0.61 | 0.68 | 0.71 | 0.60 | 0.69 | 0.86 | 0.80 | 0.75 | 0.70 | 0.75 | 0.55 | 0.64 | 0.58 | 0.53 | 0.71 | 0.83 |
| community\_index | 16 | 12 | 4 | 2 | 8 | 2 | 6 | 3 | 6 | 5 | 5 | 13 | 3 | 3 | 2 | 3 | 1 | 1 | 6 | 2 |
| max\_cluster\_coe | 0.89 | 0.89 | 0.80 | 0.71 | 0.73 | 0.81 | 0.85 | 0.73 | 0.77 | 0.90 | 0.87 | 0.87 | 0.78 | 0.81 | 0.69 | 0.74 | 0.74 | 0.68 | 0.83 | 0.89 |
| community\_index | 16 | 12 | 4 | 2 | 8 | 2 | 6 | 2 | 6 | 5 | 5 | 13 | 3 | 3 | 2 | 3 | 1 | 1 | 6 | 2 |
| max\_density | 0.87 | 0.78 | 0.67 | 0.50 | 0.61 | 0.69 | 0.75 | 0.61 | 0.71 | 0.89 | 0.83 | 0.80 | 0.71 | 0.76 | 0.55 | 0.65 | 0.59 | 0.53 | 0.75 | 0.84 |
| community\_index | 3 | 1 | 1 | 1 | 9 | 3 | 4 | 3 | 7 | 3 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| min\_aver\_degree | 0.19 | 0.08 | 0.22 | 0.35 | 0.60 | 0.56 | 0.53 | 0.59 | 0.65 | 0.63 | 0.46 | 0.10 | 0.54 | 0.58 | 0.46 | 0.45 | 0.57 | 0.49 | 0.16 | 0.60 |
| community\_index | 3 | 1 | 1 | 1 | 9 | 3 | 4 | 2 | 7 | 3 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| min\_cluster\_coe | 0.37 | 0.40 | 0.49 | 0.54 | 0.71 | 0.71 | 0.65 | 0.71 | 0.75 | 0.71 | 0.63 | 0.32 | 0.70 | 0.72 | 0.65 | 0.62 | 0.70 | 0.64 | 0.40 | 0.75 |
| community\_index | 3 | 1 | 1 | 1 | 9 | 3 | 4 | 3 | 7 | 3 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| min\_density | 0.20 | 0.08 | 0.23 | 0.36 | 0.61 | 0.58 | 0.55 | 0.59 | 0.67 | 0.65 | 0.47 | 0.10 | 0.55 | 0.60 | 0.47 | 0.45 | 0.57 | 0.49 | 0.16 | 0.61 |
| **core no** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** | **32** | **33** | **34** | **35** | **36** | **37** | **38** | **39** | **40** |
| community\_index | 1 | 1 | 5 | 1 | 1 | 6 | 1 | 3 | 1 | 5 | 1 | 5 | 2 | 3 | 1 | 3 | 2 | 4 | 3 | 23 |
| max\_aver\_degree | 0.85 | 0.79 | 0.89 | 0.88 | 0.82 | 0.79 | 0.80 | 0.87 | 0.86 | 0.72 | 0.78 | 0.75 | 0.76 | 0.77 | 0.88 | 0.85 | 0.78 | 0.79 | 0.78 | 0.86 |
| community\_index | 1 | 1 | 5 | 1 | 1 | 6 | 1 | 3 | 1 | 5 | 1 | 5 | 2 | 3 | 1 | 3 | 2 | 4 | 5 | 23 |
| max\_cluster\_coe | 0.91 | 0.84 | 0.97 | 0.91 | 0.86 | 0.84 | 0.86 | 0.94 | 0.89 | 0.80 | 0.84 | 0.82 | 0.83 | 0.83 | 0.90 | 0.89 | 0.85 | 0.84 | 0.88 | 0.93 |
| community\_index | 1 | 1 | 5 | 1 | 1 | 6 | 1 | 3 | 1 | 5 | 1 | 5 | 2 | 3 | 1 | 3 | 2 | 4 | 5 | 23 |
| max\_density | 0.90 | 0.80 | 0.97 | 0.89 | 0.83 | 0.80 | 0.81 | 0.93 | 0.86 | 0.72 | 0.78 | 0.76 | 0.77 | 0.77 | 0.88 | 0.86 | 0.79 | 0.79 | 0.80 | 0.92 |
| community\_index | 3 | 2 | 3 | 4 | 3 | 2 | 2 | 4 | 3 | 5 | 3 | 4 | 1 | 1 | 2 | 1 | 1 | 3 | 5 | 4 |
| min\_aver\_degree | 0.40 | 0.70 | 0.41 | 0.61 | 0.56 | 0.68 | 0.70 | 0.75 | 0.56 | 0.72 | 0.67 | 0.41 | 0.66 | 0.67 | 0.54 | 0.74 | 0.55 | 0.71 | 0.74 | 0.18 |
| community\_index | 3 | 2 | 3 | 4 | 3 | 2 | 2 | 4 | 3 | 5 | 3 | 4 | 1 | 1 | 2 | 4 | 1 | 3 | 3 | 2 |
| min\_cluster\_coe | 0.64 | 0.80 | 0.62 | 0.66 | 0.71 | 0.78 | 0.78 | 0.80 | 0.68 | 0.80 | 0.82 | 0.64 | 0.77 | 0.77 | 0.64 | 0.80 | 0.72 | 0.83 | 0.83 | 0.37 |
| community\_index | 3 | 2 | 3 | 4 | 3 | 2 | 2 | 4 | 3 | 5 | 3 | 4 | 1 | 1 | 2 | 4 | 1 | 3 | 6 | 4 |
| min\_density | 0.43 | 0.71 | 0.43 | 0.66 | 0.58 | 0.69 | 0.72 | 0.75 | 0.60 | 0.72 | 0.73 | 0.45 | 0.67 | 0.70 | 0.54 | 0.77 | 0.56 | 0.74 | 0.78 | 0.18 |

Community\_index means the index of community which has the maximum or minimum feature value.

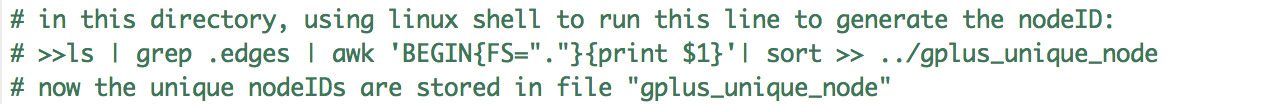
**Conclusion:**

From table 6.1 we could find out that for most personal networks all these three features points to the same community, which means the community could probably be the highest or lowest closeness community. The combination of these 3 features seems to be pretty reasonable for judging the closeness of the communities. However, without further investigation, we can’t decide which feature has the most weight. In a word, the community with highest feature values would probably be a “classmate” community or some other communities with high closeness and the one with lowest feature value seems to be an “internet friends” community or some other communities with low closeness. The result here doesn’t give us fully confidence, but it makes sense more or less.

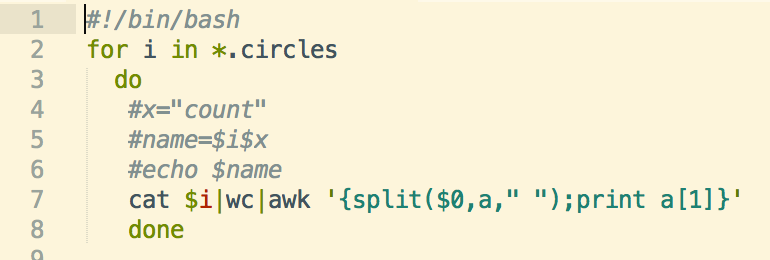
We just choose two type of community as examples, adding other features like community size, embeddedness and dispersion in the former problem may help to find out other type of communities. For example, a community with high closeness but a small community size could probably be a “family” community and a community with a large community size but also have a large dispersion and a relative low closeness may be considered as an “alumni” community and so on.

## 2.7 Problem 7

First we ran a linux shell command to generate the unique ego node ID as follow:



Then we used another linux shell script to determine the number of circles of each echo node, which looks like below:



Then for each ego node, if it has more than 2 circles, we constructed the graph by reading the edge list file and adding the ego node and corresponding edges. Then we select nodes from each community and count the percentage of those overlapped between each community and circle. The percentage means those overlapped over the total nodes number of each community. And the results between two nodes are shown in table 7.1-7.4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Selected | Circle1 | Circle2 | Circle3 | Circle4 |
| Community1 | 0 | 1.79% | 5.35% | 3.57% |
| Community2 | 1.43% | 35.51% | 61.78% | 8.92% |
| Community3 | 11.05% | 1.54% | 2.57% | 37.02% |
| Community4 | 0 | 13.16% | 80.71% | 1.75% |

Table 7.1 The percentage of overlap using walktrap algorithm of node 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| selected | Circle1 | Circle2 | Circle3 | Circle4 |
| Community1 | 4.60% | 21.91% | 43.91% | 17.58% |
| Community2 | 0 | 0 | 7.69% | 3.84% |
| Community3 | 50% | 0 | 0 | 50% |
| Community4 | 0 | 0 | 0 | 100% |

Table 7.2 The percentage of overlap using infomap algorithm of node 7

|  |  |  |  |
| --- | --- | --- | --- |
| selected | Circle1 | Circle2 | Circle3 |
| Community1 | 5.58% | 7.30% | 5.15% |
| Community2 | 0 | 0 | 0 |
| Community3 | 0 | 0 | 0 |
| Community4 | 70.54% | 31.34% | 11.29% |

Table 7.3 The percentage of overlap using walktrap algorithm of node 12

|  |  |  |  |
| --- | --- | --- | --- |
| selected | Circle1 | Circle2 | Circle3 |
| Community1 | 46.02% | 22.33% | 9.12% |
| Community2 | 4.35% | 4.35% | 0 |
| Community3 | 0 | 0 | 16.67% |
| Community4 | 0 | 25% | 0 |

Table 7.2 The percentage of overlap using infomap algorithm of node 12

Note that for node 7, it has 4 circles. If using walktrap, it has 4 communities, and 17 if using infomap algorithm. We listed all the overlap percentage for walktrap and selected ones for infomap. Similarly, node 12 has 3 circles, and we listed all communities of walktrap algorithm and selected 4 for infomap.

As we can see from the tables, the overlap varies a lot among users. Also there is a large variation between different algorithms.

# 3. Difficulty encountered

The project needs more skill on graph manipulations. When we first creating sub-graphs, we find out that the vertex ID changes when creating a graph, so when we can’t easily trace back to the original graph. After getting more knowledge on graph manipulations, we find out we could add names for the nodes in the graph, then we could trace back to the original graph with the specific name of the node, it is much easier. In addition, another interesting part is how to plot a pretty figure with certain edge and vertex highlighted, this makes the figure more readable and could better represent the information hidden in the graph.