Understanding Vulnerability of Communities in Social Networks

Student Name: Viraj Parimi
Roll Number: 2015068

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering

on November 16, 2017

BTP Track: Research

BTP Advisor
Dr. Tanmoy Chakraborty
Dr. Ponnurangam Kumaraguru

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **"Understanding vulnerability of Communities in Social Networks"** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Tanmoy Chakraborty** and **Dr. Ponnurangam Kumaraguru**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

.... **Place & Date: ....**
Viraj Parimi

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

.... **Place & Date: ....**
Dr. Tanmoy Chakraborty

.... **Place & Date: ....**
Dr. Ponnurangam Kumaraguru

Abstract

In this project I study the problem of identifying critical nodes in a network which minimize different community metrics like NMI(Normalized Mutual Information), Modularity, Adjusted Rand Index etc. and compare their results. Currently I focus on Modularity and NMI metric of communities in a social network. The ability to see such results allows us to better understand the vulnerability of the network since the nodes selected across different metrics might be different and hence seeing those results in conjunction will tell us a lot more about the network, than seeing only one metric which is what is done in some literature. Also since the problem does not restrict the use of only one community detection algorithm, hence the mechanism that I would propose should be able to work on multiple community detection algorithms. Presently, I have tried different approaches or heuristics since the problem itself is NP-Complete and attempted to achieve the ground truth results for small datasets such as Karate Club.

Keywords: (Graph Mining, Algorithms, Community Analysis)

Contents

Chapter 1

Introduction

1.1 Problem Definition

Given a graph $G(V, E)$, each node i is associated with a cost C_i, and there is a total budget K. For a vertex set $S \subseteq V$, let $G[S]$ be the graph induced by S, $\textsc{Cost}(S) = \sum_{i \in S} C_i$, and $f(S) = \textsc{Modularity}(G) - \textsc{Modularity}(G[\bar{S}])$. Here, $\bar{S} = V \setminus S$. Note that, $f(S)$ is a real-valued *set function*. We need to identify a set of nodes $S \subseteq G$ which,

$$\begin{aligned} \textbf{maximize} \quad & f(S) \\ \textbf{subject to:} \quad & \textsc{Cost}(S) \leq K \end{aligned}$$

Modularity is defined as follows:

$$\frac{1}{2m} \sum_{i,j} (A_{i,j} - \frac{k_i * k_j}{2m}) \delta(c_i, c_j)$$

$$\begin{aligned} \text{where} \quad \boldsymbol{m} &= \text{Number of edges} \\ \boldsymbol{k_i, k_j} &= \text{Degree of node i, j} \\ \boldsymbol{c_i, c_j} &= \text{Community label of node i and node j} \\ \boldsymbol{A_{i,j}} &= \text{Adjacency Matrix} \\ \boldsymbol{\delta(c_i, c_j)} &= \begin{cases} 1 & c_i = c_j \\ 0 & otherwise \end{cases} \end{aligned}$$

The unit cost formulation of the above problem has only one change. The value of $C_i = 1$ i.e each node in the graph G has unit cost. We are going to prove that this version of the above mentioned problem is NP-Complete. Since the unit cost formulation is just a trivial case of the general cost version, if we prove that the unit cost version is NP-Complete then clearly the general case would also be NP-Complete.

Chapter 2

NP-Completeness Proof

2.1 Reduction Problem

We will use *Maximum Vertex Coverage - Bipartite Graphs*(MVC-B) for reduction. This problem states the following,

Given a bipartite graph $G = (V, E), V = V_L \cup V_R, V_L \cap V_R = \emptyset$ and positive integers b, c, check if there exists a subset of vertices $S \subseteq V with$ |S|= b such that at least c edges are incident to nodes in S.

2.2 Proof

Given an instance of MVC-B with bipartite graph $G = (V, E)$, $V = V_L \cup V_R, V_L \cap V_R = \emptyset$ and positive integers b, c.
Now we construct our problem instance $G' = (V', E')$ by connecting V_L, V_R to cliques K_L, K_R, respectively. We create an edge between all $\{(u, v)|u \in K_L, v \in V_L\}$ and all $\{(u, v)|u \in K_R, v \in V_R\}$. We choose the size of K_R, K_L such that, when b vertices in V are removed with at least c edges incident to them in E, $A(Algorithm)$ will detect two communities $K_L \cup V_L \setminus S_L, K_R \cup V_R \setminus S_R$ where $S_L \in V_L$ and $S_R \in V_R$ and when we do not do any such operation it will detect only one community. By this we mean that $\mid K_L \mid$ is not necessarily equal to $\mid V_L \mid$ and similarly $\mid K_R \mid$ is not necessarily equal to $\mid V_R \mid$. Suppose when E is sparse then we would want $\mid K_L \mid$ and $\mid K_R \mid$ to be small enough so that $A(G')$ will detect only one community and $A(G'[V' \setminus S]$ will detect 2 above mentioned communities when we remove $\mid S \mid = b$ vertices. Similarly when E is dense we would want K_L and K_R to be large enough for the same reason.

Let $k = b$ and,

$$a = \min_{S_L \in V_L, S_R \in V_R, |S_L|+|S_R|=b} Modularity(A(G')) - Modularity(Y')$$

where

$$Y' = \{K_L \cup V_L \setminus S_L, K_R \cup V_R \setminus S_R\}$$

Now assume we have a solution S, $\mid S \mid = K$ to our problem. As

$$Modularity(A(G') - Modularity(A(G'[V' \setminus S])) \geq a$$

, $A(G'[V' \setminus S])$ must contain two communities. By construction, number of edges removed from E is at least c. If $S \subseteq V$ then we directly obtain the solution to MVC-B. If $\exists v \in S, v \notin V$, we can always find a vertex $u \in V, u \notin S$ and update S to S' $= S \cup \{u\} \setminus \{v\}$ while keeping the number of edges incident to S greater than c. Therefore, we have a solution $S', \mid S' \mid$=K= b for MVC-B.

Now assume we have a solution $S, \mid S \mid = b$ to MVC-B. Then at least c edges in E are incident to vertices in S. If we remove all vertices in S, by construction, $A(G'[V' \setminus S)$ will output 2 communities, $K_L \cup V_L \setminus S'_L$ and $K_R \cup V_R \setminus S'_R$. Then we have,

$$Modularity(A(G')) - Modularity(A(G'[V' \setminus S])) \geq a$$

as a is the maximum Modularity difference value for communities in the form $\{K_L \cup V_L \setminus S_L, K_R \cup V_R \setminus S_R\}$. Therefore, we have a solution $S, \mid S \mid = b = K$ for our problem.

2.3 Conclusion

Since our problem statement has a solution if and only if *Maximum Vertex coverage - Bipartite Graph* has a solution, hence our problem statement is NP-Complete.

Chapter 3

Approach

- Considering that the problem is NP-Complete I needed to a small dataset to work on. Hence, I used UCI Karate Club network with 2 communities and 34 nodes.
- Since my problem statement is not restricted by a particular community detection algorithm, I have the following,
  - Louvain
  - Edge Betweenness
  - Fast Greedy
  - Infomap
  - Label Propagation
  - Leading Eigenvector
  - Multilevel
  - Walktrap
- The value functions that I have tried to optimize are the following,
  - Modularity (as mentioned above)
  - NMI (Normalized Mutual Information)
    NMI is defined as follows:

$$\frac{2\sum_{i=1}^{c_X}\sum_{j=1}^{c_Y}(n_{ij}log(\frac{n_{ij}n}{x_i * y_j}))}{(n-k)H(X)+\bar{y}log(n-k)-\sum_{j=1}^{c_Y}(y_j log(y_j)}$$

where $\boldsymbol{c_X}$ = Number of communities in network X
$\boldsymbol{c_Y}$ = Number of communities in network Y
$\boldsymbol{n_{ij}} = |X_i \cap Y_j|$
$\boldsymbol{n}$ = Number of nodes

$$\boldsymbol{x_i} = |X_i|$$
$$\boldsymbol{y_i} = |Y_i|$$
$$\boldsymbol{\bar{y}} = \text{Total size of communities in Y}$$

3.1 Getting ground truth

First I generated the ground truth for the small dataset of Karate Club. This would allow me to compare the heuristics with the actual best results. The algorithm is as follows.

Algorithm 1 Exhaustive Algorithm

```
graph ← load_graph
partition ← generate_partition
value ← compute_value_function
make plot of the partition
n ← total_nodes, r ← budget
tryAll ← generate nPr combinations
score ← array
for all tryAll do
    copy ← graph.copy
    copy ← copy.remove_selection
    new_partition ← generate_partition
    new_value ← compute_value_function
    score ← value - new_value
sort the score array
best_nodes ← score.0
graph ← graph.remove_selection
best_partition ← generate_partition
make plot of best_partition
```

3.2 Network based Greedy approach

In this approach I greedily chose top K nodes based on different metrics such as,

- Clustering Coefficient - $C_i = \frac{2|\{e_{jk}:v_j,v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i-1)}$ where C_i is the clustering coefficient of vertex i. N_i is the neighborhood of vertex i. k_i is the degree of the vertex i.

- Degree Centrality - $C_D(v) = \deg(v)$

- Betweenness Centrality - $C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$ where σ_{st} is the total number of shortest paths from vertex s to vertex t. $\sigma_{st}(v)$ is the number of those paths that pass through v.

- Eigenvector Centrality - $x_v = \frac{1}{\lambda} \sum_{t \in M(v)} x_t$ where λ is a constant and $M(v)$ is the set of neighbors of v

- Closeness Centrality - $C(x) = \frac{1}{\sum_y d(y,x)}$ where $d(y,x)$ is the distance between vertices x and y
- Coreness - The k-core of the graph is a maximal subgraph in which each vertex has at least degree(in the subgraph) k. The coreness of a vertex is k if it is a member of the k-core but not a member of $k+1$-core.
- Diversity - It is the normalized Shannon extropy of the weights of edges incident on the vertex. Shannon entropy $[H(X)] = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$
- Eccentricity - Shortest distance from (or to) the vertex, to(or from) all the other vertices in the graph, and taking maximum.
- Constraint - Burt's value. Well defined on the internet.
- Closeness Vitality - Change in the sum of distances between all node pairs when excluding that node

Apart from the above mentioned metrics I used some other metrics as well such as,

- myMod - Selects the vertex that contributes most to the modularity of the network.
- myNMI - Selects the vertex that contributes most to the NMI of the network.

Algorithm 2 Network based Greedy Algorithm

graph ← *load_graph*
partition ← *generate_partition*
value ← *compute_value_function*
selection ← *best_nodes_greedy*
copy ← *graph.remove_selection*
new_partition ← *generate_partition*
new_value ← *compute_value_function*
score ← *value* - *new_value*
make plot of *new_partition*

The intuition behind this approach was if we remove nodes that contribute majorly to the network property, then it might help us reduce the value function itself.

3.3 Community based Greedy approach

In this approach rather than directly attacking and removing nodes that contribute heavily to some network property of the graph we first try to identify a community within the network based on a greedy metric and then within that community we identify a node based on the greedy metrics defined in the previous subsection. We then repeat the process until we hit out budget.

The metrics I chose to select a community are as follows,

- Link density - $D(G) = \frac{2E}{V(V-1)}$ where E is the number of edges in the graph and V is the number of vertices in the graph.
- Conductance - $\phi(G) = \frac{s}{v}$ where s is number of vertices with one endpoint in G and another in $\bar{G}$, v is the sum of degree of nodes in G.
- Compactness - $C(G)$ is defined as the average shortest path lengths within the graph G.

Algorithm 3 Community based Greedy Algorithm

```
procedure BEST COMMUNITY
    partition_score ← array
    for all partitions_graph do
        subgraph ← graph.partition
        value ← compute_value_function
    sort the partition_score array
    return partition corresponding to partition_score.0
procedure BEST NODE
    value ← compute_value_function
    return value
graph ← load_graph
graph_partition ← generate_partition
value ← compute_value_function
counter ← 0
while counter ≤ budget do
    subgraph ← BEST_COMMUNITY
    node ← BEST_NODE
    graph ← remove_selection
    partition ← generate_partition
    counter += 1
new_value ← compute_value_function
score ← value - new_value
make plot of partition
```

The intuition behind this approach was since we want to select nodes such that the community structure of the network breaks, hence we should first target the community itself. Having identified a community, we can then target the node as described in the previous subsection. This method is more fine grained as it considers two level scrutiny at each iteration.

3.4 Genetic Algorithm approach

This approach is similar to a general genetic algorithm which works on a population and improves its progeny at every generation. The algorithm is as follows,

Algorithm 4 Genetic Algorithm

```
procedure PROPAGATION
    mating ← array
    mating ← randomly select 2 parents and cross pollinate them
    return mating
procedure MUTATION
    mutate ← array
    mutate ← randomly select a parent and change one position
    return mutate
procedure MAKE NEXT GENERATION
    propogation ← PROPAGATE
    mutation ← MUTATION
    return chromosomes + propagation + mutation
population_size ← 100 (hyperparameter)
tryAll ← generate ^nP_r combinations
chromosomes ← select population_size from tryAll
generation ← 0
score ← array
while generation ≤ MAX_GENERATIONS do
    for all chromosomes do
        copy ← graph.copy
        copy ← delete_selection
        score ← compute_value_function
    sort the score array
    chromosomes ← select top 20 best chromosomes
    chromosomes ← MAKE NEXT GENERATION
    generation += 1
```

Genetic algorithms usually start with a random guess and through generations the random samples that it initially started with improves. The improvement is measured by a fit function which in my case is the value function itself. At each generation the current set of chromosomes are evaluated and the top ones are chosen. Now for the next generation we cross pollinate between those selected chromosomes and with some probability, sometimes even mutate them. Cross pollination is done by randomly choosing 2 parents and again randomly choose 2 positions in the parents and making a new child. Mutation is done by randomly selecting a candidate and again randomly choosing a position and change the value at that position of the candidate. Once we have a new set of chromosomes of the next generation we repeat the process again.

Chapter 4

Results

The results below are shown for the following values,

- Community Detection Algorithm - Louvain
- Budget - 4
- Value Function - Modularity and NMI
- Graph - Karate Club
- Population Size - 100
- Top chromosomes per generation - 20
- Max generations - 200

4.1 Exhaustive Results

4.1.1 Modularity

Best nodes which optimized the modularity value of the network were, **0, 3, 6, 1**. The corresponding decrease in modularity value of the network was **0.12289**.

4.1.2 NMI

Best nodes which optimized the NMI value of the network were, **32, 33, 5, 6**. The corresponding NMI value between the original network and the new network was **0.38580**.

4.2 Network based Greedy approach

4.2.1 Modularity

Metrics	Clustering Coefficient	Degree Centrality	Betweenness Centrality	Eigenvector Centrality	Closeness Centrality	Coreness	Diversity	Eccentricity	Constraint	Closeness Vitality	myMod	Intra-degree
Nodes Selected	16, 22, 7, 12	33, 0, 32, 2	0, 33, 32, 2	33, 0, 2, 32	0, 2, 33, 31	33, 32, 1, 2	33, 8, 14, 13	16, 26, 18, 20	11, 16, 26, 12	0, 16, 26, 11	17, 20, 15, 5	33, 32, 0, 1
Modularity Scores	0.03664	-0.23652	-0.23652	-0.23652	-0.23186	-0.07938	-0.01111	0.04439	0.03641	-0.00166	0.03469	-0.15885

Table 4.1: Louvain Results for different greedy metrics optimizing modularity scores.

4.2.2 NMI

Metrics	Clustering Coefficient	Degree Centrality	Betweenness Centrality	Eigenvector Centrality	Closeness Centrality	Coreness	Diversity	Eccentricity	Constraint	Closeness Vitality	myNMI	Intra-degree
Nodes Selected	16, 22, 7, 12	33, 0, 32, 2	0, 33, 32, 2	33, 0, 2, 32	0, 2, 33, 31	33, 32, 1, 2	33, 8, 14, 13	16, 26, 18, 20	11, 16, 26, 12	0, 16, 26, 11	17, 20, 15, 5	33, 32, 0, 1
NMI Scores	0.64575	0.67842	0.67842	0.67842	0.77117	0.70084	0.69480	0.85071	0.64740	0.83451	0.64568	0.62733

Table 4.2: Louvain Results for different greedy metrics optimizing NMI scores.

4.3 Community based Greedy approach

4.3.1 Modularity

Link Density

Metrics	Clustering Coefficient	Degree Centrality	Betweenness Centrality	Eigenvector Centrality	Closeness Centrality	Coreness	Diversity	Eccentricity	Constraint	Closeness Vitality	myMod	Intra-degree
Nodes Selected	25, 16, 20, 10	NA	NA	31, 24, 6, 9	NA	NA	NA	28, 30, 27, 25	25, 16, 11, 10	NA	31, 27, 16, 24	31, 16, 26, 24
Modularity Scores	0.04875	NA	NA	0.05090	NA	NA	NA	-0.01348	0.04690	NA	0.03180	0.02399

Table 4.3: Louvain Results for different greedy metrics with link density as the global metric.

Conductance

Metrics	Clustering Coefficient	Degree Centrality	Betweenness Centrality	Eigenvector Centrality	Closeness Centrality	Coreness	Diversity	Eccentricity	Constraint	Closeness Vitality	myMod	Intra-degree
Nodes Selected	30, 21, 26, 25	NA	NA	NA	NA	33, 13, 7, 14	33, 21, 19, 29	32, 16, 28, 20	9, 25, 24, 27	9, 25, 24, 21	29, 27, 2, 26	NA
Modularity Scores	-0.01510	NA	NA	NA	NA	-0.01761	-0.04365	0.02800	-0.02899	-0.01871	-0.06334	NA

Table 4.4: Louvain Results for different greedy metrics with conductance as the global metric.

Compactness

Metrics	Clustering Coefficient	Degree Centrality	Betweenness Centrality	Eigenvector Centrality	Closeness Centrality	Coreness	Diversity	Eccentricity	Constraint	Closeness Vitality	myMod	Intra-degree
Nodes Selected	25, 30, 16, 26	31, 27, 25, 24	31, 27, 25, 24	31, 27, 25, 24	31, 27, 25, 24	31, 27, 25, 24	31, 27, 25, 24	28, 16, 18, 28	25, 27, 16, 28	31, 27, 25, 24	31, 27, 16, 24	31, 16, 10, 25
Modularity Scores	0.01645	-0.00041	-0.00041	-0.00041	-0.00041	-0.00041	-0.00041	-0.04365	0.01865	0.01887	0.03180	0.04360

Table 4.5: Louvain Results for different greedy metrics with compactness as the global metric.

4.3.2 NMI

Link Density

Metrics	Clustering Coefficient	Degree Centrality	Betweenness Centrality	Eigenvector Centrality	Closeness Centrality	Coreness	Diversity	Eccentricity	Constraint	Closeness Vitality	myNMI	Intra-degree
Nodes Selected	25, 16, 20, 10	NA	NA	NA	NA	NA	NA	28, 30, 27, 25	25, 16, 11, 10	NA	24, 4, 29, 5	31, 16, 26, 24
NMI Scores	0.69975	NA	NA	NA	NA	NA	NA	0.83447	0.80276	NA	0.71725	0.82711

Table 4.6: Louvain Results for different greedy metrics with link density as the global metric.

Conductance

Metrics	Clustering Coefficient	Degree Centrality	Betweenness Centrality	Eigenvector Centrality	Closeness Centrality	Coreness	Diversity	Eccentricity	Constraint	Closeness Vitality	myNMI	Intra-degree
Nodes Selected	30, 21, 26, 25	NA	NA	NA	NA	33, 13, 7, 14	33, 21, 19, 29	32, 16, 28, 20	NA	NA	23, 8, 24, 13	NA
NMI Scores	0.83881	NA	NA	NA	NA	0.51916	0.64279	0.57182	NA	NA	0.80198	NA

Table 4.7: Louvain Results for different greedy metrics with conductance as the global metric.

Compactness

Metrics	Clustering Coefficient	Degree Centrality	Betweenness Centrality	Eigenvector Centrality	Closeness Centrality	Coreness	Diversity	Eccentricity	Constraint	Closeness Vitality	myNMI	Intra-degree
Nodes Selected	25, 30, 16, 26	31, 27, 25, 24	31, 27, 25, 24	31, 24, 6, 9	31, 27, 25, 24	31, 27, 25, 24	31, 27, 25, 24	NA	25, 27, 16, 28	31, 27, 25, 24	24, 4, 29, 5	31, 16, 10, 25
NMI Scores	0.92491	0.83880	0.83880	0.72730	0.83880	0.83880	0.83880	NA	0.90921	0.83880	0.71725	0.58198

Table 4.8: Louvain Results for different greedy metrics with compactness as the global metric.

4.4 Genetic Algorithm approach

4.4.1 Modularity

Based on the hyperparameters mentioned previously, best nodes which optimized the modularity value of the network identified by the algorithm were, **0, 6, 1, 3**. The corresponding decrease in modularity value of the network was **0.08984**.

4.4.2 NMI

Based on the hyperparameters mentioned previously, best nodes which optimized the modularity value of the network identified by the algorithm were, **7, 2, 30, 21**. The corresponding NMI value between the original network and the new network was **0.74314**.

Chapter 5

Plots

Plots shown are for louvain community detection algorithm.

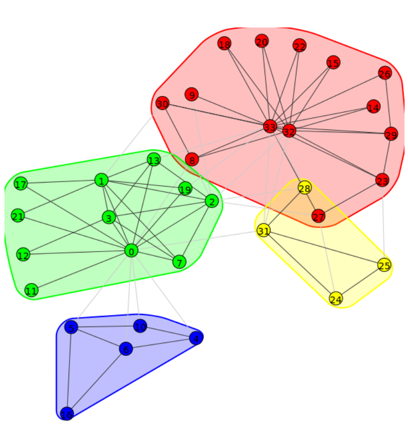

Figure 5.1: Original Network

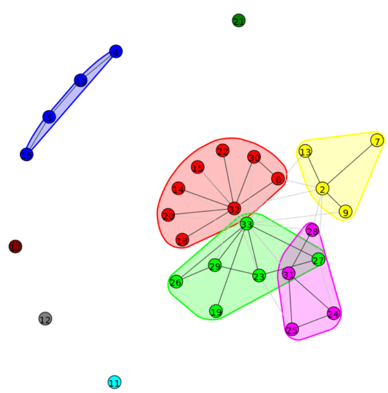

Figure 5.2: (0, 3, 6, 1) - 0.12289 - Exhaustive Modularity

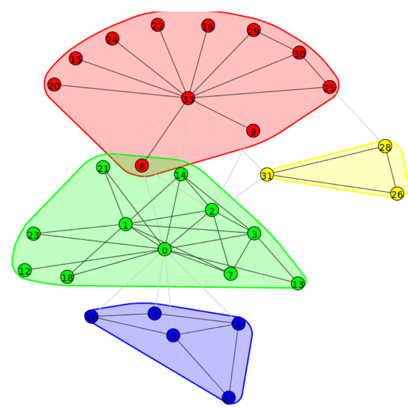

Figure 5.3: (32, 33, 5, 6) - 0.38580 - Exhaustive NMI

Figure 5.4: Original graph and Exhaustive Results

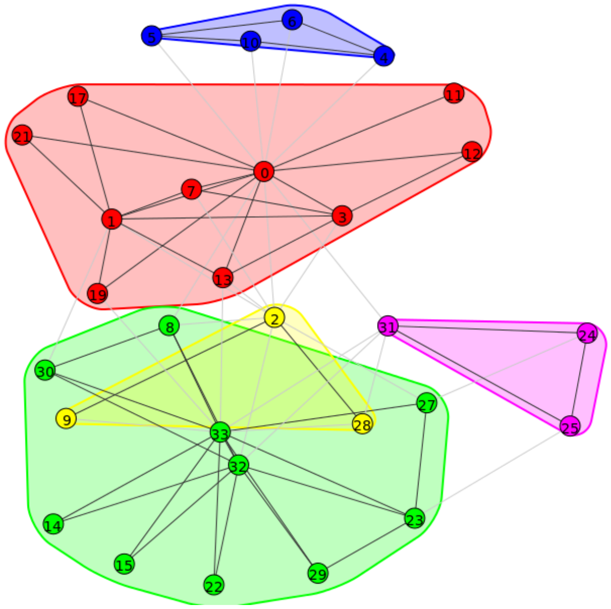

Figure 5.5: (16, 26, 18, 20) - 0.04439 - Eccentricity

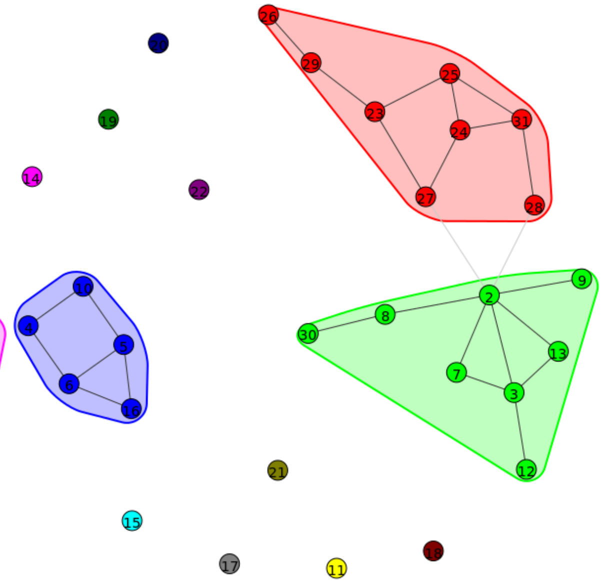

Figure 5.6: (33, 32, 0, 1) - 0.62733 - Intra Degree

Figure 5.7: Network based Greedy approach results

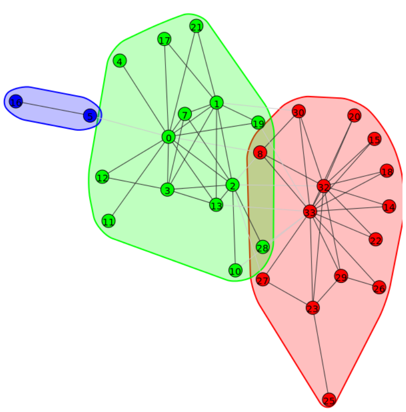

Figure 5.8: (31, 24, 6, 9) - 0.05090 - Link Density, Eigenvector Centrality

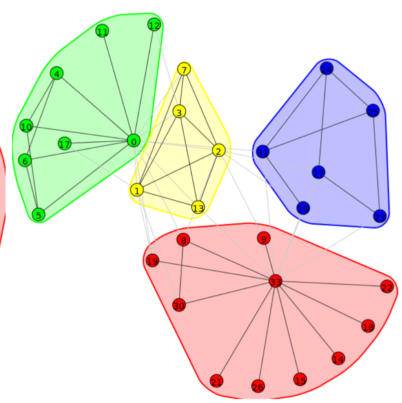

Figure 5.9: (32, 16, 28, 20) - 0.02800 - Conductance, Eccentricity

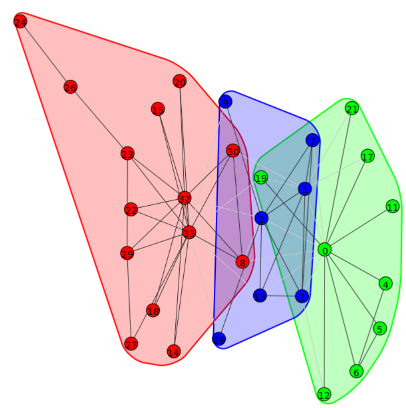

Figure 5.10: (31, 16, 10, 25) - 0.04360 - Compactness, Intra Degree

Figure 5.11: Community based Greedy approach results for Modularity

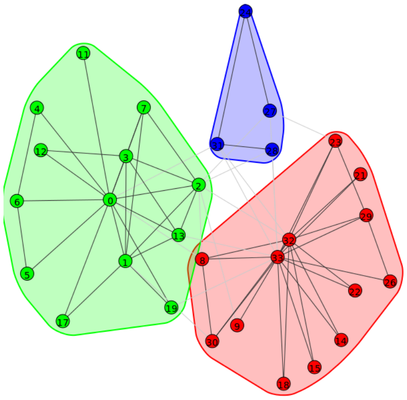

Figure 5.12: (25, 16, 20, 10) - 0.69975 - Link Density, Clustering Coefficient

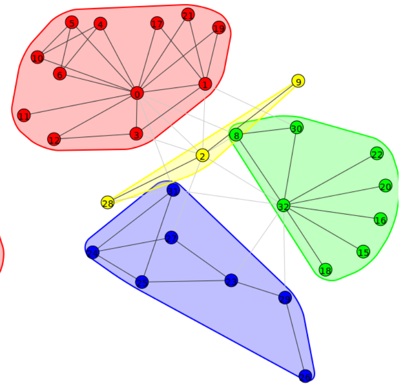

Figure 5.13: (33, 13, 7, 14) - 0.51916 - Conductance, Coreness

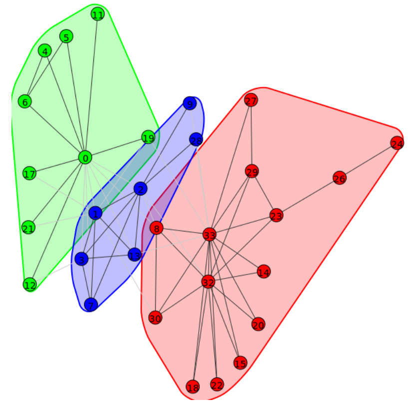

Figure 5.14: (31, 16, 10, 25) - 0.58198 - Compactness, Intra Degree

Figure 5.15: Community based Greedy approach results for NMI

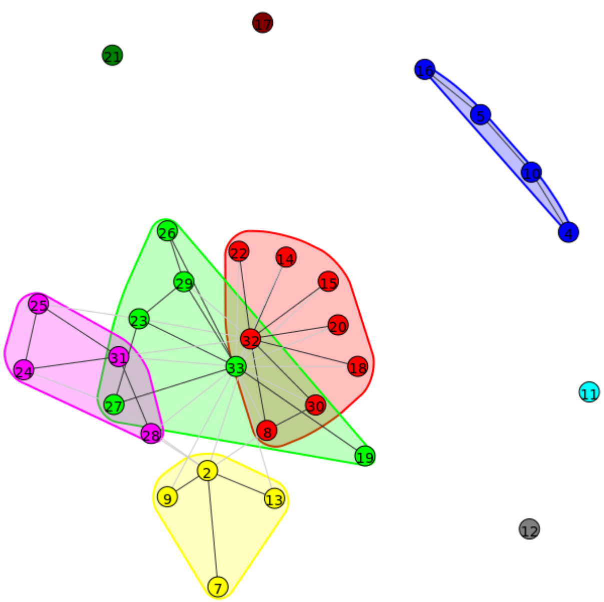

Figure 5.16: (0, 6, 1, 3) - 0.08984 - Modularity

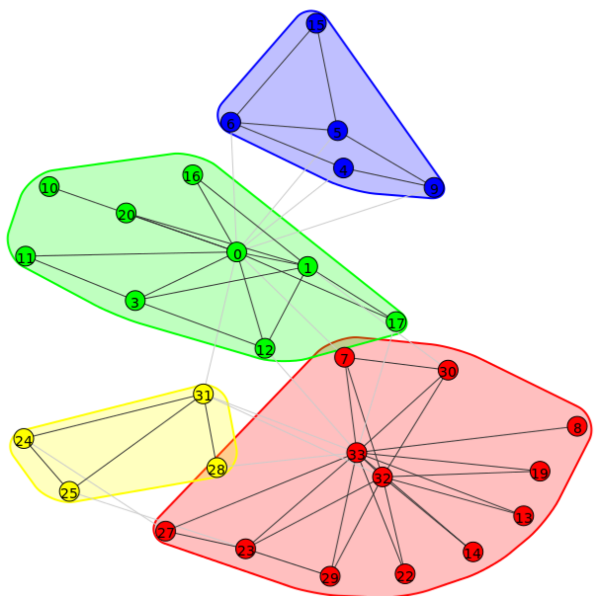

Figure 5.17: (7, 2, 30, 21) - 0.74314 - NMI

Figure 5.18: Genetic Algorithm based results

Bibliography

1. Structural Vulnerability Assessment of Community-Based Routing in Opportunistic Networks - My T. Thai
2. The Budgeted Maximum Coverage Problem - Samir Khullar
3. On the Permanence of Vertices in Network Communities - Tanmoy Chakraborty
4. Detecting Critical Nodes in Interdependent Power Networks for Vulnerability Assessment - My T. Thai
5. Community Detection in Scale-free Networks: Approximation Algorithms for Maximizing Modularity - My T. Thai
6. Scribe - Animesh Mukherjee
7. Characterizing the Community Structure of Complex Networks - Andrea Lancichinetti
8. Identifying Community Structure in Semantic Peer-to-Peer Networks - Hanhua Chen
9. Statistical Properties of Community Structure in Large Social and Information Networks - Jure Leskovec
10. Quantifying the resilience of community structures in networks - Jose E.Ramirez-Marquez
11. Community structure in social and biological networks - M. Girvan
12. Community detection in networks: Structural communities versus ground truth - Darko Hric