VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
VELLORE ▪ CHENNAI
www.vit.ac.in

School of Computing Science & Engineering

CSE3021 – Social and Information Networks

Final Review

Title of the Project: Exploiting Social Networks for Large-Scale Human Behaviour Modelling

Student Information

Sl. No.	Register No.	Name of the Student	Mobile No.	Email ID
1.	15BCE0638	Ayush Nigam	9717174800	Ayushnigam97@gmail.com
2.	15BCE0646	Ratnasambhav Priyadarshi	9087022510	ratnasambhav@gmail.com

Abstract

This project confers about various methods of discovering communities in a social network contained inside a particular location or organization. Community detection in such social networks is a difficult task because of its complexity, rapid rate of change, size and high number of linkages between different nodes. The paper proposes a Community Detection Method to discover communities in a closed social network using 2 metrics: Friendship and Colocation Metrics. At the end the efficiency of community detection can be modified by using a threshold ‘k’.

Literature Survey

In our survey we study different types of community detection algorithm used in online social network analysis. We try to find what the input parameters of these techniques are and what is the general relation of the output communities with the input parameters for each method. Which techniques are more suited towards a particular type of social network. These are some of the questions which have motivated us to conduct this survey.

The study of communities in networks draws close relation to the ideas of graph partitioning and cluster formation in sociology [8]. Owing to the fact that the number of communities in a network are unknown and their sizes/densities are varied, finding communities within a complex network often becomes computationally difficult. Despite these difficulties we have managed to develop several methods for detecting communities and it is hence needless to say that each has its own advantages and/or disadvantages. Though the clustering might seem quite intuitive at first sight, it lacks formal definition. The main components of the problem are themselves not rigorously defined, and require some extent of randomness or common sense. There are also plenty of hidden ambiguities with viable ways of resolving them. One of the most important factor for clustering is that the graphs must be sparse, i.e. the number of edges m and number of nodes n must be

approximately of the same order [8]. Community detection is broadly divided into the following categories:

A. Partitioning

These methods aim to partition the network into a particular number of groups which are usually of similar sizes and are chosen in a manner such that the edges inside the group(intra-cluster) is maximized and those between groups(inter-cluster) is minimized. These methods try to forcibly find the given number of clusters irrespective of the structure of the graph. Though these methods are not ideal for finding communities in real networks, these are simple and easy to implement [9].

B. Hierarchical clustering

Hierarchical clustering methods use a score or value which is a quantitative indication of similarity between the pair of nodes. Some of the most popular measures being the Jaccard similarity index, the cosine similarity, and the Hamming distance. Using this score we can then find the pair of nodes that are most similar and group them recursively. There are also several schemes applied to perform the grouping i..e. for example the single-linkage clustering, in which the groups are deemed to be different communities if and only if all the sets of nodes in the said groups have similarity score lower than the predefined threshold, and complete linkage clustering, where all the nodes in each group has similarity score greater than the threshold [3].

C. Modularity optimization

Modularity optimization uses a function (Modularity) that provides an estimate on the quality of a particular partitioning in the network. This method thus performs a brute force search on the possible partitioning's and finds the one with the highest modularity score. Considering that exhaustive search on all possible divisions turns out to be impractical, other algorithms that are based on near optimization methods such as greedy algorithms, simulated annealing, or spectral optimization are used [4].

D. Statistical inference

These methods attempt to fit a mathematical (generative) model onto the network data, such that it encodes the community structure. This method is regarded to be more principled in nature, and has the capacity to capture the implicit statistical properties of the network [10].

E. Clique based methods

Cliques are subgraphs which is complete which means that every node is connected to every other node. As nodes cannot be more densely packed than this, they are regarded to be a community and hence finding of communities draws analogy with the detection of cliques in a graph [2].

Proposed Problem Statement

To identify suitable candidates to form a community, we construct a similarity based graph based on social network. Each node is a user and the weight of the edge between any two user i and j is the similarity score between then. We are consulting data form friendship network and temporal collocation and co-occurrence social network (Considering all the members of a classes a social

network. Data on friendship network can be collected by asking each student whether they consider every other student (individually) as their friend or not. Temporal collocation and co- occurrence can be found using their GPS data from their phones.). We calculate the similarity score between each and every pair of users individually for each type of social network and finally use weighted average of these scores to arrive at a final similarity score for each pair of users. Edges between nodes can be filtered by rejecting all edges whose weight fall below a certain threshold value. This way we can find communities for each node by clustering all the nodes connected to it.

The number of times two users visit a common place, as done in [1] can be deceiving. For example, if A visits a music concert during the evening at a location where B had attend a book fair in the morning, A and B have a high colocation score. But these two users may be totally opposite of each others. If we use co-occurrence as well as colocation of two nodes this problem can be mitigated to some extent because if A and B are at the same location at the same time then that are most probably interested in the same topic. Hence we consider co-occurrence as well as colocation data for our implementation.

Proposed Solution

In case of friendship network, if two nodes have an edge between them, then their friendship similarity is 1 and 0 otherwise. Consider two nodes, i and j of a graph G(V,E), where V is the set of all vertices in G and E is the set of all edges in G. Mathematically,

$$if(i,j)\epsilon V, sim_f(i,j) = 1$$
$$else, sim_f(i,j) = 0,$$

where $sim_f(i,j)$ denotes the similarity value for friendship network for nodes i and j. For colocation network, we collected data of the hourly location of 50 people from 8:00 AM to 8:00 PM (12 hours). Then for any node i and j we calculated the frequency of their colocation, i.e. the number of hours when they are in the some location during the observed period. This frequency is then taken as the value of the colocation similarity of the two nodes.

$$sim_c(i,j) = \sum_{k=1}^{m} e_k,$$

where $sim_c(i,j)$ is the similarly score of the nodes i and j in colocation network.

e_k = 1 if i and j are in the same location at the k^{th} hour and 0 otherwise. m is the total number of hours observed. Finally, sum of $sim_c(i,j)$ and $sim_f(i,j)$ is taken as the final similarity measure of the two nodes i and j.

$$sim(i,j) = sim_f(i,j) + sim_c(i,j),$$

where $sim(i, j)$ is the final similarity value between i and j. Weighted average of $sim_f(i, j)$ and $sim_c(i, j)$ can also be considers. Let α be a constant such that $\alpha \epsilon [0,1)$, then we can write

$$sim(i, j) = \alpha sim_f(i, j) + (1 - \alpha) sim_c(1, j)$$

Finally we use a threshold value to filter nodes to include in the community. Each edge whose weight is less than the threshold is removed from the graph.

The implementation of the above code is done in Python using networks library to create graph. We read colocation data from a csv file ('data.csv'). Friendship network is randomly generated as a adjacency matrix with values 1 if the two nodes have a relational tie, 1 otherwise, but that too can be read for a csv file if required. The program takes three parameters. First the number of node in the graph, then the name of the file containing the the colocation data and finally the threshold for filtering nodes.

Python code:

```
import sys
import csv
from random import randint
import networkx as nx
import matplotlib.pyplot as plt


def generate_friendship_network(N):
    adj = [[0 for _ in range(0, N)] for _ in range(0, N)]
    for i in range(0, N):
        for j in range(0, i):
            adj[i][j] = adj[j][i] = randint(0, 1)
    return adj


def read_colocation_data(fileName):
    F = open(fileName, 'rb')
    reader = csv.reader(F)
    data = []
    for row in reader:
        data.append(row)
    F.close()
    return data


def colocation_similarity(data, N):
    SIMILARITY_MATRIX = [[0 for _ in range(0, N)] for _ in range(0, N)]
    for X in range(0, N):
        for Y in range(X, N):
            SIMILARITY = 0
            for i in range(0, 12):
                if data[X][i] == data[Y][i]:
```

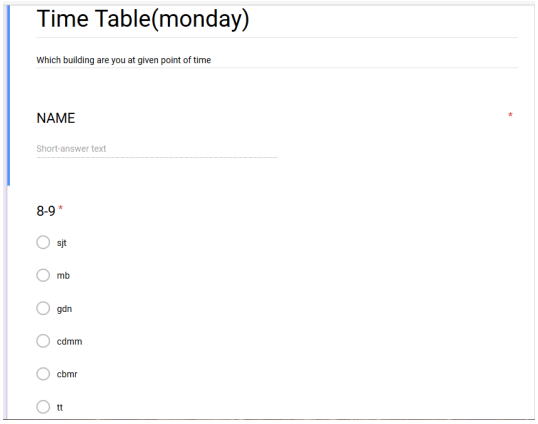
```
                    SIMILARITY += 1
                     SIMILARITY_MATRIX[X][Y] = SIMILARITY_MATRIX[Y][X] =
SIMILARITY
    for X in range(0, N):
        SIMILARITY_MATRIX[X][X] = -1
    return SIMILARITY_MATRIX


def create_network(F, S, N, THRESHOLD):
    G = nx.Graph()
    G.add_nodes_from([x for x in range(0, N)])
    # WEIGHT_MATRIX = [[0 for _ in range(0, N)] for _ in range(0, N)]
    for X in range(0, N):
        for Y in range(X, N):
            WEIGHT = F[X][Y] + S[X][Y]
            if(WEIGHT >= THRESHOLD):
                G.add_edge(X, Y)
                G[X][Y]['weight'] = WEIGHT
            # WEIGHT_MATRIX[X][Y] = WEIGHT_MATRIX[Y][X] = WEIGHT
    return G


def create_network_for_node(F, S, N, THRESHOLD, CHOICE):
    G = nx.Graph()
    G = nx.Graph()
    G.add_nodes_from([x for x in range(0, N)])
    X = CHOICE
    # WEIGHT_MATRIX = [[0 for _ in range(0, N)] for _ in range(0, N)]
    for Y in range(0, N):
        WEIGHT = F[X][Y] + S[X][Y]
        if(WEIGHT >= THRESHOLD):
            G.add_edge(X, Y)
            G[X][Y]['weight'] = WEIGHT
        # WEIGHT_MATRIX[X][Y] = WEIGHT_MATRIX[Y][X] = WEIGHT
    return G


def main(argv=None):
    if argv is None:
        argv = sys.argv
    DATA = read_colocation_data(argv[2])
    S = colocation_similarity(DATA, int(argv[1]))
    F = generate_friendship_network(int(argv[1]))

    CHOICE = raw_input(
        "Enter node number or enter All for the whole network: ")
    if CHOICE == "All":
        G = create_network(F, S, int(argv[1]), int(argv[3]))
    else:
        G = create_network_for_node(
            F, S, int(argv[1]), int(argv[3]), int(CHOICE))
```

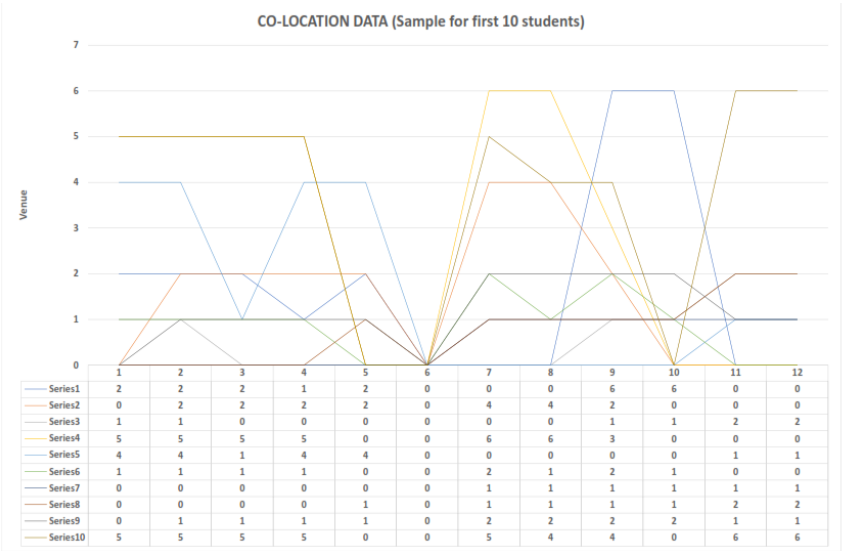
```
    plt.subplot(111)
    nx.draw(G, with_labels=True, font_weight='bold')
    plt.show()


if __name__ == '__main__':
    sys.exit(main())
```

The dataset consists of locations of 50 students inside VIT for 12 hrs (i.e. 8 a.m. to 8 p.m.). The values are as follows:

- 0 denotes Hostel
- 1 denotes SJT
- 2 denotes TT
- 3 denotes MB
- 4 denotes SMV
- 5 denotes CDMM
- 6 denotes library.

This data was collected using Google forms which can be seen below





data.csv:

2,2,2,1,2,0,0,0,6,6,0,0

0,2,2,2,2,0,4,4,2,0,0,0

1,1,0,0,0,0,0,0,1,1,2,2

5,5,5,5,0,0,6,6,3,0,0,0

4,4,1,4,4,0,0,0,0,0,1,1

1,1,1,1,0,0,2,1,2,1,0,0

0,0,0,0,0,0,1,1,1,1,1,1

0,0,0,0,1,0,1,1,1,1,2,2

0,1,1,1,1,0,2,2,2,2,1,1

5,5,5,5,0,0,5,4,4,0,6,6

4,4,5,5,5,2,4,4,4,0,0,0

3,3,0,0,0,0,5,5,5,5,0,0

5,5,5,4,4,0,4,4,4,4,6,6

0,0,6,6,5,0,5,5,0,0,0,0

2,1,1,1,0,0,0,1,1,1,0,0

3,3,4,4,0,0,3,3,3,4,4,4

0,0,0,0,0,0,3,3,3,3,3,3

5,3,3,3,5,0,1,2,6,0,0,0

1,2,2,2,0,2,1,1,1,0,0,0

3,3,0,0,0,0,3,3,3,0,0,6

2,2,2,1,1,1,0,1,1,2,1,1

4,4,4,5,6,6,0,0,0,0,3,3

2,2,1,1,0,0,0,3,3,1,1,0

5,5,5,5,0,0,6,6,0,0,0,0

2,2,2,1,2,0,0,0,6,6,0,0

3,4,4,5,1,1,0,0,6,2,2,1

1,1,0,0,1,1,1,2,2,2,6,6

4,4,4,5,6,6,6,3,3,0,0,0

2,2,2,2,0,0,6,6,4,2,6,1

3,3,3,3,0,0,3,5,3,5,5,3

0,5,3,3,6,6,1,1,2,5,5,3

0,4,4,4,0,6,2,3,2,3,5,1

0,2,2,2,6,1,3,6,3,0,5,4

2,4,5,0,4,2,1,6,5,0,1,3

3,6,6,0,2,1,4,0,5,1,1,1

4,0,4,0,4,3,4,0,4,4,6,4

0,0,0,1,1,1,1,3,5,6,6,0

2,0,2,4,5,0,1,2,6,0,0,0

1,4,1,1,4,3,0,0,0,0,1,1

1,1,3,3,3,2,2,6,6,4,0,0

1,5,0,0,0,3,4,2,6,1,4,5

1,2,2,4,4,0,6,0,0,0,3,3

2,3,3,3,0,0,0,0,5,5,1,1

5,5,0,0,0,3,1,1,1,0,1,3

4,4,4,2,3,4,5,6,4,2,2,0

0,0,0,0,1,1,1,2,0,1,1,1

1,2,1,2,1,1,2,4,4,3,0,1

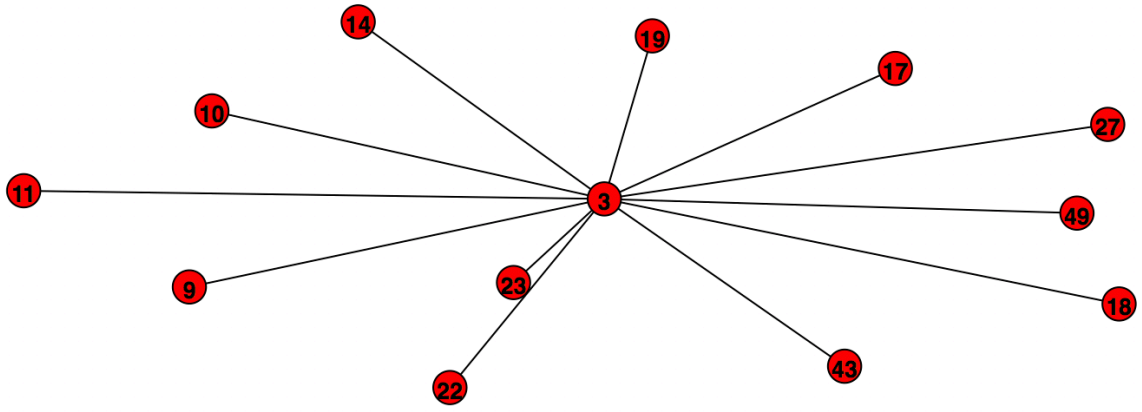
6,6,1,2,3,5,5,3,4,4,1,2

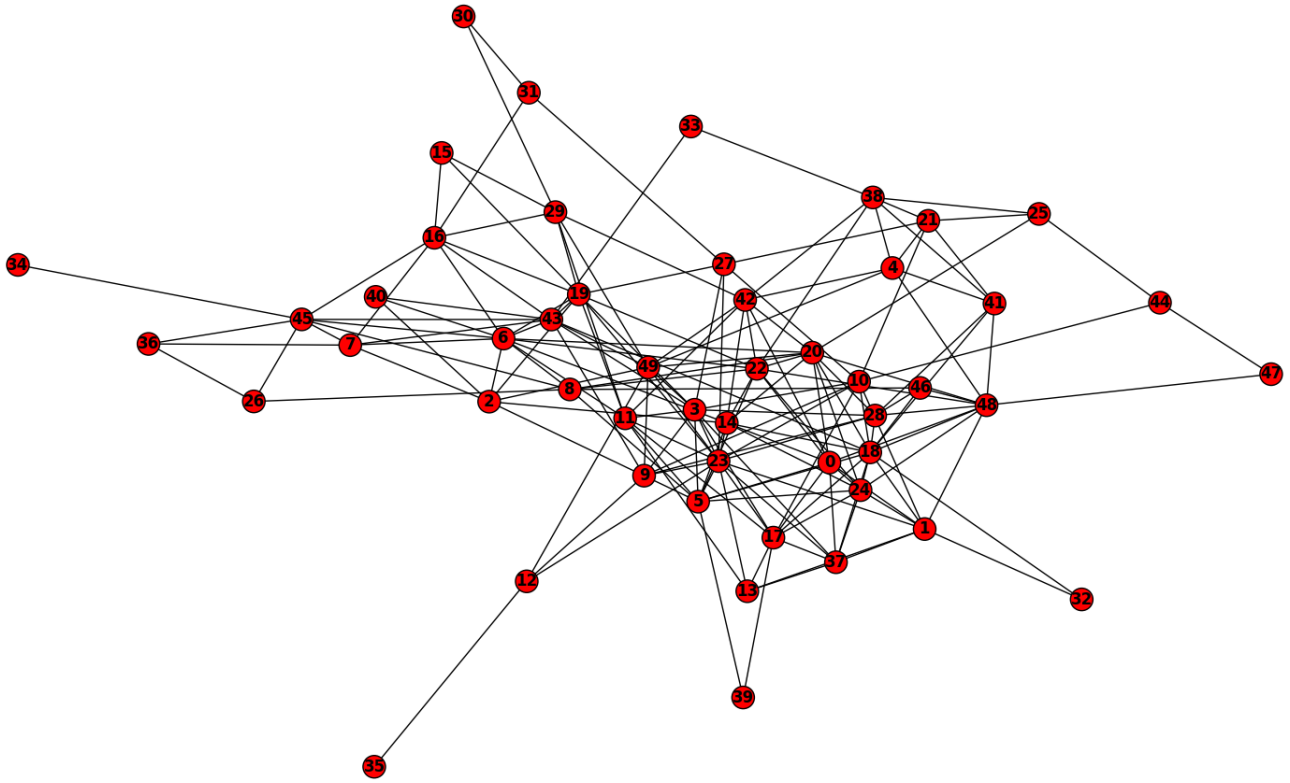
2,2,2,2,1,0,0,4,4,0,1,2

5,5,5,3,0,0,0,0,1,2,1,3

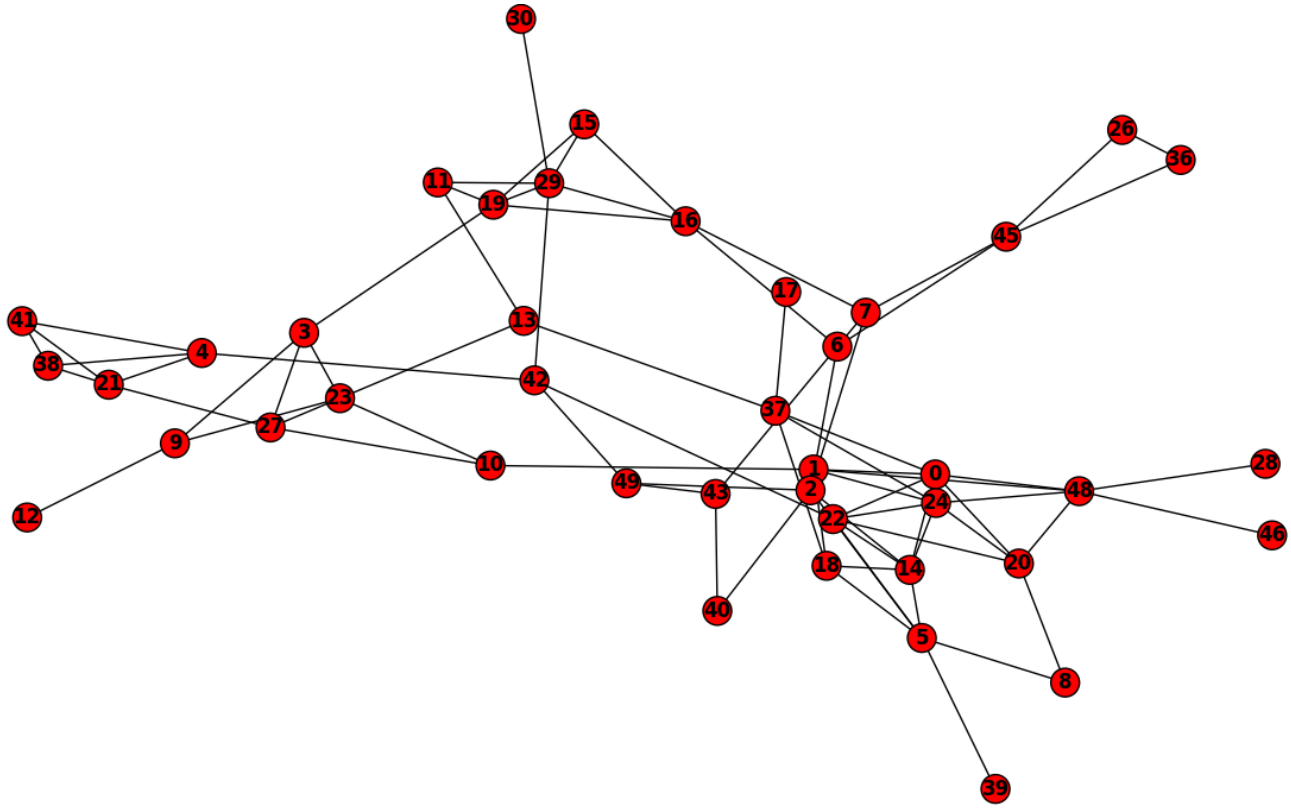
Evaluation Metrics

Below we can see network formed by all the nodes which have a edge with node 3. Threshold was set to 5 again.

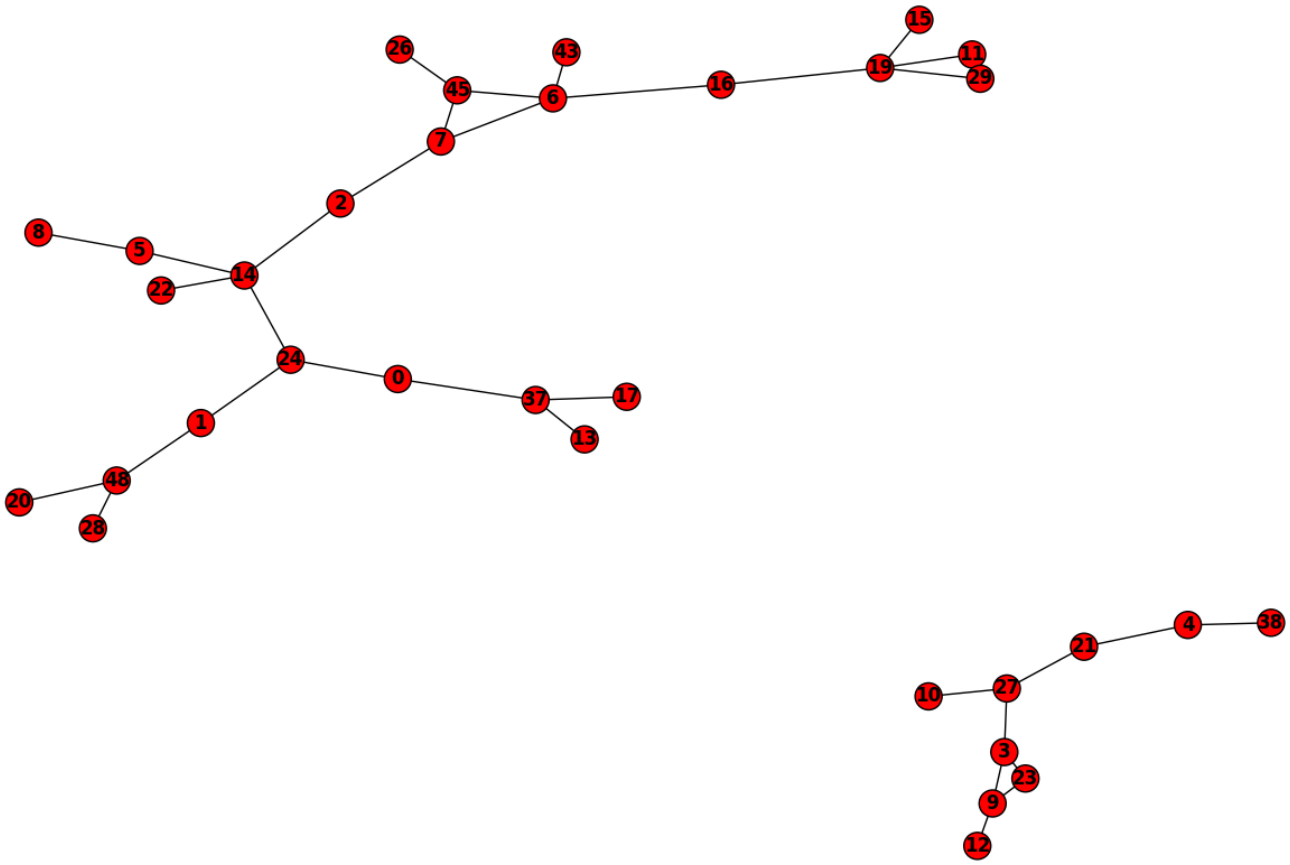




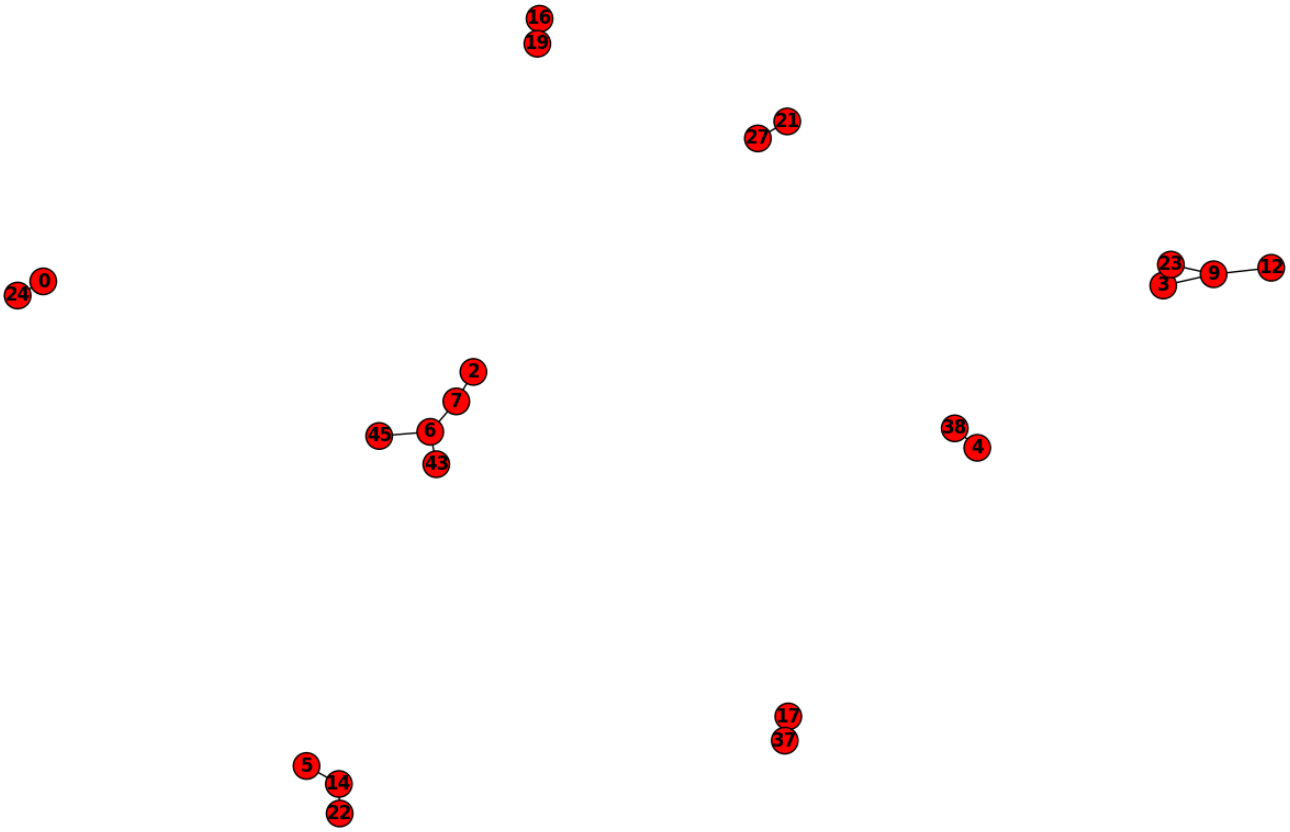
50 nodes, Threshold = 5



50 nodes, Threshold = 6



50 nodes, Threshold = 7



50 nodes, Threshold = 8

We are able to successfully segregate the members belonging to a particular community based on their weighted score which is the sum of their friendship score and the similarity score, as well as the community of each node gets successfully displayed as shown above based on the node number entered. We have also provided with a provision in order to get the community structure of the entire 50 nodes participating in the network as there is an option which allows the user to enter ‘All’ option. As a result the students who more frequently visit a particular building belong to a single community or lie in the community of a particular node.

If there is a student whose timetable does not match with any of the student among the 50 students, then the community structure of that node is completely empty. Hence the community for each node is getting successfully identified as well as for the entire network. As we can se form the results, as we increase our threshold, weaker ties are removed and only stringer and stronger ties remain.

Another application of the project is to find nodes which are connected to a particular node. Imagine we have a network of a particular department of an organization and we want to promote the node with the most number of ties within the department as the department head.. We can find the node which has the most number of relations in a node. These relations can be strong or weak and strong. With our project we can select the candidate (node) with the most number of connections in the department.

Conclusion

The paper presents an analysis of the various community detection processes that have been used on an online or localised social network affects the use of resulting community outputs e.g. target advertising, detecting information flow in social network etc. Divisive clustering methods like hierarchical clustering.

For our implementation, we have taken colocation data and used the frequency of colocation of two nodes as a metric of similarity. This model is flexible enough and can be moulded for a different application. For example, lets consider the example of tweets. We can collect tweets for all users under observation and analyse them for keywords which can act as an indicator of their interests. Now we can take the number of occurrence of these keywords in their tweets and their context (positive or negative) and compare it with other users and find their mutual similarity. Similarly, we can study the purchases of all users under observation and group users with similar purchases in one community. Such communities can be used for targeted advertisements. This method of detecting communities is useful for organizations to find which employees are more comfortable and productive in the company of another employee. Detecting sub-communities in a larger community is also a key application of this model.

Clustering based on Colocation and Friendship Metric is also a kind of Divisive clustering and does not require an explicit definition of the number of clusters are deemed to be a base principle for modern community detection algorithms. This is because algorithms of this category are fundamentally quality function based cluster detection mechanisms. The researches discussed above are focused on the type of network that it is being applied on and hence the community detection methods are specific and hence there is need for general scalable approaches that apply to any kind of online social network.

This method of detecting communities is useful for organizations to find which employees are more comfortable and productive in the company of another employee. This can also be utilized in detecting sub-communities within a larger community.

References

[1] Lane, Nicholas D., et al. "Exploiting social networks for large-scale human behavior modeling." *IEEE Pervasive Computing* 10.4 (2011): 45-53.

[2] Lane, Nicholas D., et al. "Community similarity networks." *Personal and ubiquitous computing* 18.2 (2014): 355-368.

[3] Iglesias, Josué, et al. "A ubiquitous activity-monitor to prevent sedentariness." *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. IEEE, 2011.

[4] Zhang, Yu, and Mihaela van der Schaar. "Information production and link formation in social computing systems." *IEEE Journal on Selected Areas in Communications* 30.11 (2012): 2136-2145.

[5] D. Peebles, "Community-Guided Learning: Exploiting Mobile Sensor Users to Model Human Behavior", Proc. Conf. Artificial Intelligence (AAAI 10), pp. 1600-1606, 2010.

[6] M. Stikic, K. Laerhoven, B. Schiele, "Exploring Semi-Supervised and Active Learning for Activity Recognition", Proc. 2008 Int'l Symp. Wearable Computers (ISWC 08), pp. 81-88, 2008.

[7] Longstaff, Brent, Sasank Reddy, and Deborah Estrin. "Improving activity classification for health applications on mobile devices using active and semi-supervised learning." Pervasive Computing

[8] Fortunato, S. (2010). Community detection in graphs. Physics reports.

[9] Newman, M. E. (2004). Finding and evaluating community structure in networks. Physical review E.

[10] Jain, A. K. (2000). Statistical pattern recognition: A review. IEEE Transactions on pattern analysis and machine intelligence, 4-37.