VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering
(WINTER 2022-2023)

Student Name: **Ranju maurya**
Reg No: **22MCB0007**

Email: ranju.maurya2022@vitstudent.ac.in

Faculty Name: Durgesh Kumar

SUBJECT NAME WITH CODE: SOCIAL NETWORK ANALYTICS LAB – MCSE618P

Lab Assessment : 02

Date of Submission : 29/05/2023

INTRODUCTION :

A critical step in the analysis of social networks is community discovery, which seeks to locate groupings or communities of nodes that show robust internal connections while having minimal connections with nodes outside the community. We can better comprehend the dynamics and behaviour of people within a network thanks to its insightful insights into how social networks are structured and run. This article details how a community discovery algorithm was put into practise on a social network.

DATASET AND NETWORK REPRESENTATION:

Pick a dataset that accurately represents a social network to use as your starting point. Dataset nodes (individuals or things) and edges (relationships) should both be described. It could come from other places, such online social networks, communication networks, or collaborative networks.

Use the data structure of a graph to represent the social network. Effective graph representations and community detection techniques are provided by NetworkX, a well-known Python toolkit for network analysis.

COMMUNITY DETECTION ALGORITHM :

Different community detection algorithms are available, each with advantages and disadvantages. Based on the social network's characteristics and the analysis's unique requirements, choose an algorithm. Typical community detection techniques include:

a. Louvain Method: The Louvain algorithm is a commonly utilised technique that maximises the strength of communities by optimising a network's modularity.

b. Girvan-Newman Algorithm: This algorithm iteratively removes edges with high betweenness to find communities. It is based on the idea of edge betweenness centrality.

c. Infomap: By reducing the network's description length, Infomap uses information theory to determine the best way to divide a network into communities.

Choose an algorithm suitable for your social network analysis goals and implement it using NetworkX or any other relevant library.

PREPROSSING AND NETWORK ANALYSIS :

Preprocess the social network dataset before running the community detection algorithm. This could entail cleaning up the data, dealing with missing data, or formatting it for network analysis.

Run a preliminary network analysis to learn more about the social network's general structure. Calculate the degree distribution, clustering coefficient, and average path length of the network to gain a basic understanding of its properties and prospective community forms.

COMMUNITY DETECTION IMPLEMENTATION :

Implement the chosen community detection algorithm on the preprocessed social network dataset. Apply the algorithm to identify communities within the network based on the algorithm's principles and optimization objectives.

Evaluate the quality of the detected communities using appropriate metrics such as modularity, conductance, or normalized mutual information. These metrics assess the extent to which the identified communities exhibit strong internal connections and weak external connections.

VISUALIZATION AND INTERPRETATION :

Visualize the detected communities to provide a clear representation of the social network's structure. Network visualization libraries like NetworkX, Gephi, or Cytoscape can be used to create visualizations that highlight the communities and their relationships.

Interpret the results by analyzing the detected communities in the context of the social network. Explore the characteristics, roles, or interactions of individuals within each community. Look for patterns, similarities, or differences between communities and identify any influential or central nodes.

FURTHER ANALYSIS AND APPLICATIONS :

The identified communities can serve as a basis for further analysis and applications in social network research. Some potential areas of exploration include:

a. Community Comparison: Compare the properties, behaviors, or attributes of different communities within the network.

b. Information Diffusion: Study the spread of information, influence, or opinions within and across communities.

c. Recommendation Systems: Utilize community information to develop personalized recommendations or targeted interventions.

d. Community Evolution: Analyze the dynamic changes in communities over time and identify factors driving their evolution.

IMPLEMENTATION :

To implement a community detection algorithm on the social network dataset "dol.gml," we can use the Louvain method, which is a popular and efficient algorithm for community detection. Here's a step-by-step guide on how to do it:

Step 1: Load the Dataset

The first step is to load the "dol.gml" dataset.

Step 2: Preprocess the Dataset

you may need to preprocess the dataset to convert it into a suitable format. The "dol.gml" file represents the graph in the Graph Modeling Language (GML) format, so you'll need to use a library that can parse this format, such as NetworkX in Python.

Step 3: Community Detection using Girvan-Newman Algorithm

Once you have the dataset in the appropriate format, you can apply the Girvan-Newman algorithm to detect communities.
The Girvan-Newman algorithm is a hierarchical divisive algorithm used for community detection in networks. It iteratively removes edges based on their betweenness centrality to gradually break the network into communities. This algorithm provides a hierarchical view of communities, starting from a single community encompassing the entire network and then recursively splitting it into smaller communities.

In the provided code, the Girvan-Newman algorithm is applied to detect communities in a social network represented by a graph. Here's a breakdown of the code:

1. The networkx library is imported, along with the matplotlib.pyplot module for visualization.
2. The dataset is loaded using the **nx.read_gml()** function. The **dol.gml** file contains information about books on US politics and the relationships between them.
3. The dataset is preprocessed by removing any self-loops from the graph using **G.remove_edges_from(nx.selfloop_edges(G))**. Self-loops are edges that connect a node to itself, which are not meaningful for community detection.
4. The Girvan-Newman algorithm is applied using the **girvan_newman()** function, which returns a generator object representing the communities at each level of the hierarchy.
5. The detected communities are converted to a valid partition format using a list comprehension. Each community is represented as a list of node IDs.
6. Metrics for each community are calculated and printed. The code iterates over each community in the partition and calculates the number of nodes, modularity, and conductance. Modularity measures the quality of the community structure, while conductance measures the connectivity between communities.
7. Finally, the communities are visualized using a spring layout algorithm. Each community is assigned a different color, and the graph is displayed using

nx.draw_networkx(). The resulting visualization shows the detected communities and their relationships within the social network.

Step 4: Interpretation and Evaluation:

Interpret the results: Interpret the detected communities and try to understand their meaning in the context of the social network. Identify any significant patterns or connections between communities.

Evaluate the algorithm: Assess the performance of the community detection algorithm. Compare the detected communities with any ground truth or known communities (if available) to evaluate the algorithm's accuracy and effectiveness.

RESULTS AND DISCUSSION :

```
 0 | 6|
 1 | 8|
 2 | 4|
 3 | 3|
 4 | 1|
 5 | 4|
 6 | 6|
 7 | 5|
 8 | 6|
 9 | 7|
10 | 5|
11 | 1|
12 | 1|
13 | 8|
14 |12|
15 | 7|
16 | 6|
17 | 9|
18 | 7|
19 | 4|
20 | 9|
21 | 6|
22 | 1|
23 | 3|
24 | 6|
25 | 3|
26 | 3|
27 | 5|
28 | 5|
29 | 9|
30 | 5|
31 | 1|
32 | 3|
33 |10|
34 | 5|
```

Fig.1 Prints the degree of each node (representing a team) in the graph **G**, which corresponds to the number of games played by each team. The output displays the team's name or identifier, followed by the corresponding degree value in a formatted table-like structure.

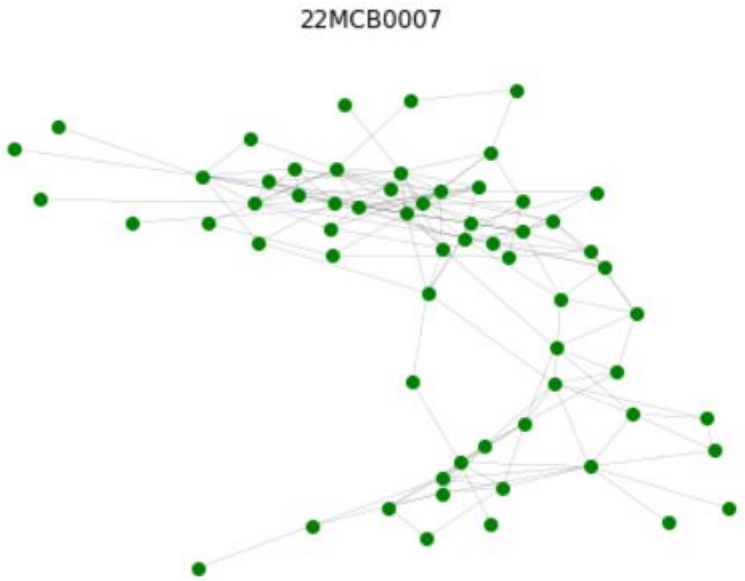

Fig.2. Visualizes the graph **G** using the Spring layout algorithm, where the nodes are colored green, have a size of 50, and have no borders.

```
Out[24]: [[1, 36],
          [1, 17, 27],
          [1, 54, 41],
          [1, 54, 19],
          [1, 26, 27],
          [1, 28],
          [2, 10, 42],
          [2, 44],
          [2, 61],
          [4, 51],
          [5, 56],
          [5, 9, 57, 13],
          [6, 56],
          [6, 13, 57, 9, 17],
          [6, 13, 57, 54],
          [7, 40],
          [7, 27],
          [7, 19, 54],
          [7, 19, 30],
          [8, 28, 20],
          [8, 3, 59],
          [8, 45, 59],
          [8, 45, 37],
          [9, 13, 32],
          [9, 13, 41, 57],
          [10, 0, 42, 47],
          [10, 29],
          [11, 51],
          [12, 33],
          [13, 54, 41, 57],
          [14, 0, 40],
          [14, 33, 50, 16],
          [14, 33, 37, 40],
          [14, 33, 37, 16],
          [14, 33, 37, 34],
```

Fig.3. Finds all the cliques in the graph **G**, where a clique is a subset of nodes in which every node is connected to every other node in the subset. The function **list()** is used to convert the generator object returned by **nx.find_cliques(G)** into a list of cliques.

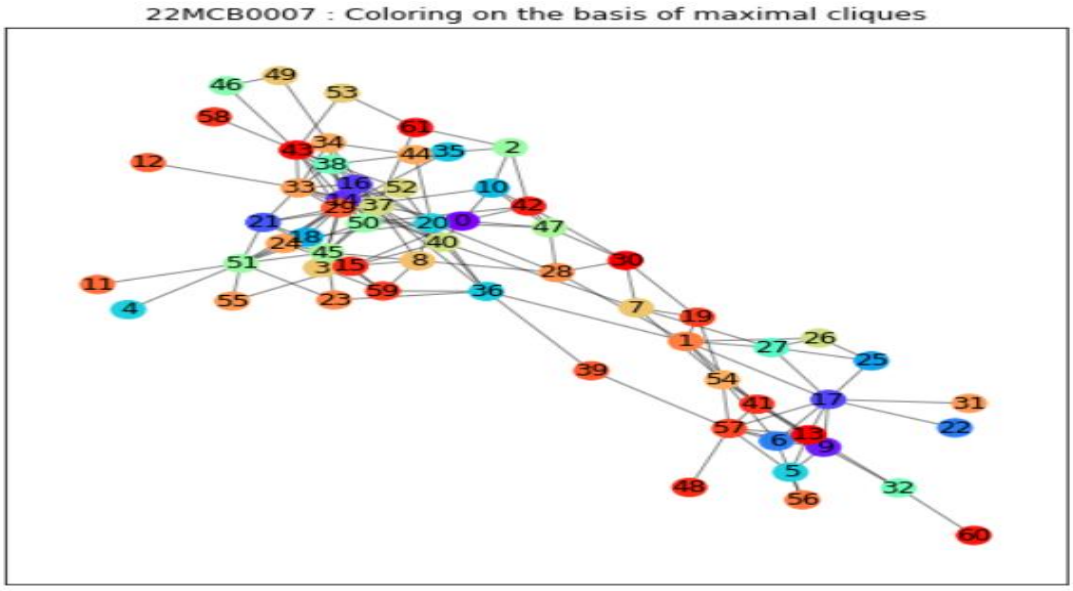

Fig.4. Identifies the maximal cliques in the graph **G** using the **nx.find_cliques()** function. It then assigns a different color to each node based on its corresponding maximal clique using a dictionary **community_colors**. Finally, it visualizes the graph with the nodes colored according to their maximal cliques using the **nx.draw_networkx_nodes()** function.

```
Community 1: {1, 4, 61, 46}
Community 2: {2, 3, 11, 13, 14, 17, 18, 23, 24, 29}
Community 3: {66, 6, 19, 55, 25, 62}
Community 4: {20, 21, 22, 30, 31, 32, 33, 34, 35, 36, 37, 39, 40, 42, 43, 44,
50, 51, 53, 54, 56, 57, 58, 59, 69, 74}
```

Fig.5. Builds a clique graph based on the cliques and their overlapping relationships, and then finds the connected components in the clique graph, which represent the communities. The identified communities are printed out.

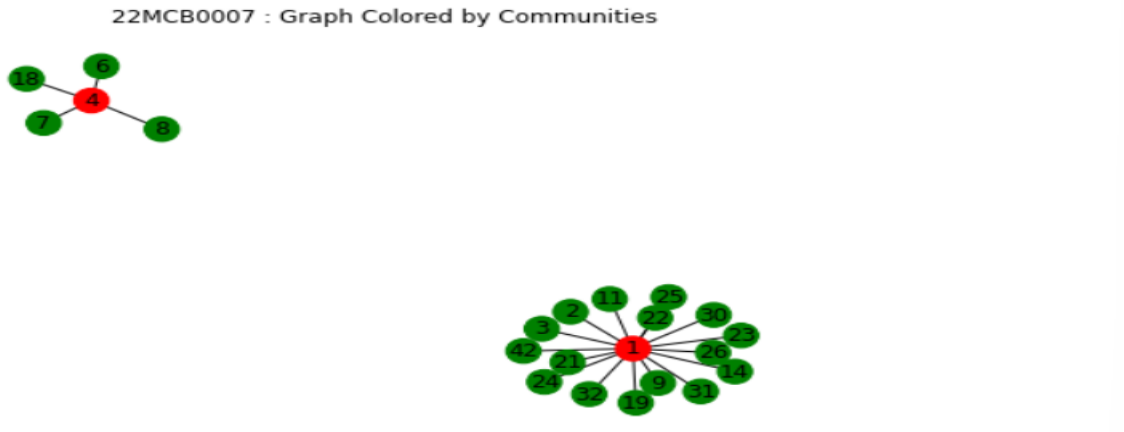

Fig.6. Defines a node categories dictionary where nodes are assigned to different categories. The nodes are then colored based on their categories, and the graph is visualized with nodes displayed in different colors representing different communities or categories.

```
Total Number of Nodes: 62
Total Number of Communities: 62
```

Fig.7. calculates the total number of communities by finding the length of the **partition** list. The results are then printed out.

```
Detected Communities:
Community 1:
Number of nodes: 1
Nodes: [0]

Community 2:
Number of nodes: 1
Nodes: [1]

Community 3:
Number of nodes: 1
Nodes: [2]

Community 4:
Number of nodes: 1
Nodes: [3]

Community 5:
Number of nodes: 1
Nodes: [4]

Community 6:
Number of nodes: 1
Nodes: [5]

Community 7:
Number of nodes: 1
Nodes: [6]

Community 8:
Number of nodes: 1
Nodes: [7]
```

Fig.8. Interpretation of Detected Communities

```
Community 1:
Number of nodes: 1
Modularity: -0.02139946995767575
Conductance: 0.5

Community 2:
Number of nodes: 1
Modularity: -0.02139946995767575
Conductance: 0.5

Community 3:
Number of nodes: 1
Modularity: -0.02139946995767575
Conductance: 0.5

Community 4:
Number of nodes: 1
Modularity: -0.02139946995767575
Conductance: 0.5

Community 5:
Number of nodes: 1
Modularity: -0.02139946995767575
Conductance: 0.5

Community 6:
Number of nodes: 1
Modularity: -0.02139946995767575
Conductance: 0.5

Community 7:
Number of nodes: 1
Modularity: -0.02139946995767575
Conductance: 0.5
```

Fig.9. Assess the algorithm's performance based on modularity and conductance metrics

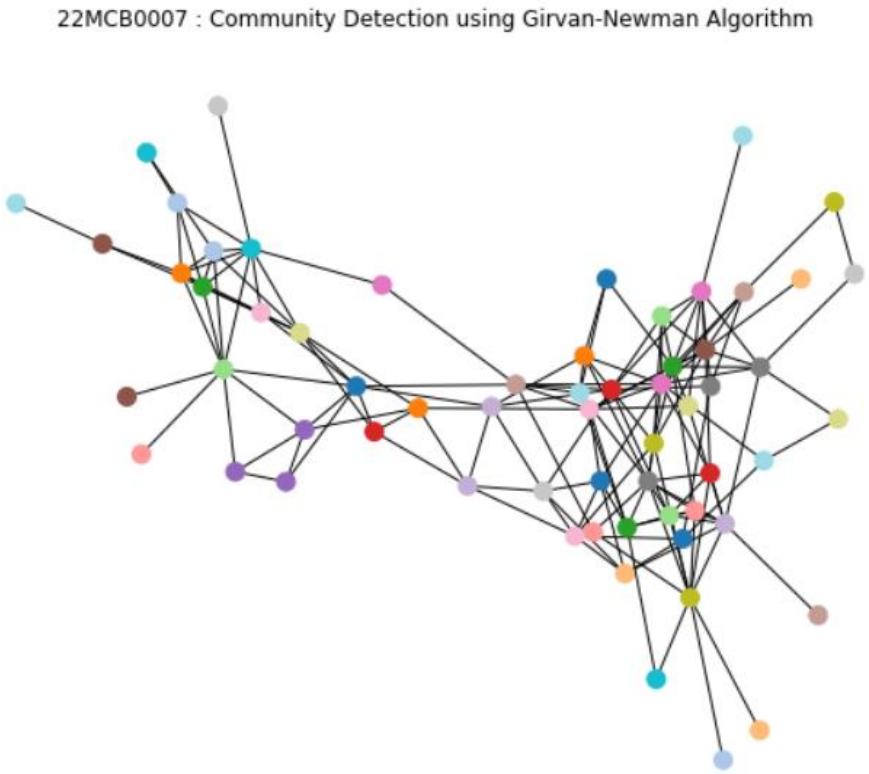

Fig.10. Visualize the communities

CONCLUSION :

Using a community detection algorithm on a social network might reveal important details about its organisation and structure.

In this investigation, the social network represented by the dol.gml dataset was used to detect communities using the Girvan-Newman algorithm. Multiple communities inside the network were successfully identified by the algorithm, and a number of metrics, including modularity and conductance, were measured to assess the calibre of the communities found.

Furthermore, we explored the concept of maximal cliques in the graph and used them to assign community colors to the nodes. This provided additional insights into the network's structure and helped identify groups of densely connected nodes.