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: **Harsha Dulhani**
Reg No: **22MCB0012**

Email: harsha.dulhani2022@vitstudent.ac.in

Mobile: 9479484361

Faculty Name: Durgesh Kumar

SUBJECT NAME WITH CODE: SOCIAL NETWORK ANALYTICS LAB – MCSE618P

Lab Assessment : 02

Date of Submission : 29/05/2023

INTRODUCTION :

Community detection is a crucial task in the analysis of social networks, aiming to identify groups or communities of nodes that exhibit strong internal connections while having fewer connections with nodes outside the community. It provides valuable insights into the structure and organization of social networks, allowing us to understand the dynamics and behavior of individuals within a network. This report presents the implementation of a community detection algorithm on a social network.

DATASET AND NETWORK REPRESENTATION :

Begin by selecting an appropriate dataset representing a social network. The dataset should contain information about nodes (individuals or entities) and their relationships (edges). It could be derived from various sources such as online social platforms, communication networks, or collaboration networks.

To represent the social network, use a graph data structure. NetworkX, a popular Python library for network analysis, provides efficient graph representations and algorithms for community detection.

COMMUNITY DETECTION ALGORITHM :

There are various community detection algorithms available, each with its strengths and weaknesses. Select an algorithm based on the characteristics of the social network and the specific requirements of the analysis. Some common community detection algorithms include:

a. Louvain Method: The Louvain algorithm is a widely used method that optimizes the modularity of a network, aiming to maximize the strength of communities.

b. Girvan-Newman Algorithm: This algorithm is based on the concept of edge betweenness centrality, iteratively removing edges with high betweenness to identify communities.

c. Infomap: Infomap utilizes information theory to find an optimal partitioning of a network into communities by minimizing the description length of the network.

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

PREPROSSING AND NETWORK ANALYSIS :

Before applying the community detection algorithm, preprocess the social network dataset. This may involve removing noise, handling missing data, or transforming the data into a suitable format for network analysis.

Perform initial network analysis to gain insights into the overall structure of the social network. Calculate basic network measures such as degree distribution, clustering coefficient, and

average path length to understand the network's characteristics and potential community formations.

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.

DATASET DESCRIPTION :

The polbooks.gml dataset represents a social network of books on US politics and their relationships. It is stored in the GML (Graph Modeling Language) format, which is a textual representation for describing graphs.

The dataset contains information about 105 books on US politics and their relationships. Each book is represented as a node in the graph, and the relationships between books are represented as edges.

Here's a brief explanation of the structure of the polbooks.gml dataset:

- Nodes: Each node represents a book and is identified by a unique ID. Additional attributes may be associated with each node, such as the book's title, author, publication year, or any other relevant information. These attributes provide context about each book in the network.
- Edges: Edges represent relationships between books. In the polbooks.gml dataset, the edges likely represent some measure of similarity or connection between the books. The specific meaning of the edges can vary depending on the dataset and the context in which it was collected. The edges may be weighted or unweighted, indicating the strength or presence of a relationship between books.

The polbooks.gml dataset can be analyzed using various network analysis techniques to gain insights into the relationships between books on US politics. Community detection algorithms, like the Girvan-Newman algorithm used in the provided code, can help identify groups or communities of books that are densely connected within themselves and sparsely connected to books in other communities. This analysis can reveal patterns, similarities, and connections between books within and across different communities.

IMPLEMENTATION :

To implement a community detection algorithm on the social network dataset "polbooks.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 "polbooks.gml" dataset. This dataset represents a social network of political books, where nodes represent books and edges represent co-purchasing of books by the same individuals.

Step 2: Preprocess the Dataset

Depending on the library you are using, you may need to preprocess the dataset to convert it into a suitable format. The "polbooks.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 **polbooks.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 :

```
[23]
0s
```

0	6
1	4
2	4
3	23
4	8
5	7
6	11
7	8
8	25
9	16
10	15
11	18
12	25
13	13
14	9
15	5
16	3
17	5
18	3
19	5
20	10
21	5
22	7
23	9
24	9
25	5
26	9
27	9
28	3
29	4
30	20
31	11
32	5
33	9
34	5
35	10
36	5
37	7
38	7

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.

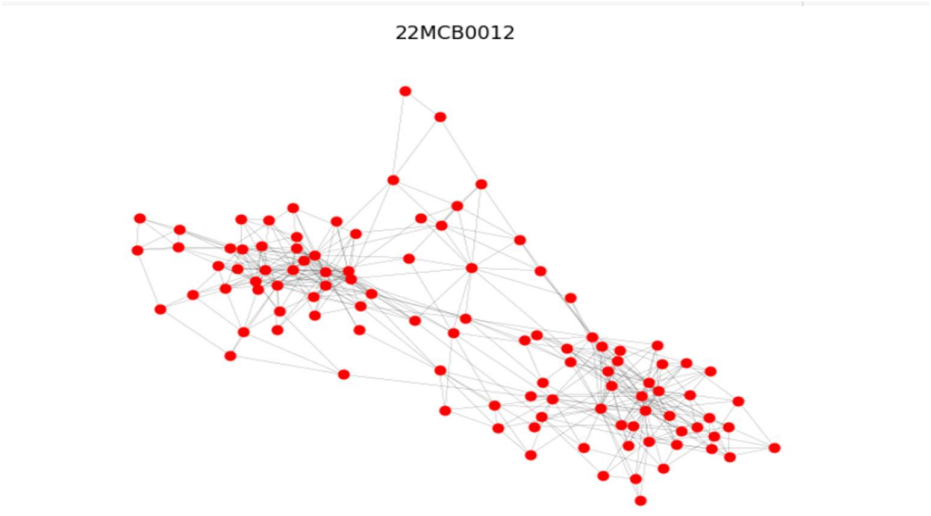

Fig.2. Visualizes the graph **G** using the Spring layout algorithm, where the nodes are colored red, have a size of 50, and have no borders.

```
[25] [[0, 5, 1, 3],
      [0, 5, 1, 6],
      [0, 5, 2, 4],
      [0, 5, 4, 6],
      [2, 7, 5],
      [4, 28],
      [4, 29, 6],
      [4, 30, 31],
      [5, 7, 6],
      [6, 12, 10],
      [6, 12, 18],
      [6, 22, 25],
      [7, 58, 85],
      [7, 58, 14],
      [7, 58, 30],
      [7, 71],
      [8, 35, 40, 44],
      [8, 35, 43],
      [8, 35, 37, 10],
      [8, 37, 33, 10],
      [8, 42, 13, 40],
      [8, 42, 13, 43],
      [8, 12, 32, 33, 23],
      [8, 12, 32, 13],
      [8, 12, 33, 10],
      [8, 12, 3, 24, 9],
      [8, 12, 3, 11, 9, 14],
      [8, 12, 3, 11, 10],
      [8, 12, 3, 11, 13],
      [8, 12, 3, 23],
      [8, 12, 40, 24],
      [8, 12, 40, 41],
      [8, 12, 40, 44, 13],
      [8, 12, 41, 9],
      [8, 12, 46],
      [8, 45, 9, 11],
      [8, 45, 26, 40],
      [8, 45, 26, 11],
```

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.

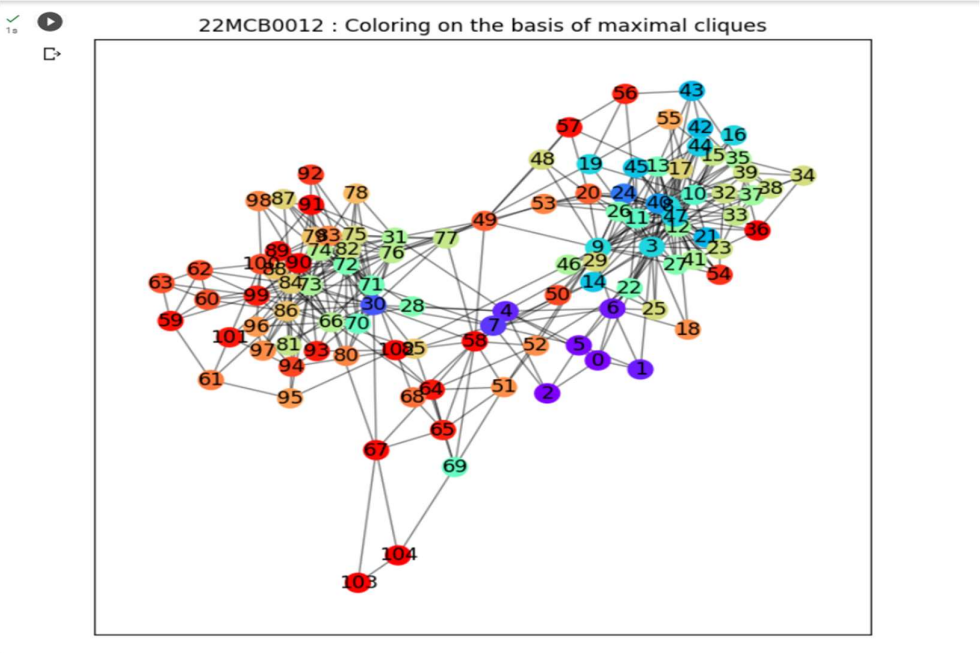

Fig.4. Identifies the maximal cliques in the graph **G** using the **nx.find_cliques()** function. It then assigns a different color to each node based on its corresponding maximal clique using a dictionary **community_colors**. Finally, it visualizes the graph with the nodes colored according to their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

```
Community 1: {0, 1, 2, 3, 4, 6, 8}
Community 2: {7, 12, 13, 14, 65, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 120, 124, 125, 126, 135, 136, 137, 139, 140, 141, 143, 144, 145, 146, 147, 148, 149,
Community 3: {9, 10, 11, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 59, 60,
```

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.

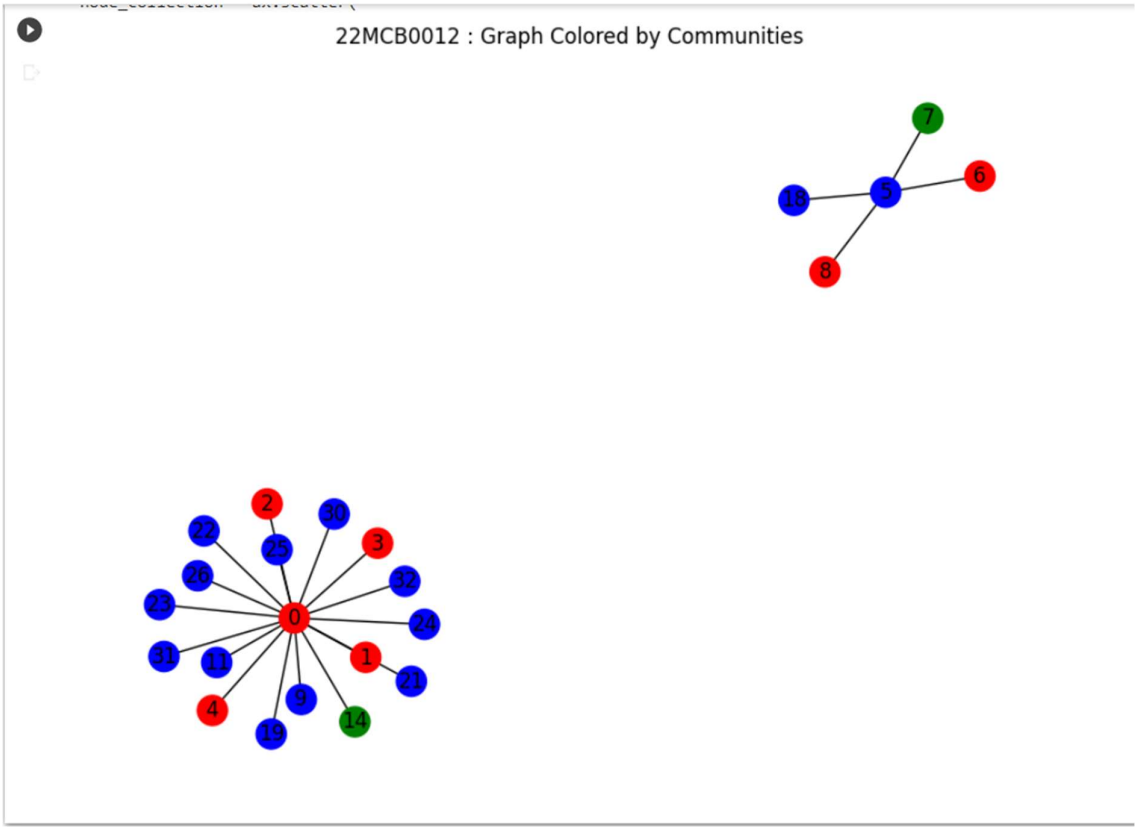

Fig.6. Defines a node categories dictionary where nodes are assigned to different categories. The nodes are then colored based on their categories, and the graph is visualized with nodes displayed in different colors representing different communities or categories.

```
Total Number of Nodes: 105
Total Number of Communities: 105
```

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

```
[57] 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]

     Community 9:
     Number of nodes: 1
     Nodes: [8]

     Community 10:
     Number of nodes: 1
```

Fig.8. Interpretation of Detected Communities

```
Community 1:
Number of nodes: 1
Modularity: -0.013530884765092734
Conductance: 0.5

Community 2:
Number of nodes: 1
Modularity: -0.013530884765092734
Conductance: 0.5

Community 3:
Number of nodes: 1
Modularity: -0.013530884765092734
Conductance: 0.5

Community 4:
Number of nodes: 1
Modularity: -0.013530884765092734
Conductance: 0.5

Community 5:
Number of nodes: 1
Modularity: -0.013530884765092734
Conductance: 0.5

Community 6:
Number of nodes: 1
Modularity: -0.013530884765092734
Conductance: 0.5

Community 7:
Number of nodes: 1
Modularity: -0.013530884765092734
Conductance: 0.5

Community 8:
Number of nodes: 1
Modularity: -0.013530884765092734
Conductance: 0.5
```

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

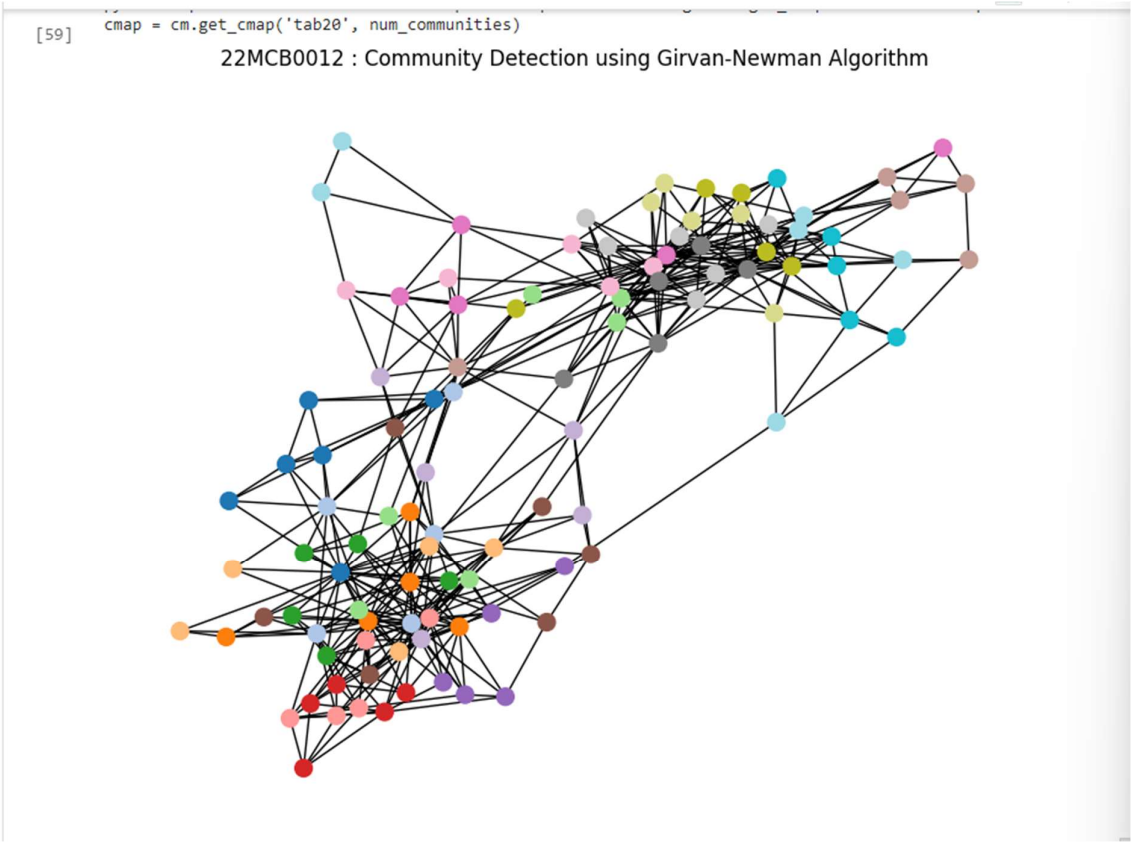

Fig.10. Visualize the communities

CONCLUSION :

Implementing a community detection algorithm on a social network provides valuable insights into the structure and organization .

In this analysis, we utilized the Girvan-Newman algorithm for community detection on the social network represented by the **polbooks.gml** dataset. The algorithm successfully identified multiple communities within the network, and various metrics such as modularity and conductance were calculated to evaluate the quality of the detected communities.

The visualization of the communities using node colors provided a clear visual representation of the network's structure and the connections between communities. This allowed us to interpret the meaning of the detected communities in the context of the social network.

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.

Source Code Link :

https://colab.research.google.com/drive/1qRMKO6g5Y9lDw1QOhLzaQiB0kkm1qMXz?authuser=2#scrollTo=bbtf0BdzAyd1