School of Computer Science and Engineering

(WINTER 2022-2023)

Student Name: Ketan Kolte

Reg No: 22MCB0016

Email: ketansanjay.kolte2022@vitstudent.ac.in

Mobile: 8329324714

Faculty Name: DURGESH KUMAR

Subject Name with code: SOCIAL NETWROK ANALYTICS LAB – MCSE618P

Date : 01/06/2023

TABLE OF CONTENTS

INTRODUCTION

This project report presents the implementation of three community detection algorithms on social network data. Specifically, Girvan-Newman method, Fast greedy method, and Modularity Maximization method have been implemented and their effectiveness in identifying communities has been evaluated. The aim of this project is to gain insights into the underlying structure of social networks through the application of these algorithms.

Social networks are complex systems that exhibit rich structural patterns and understanding these patterns can be beneficial for various applications such as marketing, recommendation systems, and information dissemination. Community detection algorithms are widely used for identifying groups of nodes within a network which are more densely connected with each other than with nodes outside the group.

In this project, we have applied three community detection algorithms on social network data and compared their performance based on metrics such as modularity score, number of detected communities, and visualization of the detected communities. The Girvan-Newman method works by iteratively removing edges from the original network to identify communities. Fast Greedy algorithm aims to efficiently identify communities in a network by iteratively merging. Modularity Maximization aims to maximize the modularity score of a partition of the network into communities.

Through this project, we aim to provide valuable information about the effectiveness of community detection algorithms on social network data and provide insights into the underlying structure of social networks.

COMMUNITY DETECTION

Community detection is a process in network analysis that aims to identify groups or clusters of nodes within a network that exhibit strong interconnectivity. It involves partitioning the network into cohesive subgroups based on the relationships between its nodes. These subgroups, known as communities, are characterized by a higher density of connections among their members compared to the connections between members of different communities.

The objective of community detection is to uncover the underlying structure or organization within a network, revealing communities that may have distinct functions, characteristics, or roles. It can provide valuable insights into the network's properties, such as identifying key players or understanding patterns of interactions.

Various algorithms and techniques exist for community detection, each with its own strengths and limitations. Some popular approaches include modularity optimization, hierarchical clustering, and spectral clustering. These methods utilize different principles, such as maximizing modularity, optimizing similarity measures, or analyzing network properties like network flows or eigenvalues.

Community detection can be applied to various types of networks, including social networks, biological networks, transportation networks, and many others. It has applications in diverse fields, such as sociology, biology, computer science, and marketing, to name a few.

COMMUNITY DETECTION TECHNIQUES

1. Modularity Optimization: This technique aims to maximize the modularity of a network, which measures the quality of the community structure. It iteratively optimizes the assignment of nodes to communities based on the difference between the actual and expected number of edges within communities.

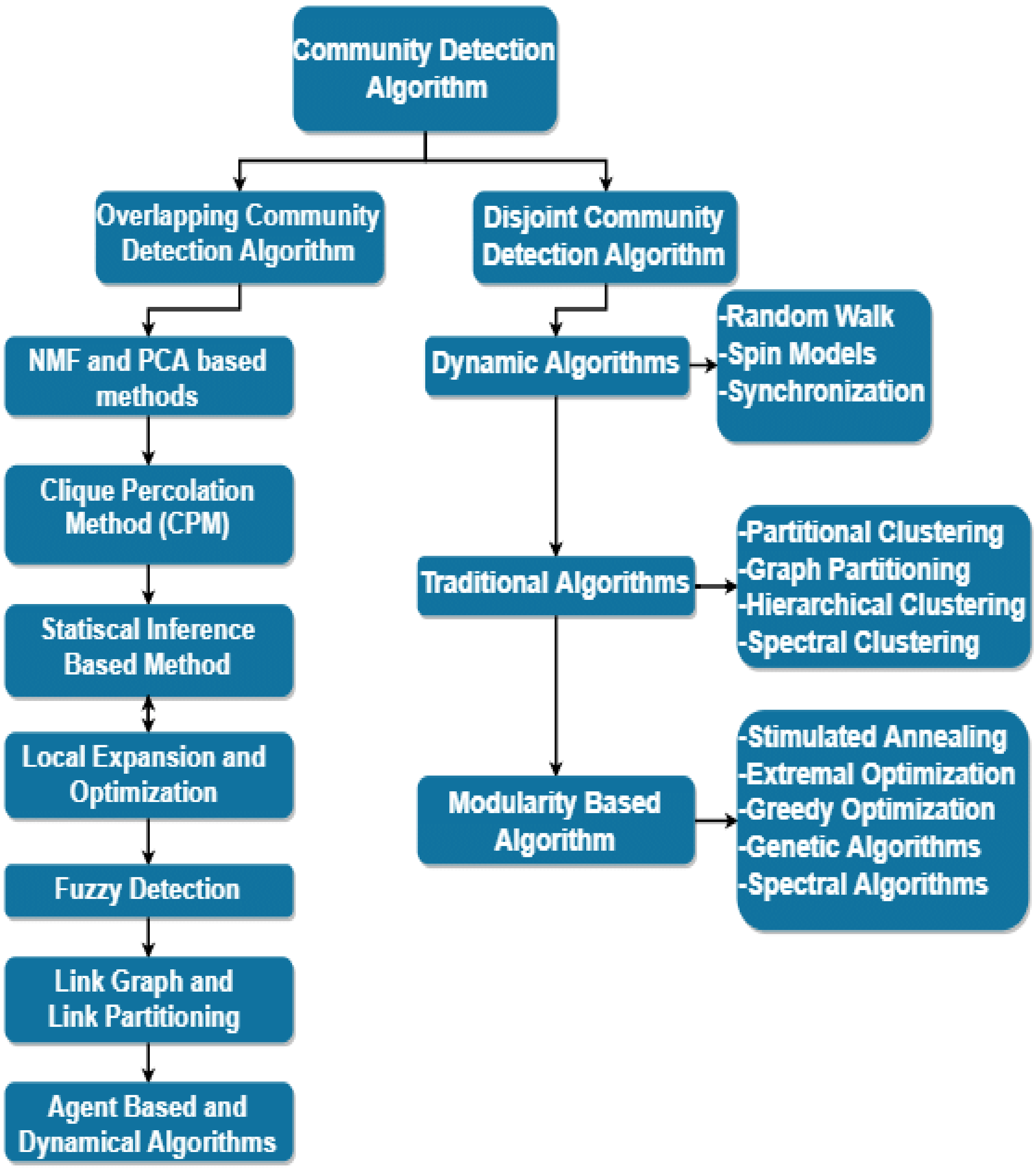
2. Hierarchical Clustering: This technique creates a hierarchy of communities by iteratively merging or splitting clusters based on certain similarity measures. It starts with individual nodes as separate communities and gradually groups them together until a desired level of granularity is reached.

3. Spectral Clustering: This technique uses the eigenvectors of the network's adjacency matrix or Laplacian matrix to detect communities. It converts the network into a lower-dimensional space and then applies clustering algorithms to identify distinct groups.

4. Louvain Method: This method is a fast and scalable algorithm that optimizes modularity. It initially assigns each node to its own community and iteratively merges or splits communities to increase modularity. It efficiently handles large networks and can identify overlapping communities.

5. Infomap: This technique treats community detection as an optimization problem where the objective is to minimize the expected description length of a random walker's trajectory on the network. It identifies communities by compressing information flow patterns within the network.

6. Edge Betweenness: This approach identifies communities by iteratively removing edges with high betweenness centrality, which indicates their importance in connecting different communities. The process continues until the network breaks into distinct components.


Community Detection Algorithm
Overlapping Community Detection Algorithm
Disjoint Community Detection Algorithm
NMF and PCA based methods
Clique Percolation Method (CPM)
Statiscal Inference Based Method
Local Expansion and Optimization
Fuzzy Detection
Link Graph and Link Partitioning
Agent Based and Dynamical Algorithms
Dynamic Algorithms
-Random Walk
-Spin Models
-Synchronization
Traditional Algorithms
-Partitional Clustering
-Graph Partitioning
-Hierarchical Clustering
-Spectral Clustering
Modularity Based Algorithm
-Stimulated Annealing
-Extremal Optimization
-Greedy Optimization
-Genetic Algorithms
-Spectral Algorithms

COMMUNITY DETECTION APPLICATIONS

1. Social Network Analysis: Community detection is widely used in social network analysis to uncover groups of individuals with similar interests, behaviors, or affiliations. It helps in identifying influential users, understanding information diffusion patterns, and detecting online communities in platforms like Facebook, Twitter, and LinkedIn.

2. Biological Networks: Community detection is applied to biological networks such as protein-protein interaction networks and gene co-expression networks. It helps in identifying functional modules or groups of genes/proteins that work together in biological processes, providing insights into disease mechanisms, drug targets, and protein interactions.

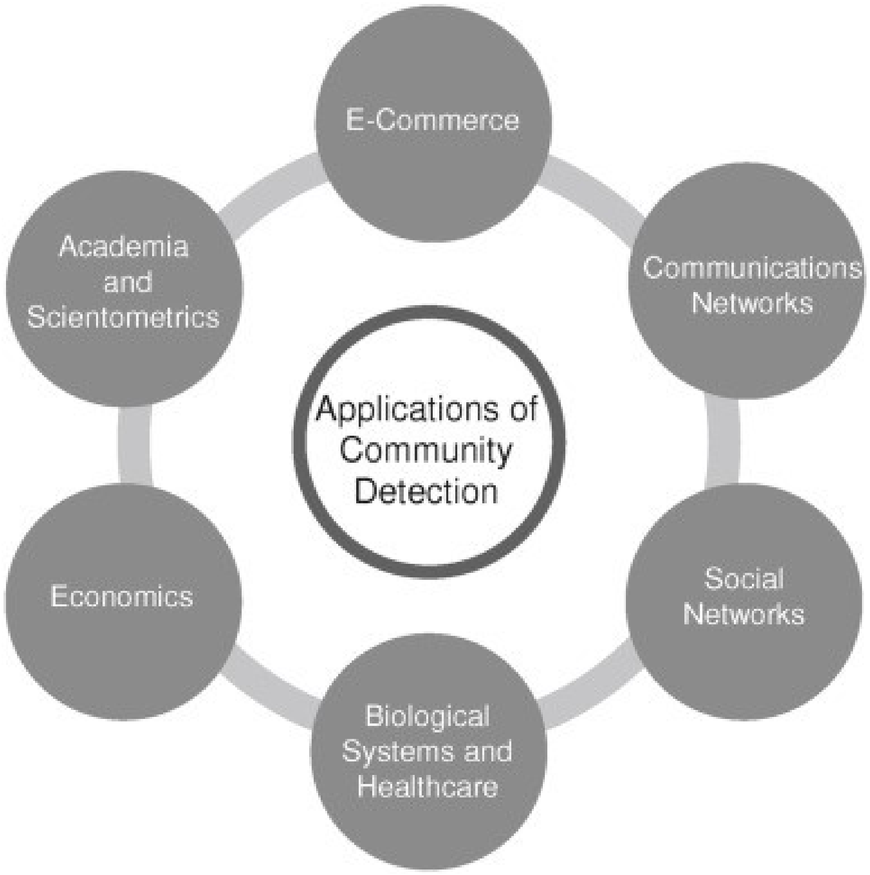
3. Recommender Systems: Community detection techniques are used in recommender systems to identify communities of users with similar preferences or behaviors. By grouping users into communities, personalized recommendations can be made based on the preferences of users within the same community.

4. Web Page Ranking: Community detection algorithms can be used to improve web page ranking algorithms like PageRank. By considering communities within a web graph, the importance of a web page can be assessed not only based on its individual links but also on its role within its community, leading to more accurate rankings.

5. Traffic Analysis and Routing: Community detection helps in understanding traffic patterns and routing in transportation and communication networks.

By identifying communities of interconnected nodes, efficient routing strategies can be developed, leading to improved traffic management and network performance.

6. Marketing and Customer Segmentation: Community detection is applied in market research to identify groups of customers with similar purchasing behaviors, preferences, or demographic characteristics. It helps in targeted marketing campaigns, product recommendations, and customer segmentation for personalized marketing strategies.


Applications of Community Detection
E-Commerce
Communications Networks
Social Networks
Biological Systems and Healthcare
Economics
Academia and Scientometrics

GIRVAN-NEWMAN ALGORITHM

The Girvan-Newman method, proposed by Mark Girvan and Michelle Newman, is based on the concept of edge betweenness centrality. It starts by calculating the betweenness centrality for all edges in the network, which measures how many shortest paths pass through each edge.

The algorithm then iteratively removes the edge with the highest betweenness centrality, which is considered as a bridge connecting different communities. The removal of this edge disrupts the flow between communities, effectively dividing the network into smaller components.

After each removal, the betweenness centrality of the remaining edges is recalculated, and the process is repeated until a stopping criterion is met. Typically, the algorithm continues until all edges are removed, or until the network breaks into a specified number of components.

The Girvan-Newman method provides a hierarchical view of the community structure by generating a dendrogram or a hierarchical tree. The dendrogram represents the network as a series of divisions at different levels, showing the progressive formation of communities.

One drawback of the Girvan-Newman method is that it can be computationally expensive for large networks since it requires calculating the betweenness centrality for all edges. However, it remains a popular method for community detection due to its effectiveness in identifying communities based on the concept of edge betweenness.

FAST GREEDY ALGORITHM

The Fast Greedy algorithm is a community detection method that aims to efficiently identify communities in a network by iteratively merging or splitting communities based on their modularity gain. Here is a brief description of the Fast Greedy algorithm for your project report:

The Fast Greedy algorithm, also known as the Clauset-Newman-Moore algorithm, starts with each node belonging to its own separate community. It iteratively merges communities that lead to the maximum increase in modularity.

The algorithm operates as follows:

1. Compute the initial modularity: Calculate the modularity of the network partition, which measures the quality of the current community structure.

2. Merge communities: Find the pair of communities that, when merged, results in the highest increase in modularity. Merge these communities together, forming a larger community.

3. Update modularity: Recalculate the modularity after the merger. This involves evaluating the change in the number of edges within and between communities.

4. Repeat steps 2 and 3: Iterate the process of merging communities and updating the modularity until no further improvement can be made.

The Fast Greedy algorithm efficiently explores the space of community partitions by considering all possible community mergers at each step. By greedily maximizing modularity gain, it quickly identifies communities that exhibit strong internal connectivity and weak external connectivity.

One advantage of the Fast Greedy algorithm is its computational efficiency, making it suitable for analyzing large networks. However, it may suffer from the resolution limit, similar to other modularity-based methods, where small communities within larger ones may not be detected.

MODULARITY MAXIMIZATION METHOD

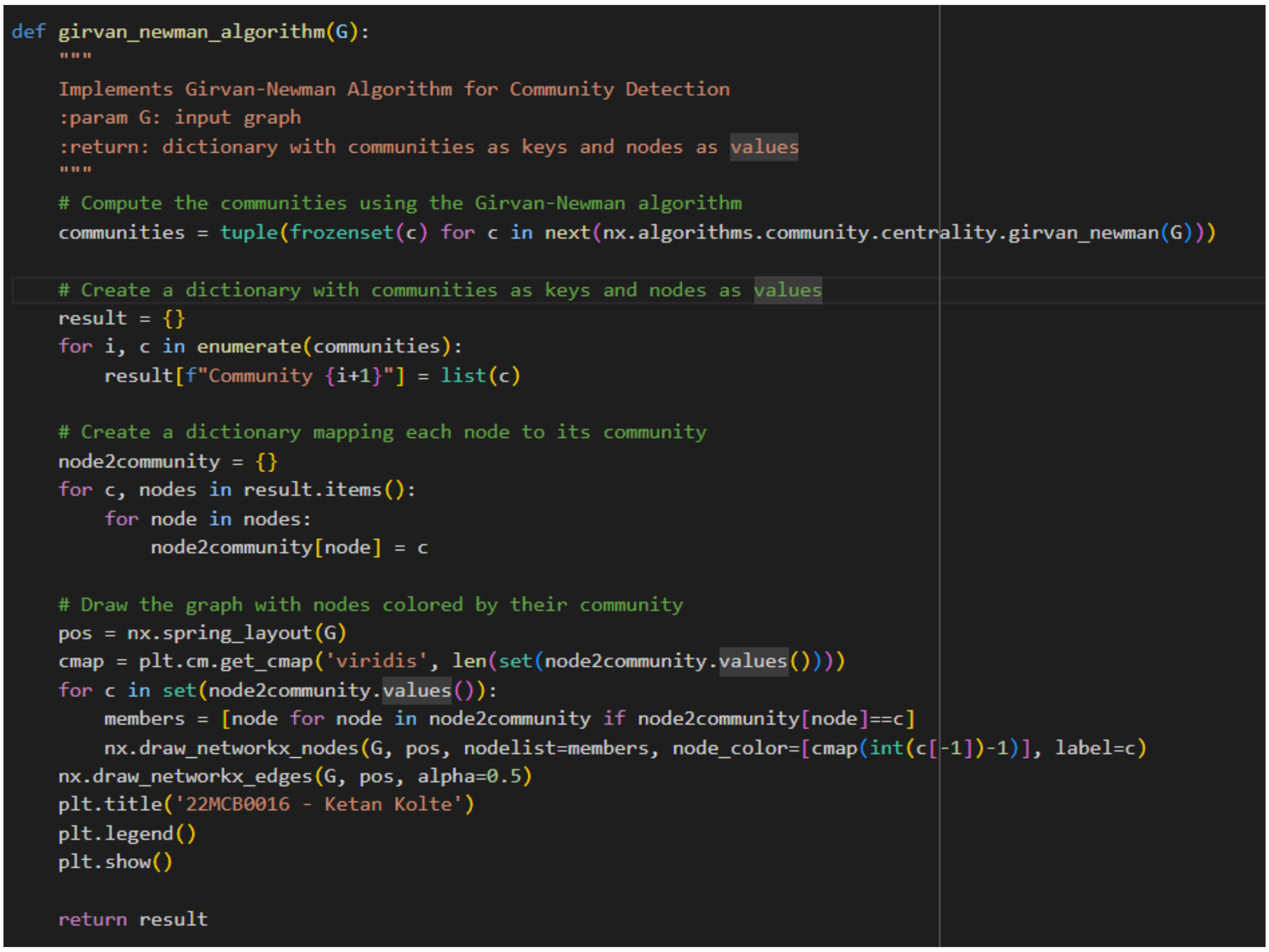
The Modularity Maximization method, proposed by Newman and Girvan, focuses on optimizing the modularity of a network. Modularity measures the difference between the number of edges within communities and the expected number of edges in a random network. Higher modularity values indicate a stronger community structure.

The algorithm begins by assigning each node to its own community. It then iteratively examines the possibility of merging communities to increase the overall modularity. This is done by evaluating the change in modularity resulting from the merger of two communities. The algorithm continues merging communities until no further improvement in modularity can be achieved.

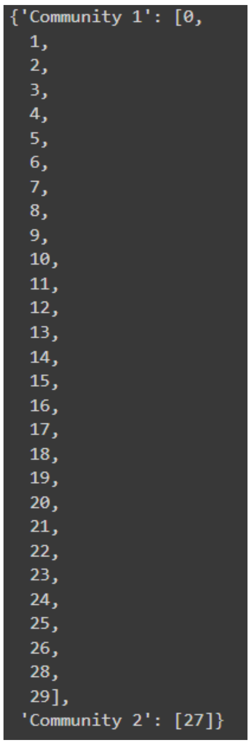
To evaluate the change in modularity, the algorithm considers the gain or loss of edges within and between communities when they are merged. The goal is to maximize the net gain in modularity by merging communities that have a higher density of internal edges compared to the expected density in a random network.

The Modularity Maximization method is known for its simplicity and effectiveness in detecting communities. It can identify communities of various sizes, shapes, and densities. However, it may suffer from the resolution limit, which means that it can fail to detect small communities when they are embedded within larger communities.

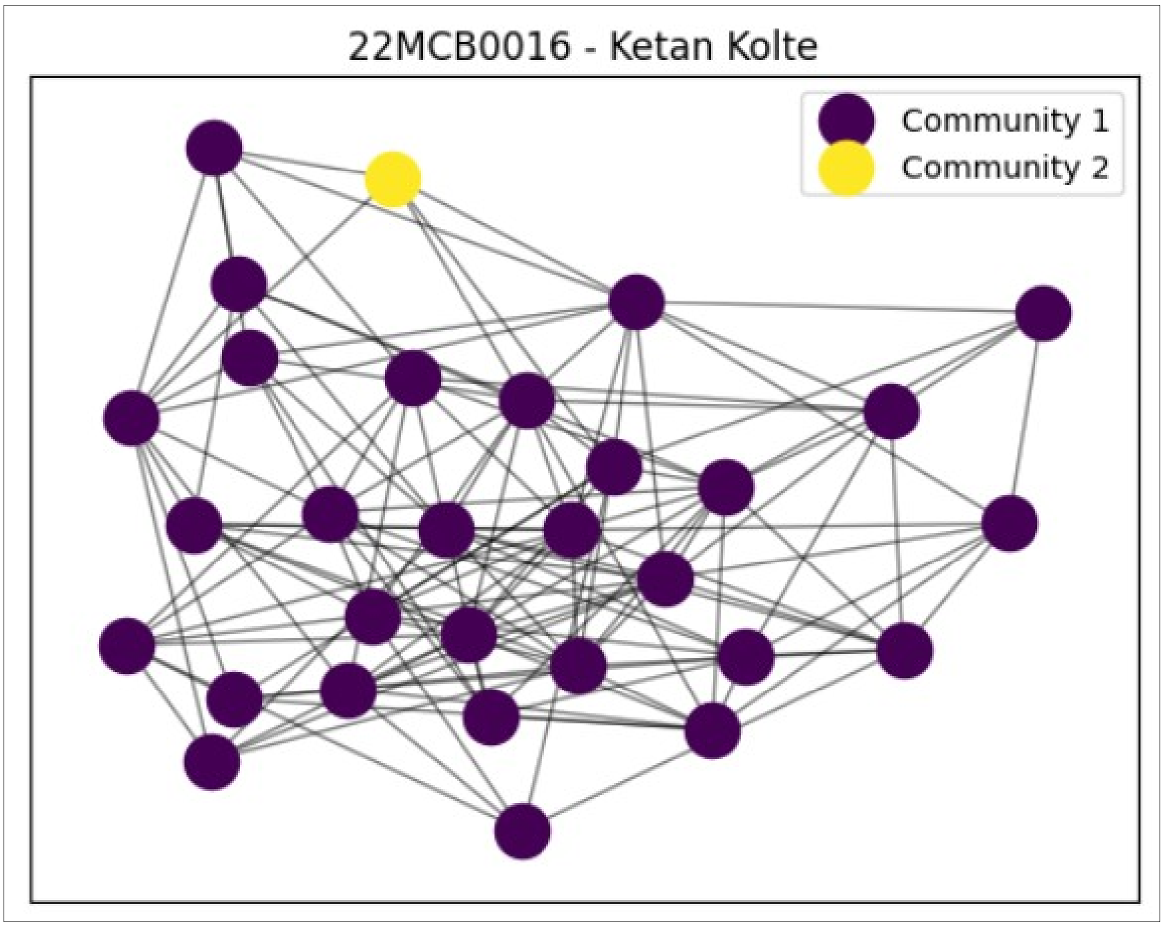
IMPLEMENTATION OF GIRVAN-NEWMAN ALGORITHM



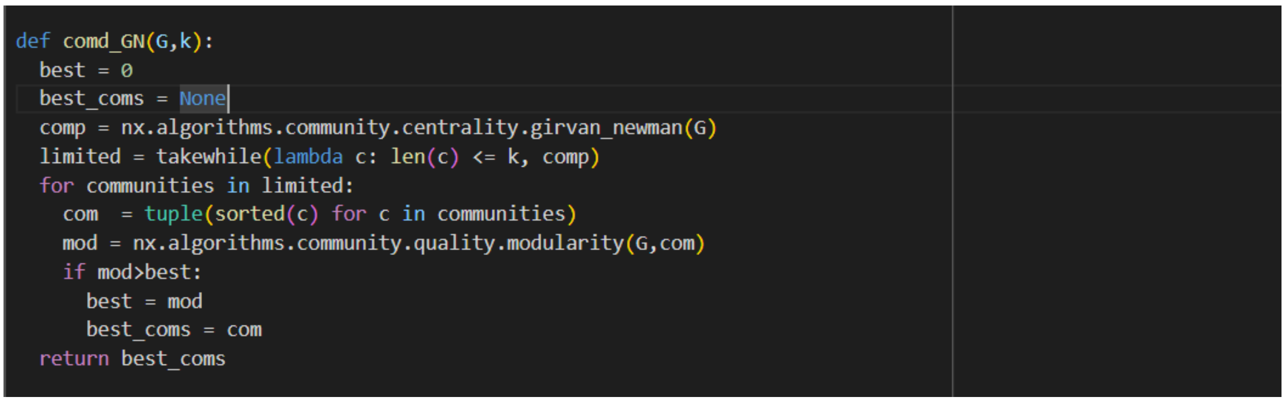
```python
def girvan_newman_algorithm(G):
    """
    Implements Girvan-Newman Algorithm for Community Detection
    :param G: input graph
    :return: dictionary with communities as keys and nodes as values
    """
    # Compute the communities using the Girvan-Newman algorithm
    communities = tuple(frozenset(c) for c in next(nx.algorithms.community.centrality.girvan_newman(G)))

    # Create a dictionary with communities as keys and nodes as values
    result = {}
    for i, c in enumerate(communities):
        result[f"Community {i+1}"] = list(c)

    # Create a dictionary mapping each node to its community
    node2community = {}
    for c, nodes in result.items():
        for node in nodes:
            node2community[node] = c

    # Draw the graph with nodes colored by their community
    pos = nx.spring_layout(G)
    cmap = plt.cm.get_cmap('viridis', len(set(node2community.values())))
    for c in set(node2community.values()):
        members = [node for node in node2community if node2community[node]==c]
        nx.draw_networkx_nodes(G, pos, nodelist=members, node_color=[cmap(int(c[-1])-1)], label=c)
    nx.draw_networkx_edges(G, pos, alpha=0.5)
    plt.title('22MCB0016 - Ketan Kolte')
    plt.legend()
    plt.show()

    return result
```



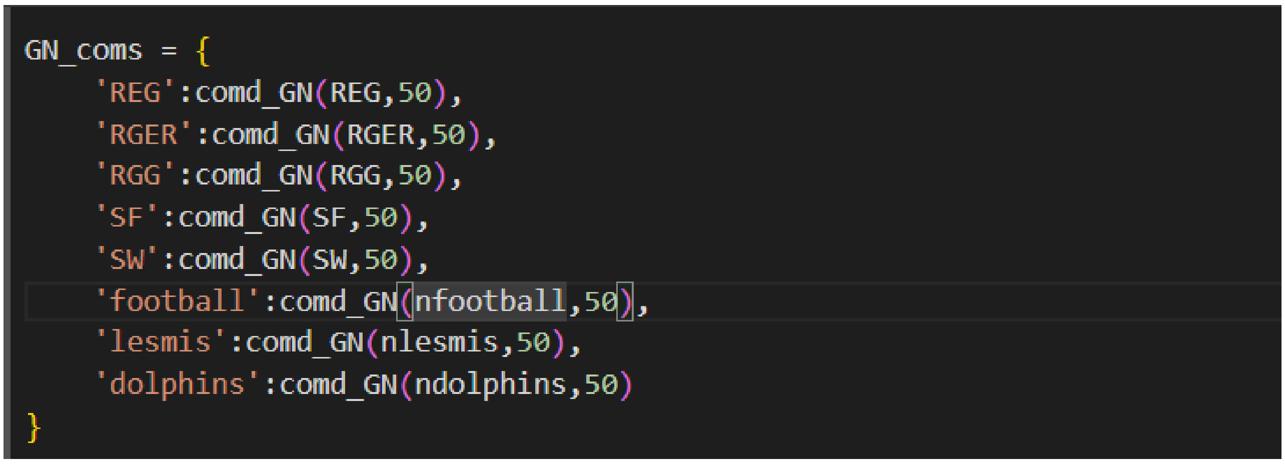
```
{'Community 1': [0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  28,
  29],
 'Community 2': [27]}
```



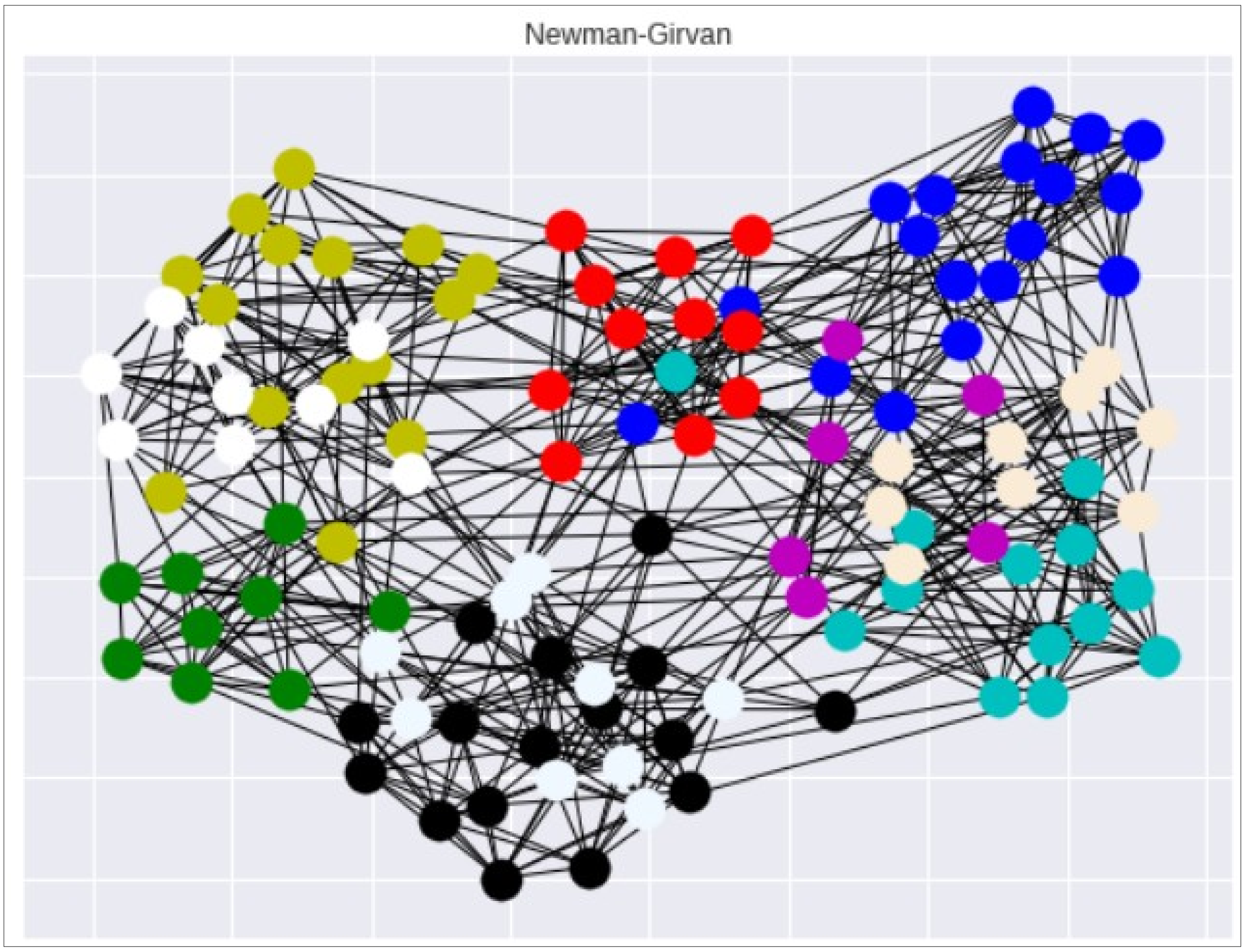
USING FOOTBALL.GML DATASET.



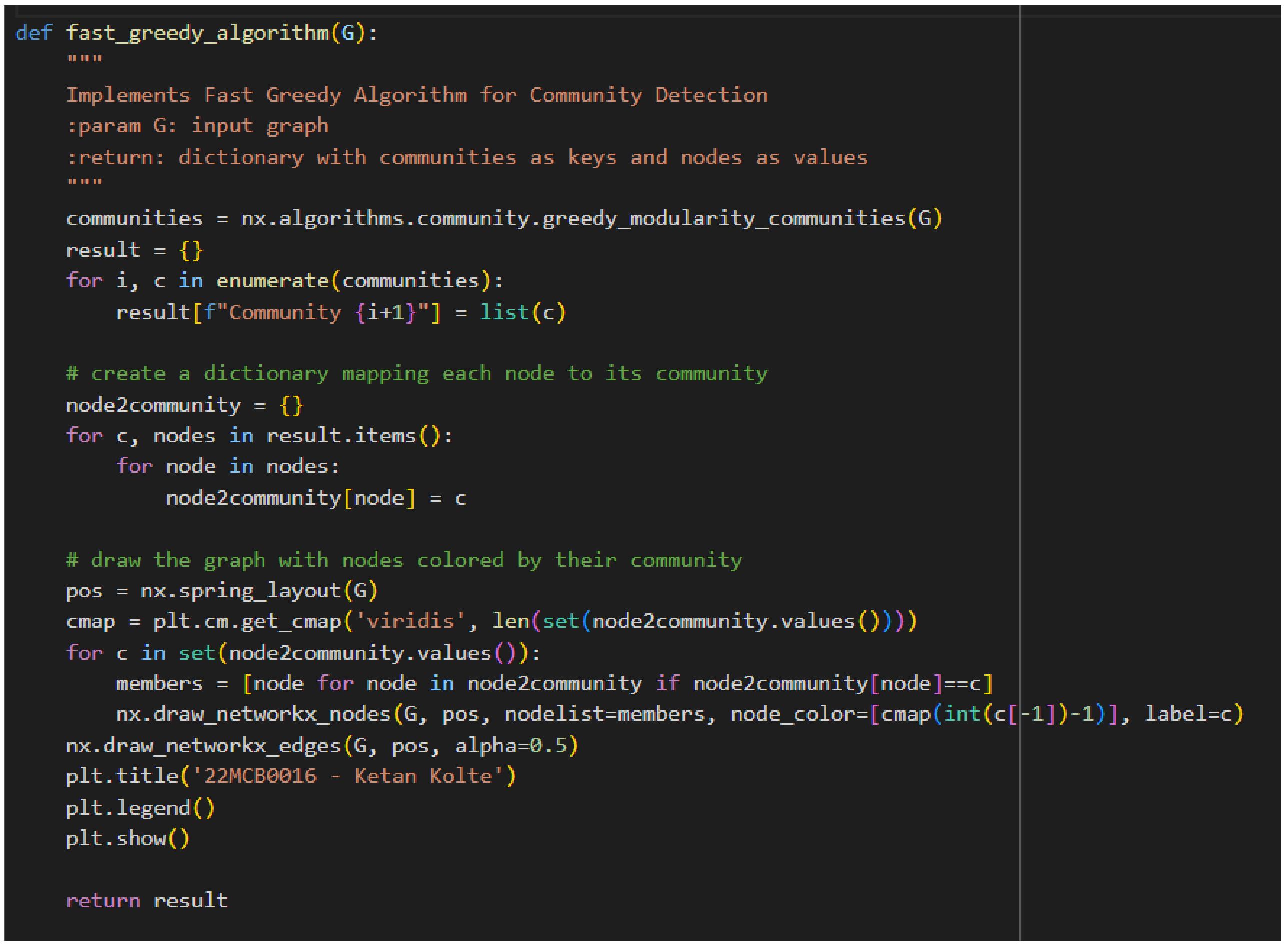
```
def comd_GN(G,k):
  best = 0
  best_coms = None
  comp = nx.algorithms.community.centrality.girvan_newman(G)
  limited = takewhile(lambda c: len(c) <= k, comp)
  for communities in limited:
    com  = tuple(sorted(c) for c in communities)
    mod = nx.algorithms.community.quality.modularity(G,com)
    if mod>best:
      best = mod
      best_coms = com
  return best_coms
```



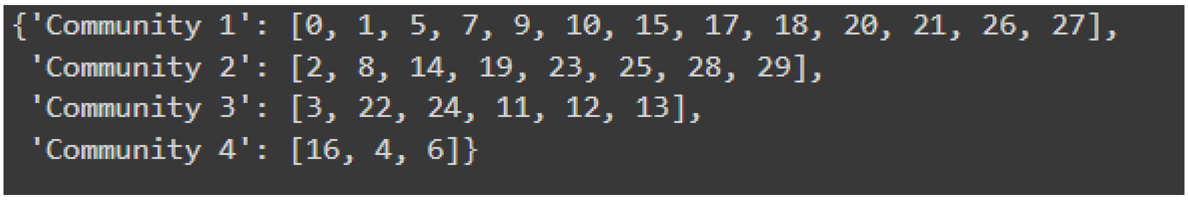
```
GN_coms = {
    'REG':comd_GN(REG,50),
    'RGER':comd_GN(RGER,50),
    'RGG':comd_GN(RGG,50),
    'SF':comd_GN(SF,50),
    'SW':comd_GN(SW,50),
    'football':comd_GN(nfootball,50),
    'lesmis':comd_GN(nlesmis,50),
    'dolphins':comd_GN(ndolphins,50)
}
```



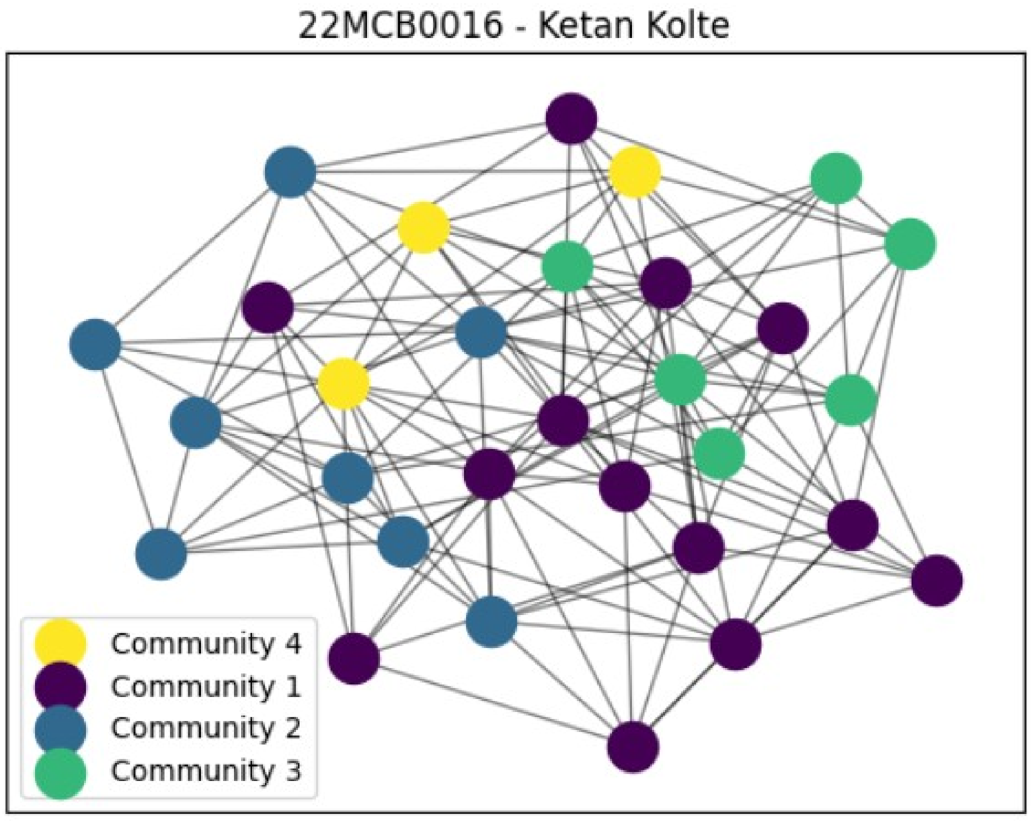
IMPLEMENTATION OF FAST GREEDY ALGORITHM



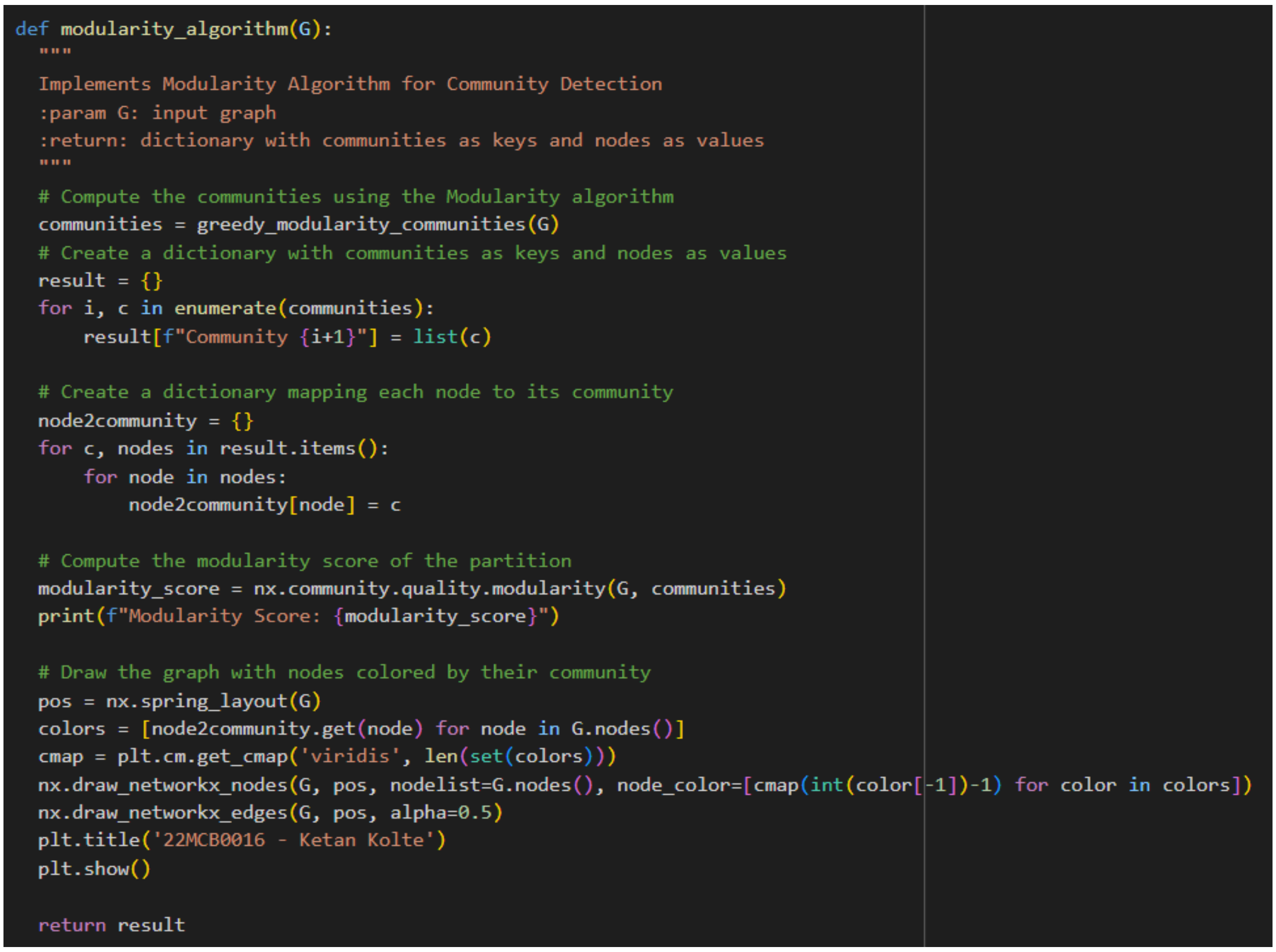
```python
def fast_greedy_algorithm(G):
    """
    Implements Fast Greedy Algorithm for Community Detection
    :param G: input graph
    :return: dictionary with communities as keys and nodes as values
    """
    communities = nx.algorithms.community.greedy_modularity_communities(G)
    result = {}
    for i, c in enumerate(communities):
        result[f"Community {i+1}"] = list(c)

    # create a dictionary mapping each node to its community
    node2community = {}
    for c, nodes in result.items():
        for node in nodes:
            node2community[node] = c

    # draw the graph with nodes colored by their community
    pos = nx.spring_layout(G)
    cmap = plt.cm.get_cmap('viridis', len(set(node2community.values())))
    for c in set(node2community.values()):
        members = [node for node in node2community if node2community[node]==c]
        nx.draw_networkx_nodes(G, pos, nodelist=members, node_color=[cmap(int(c[-1])-1)], label=c)
    nx.draw_networkx_edges(G, pos, alpha=0.5)
    plt.title('22MCB0016 - Ketan Kolte')
    plt.legend()
    plt.show()

    return result
```



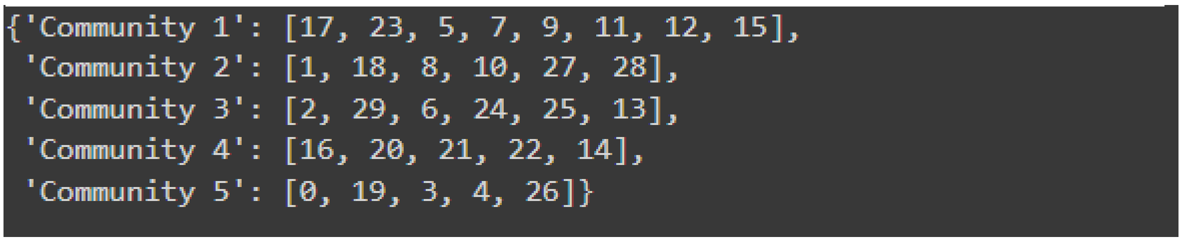
```
{'Community 1': [0, 1, 5, 7, 9, 10, 15, 17, 18, 20, 21, 26, 27],
 'Community 2': [2, 8, 14, 19, 23, 25, 28, 29],
 'Community 3': [3, 22, 24, 11, 12, 13],
 'Community 4': [16, 4, 6]}
```


22MCB0016 - Ketan Kolte
Community 4
Community 1
Community 2
Community 3

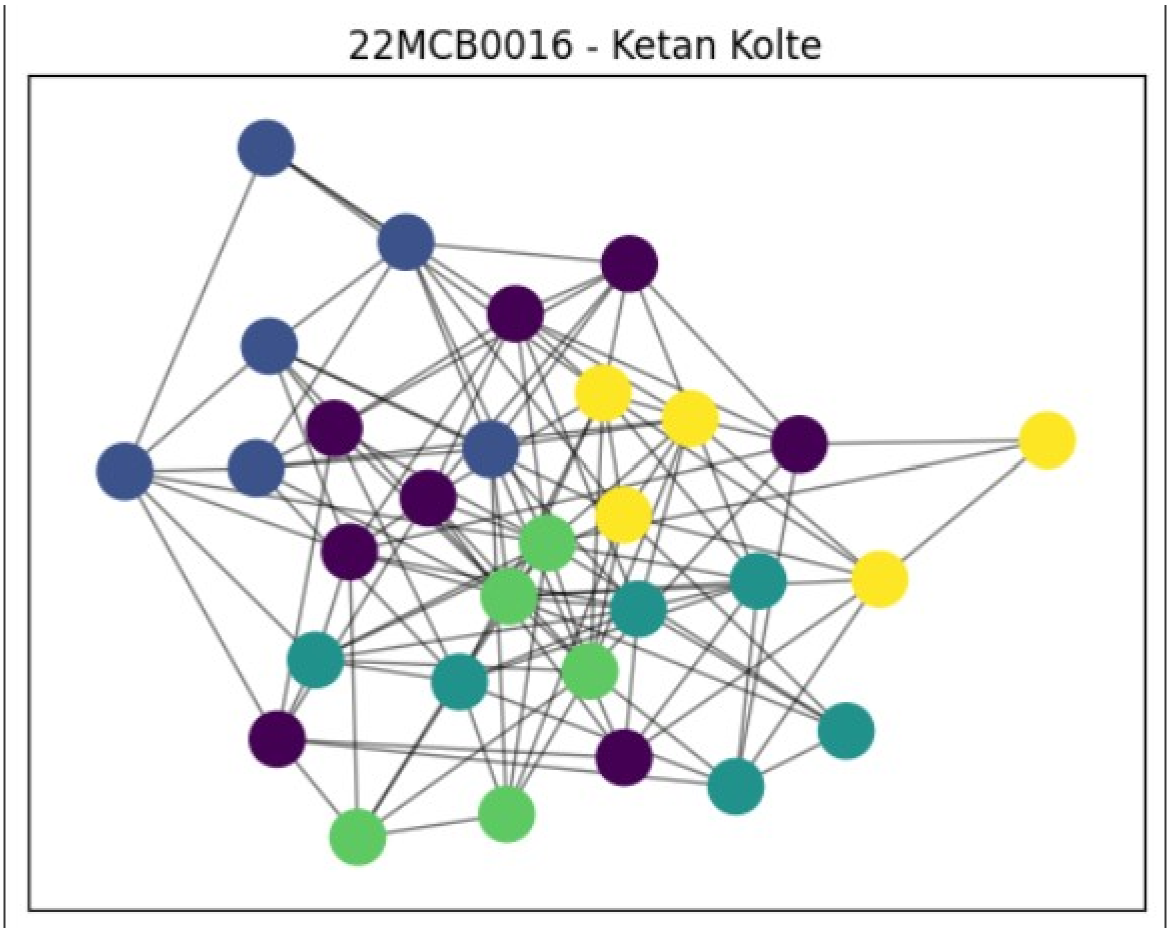
IMPLEMENTATION OF MODULARITY MAXIMIZATION METHOD



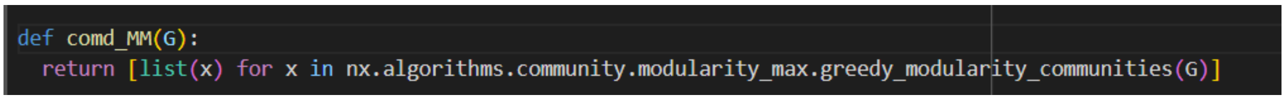
```python
def modularity_algorithm(G):
    """
    Implements Modularity Algorithm for Community Detection
    :param G: input graph
    :return: dictionary with communities as keys and nodes as values
    """
    # Compute the communities using the Modularity algorithm
    communities = greedy_modularity_communities(G)
    # Create a dictionary with communities as keys and nodes as values
    result = {}
    for i, c in enumerate(communities):
        result[f"Community {i+1}"] = list(c)

    # Create a dictionary mapping each node to its community
    node2community = {}
    for c, nodes in result.items():
        for node in nodes:
            node2community[node] = c

    # Compute the modularity score of the partition
    modularity_score = nx.community.quality.modularity(G, communities)
    print(f"Modularity Score: {modularity_score}")

    # Draw the graph with nodes colored by their community
    pos = nx.spring_layout(G)
    colors = [node2community.get(node) for node in G.nodes()]
    cmap = plt.cm.get_cmap('viridis', len(set(colors)))
    nx.draw_networkx_nodes(G, pos, nodelist=G.nodes(), node_color=[cmap(int(color[-1])-1) for color in colors])
    nx.draw_networkx_edges(G, pos, alpha=0.5)
    plt.title('22MCB0016 - Ketan Kolte')
    plt.show()

    return result
```



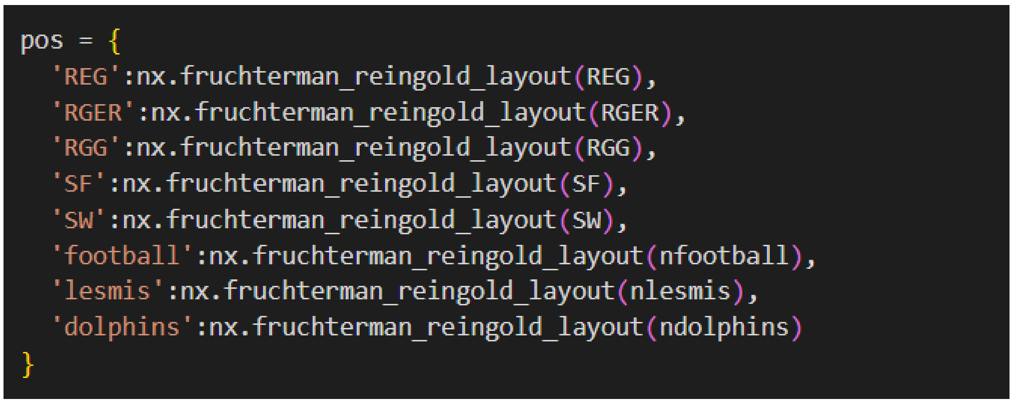
```
{'Community 1': [17, 23, 5, 7, 9, 11, 12, 15],
 'Community 2': [1, 18, 8, 10, 27, 28],
 'Community 3': [2, 29, 6, 24, 25, 13],
 'Community 4': [16, 20, 21, 22, 14],
 'Community 5': [0, 19, 3, 4, 26]}
```



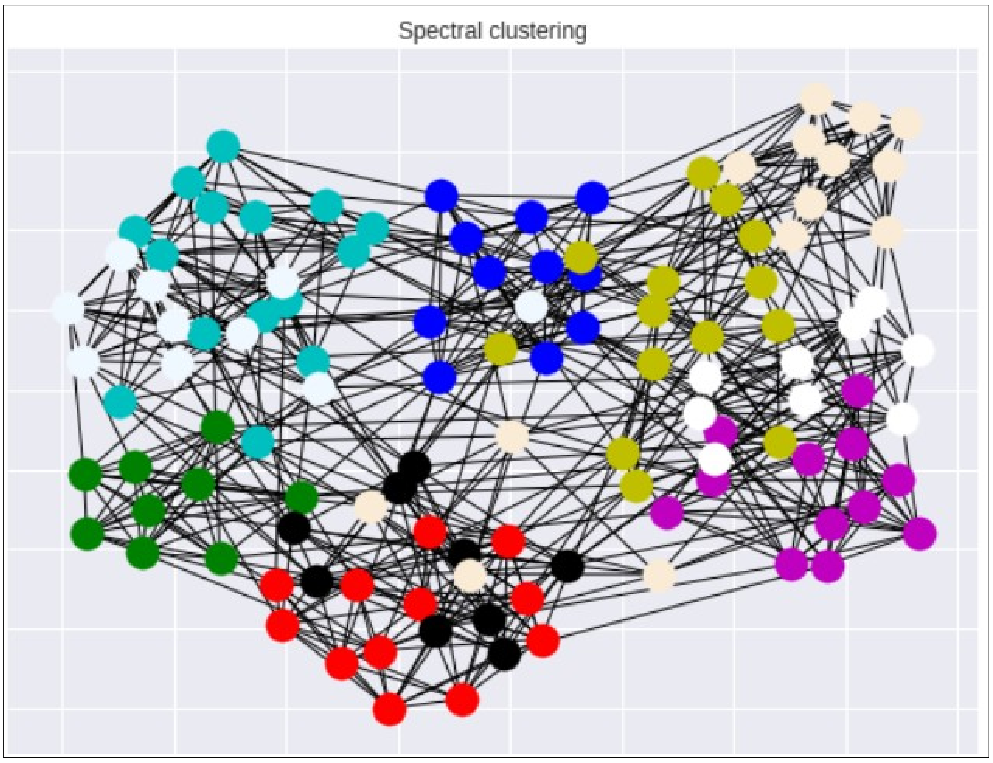
USING FOOTBALL.GML DATASET :



```
def comd_MM(G):
  return [list(x) for x in nx.algorithms.community.modularity_max.greedy_modularity_communities(G)]
```



```
pos = {
  'REG':nx.fruchterman_reingold_layout(REG),
  'RGER':nx.fruchterman_reingold_layout(RGER),
  'RGG':nx.fruchterman_reingold_layout(RGG),
  'SF':nx.fruchterman_reingold_layout(SF),
  'SW':nx.fruchterman_reingold_layout(SW),
  'football':nx.fruchterman_reingold_layout(nfootball),
  'lesmis':nx.fruchterman_reingold_layout(nlesmis),
  'dolphins':nx.fruchterman_reingold_layout(ndolphins)
}
```



DATASET USED :

The football.gml dataset is a network representation of American college football teams and their matchups. Each team is a node, and edges represent games played between teams. Analyzing this dataset using community detection helps identify groups of teams with similar opponents or performance.

CONCLUSION :

Implementing community detection algorithms both with a real-world dataset and a randomly generated graph has allowed for the identification of distinct groups within networks. These algorithms provide insights into the inherent structures and patterns of connectivity within the networks, offering valuable information about communities and their interrelationships.

CLIQUE PERCOLATION METHOD

The clique percolation method (CPM) is a graph-based clustering algorithm that identifies densely connected groups of nodes, known as cliques, within a network. It provides a way to discover overlapping communities in complex networks.

Steps involved in implementing the Clique Percolation Method:

1. Input: The algorithm takes an input graph represented by its adjacency matrix or an edge list.

2. Clique Identification: The algorithm begins by finding all maximal cliques in the graph. A maximal clique is a fully connected subgraph where no additional node can be added without breaking the connectivity.

3. Overlapping Cliques: To allow for overlapping communities, the CPM introduces the concept of overlap between cliques. Two cliques are considered overlapping if they share a predefined number of nodes.

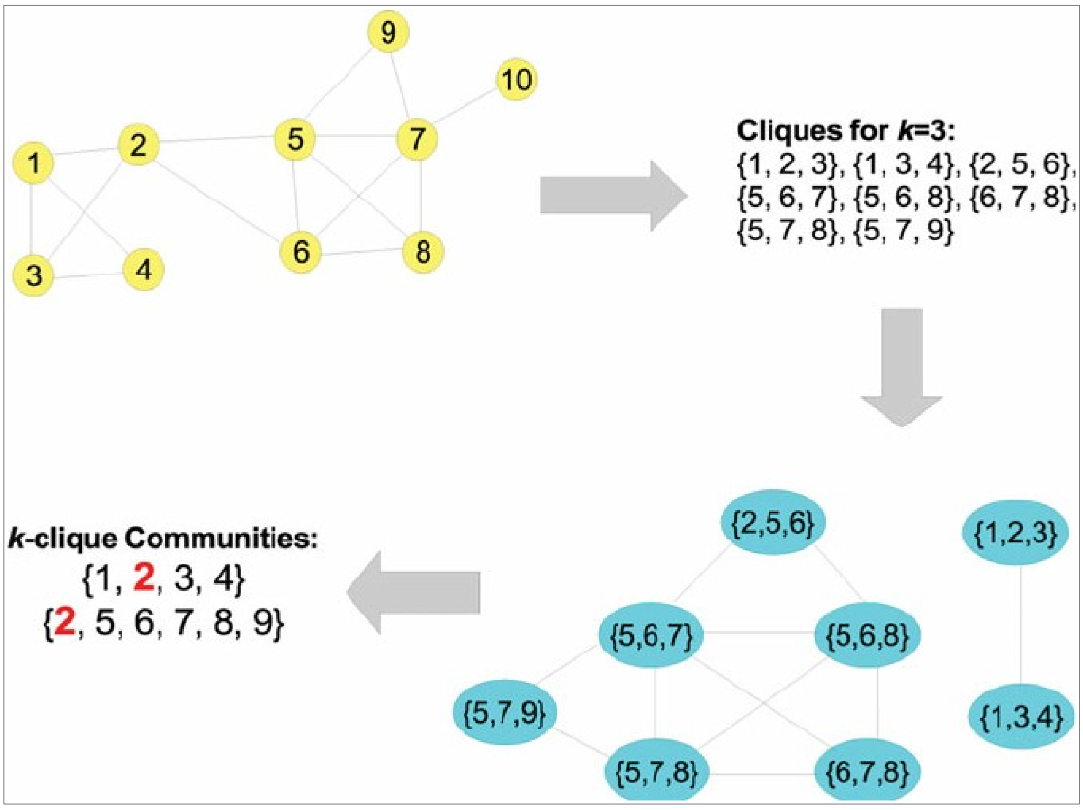
4. Clique Percolation: Cliques that overlap are merged together to form larger communities. This is done by constructing a new graph, called the clique graph, where each node represents a clique, and an edge exists between two nodes if the corresponding cliques overlap.

5. Community Extraction: The final step involves extracting the communities from the clique graph. This can be achieved by applying traditional

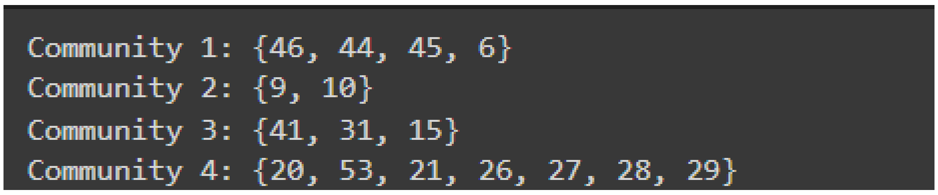
community detection algorithms, such as modularity optimization or label propagation, on the clique graph.

6. Output: The output of the algorithm is a set of overlapping communities, where each community represents a group of nodes that are densely connected.

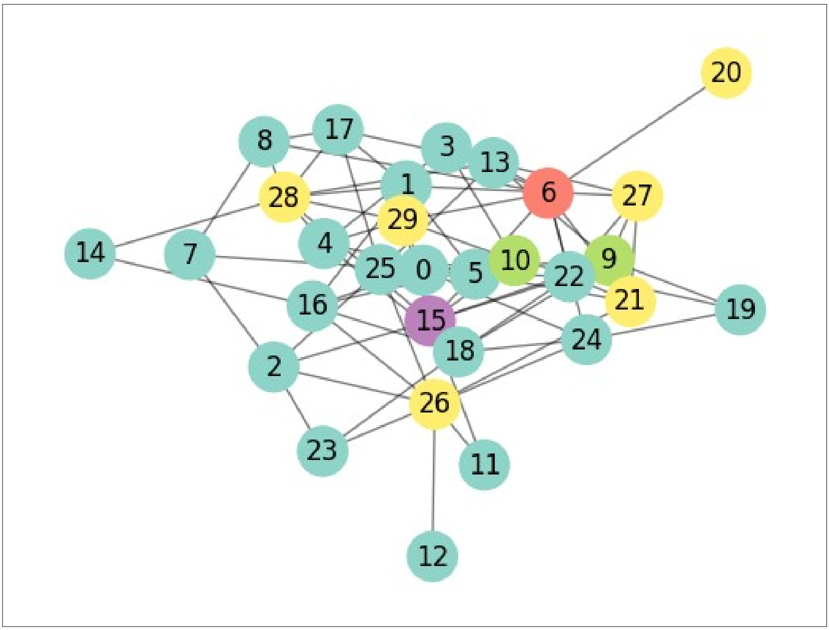
The Clique Percolation Method is a graph clustering algorithm that detects overlapping communities by identifying cliques and merging them based on their overlap. It provides a flexible approach for discovering communities in complex networks.



RESULT :



```
Community 1: {46, 44, 45, 6}
Community 2: {9, 10}
Community 3: {41, 31, 15}
Community 4: {20, 53, 21, 26, 27, 28, 29}
```


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