Clique Percolation Method (CPM)

The Clique Percolation Method (CPM) is a network analysis algorithm used to identify overlapping communities or clusters in complex networks. It was proposed by Palla et al. in 2005 as an alternative to traditional community detection algorithms such as modularity optimization and hierarchical clustering.

An overview of the Clique Percolation Method and how it works:

1. Network representation: The CPM operates on a network represented as a graph, where nodes represent entities (e.g., individuals, proteins, or websites) and edges represent connections or relationships between them.

2. Cliques: A clique is a fully connected subgraph, meaning that every node in the clique is directly connected to every other node. In other words, it is a complete subgraph. Cliques are considered as the elementary building blocks of communities in the CPM.

3. Clique overlap: Unlike traditional community detection methods, the CPM allows for overlapping communities. Overlapping communities share some nodes or cliques, indicating that nodes can belong to multiple communities simultaneously. This flexibility makes the CPM well-suited for capturing complex community structures in real-world networks.

4. Algorithm steps: The CPM algorithm consists of the following steps:

a. Finding k-cliques: The algorithm starts by identifying all cliques of size k (where k is a predefined parameter) in the network. A k-clique is a subgraph in which every node is connected to at least k-1 other nodes.

b. Creating a clique graph: A clique graph is constructed, where each node represents a k-clique, and there is an edge between two nodes if the corresponding k-cliques share k-1 nodes.

c. Percolation: In this step, the algorithm finds the connected components of the clique graph. A connected component represents a community in the network. The percolation process

involves progressively adding edges between the nodes (k-cliques) in the clique graph until the communities percolate.

d. Overlapping communities: Finally, the algorithm extracts overlapping communities from the percolated network by considering cliques that belong to multiple communities.

5. Parameter selection: The key parameter in the CPM is the clique size (k). A higher value of k leads to larger and more densely connected communities, while a lower value allows for smaller and more numerous communities. Selecting an appropriate value of k depends on the characteristics of the network and the specific analysis goals.

The Clique Percolation Method has been applied to various domains, including social networks, biological networks, and web networks. It offers a way to capture overlapping community structures, which can provide valuable insights into the organization and functioning of complex systems.

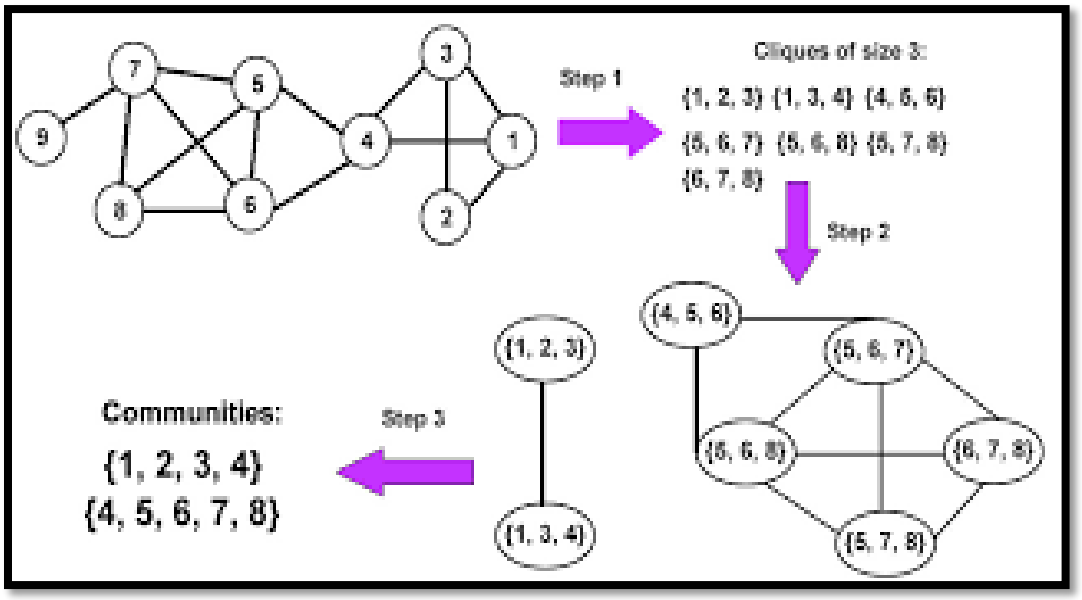

Fig. Clique Percolation Method

```
def find_communities_from_cliques(cliques, overlap_threshold):
 # Step 1: Build a clique graph
 clique_graph = nx.Graph()
 for i, clique1 in enumerate(cliques):
    for j, clique2 in enumerate(cliques):
        if i != j and len(set(clique1) & set(clique2)) >=
overlap_threshold:
            clique_graph.add_edge(i, j)

 # Step 2: Find connected components
 communities = list(nx.connected_components(clique_graph))
 return communities

# Example usage
# Provide a list of cliques identified by the clique percolation method
G = nx.erdos_renyi_graph(n=30, p=0.2)
cliques = list(nx.find_cliques(G))

# Set the overlap threshold
overlap_threshold = 2

# Find communities from cliques
communities = find_communities_from_cliques(cliques, overlap_threshold)

# Print the identified communities
for i, community in enumerate(communities):
 print(f"Community {i+1}: {community}")
```

OUTPUT:

```
Community 1: {6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 22, 23, 24, 29, 32, 34, 36, 41, 47, 48, 52, 53}
Community 2: {25, 33}
```

Visualization of Communities:

```
def visualize_communities(G, communities):
    # Create a dictionary mapping each node to its community index
    node_community = {}
    for i, community in enumerate(communities):
        for node in community:
            node_community[node] = i

    # Create a list of colors for the nodes based on their community index
    node_colors = [node_community.get(node, -1) for node in G.nodes()]

    # Draw the graph with the nodes colored by community
```

```
    pos = nx.spring_layout(G)
    nx.draw_networkx_nodes(G, pos, node_color=node_colors,
cmap=plt.cm.Set3, node_size=500)
    nx.draw_networkx_edges(G, pos, alpha=0.5)
    labels = {node: str(node) for node in G.nodes()}
    nx.draw_networkx_labels(G, pos, labels, font_size=12)

    # Show the plot
    plt.axis('off')
    plt.show()

# Usage Example
G = nx.erdos_renyi_graph(n=30, p=0.2)
cliques = list(nx.find_cliques(G))
overlap_threshold = 2
communities = find_communities_from_cliques(cliques, overlap_threshold)
visualize_communities(G, communities)
```