VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Name: Arjun Krishna S R

Reg No: 22MCB0038

Subject Name: Social Network Analytics Lab

Subject Code: MCSE618P

Lab Assessment 2

Community Detection Algorithms:

Community detection algorithms in social networks are computational methods used to identify and uncover groups or communities of nodes that exhibit dense internal connections and sparse connections with the rest of the network. These algorithms aim to reveal the underlying structure and organization of social networks by partitioning nodes into cohesive and meaningful groups.

Social networks often represent relationships between individuals, such as friendships, collaborations, or interactions. Community detection algorithms help uncover communities within these networks, where nodes within a community are more likely to interact with each other than with nodes outside the community. Communities can represent various social structures, such as groups of friends, co-authorship networks, interest-based communities, or functional clusters within an organization.

Community detection algorithms leverage the network topology and connectivity patterns to identify communities. They typically analyze the network's connectivity, node attributes, or a combination of both

The Most Commonly used community detection algorithms are:

Clique Percolation Method for community detection:

The Clique Percolation Method (CPM) is a community detection algorithm that identifies overlapping communities in a network by leveraging the concept of cliques. A clique is a subset of nodes in a network where every pair of nodes is connected by an edge, forming a complete subgraph. The CPM operates by finding and merging the cliques that share a certain number of common nodes.

Here's a brief overview of the Clique Percolation Method:

Identification of Cliques: The algorithm starts by identifying all cliques of a given size (k) in the network. A clique of size k consists of k nodes where each pair of nodes is connected by an edge.

Construction of Overlapping Cliques: The algorithm constructs a clique graph, where each node represents a clique of size k. Two nodes in the clique graph are connected if their corresponding cliques share a specified number of common nodes (often k-1).

Identification of Overlapping Communities: The algorithm identifies the connected components in the clique graph. Each connected component represents a potential overlapping community, where nodes within the same component are part of the same community.

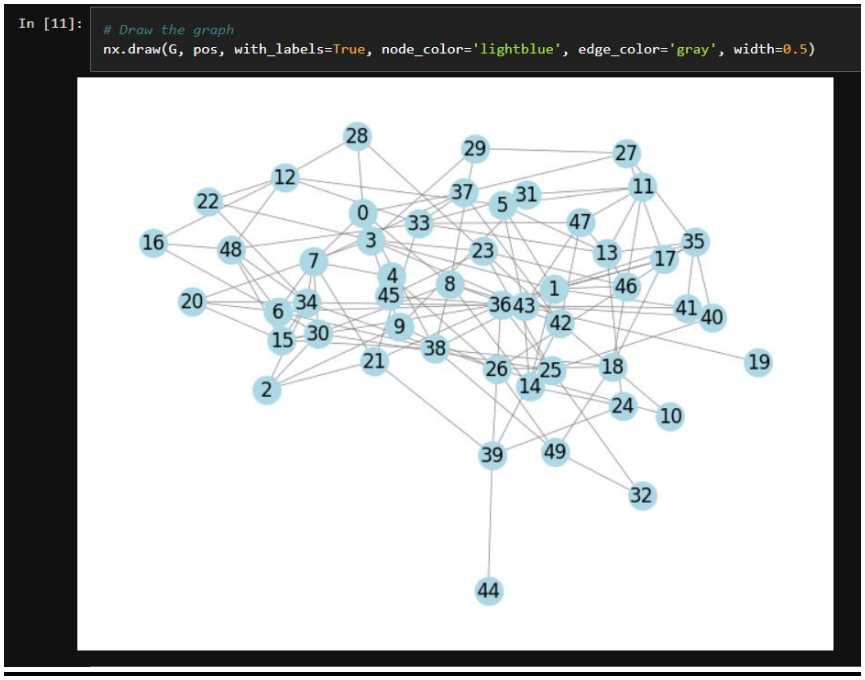
Extraction of Overlapping Communities: The algorithm extracts the overlapping communities from the identified connected components. Each overlapping community consists of nodes that belong to multiple cliques within the same connected component.

By considering the overlapping cliques and their connections, the Clique Percolation Method captures the presence of overlapping communities in a network. This is particularly useful when nodes can participate in multiple communities simultaneously, reflecting the complex nature of real-world networks where individuals often have diverse affiliations.

The Clique Percolation Method allows for varying the size of cliques (k) to control the granularity of the detected communities. Smaller values of k tend to result in larger and more inclusive communities, while larger values of k lead to smaller and more distinct communities.

Overall, the Clique Percolation Method provides a framework for identifying overlapping communities based on the presence of overlapping cliques, enabling a deeper understanding of the structural and functional characteristics of complex networks.

Screenshot of Community:



Edge Betweenness Algorithm for community detection

The Edge Betweenness Algorithm is a community detection algorithm that identifies communities in a network based on the concept of edge betweenness centrality. It operates by iteratively removing edges with high betweenness centrality to reveal the underlying community structure.

Here's a brief summary of the Edge Betweenness Algorithm:

Calculation of Edge Betweenness: The algorithm starts by calculating the betweenness centrality for each edge in the network. Betweenness centrality measures the number of shortest paths passing through an edge, indicating its importance in connecting different parts of the network.

Identification of Edge with Highest Betweenness: The algorithm identifies the edge with the highest betweenness centrality. This edge is considered a crucial bridge between different parts of the network.

Removal of Edge: The identified edge with the highest betweenness centrality is removed from the network.

Recalculation of Betweenness: The algorithm recalculates the betweenness centrality for all remaining edges in the network.

Iteration: Steps 2 to 4 are repeated until a desired number of communities is obtained or until the network is completely fragmented into isolated nodes.

Community Extraction: The resulting communities are extracted based on the connectivity patterns after the removal of edges. Nodes that are still connected after the removal of high-betweenness edges belong to the same community.

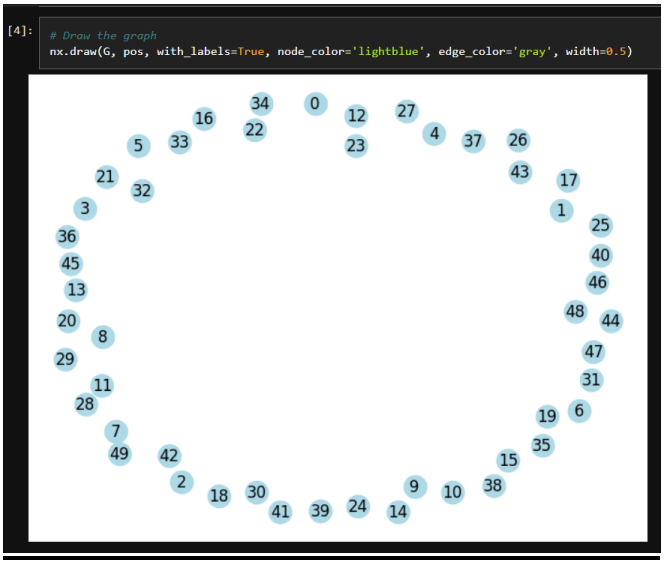
The Edge Betweenness Algorithm uncovers communities by targeting edges that play a critical role in connecting different parts of the network. By iteratively removing these edges, the algorithm gradually reveals the underlying community structure.

One limitation of the Edge Betweenness Algorithm is that it can be computationally expensive, especially for large networks, as calculating betweenness centrality for each edge requires evaluating the shortest paths between nodes. Additionally, the algorithm tends to create hierarchical community structures, where larger communities are further split into smaller ones.

Despite these limitations, the Edge Betweenness Algorithm has been widely used in community detection and has provided valuable insights into the modular organization of various types of networks, including social networks, biological networks, and transportation networks.

Screenshot of Community:

```
[4]: # Draw the graph
     nx.draw(G, pos, with_labels=True, node_color='lightblue', edge_color='gray', width=0.5)
```



Fast Greedy Algorithm for community detection

The Fast Greedy Algorithm, also known as the Clauset-Newman-Moore algorithm, is a community detection algorithm that aims to optimize the modularity of a network. It is a hierarchical agglomerative algorithm that iteratively merges communities to maximize the modularity score.

Here's a brief summary of the Fast Greedy Algorithm:

Initialization: Each node in the network is initially assigned to its own community.

Modularity Calculation: The algorithm calculates the modularity of the initial partition. Modularity measures the quality of the division of nodes into communities by comparing the density of edges within communities to the expected density if the edges were randomly distributed.

Community Merging: The algorithm iteratively merges communities based on their modularity gain. At each step, it considers all pairs of adjacent communities and calculates the change in modularity if they were to be merged. The pair with the highest modularity gain is merged into a single community.

Modularity Update: After merging communities, the algorithm updates the modularity value based on the new partition.

Iteration: Steps 3 and 4 are repeated until no further improvement in modularity is possible, or until a stopping criterion is met.

Community Extraction: The resulting partition represents the detected communities in the network.

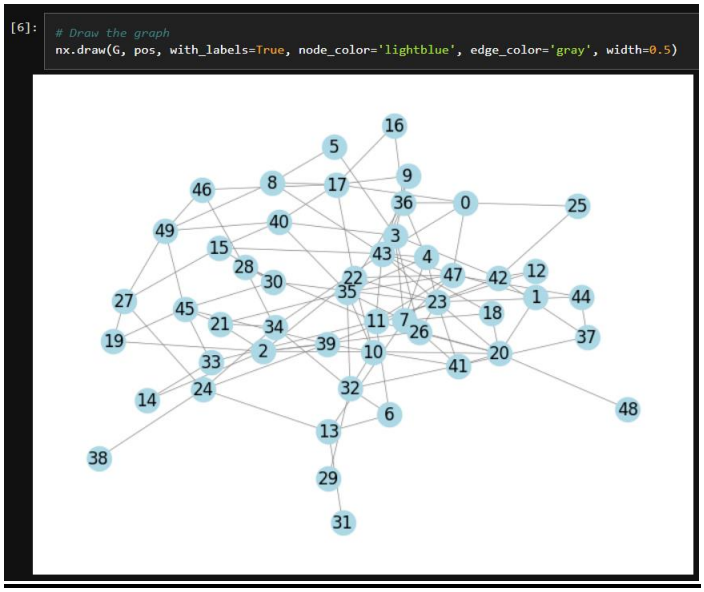
The Fast Greedy Algorithm optimizes modularity by iteratively merging communities that lead to the highest increase in modularity. It explores the network's structure to identify cohesive groups of nodes that exhibit strong internal connections.

One advantage of the Fast Greedy Algorithm is its efficiency compared to other community detection algorithms. It can handle large networks relatively quickly, making it suitable for real-world applications.

However, the Fast Greedy Algorithm may suffer from resolution limit issues, where it tends to merge small communities into larger ones, potentially overlooking finer-scale community structures. Additionally, the algorithm's results can be sensitive to the initialization of communities.

Despite these limitations, the Fast Greedy Algorithm has been widely used for community detection in various domains, including social networks, biological networks, and online networks. It provides a computationally efficient approach to uncovering the modular organization of complex networks.

Screenshot of community



Girvan Newman algorithm for community detection

The Girvan-Newman algorithm, also known as the Edge Betweenness Algorithm, is a community detection algorithm that focuses on identifying communities by iteratively removing edges with high betweenness centrality.

Here's a brief summary of the Girvan-Newman algorithm:

Calculation of Edge Betweenness: The algorithm starts by calculating the betweenness centrality for each edge in the network. Betweenness centrality measures the number of shortest paths passing through an edge, indicating its importance in connecting different parts of the network.

Identification of Edge with Highest Betweenness: The algorithm identifies the edge with the highest betweenness centrality. This edge is considered a crucial bridge between different parts of the network.

Removal of Edge: The identified edge with the highest betweenness centrality is removed from the network.

Recalculation of Betweenness: The algorithm recalculates the betweenness centrality for all remaining edges in the network.

Iteration: Steps 2 to 4 are repeated until a desired number of communities is obtained or until the network is completely fragmented into isolated nodes.

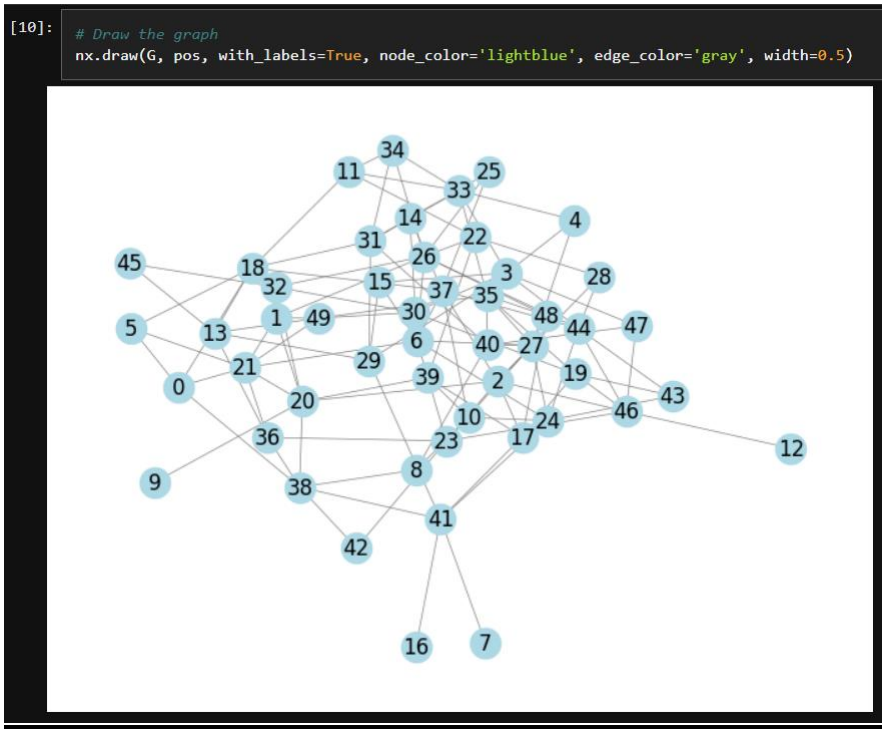
Community Extraction: The resulting communities are extracted based on the connectivity patterns after the removal of high-betweenness edges. Nodes that are still connected after the removal of these edges belong to the same community.

By iteratively removing high-betweenness edges, the Girvan-Newman algorithm gradually uncovers the community structure of a network. The intuition behind this approach is that edges with high betweenness centrality play a crucial role in connecting different communities, and their removal exposes distinct community structures.

One limitation of the Girvan-Newman algorithm is that it can be computationally expensive for large networks, as it requires calculating betweenness centrality for each edge. Additionally, the algorithm may create hierarchical community structures, where larger communities are further split into smaller ones.

Despite these limitations, the Girvan-Newman algorithm has been widely used in community detection and has provided valuable insights into the modular organization of various types of networks, including social networks, biological networks, and transportation networks. It offers an approach to identify communities based on their connectivity and the role of specific edges in bridging different parts of the network.

Screenshot of Community



Louvain Method for community detection

The Louvain Method is a community detection algorithm that focuses on optimizing modularity to identify communities in a network. It is a two-phase algorithm that efficiently detects communities by iteratively optimizing the modularity at different levels of the network's hierarchy.

Here's a brief summary of the Louvain Method:

Phase 1: Local Optimization: The algorithm starts by assigning each node to its own community. It then iterates through all nodes and considers moving each node to its neighboring community. It calculates the change in modularity that would result from moving the node to a different community and selects the move that maximizes the modularity gain. This process is repeated until no further improvement in modularity is possible.

Phase 2: Aggregation: In this phase, the algorithm creates a new network where the communities detected in Phase 1 are treated as nodes. The weights of the edges between communities are determined by the sum of the weights of the original network's edges between the nodes in the respective communities. The algorithm then iterates through the new network and performs Phase 1 (local optimization) again to detect communities at the aggregated level.

Iteration: Steps 1 and 2 are repeated iteratively until no further improvement in modularity is achieved or until a stopping criterion is met. At each iteration, the algorithm builds a new level of community hierarchy.

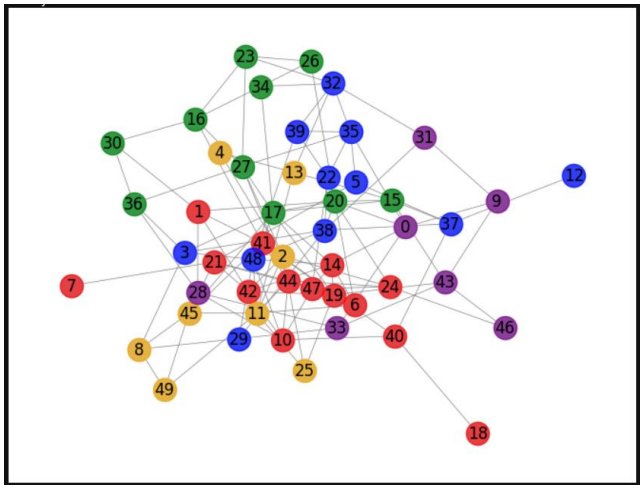
Community Extraction: The final communities are extracted from the community hierarchy obtained in the iterations. Each node belongs to the community at the finest level of the hierarchy where it achieves maximum modularity.

The Louvain Method is known for its scalability and efficiency, making it suitable for large-scale networks. It optimizes modularity by iteratively moving nodes between communities and aggregating communities at higher levels to capture the hierarchical structure of the network.

One advantage of the Louvain Method is its ability to detect both small and large communities, as well as overlapping communities. It can identify communities at different scales, capturing both the local and global structures of a network.

The Louvain Method has been widely applied to various types of networks, including social networks, biological networks, and technological networks. It provides a powerful approach to uncovering the modular organization of complex systems and has been influential in the field of community detection.

Screenshot of community



Spectral Clustering for community detection

Spectral Clustering is a community detection algorithm that leverages the spectral properties of a network to identify communities. It views the network as a graph and uses the eigenvectors of the graph Laplacian matrix to partition the nodes into distinct communities.

Here's a brief summary of Spectral Clustering:

Graph Representation: The algorithm starts by representing the network as a graph, where nodes represent entities (e.g., individuals, vertices, or data points) and edges represent connections or relationships between them.

Graph Laplacian: The Laplacian matrix is constructed from the graph, which captures the relationships between nodes. There are different types of Laplacian matrices, such as the unnormalized Laplacian, normalized Laplacian, or the symmetric normalized Laplacian, each with its own advantages.

Eigenvector Decomposition: The algorithm performs an eigenvector decomposition of the Laplacian matrix to obtain its eigenvectors and eigenvalues. The eigenvectors represent the embedding of the nodes in a lower-dimensional space.

K-means or Normalized Cut: The eigenvectors are used as inputs to traditional clustering algorithms like K-means or normalized cut to partition the nodes into communities. K-means is applied to the eigenvectors directly, while normalized cut divides the eigenvectors into two or more clusters.

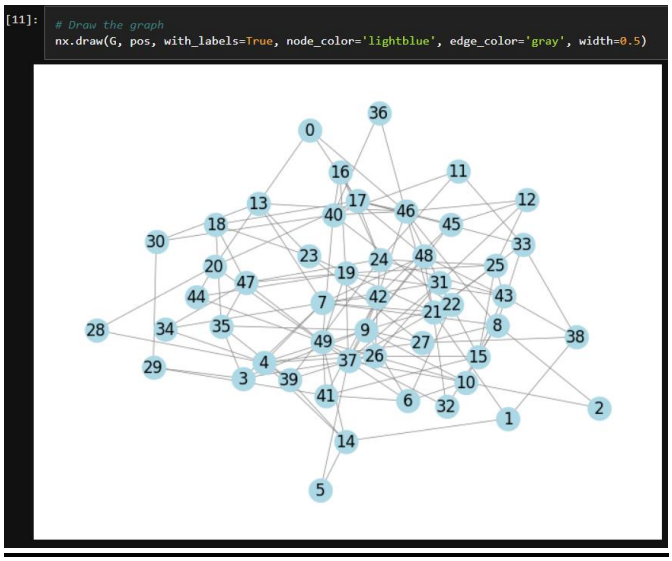
Community Extraction: The resulting clusters obtained from the clustering algorithm represent the communities in the network.

Spectral Clustering is effective in identifying communities that exhibit clear structural patterns in the network. It can handle networks with irregular shapes and is not restricted to detecting communities of similar sizes or densities.

One advantage of Spectral Clustering is its ability to capture non-linear relationships and identify communities even when they are not well-separated. It is particularly useful in applications where communities may have complex interconnections or overlap with each other.

Spectral Clustering has been successfully applied to various domains, including social network analysis, image segmentation, and document clustering. It provides a powerful approach to uncovering communities based on the spectral properties of a network and has gained popularity in the field of community detection.

Screenshot of Community



Walktrap Algorithm for community detection

The Walktrap Algorithm is a community detection algorithm that uses random walks to identify communities in a network. It measures the similarity between nodes based on their random walk behavior and clusters them into communities based on this similarity.

Here's a brief summary of the Walktrap Algorithm:

Random Walks: The algorithm starts by performing random walks on the network. It simulates a random walker moving from one node to another through the network, selecting each neighboring node with equal probability. The length of the random walk is typically determined by a predefined parameter.

Similarity Calculation: The algorithm calculates the similarity between nodes based on the number of times they are visited together during random walks. Nodes that are frequently visited together are considered more similar and likely to belong to the same community.

Distance Calculation: The similarity between nodes is used to calculate a distance metric that reflects the structural similarity between nodes. This distance metric can be defined in various ways, such as the commute time or the number of steps required to transition between nodes.

Hierarchical Clustering: The algorithm applies hierarchical clustering techniques, such as agglomerative clustering, to group nodes into communities based on the calculated distances. It starts by considering each node as a separate community and iteratively merges communities based on their proximity until a stopping criterion is met.

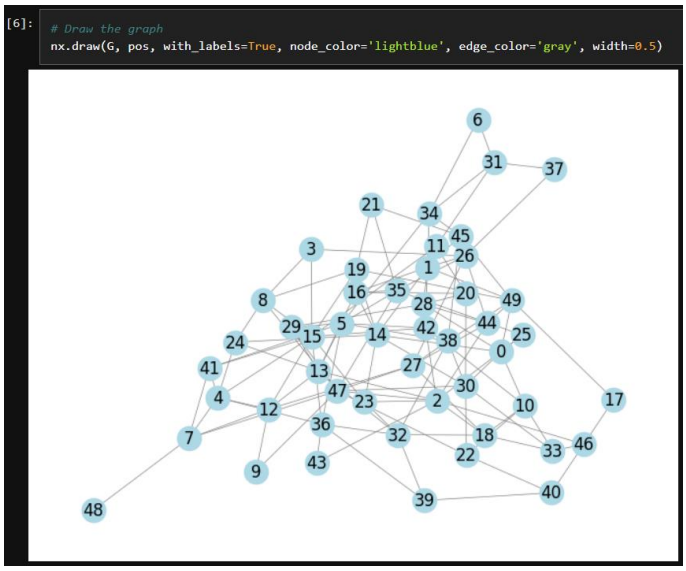
Community Extraction: The resulting hierarchical clustering structure is used to extract communities at different levels of granularity. Different levels of the hierarchical structure correspond to communities of different sizes and densities.

The Walktrap Algorithm is effective in capturing the local connectivity patterns of nodes by considering the shared random walk behavior. It identifies communities based on the idea that nodes that are frequently visited together are more likely to belong to the same community.

One advantage of the Walktrap Algorithm is its ability to handle networks with complex community structures, including networks with overlapping or hierarchical communities. It can uncover communities of different sizes and densities.

The Walktrap Algorithm has been applied in various domains, including social network analysis, biological networks, and web link analysis. It provides a valuable approach to community detection by leveraging the random walk behavior of nodes in a network.

Screenshot of community



Conclusion:

Community detection algorithms play a crucial role in analyzing the structure and organization of complex networks. They provide insights into the modular nature of networks and help identify cohesive groups of nodes with similar characteristics or patterns of connections. Here is a conclusion about community detection algorithms:

Diverse Approaches: Community detection algorithms employ various techniques to identify communities, including optimizing modularity, analyzing network connectivity, measuring similarity, or leveraging random walks. Each algorithm has its own strengths and limitations, making it important to choose the most suitable algorithm for a given network and analysis goal.

Scalability: The scalability of community detection algorithms is a significant consideration, especially for large-scale networks. Some algorithms, like the Louvain Method and Spectral Clustering, are known for their efficiency and ability to handle large networks, while others may struggle with computational complexity.

Resolution and Granularity: Community detection algorithms offer different levels of resolution and granularity in identifying communities. Some algorithms focus on finding large, densely connected communities, while others uncover smaller, more distinct groups.

The choice of algorithm should align with the desired level of resolution and the nature of the network being analyzed.

Overlapping and Hierarchical Communities: Some algorithms can identify overlapping communities, where nodes can belong to multiple communities simultaneously. Additionally, hierarchical community structures can be discovered, revealing nested communities at different levels. These capabilities are particularly useful for networks with complex community structures.

Application Specific: The choice of community detection algorithm should also consider the specific application or domain. Different algorithms may be more suitable for social networks, biological networks, transportation networks, or other types of networks. Understanding the characteristics of the network and the analysis goals is crucial in selecting the appropriate algorithm.

Algorithm Selection: There is no one-size-fits-all community detection algorithm. It is often recommended to try multiple algorithms and compare their results to gain a comprehensive understanding of the network's community structure. Ensemble methods or combining multiple algorithms can also provide robust and accurate community detection.

Overall, community detection algorithms are powerful tools for uncovering the underlying structure and organization of complex networks. They offer insights into the modular nature of networks, facilitate understanding of network dynamics, and have applications in diverse fields ranging from social sciences to biology and computer science.