DA Assignment 1

Shreeman Agrawal

August 2024

1 Introduction

The Girvan-Newman algorithm is a method for detecting community structure in complex networks. It identifies communities by progressively removing edges from the original network. The algorithm is based on the concept of *edge betweenness*, which measures the importance of an edge in terms of the number of shortest paths that pass through it.

2 Algorithm Overview

The Girvan-Newman algorithm operates as follows:

1. **Calculate edge betweenness:** For each edge in the network, calculate its betweenness. The betweenness of an edge is defined as the number of shortest paths between pairs of nodes that run along that edge. function - calculate_edge_betweenness
2. **Remove the edge with the highest betweenness:** Identify the edge with the highest betweenness and remove it from the network. function - remove_edge
3. **Recalculate betweenness:** After removing the edge, recalculate the betweenness of all remaining edges, as the network's structure has changed.
4. **Repeat:** Repeat the process of removing edges and recalculating betweenness until the network breaks down into distinct communities.

3 Edge Betweenness

Edge betweenness $B(e)$ for an edge e is given by:

$$B(e) = \sum_{s \neq t \neq v} \frac{\sigma_{st}(e)}{\sigma_{st}}$$

where:

- σ_{st} is the total number of shortest paths from node s to node t.
- $\sigma_{st}(e)$ is the number of those shortest paths that pass through edge e.

The algorithm's effectiveness is rooted in the assumption that edges connecting different communities will have high betweenness because many shortest paths will pass through them. The Girvan-Newman algorithm is a popular method for detecting community structure in networks. However, determining the optimal partition of the network into communities requires an effective stopping criterion. This document presents an automated algorithm that uses modularity to identify the best community structure and provides a discussion on suitable stopping criteria.

I have implemented the Brandes' algorithm for finding the edge betweenness of each node.

4 Algorithm Description

The algorithm proceeds as follows:

1. **Initialization:**
   - Compute the initial modularity Q of the entire network, considering it as a single community.
2. **Iterative Process:**
   (a) **Calculate Edge Betweenness:** For each edge in the network, compute its betweenness centrality.
   (b) **Remove Edge:** Identify and remove the edge with the highest betweenness.
   (c) **Update Community Structure:** Recalculate the connected components in the network, which now represent the updated communities. function - identify_communities (implemented simple bfs for community detection)
   (d) **Compute Modularity:** Calculate the modularity Q of the current partition. function - calculate_modularity_gain
   (e) **Track Best Modularity:** Maintain the maximum modularity $Q_{\max}$ observed so far and store the corresponding community structure.
3. **Stopping Criterion:**
   - Stop when the modularity Q reaches its peak and begins to decrease after further edge removals.
4. **Output:**
   - Return the community structure corresponding to the maximum modularity $Q_{\max}$.

5 Stopping Criteria

The choice of stopping criterion is essential to ensure that the algorithm identifies the most meaningful community structure. Below are possible criteria:

5.1 Maximum Number of Communities

Criterion: Stop when the number of detected communities exceeds a specified maximum $C_{\max}$.

Justification: This criterion can prevent over-segmentation and is useful when a specific number of communities is desired.

6 Importing and cleaning data

6.1 `import_wiki_vote_data`

This function reads the Wikipedia Vote dataset from a specified file path, constructs an undirected graph, and removes duplicate edges. The process is as follows:

- **Input Data Reading:** The function reads the dataset using `pandas` with tab-delimited values. The dataset contains edges between nodes labeled `FromNodeId` and `ToNodeId`.
- **Node Set Creation:** A set of unique nodes is created from the dataset to ensure no duplicates.
- **Graph Construction:** An undirected graph is constructed using a custom `Graph` class. Each node is assigned a unique identifier, and edges are added between nodes. Duplicate edges are automatically removed due to the set-based nature of the graph.

The function returns the constructed graph and the total number of unique nodes.

6.2 `import_lastfm_asia_data`

Similar to the `import_wiki_vote_data` function, this function reads the Last.fm Asia dataset, constructs an undirected graph, and removes duplicate edges. The main difference lies in the input format, which is read directly as a CSV file.

- **Input Data Reading:** The function reads the dataset using `pandas`. The dataset contains pairs of nodes representing connections between users.
- **Node Set Creation:** A set of unique nodes is extracted to ensure that no duplicates exist.

- **Graph Construction:** An undirected graph is built using the same process as the previous function, with unique node identifiers and edge creation. Duplicate edges are not added.

This function also returns the constructed graph and the number of unique nodes.

7 Louvain Algorithm Results

The Louvain algorithm was applied to detect communities in the datasets. After all iterations, the resulting communities were saved.

7.1 Wiki-Vote Dataset

- **Number of communities found:** 28, Fig. 1

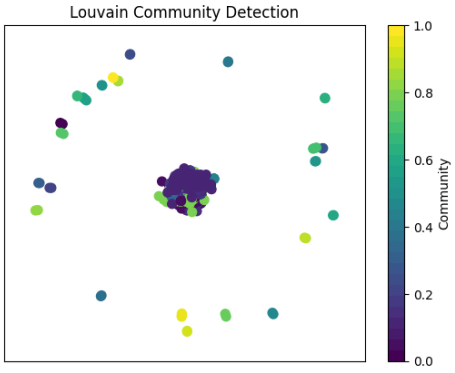

Figure 1: Result of Louvian on Wiki-Vote

7.2 LastFM Asia Dataset

- **Number of communities found:** 27, Fig. 2

8 Time Complexity

The Girvan-Newman and Louvain algorithms are both used for community detection, but they differ in their computational complexity and scalability. The Girvan-Newman algorithm detects communities by iteratively removing edges with the highest betweenness centrality and finding connected components. Its time complexity is $O(k \cdot (n \cdot (m + n \log n)))$, where k is the number of iterations,

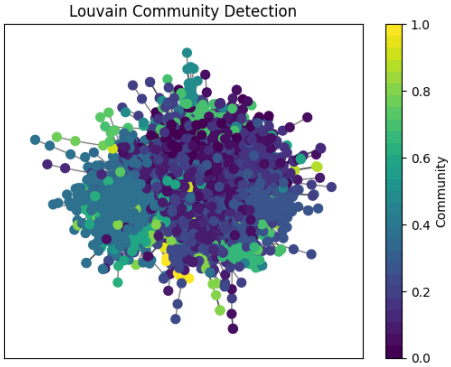

Figure 2: Result of Louvian on LastFM Asia Dataset

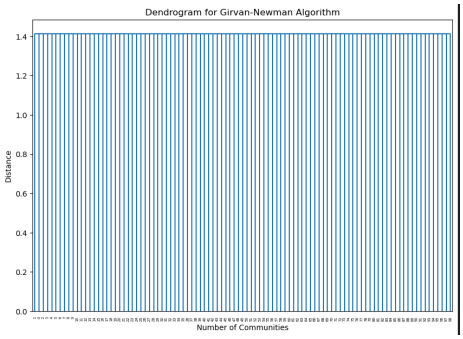

Figure 3: Dendogram of Girvan on LastFM Asia Dataset for 100 levels

n is the number of nodes, and m is the number of edges. This can be computationally expensive for large networks. In contrast, the Louvain algorithm operates through modularity optimization and community aggregation. Its time complexity is $O(n \cdot m)$ per iteration, with a generally small number of iterations required. The Louvain algorithm is more scalable and efficient, making it better suited for large networks, while the Girvan-Newman algorithm may provide more precise community detection in smaller or less complex networks.

9 Comparison

The Girvan-Newman algorithm and the Louvain algorithm are both effective for community detection but differ significantly in their approach and efficiency. The Girvan-Newman algorithm is known for its precision in identifying well-defined communities by iteratively removing edges with the highest betweenness centrality, which can be computationally expensive and less scalable for large networks. In contrast, the Louvain algorithm is designed to be efficient

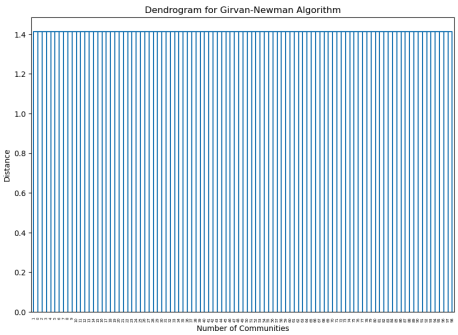

Figure 4: Dendogram of Girvan on Wiki-Vote for 100 levels

and scalable, optimizing modularity to reveal meaningful community structures even in large networks. While the Louvain algorithm often provides practical performance and handles large datasets more effectively, it may occasionally get trapped in local optima, which can impact the quality of community detection. Overall, the choice between these algorithms depends on the size of the network and the need for precision versus computational efficiency.