Mohamed V University

École Nationale Supérieure d'Informatique et d'Analyse des Systèmes

Data Engineering Project

Community Detection On Citation Network Of DBLP

Authors
Fatiha BARRADE
Fatima NOUTFI

Supervisors
Pr.Mohamed LAZAAR

May 26, 2024

Abstract

This study explores the role of community detection in understanding academic citation networks, specifically focusing on computer science literature. Using the comprehensive DBLP dataset, advanced algorithms are employed to reveal thematic clusters and collaboration patterns among scholarly documents and authors. The research highlights the significance of community detection in navigating vast bodies of literature, identifying influential works, and tracking emerging research trends. Through rigorous analysis, the study demonstrates the effectiveness of community detection algorithms in uncovering meaningful insights from citation networks, ultimately contributing to the advancement of knowledge in computer science.

Keywords : Community Detection - Academic Citation Networks-DBLP Dataset - Louvain Method -Girvan-Newman Algorithm - Advanced Algorithms

Contents

1 Introduction

The modern era of scientific research is characterized by the proliferation of vast networks of interconnected scholarly documents, facilitating the exchange and dissemination of knowledge across diverse disciplines. Among these networks, citation networks stand out as invaluable repositories of intellectual discourse, reflecting the intricate web of references and influences that underpin academic scholarship. In this context, community detection emerges as a vital computational task aimed at uncovering the underlying structure and thematic cohesion within citation networks.

Community detection algorithms play a pivotal role in deciphering the complex landscape of scholarly communication by identifying cohesive clusters or communities of interrelated papers or authors. These communities not only reveal distinct research topics and collaboration patterns but also provide insights into the evolution and dynamics of academic discourse over time. Understanding the community structure within citation networks is essential for researchers seeking to navigate the vast corpus of literature, identify influential works, and track emerging research trends.

The significance of community detection in citation networks is further underscored by the availability of comprehensive bibliographic databases such as DBLP (Digital Bibliography & Library Project). DBLP stands as a cornerstone resource in the field of computer science, offering a rich collection of metadata on academic papers, conferences, and authors. Leveraging the DBLP dataset, researchers are empowered to explore the intricate interconnections between papers, authors, and research topics, thereby advancing our understanding of the underlying dynamics shaping the computer science landscape.

In this study, we embark on a comprehensive exploration of community detection in academic citation networks, drawing upon the wealth of data provided by the DBLP dataset. By employing state-of-the-art community detection algorithms and rigorous evaluation metrics, we aim to unveil the structural properties, thematic clusters, and collaborative networks inherent in computer science literature. Through our analysis, we seek to elucidate the fundamental role of community detection in unraveling the complex fabric of scholarly communication and its implications for advancing knowledge in the field of computer science.

2 Problem Statement

The proliferation of interconnected scholarly documents in modern scientific research has led to the emergence of citation networks as invaluable repositories of knowledge exchange. Within these networks, community detection algorithms play a crucial role in uncovering the underlying structure and thematic cohesion, thereby facilitating navigation and understanding of scholarly communication.

However, despite the significance of community detection in citation networks, challenges persist in accurately identifying cohesive clusters or communities within these vast networks. These challenges stem from the dynamic nature of scholarly discourse, the inherent noise and heterogeneity in citation data, and the need for scalable and efficient algorithms to handle large-scale datasets.

Therefore, the primary problem statement of this project is to explore and address the challenges associated with community detection in academic citation networks, particularly focusing on the DBLP dataset. Specifically, we aim to:

- 1- Develop and evaluate state-of-the-art community detection algorithms capable of accurately identifying cohesive clusters within citation networks.
- 2- Assess the performance and scalability of these algorithms in handling large-scale citation datasets such as DBLP.
- 3- Investigate the structural properties and thematic clusters revealed by community detection algorithms, providing insights into the dynamics of scholarly communication and collaboration within the field of computer science

2.1 Related Work

The field of Big Scholarly Data, encompassing vast amounts of academic information and collaboration networks, has garnered significant attention from researchers in recent years, owing to its potential to enhance various scholarly activities. A comprehensive survey conducted by [2],[3] offers valuable insights into the state-of-the-art techniques and technologies utilized in handling Big Scholarly Data, while also delving into research areas such as scientific impact evaluation, academic recommendation systems, and expert identification.

One notable contribution by involves modeling Big Scholarly Data as an academic Social Network, comprising heterogeneous entities such as researchers, venues, and publications, along with their intricate relationships. Analyzing this network structure enables researchers to optimize resource utilization across different application domains. Furthermore, relationship-oriented applications, including academic recommendation systems and community detection algorithms, leverage academic social networks to derive actionable insights.

[4] propose a model that incorporates three types of relations (Researcher-Researcher, Researcher-Article, and Article-Article) to assess academic influence for both researchers and articles. Their algorithm, based on random walk with restart, dynamically considers temporal variations in academic influence, thereby aiding researchers in navigating research collaborations and identifying emerging trends. Additionally, collaboration intensity between scholars is evaluated through the Collaboration Intensity Index (CII) introduced by [4], shedding light on the dynamics of scholarly interactions within the network.

Moreover, [5] present an infrastructure leveraging the Amazon EC2 platform to construct a graph-based knowledge base from Big Scholarly Data. By defining entities such as papers, authors, and venues, along with relationships such as recommendation and collaboration, this framework facilitates various tasks including potential collaborator discovery and research paper recommendation. These endeavors underscore the importance of leveraging network-oriented approaches to glean valuable insights from Big Scholarly Data and foster advancements in academic research and collaboration.

3 Dataset

The dataset encompasses 5,259,858 papers and 36,630,661 citation relationships as of January 31, 2023. It is exclusively designed for research applications. Drawing from diverse sources like DBLP, ACM, MAG (Microsoft Academic Graph), and others, the citation data is meticulously curated. The initial release comprises 629,814 papers and 632,752 citations, with each paper entry containing essential metadata such as abstract, authors, year, venue, and title.

This dataset caters to a wide array of research endeavors, including clustering analysis incorporating network and side information, exploration of influence dynamics within the citation network, identification of influential papers, and facilitation of topic modeling analysis.

.

Attribute	**Description**	**Example**
#* Paper Title	Title of the paper	Information Geometry of U-Boost and Bregman Divergence
#@ Authors	Authors of the paper	Noboru Murata, Takashi Takenouchi, Takafumi Kanamori, Shinto Eguchi
#t Year	Year of publication	2004
#c Publication Venue	Venue where the paper was published	Neural Computation
#index 00 Index ID	Index ID of the paper	436405
#% References	IDs of referenced papers in the paper	94584, 282290, 605546, 620759, 564877, 564235, 594837, 479177, 586607
#! Abstract	Abstract of the paper	We aim at an extension of AdaBoost to U-Boost...

3.1 Data Version

The following table presents a summary of the citation networks, including their size and date of data collection, providing a comprehensive overview of the available datasets for academic research and analysis.

Citation Network	Papers	Citation Relationships (Date)
Citation-network V1	629,814	632,752 (2010-05-15)
Citation-network V2	1,397,240	3,021,489 (2010-09-13)
DBLP-Citation-network V3	1,632,442	2,327,450 (2010-10-22)
DBLP-Citation-network V4	1,511,035	2,084,019 (2011-01-08)
DBLP-Citation-network V5	1,572,277	2,084,019 (2011-01-08)
DBLP-Citation-network V6	2,084,055	2,244,018 (2013-09-29)
DBLP-Citation-network V7	2,244,021	4,354,534 (2014-05-25)
DBLP-Citation-network V8	3,272,991	8,466,859 (2016-07-14)
ACM-Citation-network V8	2,381,688	10,476,564 (2016-04-02)
ACM-Citation-network V9	2,385,022	9,671,893 (2017-01-20)
DBLP-Citation-network V9	3,680,007	1,876,067 (2017-07-03)
DBLP-Citation-network V10	3,079,007	25,166,994 (2017-10-27)
DBLP-Citation-network V11	4,107,340	36,624,464 (2019-05-05)
DBLP-Citation-network V12	4,894,081	45,564,149 (2020-04-09)
DBLP-Citation-network V13	5,354,309	48,227,950 (2021-05-14)
DBLP-Citation-network V14	5,259,858	36,630,661 (2023-01-31)

3.2 Literature Review

This literature review section aims to explore and synthesize key concepts and theories related to network analysis, focusing on unsupervised learning methods, community detection, network citation analysis, modularity, edge betweenness, and graph networks.

3.2.1 Unsupervised Learning Methods

Unsupervised learning methods involve extracting patterns or structures from data without the need for labeled outcomes. Common techniques include clustering, dimensionality reduction, and anomaly detection. These methods play a crucial role in exploratory data analysis and uncovering hidden patterns within datasets.

3.2.2 Community Detection

Community detection refers to the process of identifying densely connected groups or communities within a network. These communities often represent clusters of nodes that share similar attributes or interact frequently. Community detection algorithms help reveal the underlying structure and organization of complex networks, facilitating insights into network dynamics and behavior.

3.2.3 Network Citation Analysis

Network citation analysis examines how publications reference and cite each other within a network, providing insights into scholarly communication. It reveals citation patterns, identifies influential works, and uncovers thematic clusters within research domains. This analysis aids researchers in navigating scholarly literature, understanding research trends, and assessing the impact of publications. Network citation is crucial for identifying emerging topics, collaboration patterns, and interdisciplinary connections, fostering innovation and knowledge dissemination in academia.

3.2.4 Modularity

Definition and Expression

Modularity is a measure of the structure of networks or graphs, specifically quantifying the strength of division of a network into modules (also called clusters or communities). It assesses whether the edges within communities are more frequent than what would be expected on the basis of chance.

The modularity of a network partition is given by the formula:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where:

- Q is the modularity score.
- A_{ij} is the adjacency matrix of the graph, where $A_{ij} = 1$ if there is an edge between nodes i and j, and 0 otherwise.
- k_i and k_j are the degrees of nodes i and j, respectively.
- m is the total number of edges in the graph.
- $\delta(c_i, c_j)$ is the Kronecker delta, which is 1 if nodes i and j are in the same community and 0 otherwise.

Range and Interpretation of Values

The value of modularity Q ranges from -1 to 1, where:

- **High Modularity (Close to 1)**: Indicates a strong community structure, with many edges within communities and few between them. This suggests well-defined and cohesive communities.

- **Moderate Modularity (Around 0)**: Indicates that the community structure is weak, with edges distributed in a manner similar to random distribution. Communities are not well-defined.
- **Low Modularity (Close to -1)**: Indicates that the network structure has fewer edges within communities than expected by chance, suggesting a counterintuitive division of the network.

Significance of Modularity

- **Community Cohesion**: High modularity scores indicate strong, cohesive communities where nodes within the same community are densely connected.
- **Insight into Network Function**: Identifying communities with high modularity can reveal functional units within the network, such as groups of webpages on similar topics, social circles, or biological complexes.

3.2.5 Silhouette Score

Definition and Expression

The silhouette score is a measure of how similar an object is to its own cluster compared to other clusters. It combines both cohesion (how close data points in a cluster are) and separation (how distinct a cluster is from others). The silhouette score ranges from -1 to 1, where a higher value indicates better-defined and more distinct clusters.

The silhouette coefficient for a single node i is calculated as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where:

- $a(i)$ is the average distance between node i and all other nodes in the same community.
- $b(i)$ is the average distance between node i and all nodes in the nearest community to which i does not belong.

The overall silhouette score is the mean silhouette coefficient for all nodes.

Range and Interpretation of Values

The value of the silhouette score $s(i)$ ranges from -1 to 1, where:

- **High Silhouette Score (Close to 1)**: Indicates that nodes are well-clustered within their communities and are distinctly separated from nodes in other communities. This suggests well-defined and cohesive clusters.
- **Moderate Silhouette Score (Around 0)**: Indicates that nodes are on or near the decision boundary between communities. Clusters are not well-defined.
- **Low Silhouette Score (Close to -1)**: Indicates that nodes are poorly clustered, possibly assigned to the wrong communities. Nodes are closer to nodes in other communities than to nodes within their own community.

Significance of Silhouette Score

- **Cluster Quality**: High silhouette scores indicate good clustering quality, where nodes are appropriately assigned to distinct communities.
- **Algorithm Comparison**: Silhouette scores provide a quantitative measure for comparing the effectiveness of different community detection algorithms, offering insight into which algorithm best captures the community structure of the network.

3.2.6 Edge Betweenness

Edge betweenness is a centrality measure that quantifies the importance of edges in facilitating communication between nodes within a network. Edges with high betweenness centrality serve as critical bridges or connectors between different parts of the network. Analyzing edge betweenness helps identify bottleneck edges and understand the flow of information or influence within the network. The edge betweenness of an edge e is given by the formula:

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

3.2.7 Graph Networks

Graph networks, also known as network graphs or complex networks, represent entities (nodes) and their relationships (edges) in a structured and interconnected manner. Graph networks provide a versatile framework for modeling and analyzing various systems, including social networks, biological networks, and transportation networks. They enable the exploration of network properties, such as connectivity, centrality, and community structure, to uncover underlying patterns and dynamics.

This literature review provides a comprehensive overview of key concepts and theories in network analysis, laying the foundation for further exploration and research in this field.

4 Implementation

4.1 System Design

Throughout this project, we will undertake several key activities. First, we will begin with a literature review to define key terminology and concepts related to community detection. Next, we will define the problem statement, outlining the need for community detection within our context.

Following the problem definition, we will move on to the critical task of data cleaning and processing, leveraging the DBLP dataset for our analysis. Once the data is prepared, we will apply various community detection algorithms to identify the underlying community structure within the dataset.

After detecting communities, we will proceed to model community networks, constructing new networks based on the detected communities. Subsequently, we will conduct centrality measurement, assessing the centrality of each community within the network. This comprehensive approach will enable us to gain deeper insights into the structure and dynamics of the dataset's community networks.

Organised as following :

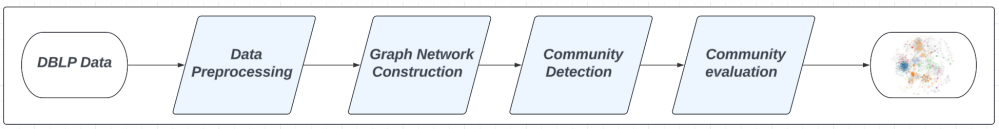

Figure 4.1: System Design

4.2 Data Preprocessing

In this section, we describe the preprocessing steps performed on the DBLP dataset to prepare it for network analysis. The DBLP dataset contains bibliographic information about computer science publications and authors, including titles, authors, publication venues, and citation relationships.

4.2.1 Dataset Format Conversion

In this section, we describe the process of converting the dataset format from `data.txt` to `data.csv` for ease of handling and analysis. The original dataset provided in `data.txt` contains bibliographic information in a delimited text format, which may not be optimal for data manipulation and analysis. Therefore, we performed the following steps to convert the dataset into a comma-separated values (CSV) format (`data.csv`):

- Data Extraction We extracted the bibliographic information from `data.txt`, including titles, authors, publication venues, and citation relationships. Each record in the dataset represents a publication entry, with relevant metadata stored in tabular format.

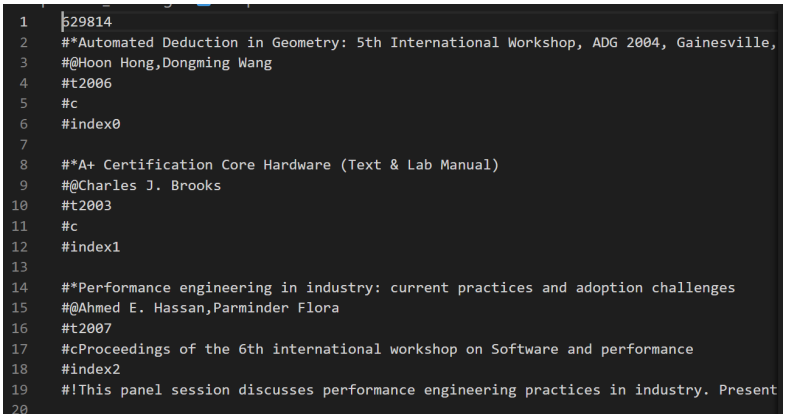

Figure 4.2: Initiale DBLP

- Data Transformation To prepare the dataset for conversion to CSV format, we transformed the data into a structured tabular format. This involved organizing the bibliographic information into rows and columns, with each column corresponding to a specific attribute (e.g., title, authors, venue) and each row representing a publication entry.
- Data Export After organizing the dataset in tabular format, we exported the data to a CSV file (`data.csv`). The CSV file format is widely supported by data analysis tools and platforms, making it easier to import and manipulate the dataset for further analysis.

4.2.2 Data Cleaning

The DBLP dataset may contain inconsistencies, missing values, or noise that could affect the quality of the network analysis. Therefore, we conducted data cleaning to ensure the integrity and consistency of the dataset. This process involved:

```
data.head()
```

	title	authors	year	venue	index	abstract	references
0	Performance engineering in industry: current p...	['Ahmed E. Hassan', 'Parminder Flora']	2007	Proceedings of the 6th international workshop ...	2	panel session discus performance engineering p...	NaN
1	Dude, You Can Do It! How to Build a Sweeet PC	['Darrel Creacy', 'Carlito Vicencio']	2005	NaN	3	whether youre frustrated current pc offering i...	NaN
2	Interpreting Kullback-Leibler divergence with ...	['Shinto Eguchi', 'John Copas']	2006	Journal of Multivariate Analysis	5	kullbackleibler divergence neymanpearson lemma...	436405
3	TOPP---the OpenMS proteomics pipeline	['Oliver Kohlbacher', 'Knut Reinert', 'Clemens...	2007	Bioinformatics	7	motivation experimental technique proteomics s...	NaN
4	Webbots, Spiders, and Screen Scrapers	['Michael Schrenk', 'Michael Shrenk']	2007	NaN	12	internet bigger better mere browser allows web...	NaN

Figure 4.3: DBLP Dataset

– Handling missing values: We addressed missing values by imputation or removal, depending on the context and impact on the analysis.

```
Modularity using Louvain method 0.9538625381110761
Modularity using Girvan-Newman 0.9436172788436755
```

Figure 4.4: NaN values

– Correcting errors: We corrected any errors or inconsistencies in the dataset, such as misspelled names or incorrect references.

this is dataset after cleaning :

	title	authors	year	venue	index	abstract	references
0	Performance engineering in industry: current p...	['Ahmed E. Hassan', 'Parminder Flora']	2007	Proceedings of the 6th international workshop ...	2	panel session discus performance engineering p...	246511
2	Interpreting Kullback-Leibler divergence with ...	['Shinto Eguchi', 'John Copas']	2006	Journal of Multivariate Analysis	5	kullbackleibler divergence neymanpearson lemma...	436405
3	TOPP---the OpenMS proteomics pipeline	['Oliver Kohlbacher', 'Knut Reinert', 'Clemens...	2007	Bioinformatics	7	motivation experimental technique proteomics s...	246511
5	Approximating fluid schedules in crossbar pack...	['Michael Rosenblum', 'Constantine Caramanis',...	2006	IEEE/ACM Transactions on Networking (TON)	17	consider problem motivated desire provide flex...	357875,214023,317448,319987,334185,95255,29412...
6	On product covering in 3-tier supply chain mod...	['Jianer Chen', 'Fenghui Zhang']	2006	Theoretical Computer Science	24	field supply chain management growing rapid pa...	251778,436906,623227,287885

Figure 4.5: DBLP after Cleaning

4.3 Network Construction

In this section, we describe the process of constructing a network representation of the DBLP dataset, focusing on transforming paper entries into nodes and establishing citation relationships between papers. We outline two approaches: based on references and based on titles. Each paper entry in the dataset is transformed

into a node within the network graph. The following attributes are assigned to each node:

- **Index**: The unique identifier of the paper.
- **Title**: The title of the paper.
- **Authors**: The authors of the paper.
- **Year**: The publication year of the paper.
- **Venue**: The publication venue of the paper.
- **Abstract**: The abstract of the paper.

These attributes are associated with each node, enabling comprehensive analysis and visualization of the network.

4.3.1 Reference based Approach

For every paper-reference pair in the dataset, an edge is established between the corresponding nodes. This edge signifies the citation relationship between the citing paper and the referenced paper.

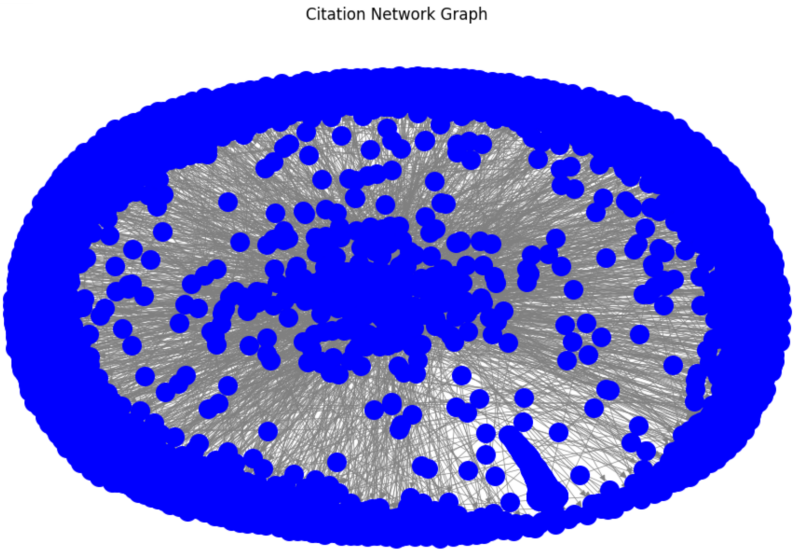

Figure 4.6: Data Network

The preprocessing steps outlined above ensure that the DBLP dataset is clean, consistent, and suitable for network analysis, enabling us to derive meaningful insights from the network structure and citation patterns.

4.3.2 Title similarity Approach

Text Preprocessing

To prepare the titles for analysis, we performed several preprocessing steps. Initially, we converted all text to lowercase to ensure uniformity. We then removed any non-alphanumeric characters and punctuation, which could introduce noise into the dataset. Additionally, we removed common stopwords that do not contribute to the semantic meaning of the text.

Tokenization and Lemmatization

Tokenization was performed to break down the titles into individual words or tokens. This step is crucial for subsequent text analysis as it allows us to handle each word independently. After tokenization, we applied lemmatization to reduce each word to its base or root form. Lemmatization helps in addressing the issue of word inflections and variations, ensuring that words like "running" and "run" are treated as the same token.

Vectorization using TF-IDF

Following the preprocessing steps, we transformed the textual data into numerical vectors using the Term Frequency-Inverse Document Frequency (TF-IDF) method. TF-IDF is a widely used vectorization technique that reflects the importance of a word in a document relative to a corpus. The term frequency (TF) represents the frequency of a word in a document, while the inverse document frequency (IDF) measures the rarity of the word across the entire corpus. The product of TF and IDF provides a weighting scheme that highlights significant words while diminishing the impact of commonly occurring terms.

Mathematically, the TF-IDF value for a term t in a document d within a corpus D is calculated as:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

where

$$\text{TF}(t, d) = \frac{\text{Frequency of } t \text{ in } d}{\text{Total number of terms in } d}$$

and

$$\text{IDF}(t, D) = \log\left(\frac{\text{Total number of documents in } D}{\text{Number of documents containing } t}\right)$$

Cosine Similarity Calculation

To quantify the similarity between the titles, we computed the cosine similarity between the TF-IDF vectors of each pair of documents. Cosine similarity is a measure that calculates the cosine of the angle between two non-zero vectors, providing a metric for their orientation and not their magnitude. This metric is particularly useful for text similarity as it ranges from 0 (no similarity) to 1 (identical documents).

The cosine similarity between two vectors $\mathbf{A}$ and $\mathbf{B}$ is defined as:

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

where $\mathbf{A} \cdot \mathbf{B}$ is the dot product of the vectors, and $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ are the magnitudes of the vectors.

4.4 Louvain Method

4.4.1 Definition and Principle

The Louvain method, proposed by Blondel et al. **blondel2008fast**, is a popular greedy algorithm used for community detection in networks. Its principle is rooted in iteratively optimizing the modularity measure to identify meaningful community structures within the network. Modularity quantifies the quality of a community partition by comparing the density of edges within communities to that expected in a random network.

The algorithm operates in two phases:

1. **Initial Assignment**: Each node is initially assigned to its own community.
2. **Iterative Improvement**: The algorithm iterates through each node and evaluates the potential gain in modularity by moving the node to neighboring communities. If such a move would increase modularity, the node is relocated accordingly. This process continues iteratively until no further improvement in modularity can be achieved.

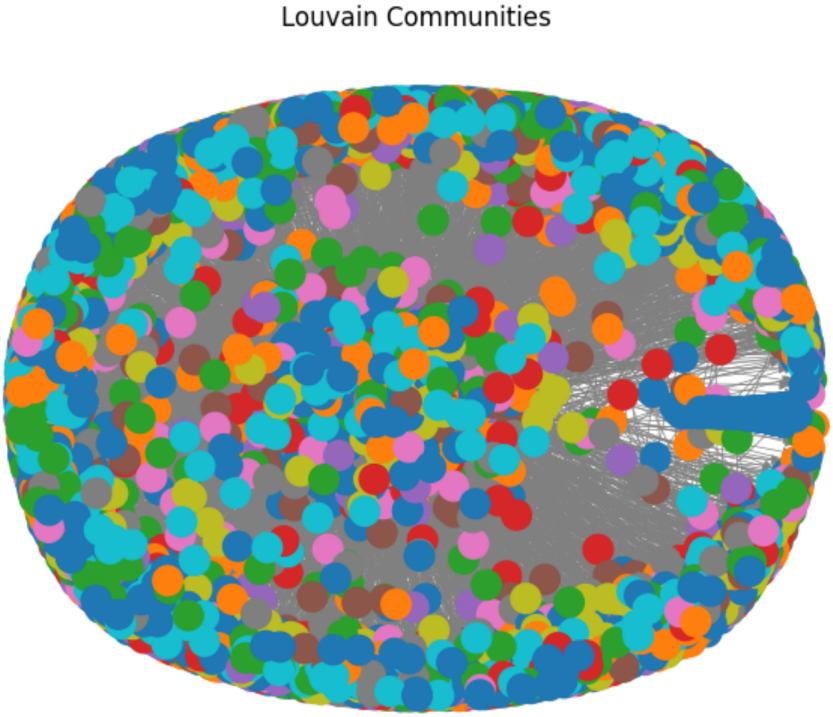

Figure 4.7: Louvain communities

4.5 Girvan-Newman Algorithm

Since its inception in 2002, the Girvan-Newman algorithm has been a widely used method for detecting communities in complex systems. It operates on unweighted graphs and heavily relies on the concept of edge betweenness to identify communities.

4.5.1 Principle

The Girvan-Newman algorithm works in two major steps, repeating until no edges are left:

1. Calculate betweenness of edges.
2. Remove the edge with the highest betweenness.

It is crucial to recalculate the betweenness of edges every time an edge is removed.

Computing Edge Betweenness

Edge betweenness is a measure of centrality in graphs. It represents the number of shortest paths between pairs of vertices that pass through a particular edge.

To compute edge betweenness, a breadth-first search is performed from each vertex to all other vertices. The number of times each edge is traversed is recorded, and the edge with the highest betweenness is removed iteratively until all edges are removed.

4.5.2 Pseudo Code

Here's a simplified pseudo code for the Girvan-Newman algorithm:

1. Calculate betweenness of all edges.
2. Remove the edge with the highest betweenness.
3. Recalculate betweenness of edges.
4. Repeat steps 2-3 until no edges remain.

4.5.3 Selecting Number of Communities

To determine the number of communities, one approach is to stop the algorithm when a desired number of distinct groups is reached. Alternatively, modularity can be computed for different groupings, and the one with the highest modularity is chosen as the final community structure.

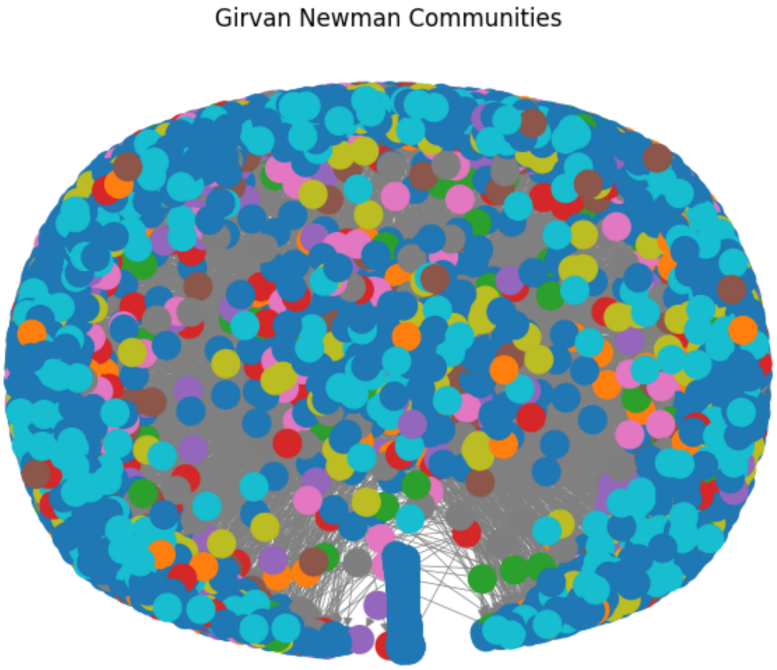

Figure 4.8: Girvan newman communities

4.6 Label Propagation Algorithm

The Label Propagation Algorithm is a semi-supervised machine learning algorithm used to assign labels to unlabeled data observations and partition classification of data observations within a dataset. In community networks, it aims to change the label of each node based on the community label of connected nodes.

4.6.1 Algorithm Principle

1. Each distinct node has a corresponding label representing the community it belongs to.
2. Nodes update their community label based on the community labels of their neighbor nodes until nodes with the same label are attributed to the same community.
3. The updated community of each node is the community with the maximum number of nodes.
4. Densely connected nodes reach a common label community.

4.6.2 Mathematical Essence

There are two kinds of label update processes within label propagation:

Synchronization Update

At each stage t, the community label of node x is updated based on the community labels of its neighbor nodes at stage $t-1$:

$$C_x(t) = \text{argmax}_j \sum_{i=1}^{k} \delta(C_{x_i}(t-1), j)$$

Asynchronous Update

At each stage t, the community label of node x is updated based on the maximum number of community labels of its neighbor nodes at stage t and stage $t-1$:

$$C_x(t) = \text{argmax}_j \left(\sum_{i=1}^{m} \delta(C_{x_i}(t), j) + \sum_{i=m+1}^{k} \delta(C_{x_i}(t-1), j) \right)$$

4.6.3 Algorithm Process

1. Initialize distinct labels for each node within the network community.
2. Set $t = 1$.
3. Arrange nodes within the network in a random order and set them within set X.
4. For each node in set X, update its community label based on the labels of its neighbors.
5. Repeat steps 3-4 until each node has a label such that the maximum number of their neighbors have the same label.

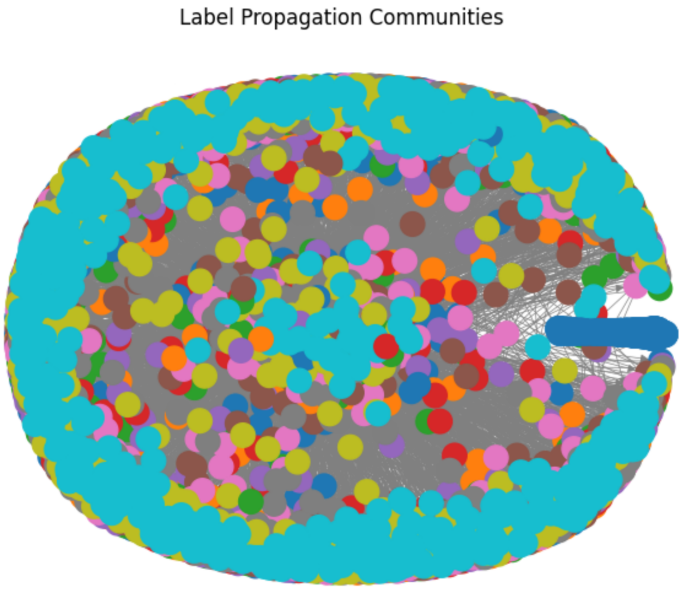

Figure 4.9: Label Propagation communities

4.7 Leiden Algorithm

The Leiden algorithm improves upon the Louvain method by ensuring well-connected communities and enhancing computational efficiency. It introduces an additional refinement phase that addresses the limitations of Louvain, particularly with respect to disconnected communities.

The Leiden algorithm consists of three phases. The first phase involves the local moving of nodes. Nodes are moved between communities to maximize a quality function such as modularity or the Constant Potts Model (CPM). Unlike the Louvain algorithm, Leiden only considers nodes whose neighbors have changed communities, reducing unnecessary recalculations and enhancing efficiency.

The second phase is the refinement of the partition. Each community from the first phase is divided into sub-communities. Nodes within these communities are allowed to move to sub-communities to increase the overall quality function. This phase ensures that communities are well-connected by avoiding the creation of disconnected substructures. It adds a layer of refinement that helps in identifying more precise community structures.

The third phase is the aggregation of the network. The refined communities are aggregated into a new network where each node represents a sub-community. This process is repeated iteratively, using the refined partition as the initial partition for

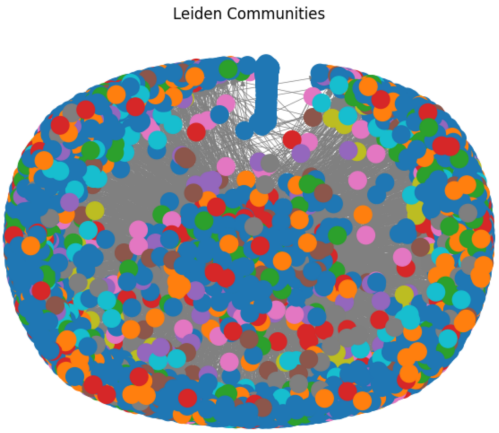

Figure 4.10: Leiden communities

the next iteration. This iterative approach ensures that the community detection process continually improves until an optimal structure is found.

Guarantees Provided by the Leiden Algorithm

The Leiden algorithm provides several guarantees that address the limitations of the Louvain method. First, each iteration of the Leiden algorithm ensures that all resulting communities are connected. This directly addresses the issue in Louvain where communities might be internally disconnected. Second, the refinement phase guarantees that communities are not only connected but also well-separated, which improves the quality and interpretability of the partitions. Finally, over successive iterations, the algorithm ensures that all subsets of all clusters are optimally assigned, leading to highly refined and optimal community structures.

Objective Function

The Leiden algorithm optimizes a modularity-like objective function defined as:

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \gamma \frac{k_i k_j}{2m} \right) \delta(\sigma_i, \sigma_j)$$

where m is the total edge weight, A_{ij} is the weight of the edge between nodes i and j, γ is the resolution parameter, k_i is the degree of node i, σ_i is the community of node i, and $\delta(\sigma_i, \sigma_j)$ is 1 if i and j are in the same community, and 0 otherwise.

4.8 Walktrap Algorithm for Community Detection

The Walktrap algorithm, developed by Pascal Pons, is a prominent method in graph theory designed to identify communities in large networks through the utilization of random walks. The core idea behind Walktrap is that short random walks tend to stay within the same community, thereby providing a natural mechanism for community detection.

Methodology

In Walktrap, the graph G is represented by its adjacency matrix A, where $A_{ij} = 1$ if vertices i and j are connected, and $A_{ij} = 0$ otherwise. The degree $d(i) = \sum_j A_{ij}$ of a vertex i is the number of its neighbors. For simplicity, we assume the graph is connected and unweighted, though the algorithm can be extended to weighted graphs.

The algorithm involves a discrete random walk process on the graph G. At each time step, a walker on a vertex moves to a randomly chosen neighbor. This process is described by the transition matrix P, where the transition probability from vertex i to vertex j is given by:

$$P_{ij} = \frac{A_{ij}}{d(i)}$$

Thus, $P = D^{-1}A$, where D is the diagonal matrix of degrees.

Properties of Random Walks

Two important properties of random walks underpin the Walktrap algorithm:

1. **Stationary Distribution:** As the length t of the random walk increases to infinity, the probability of being at vertex j depends only on the degree of j:

$$\lim_{t\to\infty} P_{ij}^t = \frac{d(j)}{\sum_k d(k)}$$

2. **Symmetry in Transition Probabilities:** The probabilities of transitioning between two vertices i and j in a fixed number of steps t are related by their degrees:

$$d(i)P_{ij}^t = d(j)P_{ji}^t$$

Distance Measure for Community Detection

To group vertices into communities, Walktrap introduces a distance measure r between vertices based on random walks. This distance should be small for vertices within the same community and large for those in different communities. The distance is defined as:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}} = \|D^{-1/2}P_i^t - D^{-1/2}P_j^t\|$$

This distance is then generalized to measure the distance between communities:

$$r_{C1C2} = \sqrt{\sum_{k=1}^{n} \frac{(P_{C1k}^t - P_{C2k}^t)^2}{d(k)}}$$

where P_{Cj}^t is the probability of transitioning from community C to vertex j in t steps.

Hierarchical Clustering with Walktrap

The Walktrap algorithm uses a bottom-up hierarchical clustering approach:

1. **Initialization:** Start with each vertex in its own community. 2. **Merge Step:** Compute distances between all pairs of communities and merge the pair with the smallest distance. 3. **Update Distances:** Recompute distances between the new community and all other communities. 4. **Repeat:** Continue the merge step until a single community remains.

This hierarchical clustering results in a dendrogram, from which the optimal community structure can be extracted.

Computational Complexity

The computational complexity of Walktrap depends on the network's properties. For sparse networks, it is typically $O(n^2 \log n)$, where n is the number of vertices. In the worst case, the complexity is $O(mn^2)$, where m is the number of edges.

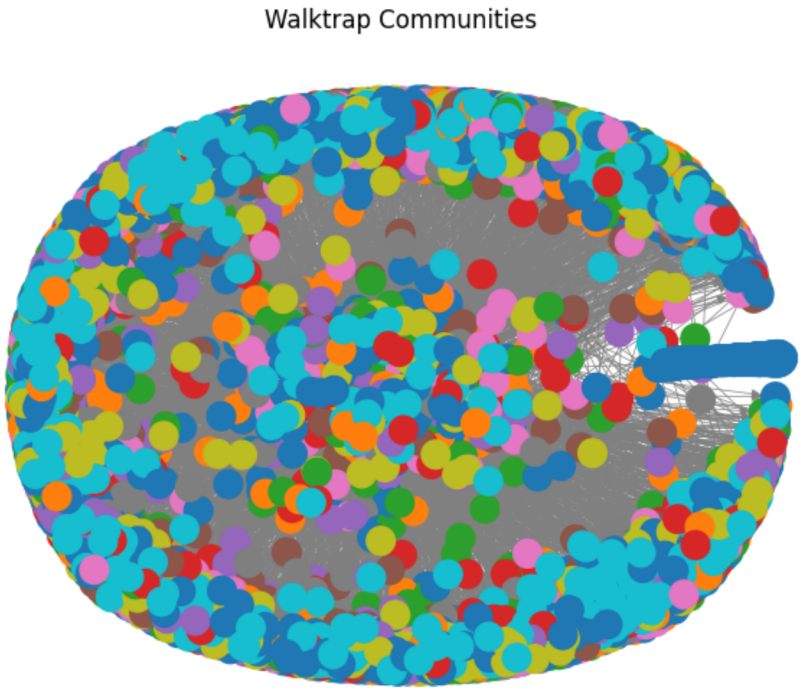

Figure 4.11: walk trap community

4.9 Comparative Analysis of Community Detection Algorithms

In this section, we will compare five community detection methods based on their modularity, silhouette score, and execution time. The methods under consideration are the Louvain method, Girvan-Newman method, Label Propagation method, Walktrap method, and Leiden method.

Modularity measures the strength of division of a network into communities, with higher modularity indicating a better division. The silhouette score measures the quality of the communities discovered, with higher silhouette scores indicating denser and more well-separated communities.

Additionally, we will assess the execution time of each method to evaluate their computational efficiency. By comparing these metrics across the different methods, we aim to assess their effectiveness in identifying meaningful communities within the network while considering computational resources.

Each of these methods comes with its own set of advantages and disadvantages

Table 4.1: Pros and Cons of Community Detection Methods

Method	**Pros**	**Cons**
Louvain	– Efficient for large networks – Optimizes modularity – Identifies well-connected communities	– May produce fragmented communities – Sensitivity to initial conditions
Girvan-Newman	– Identifies communities based on edge betweenness – Detects communities of varying sizes	– Computationally intensive for large networks – May produce overlapping communities
Label Propagation	– Simple and scalable – Fast execution time	– Less optimal community structures – May yield lower silhouette scores
Walktrap	– Captures underlying community structure through random walks – Suitable for analyzing complex networks	– Computational complexity depends on network properties – Performance may vary based on network density
Leiden	– Enhances Louvain method by refining community structures – Ensures well-connected and separated communities	– Slightly higher execution time compared to Louvain – Additional refinement phase may not always improve results

4.10 Network Citation Results Based On References

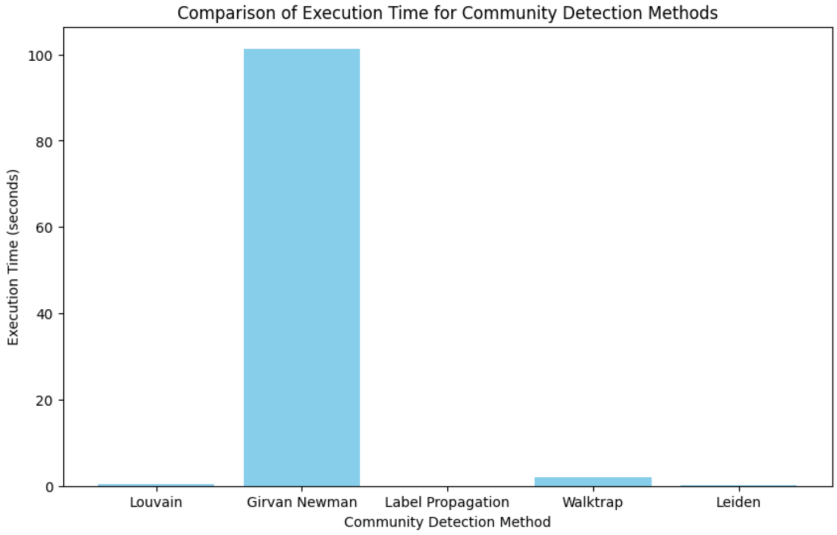

Figure 4.12: Time Execution Comparaison

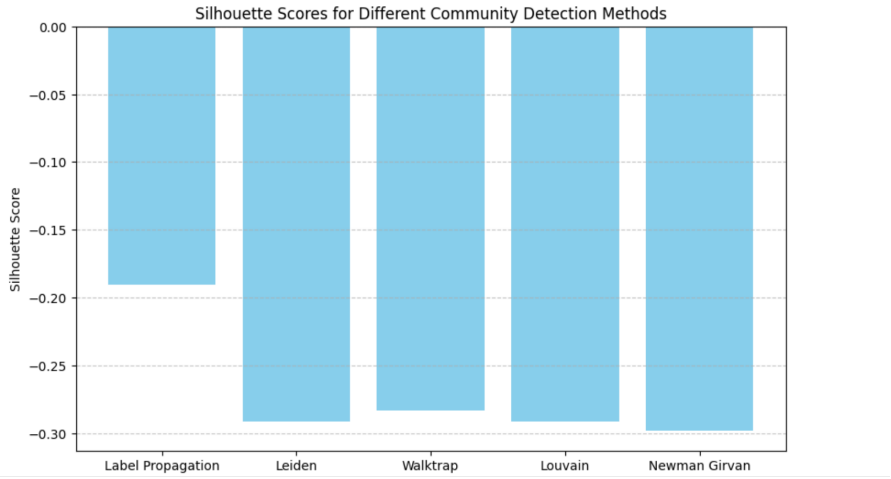

Figure 4.13: Silhouette Value

Based on the execution time, modularity, and silhouette scores obtained for the five community detection methods, a comparative analysis reveals distinct characteristics and performance metrics for each approach.

The Girvan-Newman algorithm exhibited the longest execution time among the methods, taking 101.37 seconds. This extended runtime may limit its suitability for real-time or large-scale network analysis. However, despite its computational demands, Girvan-Newman achieved a respectable modularity score of 0.7728, indi-

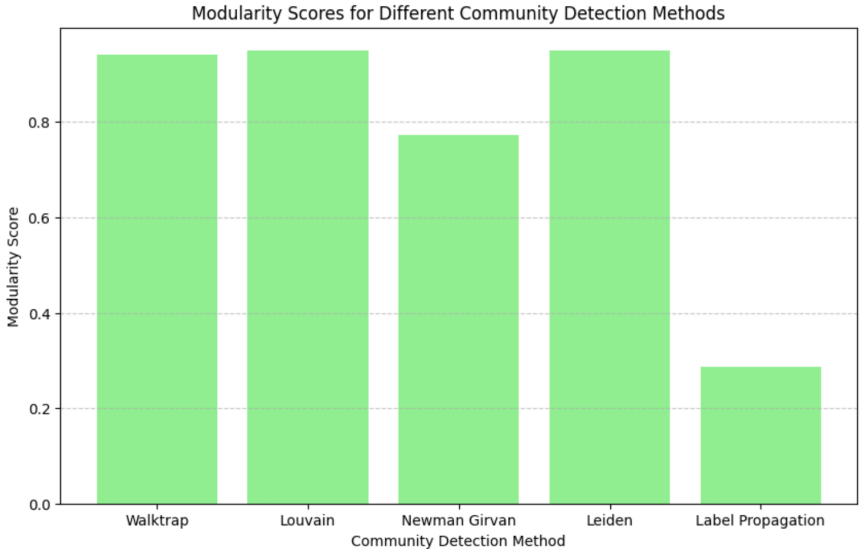

Figure 4.14: Modularity

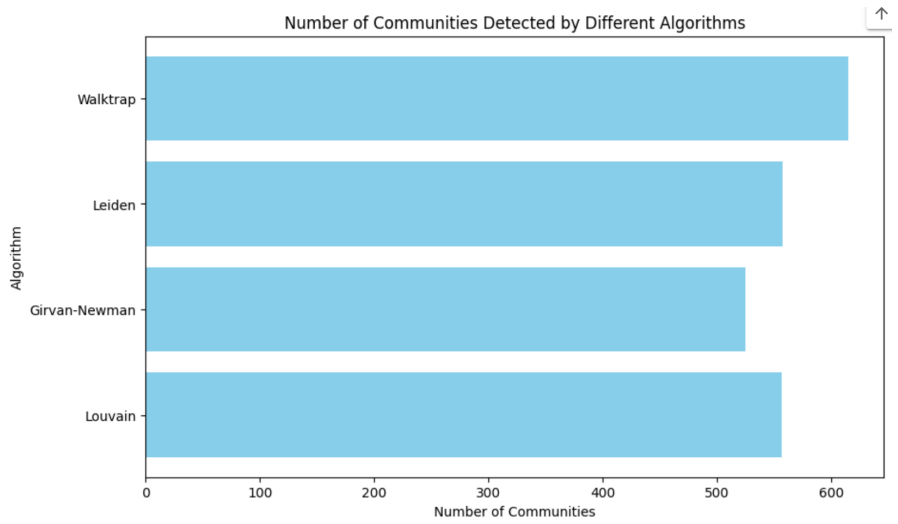

Figure 4.15: Number Of Communities

cating a meaningful division of the network into communities.

In contrast, the Label Propagation Algorithm demonstrated the shortest execution time at merely 0.03 seconds, making it highly efficient for rapid community detection tasks. Despite its swift performance, the algorithm yielded a modest modularity score of 0.2861.

The Leiden algorithm struck a balance between execution time and performance metrics, with an execution time of 0.10 seconds and a competitive modularity score

of 0.9504. This suggests that Leiden efficiently identifies well-connected communities.

Both the Walktrap and Louvain methods exhibited moderate execution times of approximately 2 seconds and 0.35 seconds, respectively. Walktrap achieved a commendable modularity score of 0.9420.

Regarding the silhouette scores, all methods produced negative values. This suggests that the nodes within the communities are not particularly similar to each other, likely because the communities were based on citation references rather than intrinsic node similarities.

In summary, while each community detection method has its strengths and weaknesses, the choice of method may depend on the specific requirements of the network analysis task. The Leiden algorithm appears to offer the best balance of efficiency and performance, whereas Label Propagation excels in speed but may compromise on the quality of community detection. Girvan-Newman, though computationally intensive, can still be valuable for scenarios where accuracy in community division is crucial. The Walktrap and Louvain methods provide a middle ground in terms of both execution time and modularity. The negative silhouette scores across all methods highlight the importance of considering the nature of the data when interpreting community detection results.

4.11 Network Citation Results Based On Title & References

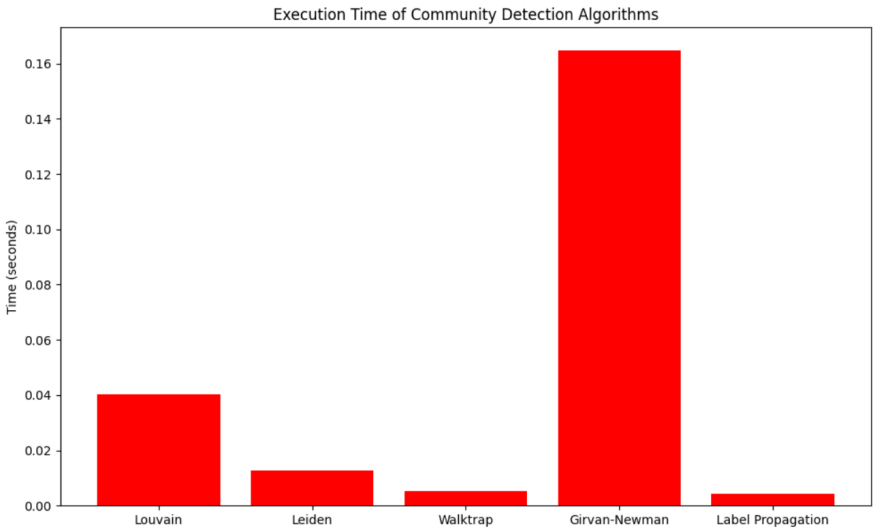

Figure 4.16: Time Execution

Based on the execution time, modularity, and silhouette scores obtained for the five community detection methods, a comparative analysis reveals distinct characteristics

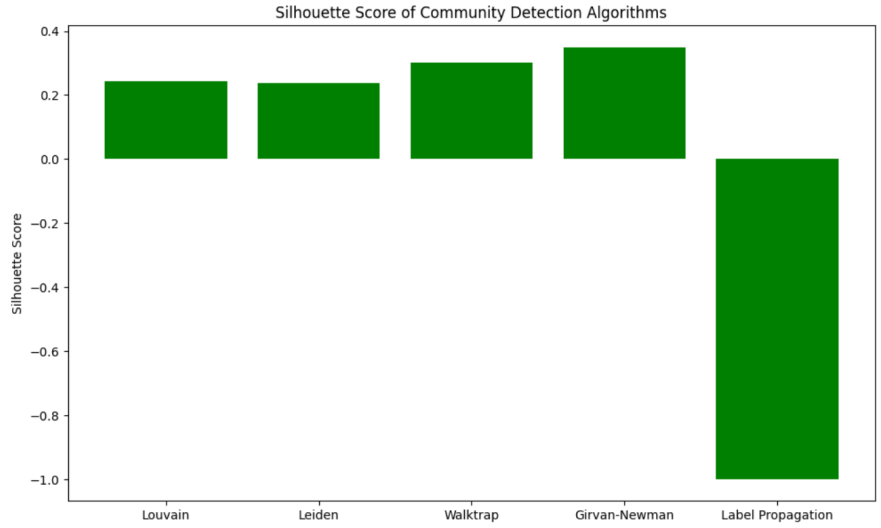

Figure 4.17: Silhouette

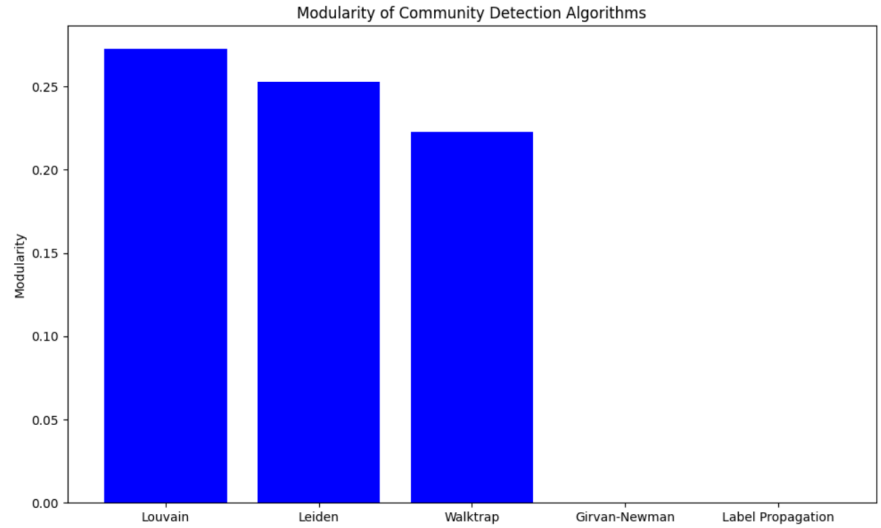

Figure 4.18: Modularity

and performance metrics for each approach.

The Girvan-Newman algorithm exhibited a relatively longer execution time among the methods, taking 0.1625 seconds. This extended runtime may limit its suitability for real-time or large-scale network analysis. Despite its computational demands, Girvan-Newman did not perform well in terms of modularity, achieving a score of -0.0000, indicating no meaningful division of the network into communities. However, it achieved a silhouette score of 0.3497, suggesting that the detected communities, though few (2 communities), had a reasonable level of internal cohesion.

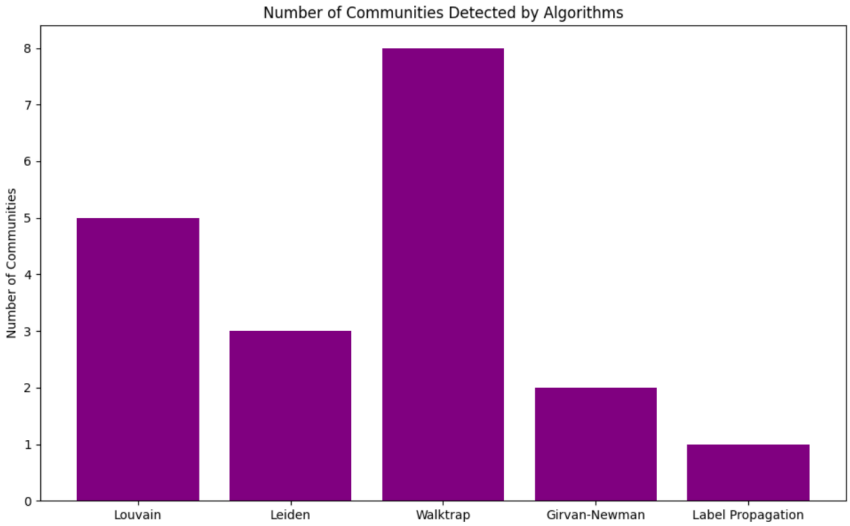

Figure 4.19: Number of Communities

In contrast, the Label Propagation Algorithm demonstrated the shortest execution time at merely 0.0054 seconds, making it highly efficient for rapid community detection tasks. Despite its swift performance, the algorithm yielded a modularity score of -0.0000 and a silhouette score of -1.0000, indicating poor community detection quality, with only 1 community detected.

The Leiden algorithm struck a balance between execution time and performance metrics, with an execution time of 0.0121 seconds and a competitive modularity score of 0.2533. This suggests that Leiden efficiently identifies well-connected communities. It also achieved a silhouette score of 0.2417 and detected 3 communities.

Both the Walktrap and Louvain methods exhibited short execution times of approximately 0.0050 seconds and 0.0395 seconds, respectively. Walktrap achieved a modularity score of 0.2229 and the highest silhouette score of 0.3004 among the methods, indicating good community structure. It detected 8 communities. Louvain had a modularity score of 0.2620, indicating a meaningful division of the network into communities, with a silhouette score of 0.2316 and 5 communities detected.

In summary, while each community detection method has its strengths and weaknesses, the choice of method may depend on the specific requirements of the network analysis task. The Leiden algorithm appears to offer the best balance of efficiency and performance, whereas Label Propagation excels in speed but may compromise on the quality of community detection. Girvan-Newman, though computationally intensive, can still be valuable for scenarios where accuracy in community division is crucial. The Walktrap and Louvain methods provide a middle ground in terms of both execution time and modularity. The negative silhouette scores across some

methods highlight the importance of considering the nature of the data when interpreting community detection results.

4.12 Conclusion and Future work

Conclusion

In conclusion, the comparative analysis of community detection algorithms applied to citation networks reveals a nuanced landscape where the choice of method depends heavily on the specific requirements and constraints of the network analysis task. Each algorithm brings its own strengths and weaknesses to the table, which must be carefully weighed against the desired outcomes.

The Leiden algorithm emerges as a robust performer, striking an excellent balance between execution time and the quality of community detection. With a modest execution time and competitive modularity and silhouette scores, Leiden is well-suited for tasks that require both efficiency and effective community identification.

Label Propagation, on the other hand, excels in speed, making it highly suitable for real-time or large-scale analyses where rapid results are essential. However, this comes at the cost of community detection quality, as evidenced by its lower modularity and silhouette scores, and the detection of only a single community.

The Girvan-Newman algorithm, despite its longer execution time, provides a reasonable silhouette score, indicating that it can identify cohesive communities, albeit fewer in number. This method may be beneficial in scenarios where computational resources are available, and the emphasis is on accurately discerning distinct community structures, even if they are not numerous.

Both Walktrap and Louvain methods offer a middle ground, providing moderate execution times along with respectable modularity scores. Walktrap, in particular, demonstrates the highest silhouette score, suggesting that it identifies well-defined community structures. Louvain also shows a meaningful division of the network into communities, making both methods suitable for tasks requiring a balance of speed and quality.

The generally low silhouette scores across several methods indicate that the detected communities might not be highly cohesive, reflecting the nature of the citation network where papers may not be strongly related. This insight is crucial, as it highlights the importance of considering the intrinsic characteristics of the data when interpreting community detection results.

Overall, this study underscores the critical role of community detection in understanding the structure and dynamics of citation networks. By leveraging the

strengths of different algorithms, researchers can gain valuable insights into the interconnected landscape of scholarly communication, thereby advancing the frontiers of knowledge in various academic fields. The choice of community detection algorithm should be guided by the specific goals of the analysis, the available computational resources, and the inherent properties of the data, ensuring the most effective and insightful outcomes.

4.12.1 Future Work

While this study significantly contributes to the understanding of community detection methods, there are several avenues for future exploration and enhancement.

Firstly, future work could delve into the integration of advanced techniques such as Graph Neural Networks (GNN) for community detection. Leveraging GNNs can potentially enhance the ability to capture intricate patterns and dependencies within complex networks, leading to more accurate community detection results.

Furthermore, there is a scope for improving computational efficiency by harnessing parallelism, especially for handling large-scale networks. Developing parallel algorithms tailored for community detection could significantly accelerate the analysis process and enable the exploration of even larger datasets.

Additionally, the development of a versatile framework capable of accommodating various data types in real-time would be instrumental for practical applications. Such a framework would facilitate the seamless integration of community detection algorithms into diverse domains, enabling real-time analysis and decision-making.

Lastly, future research endeavors could focus on exploring alternative methods for graph construction, beyond the reliance on references. Investigating novel approaches for constructing graphs from raw data sources could provide valuable insights into handling diverse datasets effectively and optimizing the community detection process.

By addressing these avenues for future work, we can further advance the field of community detection, enabling more robust and scalable solutions for analyzing complex networks across diverse domains.

Bibliography

[1] Lazaar, M. "Unsupervised Learning & SNA ."

[2] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, "Fast unfolding of communities in large networks," J. Stat. Mech. P10008, 2008.

[3] Fortunato, S. "Community detection in graphs," Phys. Rep. 486, 2010.

[4] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," Phys. Rev. E 69 (6), 2004.

[5] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, Dorothea Wagner, "On Modularity Clustering," IEEE Trans' on Knowledge and Data Engineering, vol. 20, 2008.

[6] Darby Riley, Kaitlin Mallouk, Courtney Faber, Alexandra Coso Strong. *Adoption of Pedagogical Innovations: Social Networks of Engineering Education Guilds.*

[7] Jicun Zhang, Jiyou Fei, Xueping Song, Jiawei Feng. *An Improved Louvain Algorithm for Community Detection.*

[8] Blaž Skrlj, Jan Kralj, Nada Lavra˘c. *Embedding-based Silhouette Community Detection.*

[9] X. Su et al. *A Comprehensive Survey on Community Detection With Deep Learning. IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 4682-4702, April 2024. doi: 10.1109/TNNLS.2021.3137396.

[10] Muhammad Aqib Javed, Muhammad Shahzad Younis, Siddique Latif, Junaid Qadir, Adeel Baig. *Community detection in networks: A multidisciplinary review.*