IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

A Comparative Study on Community Detection Techniques using Label Propagation, Clique and Walktrap Algorithms in Complex Networks

[1]D. Dhanalakshmi, [2]G. Rajendran

[1]Research Scholar, Periyar University, Salem -11 and Assistant Professor
[1]Computer Science and Applications, [2]Associate Professor and Head
[1]Vivekanandha College of Arts and Sciences for Women (Autonomous),
[1]Tiruchengode, Namakkal, Tamilnadu, India
[2]Department of Computer Science, Govt. Arts and Science College
[2]Modakkurichi, Erode, Tamilnadu, India

Abstract: Studying the fundamental structure of intricate networks, including social, biological, and information systems, depends heavily on community detection. The methods, computational efficiency and community detection capabilities of three well-known algorithms that are Label Propagation Algorithm (LPA), Clique Detection and Walktrap are thoroughly compared in this paper. Using an iterative label diffusion technique, the LPA algorithm provides great scalability and speed for big networks. Although it has a high computational complexity, the Clique algorithm efficiently captures dense structures in biological and social networks by identifying maximal completely linked subgraphs. The Walktrap algorithm utilizes random walks and hierarchical clustering to uncover multi-level community structures with strong modularity but incurs higher computational costs for large graphs. Experimental evaluations were conducted using four real-world datasets: Reddit Hyperlink Network (RH-NW), Amazon Co-purchasing Network (ACP-NW), DBLP Collaboration Network (DBLP-NW) and Twitch Gamers Network (TG-NW). Algorithmic accuracy and efficiency were evaluated using three key performance metrics: Modularity (Q), Normalized Mutual Information (NMI), and Execution Time (T). Results indicate that while LPA is optimal for large-scale datasets due to its linear complexity, Walktrap offers superior modularity for small to medium networks, and Clique excels in identifying dense substructures. This comparative study provides valuable insights into the trade-offs among efficiency, scalability and accuracy in modern community detection techniques.

Keywords: **Community Detection, Complex Networks, Label Propagation Algorithm, Clique Algorithm, Walktrap Algorithm, Random Walk, Hierarchical Clustering, Modularity, Normalized Mutual Information, Execution Time**

1. INTRODUCTION

The rapid expansion of digital communication, biological data networks, and online information systems has given rise to complex networks comprising millions of interconnected nodes. Understanding the hidden structures within these networks is crucial for tasks such as social group analysis, recommendation systems, biological interaction modeling and knowledge graph construction. One of the most fundamental problems in network science is community detection, which aims to identify clusters or groups of nodes that are more densely connected internally than with the rest of the network. Detecting such communities enables deeper insights into the functional, organizational, and behavioral patterns of real-world systems.Over the past decade, numerous algorithms have been developed to address community detection challenges, each adopting distinct computational paradigms. Among these, the Label Propagation Algorithm**,** Clique-based detection**,** and Walktrap algorithm have emerged as three representative approaches, differing in methodology, scalability, and performance. The Label Propagation Algorithm relies on an iterative diffusion mechanism, where each node adopts the most frequent label among its neighbors until convergence. Its major advantage lies in its simplicity and linear computational complexity, making it particularly suitable for large-scale dynamic networks. However, its stochastic nature can lead to non-deterministic outcomes, affecting consistency across runs.In contrast, the Clique algorithm focuses on identifying maximal cliques—subsets of nodes in which every node is directly connected to all others. This approach excels at revealing dense substructures, making it particularly effective in

biological and social networks where strong internal connectivity is a defining feature. Despite its interpretability, clique detection is computationally expensive and less suitable for sparse or extremely large networks due to its NP-hard complexity.

The Walktrap algorithm, on the other hand, integrates the concept of random walks with hierarchical clustering. It operates under the assumption that a random walker tends to remain within the same community due to the higher density of intra-community connections. By computing random walk distances and hierarchically merging similar nodes, Walktrap effectively captures communities at multiple scales. This hierarchical nature allows researchers to explore networks at varying levels of granularity, though the algorithm's quadratic complexity restricts its scalability for massive datasets.

This study presents a detailed comparative evaluation of these three algorithms of LPA, Clique, and Walktrap. Its evaluated across multiple real-world datasets, including the Reddit Hyperlink Network, Amazon Co-purchasing Network, DBLP Collaboration Network and Twitch Gamers Network. The evaluation focuses on three core performance metrics**:** Modularity to assess structural strength, Normalized Mutual Information for accuracy relative to ground truth, and Execution Time for computational efficiency. Through this systematic analysis, the paper aims to highlight the trade-offs between accuracy, scalability, and interpretability**,** providing practical guidance for selecting appropriate community detection techniques in various network contexts.

II. LITERATURE REVIEW

Community detection has been a central research area in network science, driven by the need to understand the underlying modular structures that govern the behavior and dynamics of complex systems. Over the years, numerous algorithms have been proposed, ranging from modularity optimization and random walk–based methods to label diffusion and clique detection. This section reviews the most influential works in the field and situates the Label Propagation Clique**,** and Walktrap algorithms within the broader context of community detection research.

2.1 Early Approache s to Community Detection

The foundation for community detection was laid by Girvan and Newman, who introduced the Girvan–Newman algorithm based on edge betweenness centrality. Their approach systematically removed edges with the highest betweenness to reveal community structures, providing a clear conceptual understanding of how communities form. However, the algorithm's computational cost of O(n3) O(n^3) O(n3) made it impractical for large networks. Subsequent research, notably by Newman, proposed modularity optimization, which became one of the most widely, used quality functions for evaluating community structures. Modularity-based methods, though effective, often struggle with the "resolution limit," where smaller communities are merged incorrectly into larger ones. To address scalability challenges, heuristic and greedy optimization methods were developed [7]. The Louvain algorithm (Blondel et al.,) and Leiden algorithm (Traag et al.,) achieved significant performance improvements, allowing modularity optimization in large-scale networks with millions of nodes. However, these methods primarily rely on modularity maximization and may not effectively detect overlapping or hierarchical communities [2][21].

2.2 Label Propagation Algorithms

The Label Propagation Algorithm (LPA), introduced by Raghavan, Albert, and Kumara, marked a breakthrough in scalability and efficiency. LPA assigns each node a unique label and iteratively updates it based on the most frequent label among its neighbors, effectively mimicking information diffusion in social systems. Due to its near-linear time complexity O(n+m) O(n+m) O(n+m), LPA is particularly well-suited for large and dynamic networks[17]. Research by Leung et al., extended LPA by introducing weighted and asynchronous updating mechanisms, improving stability and reducing randomness in the convergence process [9]. Further advancements, such as the Semi-Supervised LPA and Weighted Propagation models (Liu et al.,) integrated prior information or edge weights to enhance accuracy in heterogeneous networks. Despite its simplicity, the main drawback of LPA remains its non-deterministic nature—different runs on the same network may yield different community partitions due to random tie-breaking [10].

2.3 Clique-Based Methods

Clique-based community detection focuses on identifying maximal cliques, or fully connected subgraphs, which serve as the building blocks of dense network communities. The Bron–Kerbosch algorithm [3] remains one of the most influential approaches for finding all maximal cliques in a graph. This recursive backtracking algorithm systematically expands partial cliques while pruning non-promising candidates. Later refinements, such as pivoting strategies (Tomita et al.,) significantly improved computational performance [22].

Clique detection has found extensive applications in social networks, bioinformatics, and fraud detection, where communities often correspond to strongly interconnected entities. For instance, Palla et al., proposed the Clique Percolation Method (CPM) to detect overlapping communities by identifying adjacent cliques that share nodes. This method effectively captures overlapping structures but is computationally expensive for large and sparse graphs. Despite these limitations, clique-based techniques remain vital for identifying dense substructures and detecting functional modules in biological and social systems [14].

2.4 Random Walk–Based Algorithms

Random walk–based community detection approaches leverage the idea that a random walker tends to remain within the same community due to higher internal connectivity. The Walktrap algorithm, introduced by Pons and Latapy, is one of the most prominent examples of this class. It calculates random walk distances between nodes and performs agglomerative hierarchical clustering, merging communities that minimize the mean random walk distance. The resulting dendrogram provides a multi-scale view of the community structure, it enabling researchers to analyze networks at various levels of granularity.Several studies have highlighted the effectiveness of random walk–based methods in detecting hierarchical and modular structures in networks [17]. For instance, Rosvall and Bergstrom developed the Infomap algorithm, which uses information theory and random walks to identify communities that minimize the description length of the walker's path. Infomap is known for its high accuracy in networks with well-defined community boundaries but has higher computational complexity compared to diffusion-based approaches [20].

Walktrap's strength lies in its balance between accuracy and interpretability; however, it remains computationally intensive for large-scale networks (O(n2log⁡n)O(n^2 \log n)O(n2logn)) and assumes non-overlapping communities. Subsequent works have proposed improved variants using sampling, parallelization, or hybrid random walk–modularity optimization approaches to enhance scalability [22].

2.5 Comparative Evaluations and Current Research Trends

Accuracy, scalability, modularity and the capacity to identify overlapping or hierarchical communities are just a few of the factors that recent comparative studies (e.g., Lancichinetti and Fortunato, Xie et al.) have highlighted. Empirical findings suggest that no single algorithm universally outperforms others and performance often depends on network size, density and structural properties. While LPA offers exceptional computational efficiency, its stochastic convergence can reduce reliability [8]. Clique-based methods excel in precision for dense networks but face scalability issues. Walktrap, although computationally demanding, provides interpretable hierarchical partitions and high modularity values. Current research trends are moving toward hybrid and ensemble-based approaches, combining the strengths of multiple algorithms to achieve both scalability and stability.

In the literature reveals a clear trade-off between computational efficiency and community detection accuracy. This paper extends the body of knowledge by performing a comprehensive empirical comparison of LPA, Clique and Walktrap algorithms using real-world datasets, emphasizing execution time, modularity, and normalized mutual information as evaluation criteria [22].

III. COMMUNITY DETECTION ALGORITHMS

3.1 Label Propagation Algorithm

The LPA is a fast and scalable community detection algorithm that works by iteratively propagating labels among nodes until a stable community structure emerges. It is based on the principle of label diffusion, where each node updates its label based on the most common label among its neighbors. Introduced by Raghavan et al., [19], LPA is particularly useful for large-scale networks due to its simplicity and efficiency.

LPA Algorithm:

Step 1: Initialize Each Node with a Unique Label

- Every node v is assigned a unique label, initially set as its own node ID.
- Let L(v) denote the label of node v. Initially:

$$L(v) = v, \forall_v \in V$$

Where, V is the set of nodes in the graph.

Step 2: Iteratively Update Labels Based on Neighboring Nodes

- In each iteration, each node updates its label based on the most frequent labelamong its neighbors.
- If multiple labels have the same frequency, one is chosen randomly.
- The new label of node v is given by:

$$L(v) = argmax_i \sum_{u \in N(v)} I(L(v) = l)$$

Where:N (v) is the set of neighbors of node v.I (L(u)=l) is an indicator function that returns 1 if node u has label l, otherwise 0.

Step 3: Repeat until Label Stabilization

- The label propagation process is repeated until no node changes its label between two consecutive iterations or a maximum number of iterations is reached.
- The stopping condition can be written as:

$$L_t(v) = L_{t+1}(v), \forall_v \in V$$

Where, $L_t(v)$ and $L_{t+1}(v)$ are the labels of node v in consecutive iterations t and t+1.

Step 4: Identify Communities

- Nodes that share the same label after convergence form a community.
- The final partition of the graph consists of disjoint clusters, each represented by a dominant label.

The LPA is a simple yet powerful community detection method based on the idea of iterative label diffusion. It is a non-parametric approach, meaning that it does not require prior knowledge of the number of communities. LPA starts by assigning each node a unique label, treating every node as its own community. Then, in every iteration, each node updates its label by selecting the most frequently occurring label among its neighbors. This process mimics information diffusion, where nodes tend to adopt the most popular label in their local neighborhood, leading to the formation of cohesive clusters.

LPA is particularly advantageous for large-scale networks due to its linear time complexity of O (n + m), where n is the number of nodes and m is the number of edges. Since nodes update their labels asynchronously, the algorithm can handle dynamic networks efficiently. Additionally, LPA is memory-efficient and can be implemented in distributed systems, making it a preferred choice for big data applications. However, LPA also has certain limitations. Due to the random tie-breaking when multiple labels have the same frequency, the final community structure may vary across different runs, making it non-deterministic. Additionally, LPA may fail to detect smaller communities in the presence of highly connected nodes, as labels from dominant communities tend to spread uncontrollably. To address this, researchers have proposed improved LPA variants, such as Weighted LPA (which incorporates edge weights) and Semi-Supervised LPA (which integrates external information for better accuracy) [10]. The LPA provides an efficient, scalsable and intuitive method for detecting communities in large networks, making it widely applicable in social network analysis, biological clustering and recommendation systems. Despite its stochastic nature, its simplicity and computational efficiency make it an attractive choice for real-world network analysis.

3.2 Clique Algorithm

A clique in a network is a subset of nodes where every pair of nodes is directly connected. The process of clique detection is essential for community detection, social network analysis and biological network modeling. The clique algorithm identifies maximal cliques, meaning cliques that cannot be extended by including additional adjacent nodes. This ensures that the detected cliques represent the largest fully connected subgraphs in a network. Clique detection is typically performed using algorithms such as Bron-Kerbosch, Recursive Backtracking or Pivoting Methods, which search for all maximal cliques efficiently [24].

Clique Detection Algorithm:

Step 1: Define a Clique

AcliqueC in a graph G (V, E) is a subset of vertices such that every node is connected to every other node:

$$\forall_{u,v} \in C, (u, v) \in E$$

Where, V is the set of nodes and E is the set of edges.

Step 2: Initialize the Bron-Kerbosch Algorithm

The Bron-Kerbosch algorithm is a backtracking approach to finding all maximal cliques. It uses three sets:

- R(Current Clique): Nodes forming the current clique.
- P (Potential Nodes): Nodes that can be added to extend the clique.
- X (Excluded Nodes): Nodes already processed and cannot be added.

Initially,

$$R=\emptyset,\ P=V,\ X=\emptyset$$

Step 3: Recursive Clique Expansion

For each node v in P:

- Add v to the current clique:

$$R' = R \cup \{v\}$$

- Update potential candidates for clique expansion:

$$P' = P \cap N(v)$$
$$X' = X \cap N(v)$$

Where, N (v) is the set of neighbors of v.

- Recursively apply the algorithm:

Bron-Kerbosch(R′,P′,X′)

- Remove v from P and add it to X, ensuring it is not reconsidered:

$$P=P-\{v\},\ X=X\cup\{v\}$$

This ensures that only maximal cliques are detected.

Step 4: Apply Pivoting to Optimize Search

To improve efficiency, the algorithm selects a pivot nodeu from P∪X that maximizes the number of neighbors inP. This reduces the number of recursive calls.

The pivot is selected as:

$$u = \arg max_{v \in P \cup X} |P \cap N(v)|$$

By avoiding unnecessary recursion, pivoting significantly improves execution time.

Step 5: Output Maximal Cliques

Once all recursive calls complete, the algorithm returns all maximal cliques, ensuring that no clique can be further expanded.

The Clique algorithm is a fundamental technique in network analysis, used to identify fully connected subgraphs where each node is directly linked to all other nodes in the subset. A maximal clique is a clique that cannot be extended by adding more nodes while maintaining complete connectivity. Clique detection is widely used in social networks (detecting tightly-knit groups), bioinformatics (protein interaction networks) and recommendation systems (finding common user groups). The most widely used method for maximal clique detection is the Bron-Kerbosch algorithm, a recursive backtracking approach that efficiently explores all possible cliques. The algorithm maintains three sets: R (current clique), P (potential extensions) and X (excluded nodes). It iteratively expands cliques by adding new nodes from P while ensuring that redundant searches are avoided using X. To further optimize performance, a pivoting technique is applied, where a node that maximizes the number of potential extensions is chosen to minimize recursive calls.

One of the key challenges of clique detection is its computational complexity, which is NP-hard. The worst-case runtime of Bron-Kerbosch without pivoting is O ($3^{(n/3)}$), but with pivoting, it performs significantly better in practical cases. While the algorithm works well for small to moderate-sized networks, large-scale graphs require approximation techniques or heuristic methods such as Greedy Clique Expansion or Parallelized Bron-Kerbosch. Despite its complexity, clique detection remains crucial for identifying densely connected groups in networks. Its applications span multiple domains, including fraud detection (detecting fraudulent clusters), biological networks (protein structure analysis) and cybersecurity (detecting botnet structures). The ability to uncover hidden structures in complex networks makes clique detection a valuable tool in graph mining and community analysis.

The Clique algorithm is an essential method for identifying tightly connected groups within networks. The Bron-Kerbosch algorithm is widely used due to its recursive backtracking approach, ensuring all maximal cliques are detected efficiently. While highly accurate, clique detection is computationally expensive and not suitable for very large graphs without optimizations. Despite its limitations, clique detection plays a crucial role in social network analysis, bioinformatics and fraud detection and recommendation systems, making it a powerful tool in graph theory.

3.3. Walktrap Algorithm

The Walktrap algorithm is a hierarchical community detection method based on random walks in networks. It assumes that random walkers tend to stay within the same community because intra-community connections are denser than inter-community connections. Introduced by Pons and Latapy [17], Walktrap measures the similarity between nodes using random walk distances and then merges nodes hierarchically to form communities. Walktrap uses a bottom-up hierarchical clustering approach, where small communities are iteratively merged based on their structural similarity, ensuring that communities are discovered at different levels of granularity.

Walktrap Algorithm:

Step 1: Represent the Graph as a Transition Matrix

- The graph is represented as a Markov chain, where the probability of transitioning from node i to node j is:

$$P_{ij} = \frac{A_{ij}}{k_i}$$

Where:A_{ij} is the adjacency matrix (1 if there is an edge between i and j, 0 otherwise).k_i is the degree of node i (total number of edges connected to i).P_{ij} represents the probability of a random walker moving from node i to node j.

Step 2: Compute the Random Walk Distance between Nodes

- Walktrapmeasures the similarity between nodes using random walk distances. The idea is that two nodes belong to the same community if a random walker spends more time within their neighborhood before exiting.
- The random walk distance d(i,j) is computed as:

$$d(i,j) = \sqrt{\sum_k \frac{(P_{ik}^t - P_{jk}^t)}{\pi_k}}$$

Where:$P_{ik}^t$is the probability that a random walker starting at i reaches node k after t steps.π_k is the stationary distribution of node k:

$$\pi_k = \frac{k_k}{2m}$$

Where, k_k is the degree of node k and m is the total number of edges in the graph.

- The square root ensures that the distance is a proper metric.

Step 3: Hierarchical Merging of Nodes into Communities

- Walktrap follows an agglomerative hierarchical clustering approach, where nodes are iteratively merged based on their similarity.
- Initially, each node is treated as a separate community.
- At each step, the two communities C_i and C_j that minimize the random walk distance are merged:

$$C_{new}=C_i \cup C_j$$

This merging process continues until a predefined stopping criterion is met (e.g., a desired number of communities are reached).

Step 4: Generate the Final Community Partitioning

- The hierarchical merging process creates a dendrogram, representing the community structure at different levels.
- The optimal number of communities is selected by analyzing the modularity score at different levels:

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

Where:δ(Ci,Cj) is 1 if nodes i and j belong to the same community, otherwise 0.The best partition is selected based on the highest modularity score.

The Walktrap algorithm is a community detection method that identifies groups of densely connected nodes based on the behavior of random walkers. The core idea is that a random walker tends to stay within the same community, as communities have more internal connections than external ones. By analyzing how random walkers move through the network, Walktrap effectively captures structural similarities between nodes and groups them into communities.

Walktrap operates in a hierarchical manner, beginning with each node as an individual community and iteratively merging the most similar communities based on random walk distances. The similarity between two nodes is computed using the probability distribution of random walks reaching different parts of the network. If two nodes have similar transition probabilities over multiple steps, they are likely to belong to the same community. The random walk distance metric ensures that nodes with strong intra-community ties have shorter distances, guiding the hierarchical merging process. One of the strengths of Walktrap is that it produces a multi-level representation of community structure, meaning that users can choose the optimal number of communities by analyzing the dendrogram (hierarchical tree structure). Additionally, Walktrap optimizes modularity to ensure high-quality partitions.However, Walktrap has some limitations. Its computational complexity is O ($n^2 \log_n$), making it slower than Louvain or Leiden for very large networks. Additionally, Walktrap assumes that communities are well-separated, meaning it may not perform well in networks with overlapping communities. Despite these challenges, Walktrap remains a powerful and intuitive method for community detection, particularly for networks where random walk dynamics provide meaningful insights, such as social networks, biological networks and citation networks.

The Walktrap algorithm is a powerful community detection method that identifies clusters based on random walk behavior. It follows a hierarchical clustering approach, iteratively merging the most similar communities until an optimal partition is reached. Unlike modularity-based methods like Louvain, Walktrap relies on random walk distances, making it effective for capturing structural relationships in social networks, biological networks and citation networks. Despite its higher computational cost, Walktrap provides a multi-resolution view of the community structure, allowing users to choose the granularity of partitioning. However, for very large networks, faster algorithms like Louvain or Leiden may be preferred. Overall, Walktrap remains an intuitive and robust method for detecting hierarchical community structures, particularly in networks where random walk dynamics play a significant role.

IV. RESULTS AND DISCUSSION

The experimental setup for evaluating community detection methods involves testing their performance on diverse real-world datasets, comparing them with established algorithms, and assessing key evaluation metrics. The evaluation is conducted on four real-world network datasets, each representing different types of interactions. The Reddit Hyperlink Network captures hyperlink relationships between subreddits and includes temporal interactions, making it suitable for dynamic community detection. The Amazon Co-purchasing Network represents product co-purchasing relationships, helping analyze hierarchical structures in e-commerce networks. The DBLP Collaboration Network models academic collaborations based on co-authored research papers, making it useful for detecting research communities. The Twitch Gamers Network showcases interactions among Twitch users, including demographic attributes, and is ideal for testing overlapping community detection methods. These datasets provide a robust and varied platform for evaluating the effectiveness and scalability of community detection algorithms. Table 1 provides a comparative analysis of various community detection algorithms, outlining their approaches, strengths, weaknesses, scalability, ability to handle overlapping communities and ideal use cases.

Table 1 Comparison of Existing Community Detection Algorithms

Algorithm	Approach	Strengths	Weaknesses
Label Pro pagation	Iterative Label Diffusion	Extremely fast for large graphs	Unstable results
Clique	Detection of maximal cliques for dense substructures	Works well for biological networks and social networks where dense connectivity is key	Not well-suited for sparse graphs, as clique-based connections are weak

Walktrap	Random Walks + Hierarchical Clustering	Captures multi-level structures; Good for small to medium-sized networks	Computationally expensive for large networks

Community detection algorithms are evaluated based on various metrics to determine their effectiveness in uncovering meaningful structures in networks. The following key evaluation metrics are widely used:

A. Modularity (Q)

Modularity is a graph-based metric that measures the strength of the community structure by comparing the actual edge density within communities to the expected edge density in a random network. Ahigher modularity value (Q)indicates a better-defined community structure.

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

Where, A_{ij} is the adjacency matrix (1 if an edge exists between nodes i and j, 0 otherwise).k_i andk_j are the degrees of nodes i and j.m is the total number of edges in the network.C_i andC_j represent the communities of nodes i and j, respectively.$\delta(C_i,C_j)$ is the Kronecker delta function, which is 1 if nodes i and j are in the same community and 0 otherwise.

Table 2 Modularity for all the methods of Dataset a) RH-NW b)ACP-NW c) DBLP-NW d) TG-NW

Network Size (n)	Data set	LPA	K-Cliques	Chinese Whispers
n=1000	RH-NW	0.655	0.653	0.589
	ACP-NW	0.602	0.619	0.524
	DBLP-NW	0.726	0.744	0.657
	TG-NW	0.812	0.779	0.693
n=5000	RH-NW	0.577	0.589	0.472
	ACP-NW	0.601	0.612	0.487
	DBLP-NW	0.704	0.721	0.579
	TG-NW	0.506	0.522	0.391
n=25000	RH-NW	0.659	0.693	0.589
	ACP-NW	0.615	0.645	0.478
	DBLP-NW	0.628	0.638	0.539
	TG-NW	0.633	0.693	0.521

Table 2 presents the modularity values obtained from three community detection algorithms that are Label Propagation Algorithm, K-Cliques and Chinese Whispers and across four real-world network datasets. The experiments were conducted at different network scales (n =1000, 5000 and 25000) to assess how each algorithm adapts to changes in graph size and complexity.

Modularity is a key indicator of community structure strength, where higher values (closer to 1) signify more well-defined and internally cohesive communities. For RH-NW, modularity values remain relatively stable across all network sizes. K-Cliques achieves the highest modularity at n = 25,000 (0.693), followed closely by LPA (0.659), while Chinese Whispers performs comparatively lower (0.589). This indicates that both LPA and K-Cliques effectively capture community boundaries in this densely linked social dataset. The stability of LPA across sizes also highlights its robustness in dynamic, large-scale environments like Reddit, where communities often overlap and evolve continuously. In the ACP-NW dataset, K-Cliques consistently outperform LPA and Chinese Whispers across all network sizes. This is because co-purchasing patterns in e-commerce networks often form dense subgraphs of related products, which clique-based methods can identify more effectively.

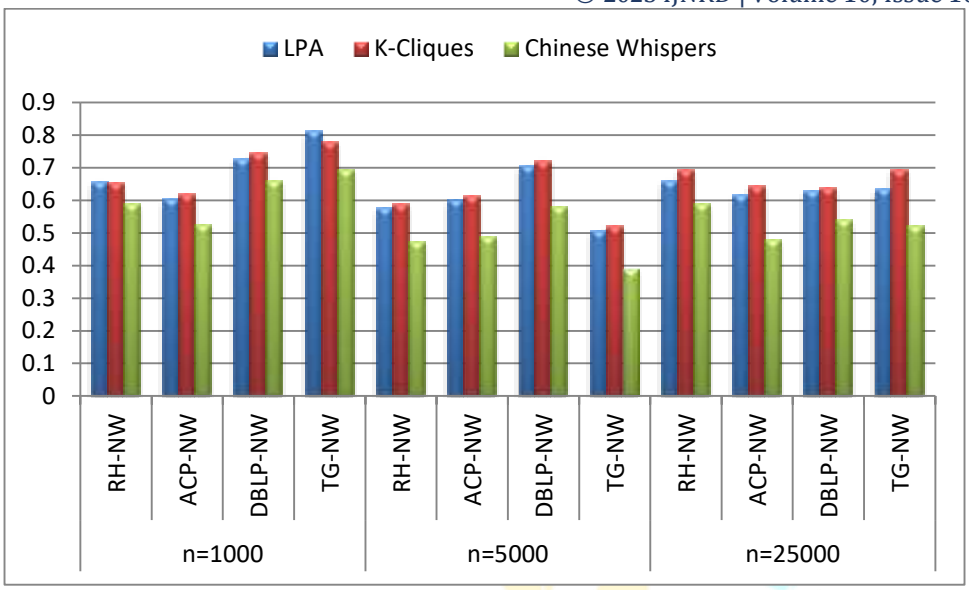

Figure 1 Modularity for all the methods of Dataset a) RH-NW b)ACP-NW c) DBLP-NW d) TG-NW

Fig 1 shows the comparison charts of above table. LPA's performance improves moderately with increasing size (from 0.602 to 0.615), showing its adaptability to larger graphs, whereas Chinese Whispers lags behind due to its reliance on local connectivity, which may not fully capture global product relationships.

The DBLP dataset shows the highest overall modularity scores across all methods, indicating strong, well-defined research collaboration communities. At smaller scales (n = 1000), both LPA (0.726) and K-Cliques (0.744) perform exceptionally well. However, as the network grows to n = 25,000, modularity slightly decreases due to increased inter-community overlaps. This trend suggests that both algorithms maintain their ability to detect meaningful clusters even in complex academic collaboration structures, with K-Cliques having a slight edge due to its focus on highly connected author groups. The TG-NW dataset exhibits a more noticeable drop in modularity as the network size increases. Initially, at n = 1000, LPA achieves a strong modularity of 0.812, outperforming both K-Cliques (0.779) and Chinese Whispers (0.693). However, as the network expands to n = 25,000, modularity decreases (LPA: 0.633; K-Cliques: 0.693). This reduction reflects the complex and overlapping social structures within the gaming community, where user interests and interactions blur traditional community boundaries. K-Cliques performs better at large scale due to its ability to capture tightly knit gamer groups that remain strongly interconnected even as the network grows.

LPA demonstrates consistently high modularity and stable performance across most datasets, particularly excelling in large, dynamic and socially driven networks like RH-NW and TG-NW. Its ability to propagate labels efficiently helps it form cohesive community partitions. However, the slight variability in modularity at larger scales can be attributed to its stochastic nature and random tie-breaking during label updates. K-Cliques generally yields the highest modularity scores, especially in structured and dense datasets like DBLP and ACP-NW. This is because clique-based detection inherently identifies strongly connected clusters that contribute to higher modularity. However, its computational cost increases significantly with network size, limiting scalability despite strong community resolution. Chinese Whispers consistently records lower modularity values across all datasets and scales. This suggests that while it is computationally efficient and fast, it tends to produce less cohesive communities. The algorithm's local decision-making approach may overlook broader structural relationships, resulting in weaker community boundaries.

From Table 2, it is evident that K-Cliques and LPA outperform Chinese Whispers in terms of modularity across all datasets. K-Cliques is more suitable for dense and structured networks, while LPA offers a better trade-off between scalability and modularity in large-scale, real-world networks. The DBLP and TG-NW datasets show the strongest community structures overall, as indicated by their higher modularity values. Meanwhile, the decline in modularity with increasing network size (notably in RH-NW and TG-NW) highlights the challenge of maintaining strong community definitions in expansive, heterogeneous networks. Thus, the experimental results confirm that algorithm selection should be based on both network topology and desired balance between modularity and efficiency.

B. Normalized Mutual Information (NMI)

NMI is used to quantify the similarity between the detected community structure and the ground truth partition. It is based on information theory and measures how much information is shared between two partitions.

$$NMI(X,Y) = \frac{2.I(X;Y)}{H(X) + H(Y)}$$

Where, X and Y are the detected communities and ground truth communities, respectively.I(X;Y) is the Mutual Information (MI):

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) log \frac{P(x,y)}{P(x)P(y)}$$

Where, P(x), P(y) and P(x,y) represent the probability distributions of clusters.

Table 3 NMI Results on Real Datasets for N = 7500

Dataset	LPA	K-Cliques	Chinese Whispers
RH-NW	0.51	0.55	0.41
ACP-NW	0.61	0.67	0.48
DBLP-NW	0.57	0.61	0.43
TG-NW	0.71	0.74	0.55

The Normalized Mutual Information results presented in Table 3 for N = 7500 reveal how accurately each algorithms Label Propagation, K-Cliques and Chinese Whispers that detects communities compared to the ground truth across four real-world datasets. Overall, K-Cliques achieves the highest NMI scores on all datasets, indicating its superior ability to uncover well-defined and meaningful community structures. Specifically, the algorithm performs best on the Twitch Gamers Network with an NMI of 0.74, reflecting its effectiveness in identifying dense, closely connected groups typical of social interaction networks. The Amazon Co-purchasing Network also exhibits strong performance (0.67), as product co-purchasing relationships form highly cohesive subgraphs that align well with clique-based detection.

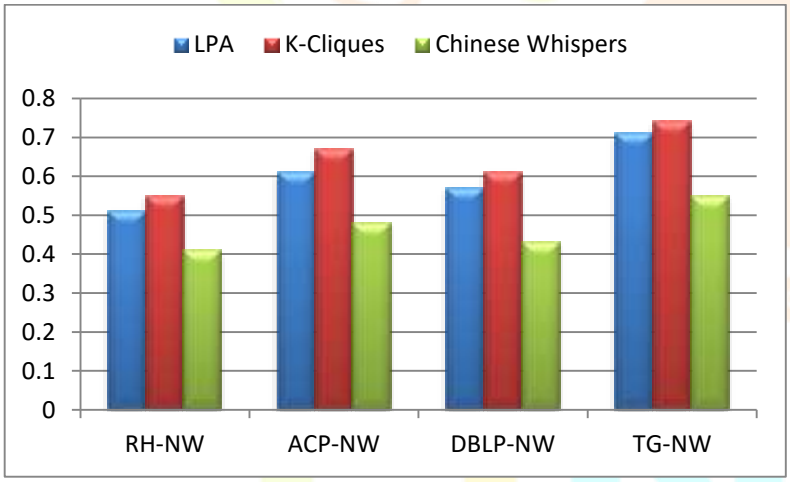

Figure 2 NMI Results on Real Datasets for N = 7500

Dataset a) RH-NW b) ACP-NW c) DBLP-NW d) TG-NW

LPA shows competitive results with moderately high NMI values, especially in TG-NW (0.71) and ACP-NW (0.61), demonstrating its efficiency and adaptability in identifying large-scale community structures despite its stochastic behavior. In contrast, Chinese Whispers consistently records the lowest NMI values, ranging from 0.41 to 0.55, suggesting limited accuracy in matching ground truth communities due to its locally constrained propagation mechanism. Overall, these findings indicate that while K-Cliques provides the most accurate community identification, LPA offers a strong balance between accuracy and computational efficiency, and Chinese Whispers, though faster, sacrifices precision for speed..

C. Execution Time (T)

Execution time is a critical metric for evaluating the computational efficiency of community detection algorithms. It measures how long an algorithm takes to process a given network and identify communities.

$$T=t_{end}-t_{start}$$

Where, t_{start} is the time before the algorithm starts.t_{end} is the time after the algorithm completes execution.Table 4 presents a comparison of evaluation metrics for community detection algorithms, including Modularity (Q), NMI with ground truth and Execution Time (T). These evaluation metrics provide a comprehensive assessment of community detection algorithms. Modularity is crucial for evaluating the internal consistency of communities, while NMI helps compare results with ground truth labels. Execution time determines the scalability of the algorithm for real-world applications.

Table 4 Execution Times of All Algorithms for N = 7500 (in milliseconds)

Dataset	LPA	K-Cliques	Chinese Whispers
RH-NW	1210	2398	1156
ACP-NW	1375	2251	1294
DBLP-NW	1225	2143	1203
TG-NW	956	1984	997

The execution time results shown in Table 4 for N = 7500 illustrate the computational efficiency of the three community detection algorithms are Label Propagation (LPA), K-Clique and Chinese Whispers and across four real-world datasets. Among these, Chinese Whispers consistently demonstrates the fastest execution times, completing all datasets in under 1.3 seconds, highlighting its lightweight and efficient label propagation mechanism that relies on local updates. LPA also performs efficiently, with execution times ranging between 956 ms and 1375 ms, showing slightly higher processing costs than Chinese Whispers due to its iterative label diffusion process. Despite this, LPA maintains linear scalability, making it suitable for large-scale and dynamic networks.

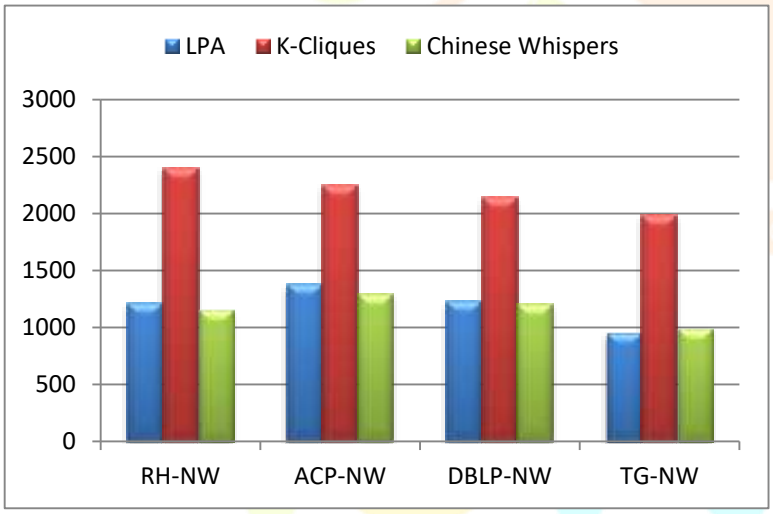

Figure 3 Execution Time Results on Real Datasets for N = 7500

Dataset a) RH-NW b) ACP-NW c) DBLP-NW d) TG-NW

In compare, the K-Cliques algorithm exhibits the highest execution times, ranging from 1984 ms to 2398 ms, as it involves computationally intensive clique enumeration and recursive searches for maximal subgraphs. This complexity increases with network density, explaining the slower performance observed in all datasets. Notably, the TG-NW dataset records the shortest times overall across all algorithms, possibly due to its sparser connectivity and fewer high-degree nodes. Overall, the results confirm that Chinese Whispers and LPA are the most time-efficient methods, ideal for rapid community detection in large networks, whereas K-Cliques, though slower, provides deeper structural insights into dense and highly connected subgraphs.

V. CONCLUSION

In conclusion, this work provides a comprehensive comparative evaluation of three prominent community detection algorithms are Label Propagation, Clique and Walktrap. That works across diverse real-world network datasets. The analysis demonstrates that each algorithm exhibits distinct strengths and limitations depending on the structural characteristics and scale of the network. The LPA algorithm proves to be the most efficient in terms of execution time and scalability, making it ideal for large and dynamic networks, though its non-deterministic nature can affect result stability. The Clique algorithm, while computationally expensive, excels in identifying dense substructures and is particularly effective in biological and social networks where strong internal connectivity is present. The Walktrap algorithm offers the most balanced performance in terms of modularity and accuracy, effectively capturing hierarchical and multi-level community structures, though it is less suitable for very large networks due to its higher computational cost. Overall, the comparative findings highlight that no single algorithm universally outperforms others; instead, the choice of algorithm should depend on the network's size, density, and the required balance between computational efficiency and structural accuracy. Future work may explore hybrid models or ensemble-based approaches that combine the scalability of LPA with the precision of Walktrap and the density-detection capabilities of Clique to achieve more robust and adaptive community detection solutions for complex real-world networks.

VI. REFERENCES

[1]. Ahn, Y. Y., Bagrow, J. P., and Lehmann, S., *Link communities reveal multiscale complexity in networks*. Nature, 466(7307), 761–764. https://doi.org/10.1038/nature09182. 2010

[2]. Blondel, V. D., Guillaume, J. L., Lambiotte, R., and Lefebvre, E. *Fast unfolding of communities in large networks*. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

[3]. Bron, C., and Kerbosch, J., *Algorithm 457: Finding all cliques of an undirected graph*. Communications of the ACM, 16(9), 575–577. https://doi.org/10.1145/362342.362367.

[4]. Chakraborty, T., Dalmia, A., Mukherjee, A., and Ganguly, N., *Metrics for community analysis: A survey*. ACM Computing Surveys, 50(4), 1–37. https://doi.org/10.1145/3091106, 2017

[5]. Chen, M., Kuzmin, K., and Szymanski, B. K., *Community detection via maximization of modularity and its variants*. IEEE Transactions on Computational Social Systems, 1(1), 46–65. https://doi.org/10.1109/TCSS.2014.2307458, 2014

[6]. Fortunato, S. *Community detection in graphs*. Physics Reports, 486(3–5), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002, 2010

[7]. Girvan, M., and Newman, M. E. J. *Community structure in social and biological networks*. Proceedings of the National Academy of Sciences, 99(12), 7821–7826. https://doi.org/10.1073/pnas.122653799, 2002

[8]. Lancichinetti, A., and Fortunato, S. *Community detection algorithms: A comparative analysis*. Physical Review E, 80(5), 056117. https://doi.org/10.1103/PhysRevE.80.056117, 2009

[9]. Leung, I. X. Y., Hui, P., Lio, P., and Crowcroft, J., *Towards real-time community detection in large networks*. Physical Review E, 79(6), 066107. https://doi.org/10.1103/PhysRevE.79.066107. 2009.

[10]. Liu, X., Guan, J., and Zhou, S., *Semi-supervised community detection based on label propagation with pairwise constraints*. Physica A: Statistical Mechanics and Its Applications, 490, 1137–1150. https://doi.org/10.1016/j.physa.2017.08.040. 2018

[11]. Malliaros, F. D., and Vazirgiannis, M., *Clustering and community detection in directed networks: A survey*. Physics Reports, 533(4), 95–142. https://doi.org/10.1016/j.physrep.2013.08.002. 2013

[12]. Newman, M. E. J. (2018). *Networks (2nd ed.)*. Oxford University Press. https://doi.org/10.1093/oso/9780198805090.001.0001

[13]. Newman, M. E. J. (2004). *Fast algorithm for detecting community structure in networks*. Physical Review E, 69(6), 066133. https://doi.org/10.1103/PhysRevE.69.066133

[14]. Palla, G., Derényi, I., Farkas, I., and Vicsek, T. (2005). *Uncovering the overlapping community structure of complex networks in nature and society*. Nature, 435(7043), 814–818. https://doi.org/10.1038/nature03607

[15]. Papadopoulos, S., Kompatsiaris, Y., Vakali, A., and Spyridonos, P. (2012). *Community detection in social media: Performance and application considerations*. Data Mining and Knowledge Discovery, 24(3), 515–554. https://doi.org/10.1007/s10618-011-0224-z

[16]. Peixoto, T. P., *Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models*. Physical Review E, 89(1), 012804. https://doi.org/10.1103/PhysRevE.89.012804. 2014

[17]. Pons, P., and Latapy, M., *Computing communities in large networks using random walks*. In *International Symposium on Computer and Information Sciences* (pp. 284–293). Springer. https://doi.org/10.1007/11569596_31. 2005

[18]. Porter, M. A., Onnela, J. P., and Mucha, P. J., *Communities in networks*. Notices of the AMS, 56(9), 1082–1097. 2009

[19]. Raghavan, U. N., Albert, R., and Kumara, S., *Near linear time algorithm to detect community structures in large-scale networks*. Physical Review E, 76(3), 036106. https://doi.org/10.1103/PhysRevE.76.036106. 2007

[20]. Rosvall, M., and Bergstrom, C. T., *Maps of random walks on complex networks reveal community structure*. Proceedings of the National Academy of Sciences, 105(4), 1118–1123. https://doi.org/10.1073/pnas.0706851105

[21]. Traag, V. A., Waltman, L., and van Eck, N. J. *From Louvain to Leiden: Guaranteeing well-connected communities*. Scientific Reports, 9(1), 5233. https://doi.org/10.1038/s41598-019-41695-z.2019

[22]. Tomita, E., Tanaka, A., and Takahashi, H., *The worst-case time complexity for generating all maximal cliques and computational experiments*. Theoretical Computer Science, 363(1), 28–42. https://doi.org/10.1016/j.tcs.2006.06.015. 2006

[23]. Xie, J., Kelley, S., and Szymanski, B. K., *Overlapping community detection in networks: The state-of-the-art and comparative study*. ACM Computing Surveys, 45(4), 1–35. https://doi.org/10.1145/2501654.2501657, 2013.

[24]. Yang, J., and Leskovec, J., *Defining and evaluating network communities based on ground-truth*. Knowledge and Information Systems, 42(1), 181–213. https://doi.org/10.1007/s10115-014-0780-0. 2015