Network Science Project

An In-Depth Exploration of Label Propagation Algorithm in Complex Networks

Prakhar Gupta | Shrutya Chawla | Vimansh Mahajan
2021550 2022487 2022572

Research Paper Referred

Near linear time algorithm to detect community structures in large-scale networks

Paper: LINK

Community detection and analysis is an important methodology for understanding the organization of various real-world networks and has applications in problems as diverse as consensus formation in social communities or the identification of functional modules in biochemical networks. Currently used algorithms that identify the community structures in large-scale real-world networks require a priori information such as the number and sizes of communities or are computationally expensive. In this paper we investigate a simple label propagation algorithm that uses the network structure alone as its guide and requires neither optimization of a pre-defined objective function nor prior information about the communities. In our algorithm every node is initialized with a unique label and at every step each node adopts the label that most of its neighbors currently have. In this iterative process, densely connected groups of nodes form a consensus on a unique label to form communities. We validate the algorithm by applying it to networks whose community structures are known. We also demonstrate that the algorithm takes an almost linear time and hence it is computationally less expensive than what was possible so far.

Deliverable 1: *Label Propagation Algorithm Scratch Implementation*

Objective

The goal of this deliverable was to implement the Label Propagation Algorithm (LPA) from scratch to detect communities in a network, and to evaluate and visualize the results using various metrics and plots.

1. Reading and Building the Graph

We start by reading an edge list from a file and constructing an undirected graph using an adjacency list:

- Each line in the file represents an edge between two nodes.
- The graph is stored as a dictionary where each key is a node, and the value is the list of its neighbors.

2. Label Propagation Algorithm

We implemented LPA from scratch. Here's how it works:

- Initially, each node is assigned a unique label (its own ID).
- At each iteration:
    - Nodes are visited in random order.
    - Each node adopts the most frequent label among its neighbors.
    - If no label changes in an iteration, the algorithm stops early.
- After convergence, nodes with the same final label are grouped into communities.

3. Evaluation Metrics

To assess the quality and consistency of the communities, we used the following metrics:

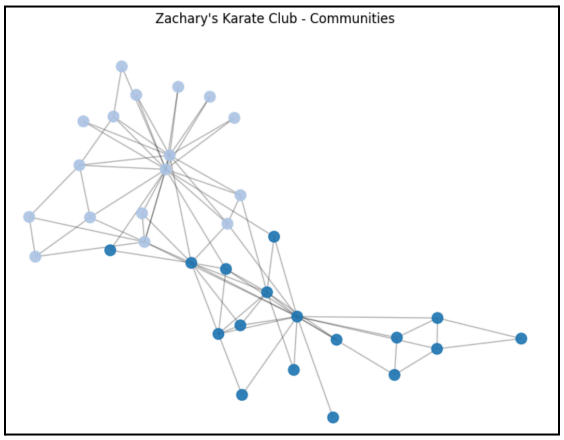
- **Modularity**: Measures how well the network is divided into communities.
- **fsame**: Measures the overlap quality between two sets of communities.
- **Jaccard Index**: Measures similarity between two community partitions using pairwise node combinations.

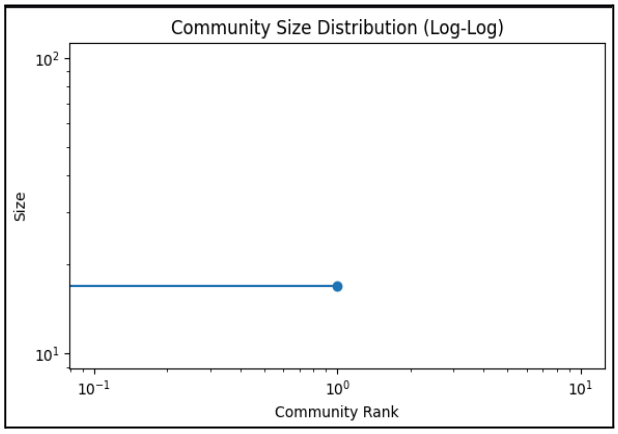
4. Visualization

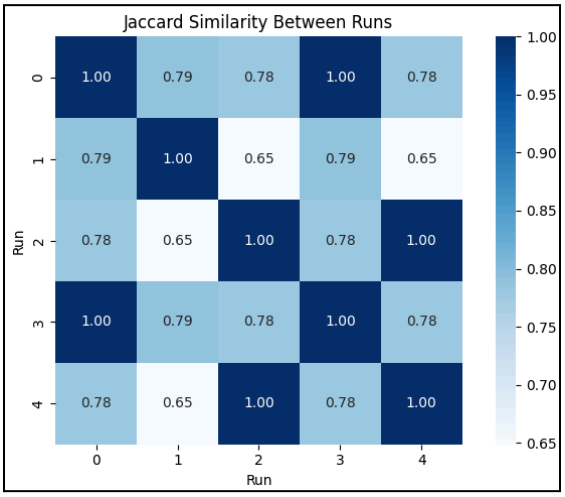
We created several visual tools to interpret results:

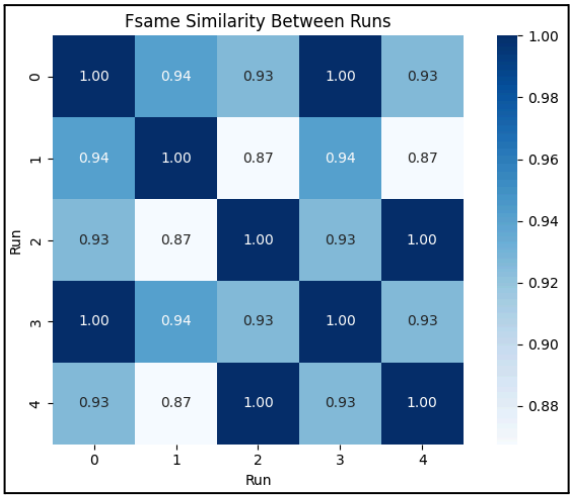
- **Community Graph**: Visualizes the network with each community in a different color.
- **Community Size Distribution**: Bar chart or log-log plot showing the size of each community.
- **Similarity Heatmap**: Compares multiple community detection results using either fsame or Jaccard similarity.

Zachary's Karate Club







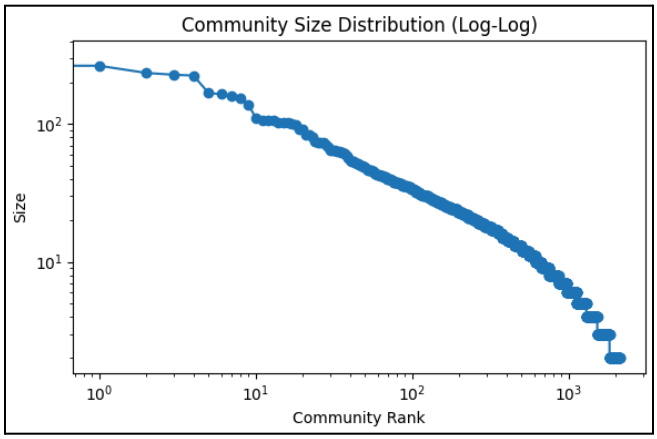


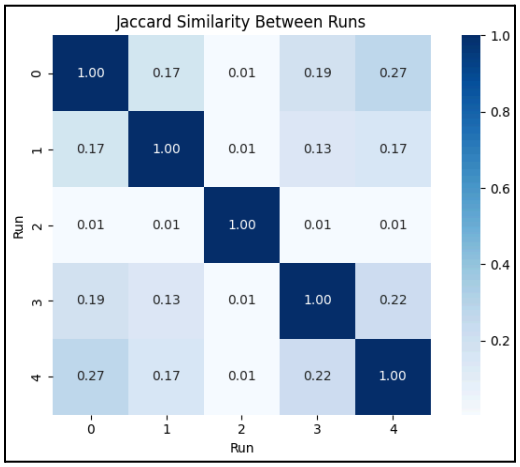
Q-VALUE

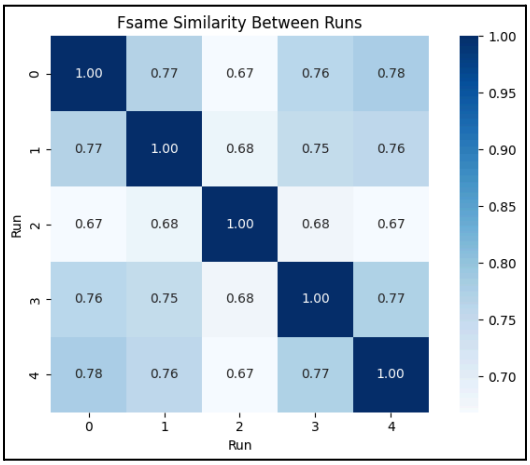
Our Value: 0.3718

Paper Range: 0.355 - 0.399

Co-Authorship Network





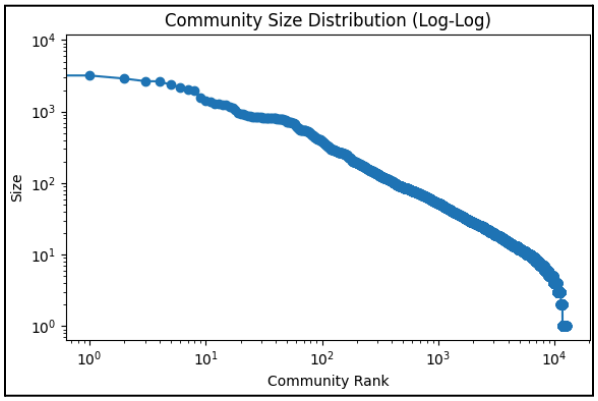


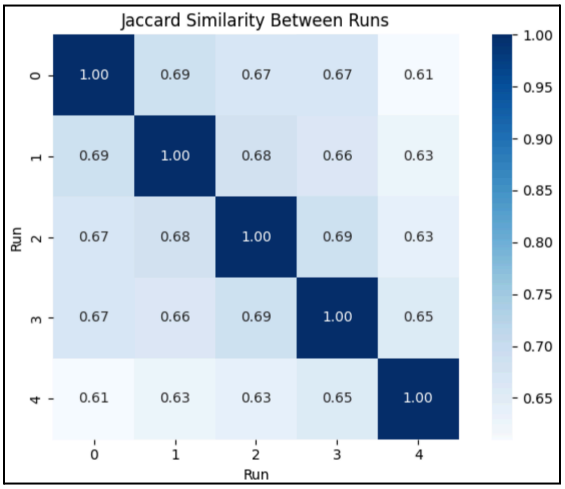
Q-VALUE

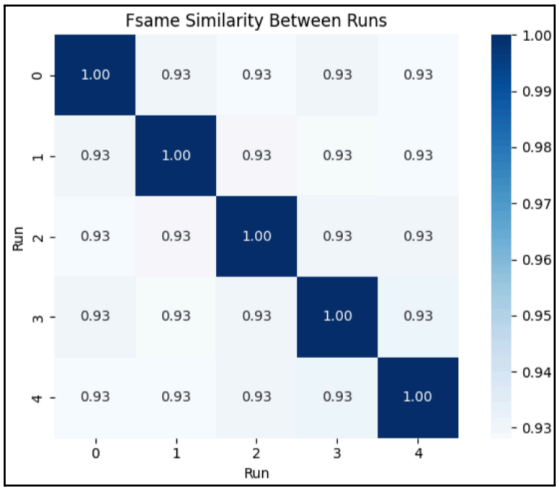
Our Value : 0.6352

Paper Range: 0.720 - 0.722

WWW Network







Q-VALUE

Our Value: 0.8435

Paper Range: 0.857 - 0.864

Conclusion for Zachary Karate Club Graph

The graph has two large community. The Jaccard similarity scores (middle chart) show that the results of the community detection algorithm change a bit between runs. However, the Fsame scores are very high,

meaning that even though the exact communities may change, most nodes still stay in the same groups with each other. So, the grouping is mostly stable, even if the details change slightly.

Conclusion for Coauthorship Network

This graph has many communities of different sizes, as seen in the top chart. It shows a more complex and varied community structure. However, the Jaccard similarity (middle chart) between different runs is very low, meaning the community detection algorithm gives very different results each time. The Fsame scores (bottom chart) are a bit better but still not very high, showing that even the grouping of nodes changes quite a bit between runs. Overall, the algorithm is unstable on this graph and doesn't produce consistent community results.

Conclusion for WWW

This graph has many communities of different sizes, ranging from large to very small ones, as seen in the top chart. The Jaccard similarity (middle chart) between runs is moderate (around 0.6–0.7), which means the exact community assignments change somewhat across runs. However, the Fsame scores (bottom chart) are very high (around 0.93), showing that most node pairs tend to stay grouped together even if the communities shift slightly. Overall, the algorithm is fairly stable in terms of grouping nodes, even if the exact community structure changes.

Deliverable 2: *LPA v/s Community Detection using Louvian Algorithm*

1. Objective

To compare the community structures identified by the **Label Propagation Algorithm (LPA)** and **Louvain method** on graphs like *Karate Club* and *web-NotreDame*, using metrics such as **modularity**, **Jaccard similarity**, and **pairwise node agreement (fsame)**.

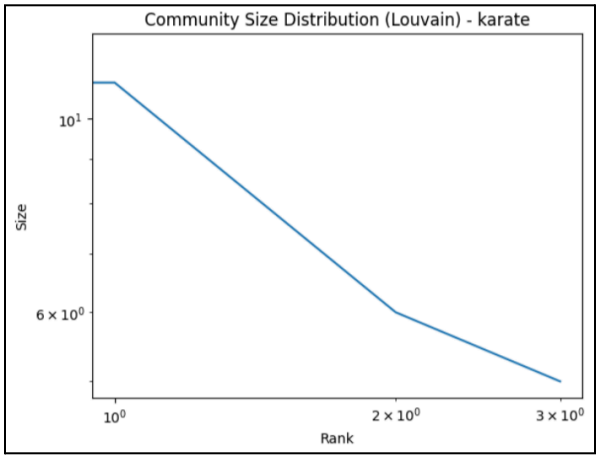
2. Methodology

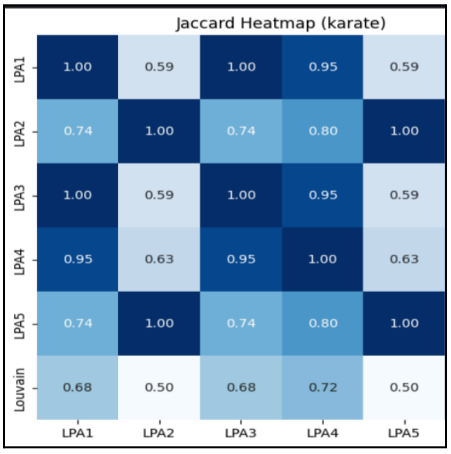
- **Algorithms Applied:**
    - **Label Propagation Algorithm (LPA)**: Run 5 times due to its randomness.

- **Louvain**: Single deterministic run using modularity maximization.

- **Metrics Used:**
  - **Modularity (Q)**: Measures community quality.
  - **Jaccard Similarity**: Measures best overlap between detected communities.
  - **fsame**: Fraction of node pairs assigned to the same community in both methods.
- **Visualization:**
  - **Community Size Distribution** (log-log plots).
  - **Spring Layout Graphs** with colored communities.
  - **Heatmaps** for pairwise similarity between runs (Jaccard and fsame).

Zachary Karate Club Network





Louvian Q-Value : 0.4188

1. Community Size Distribution (Louvain)

The community size distribution reveals that the Louvain algorithm identifies **three communities** with a **highly skewed size profile**. One community is significantly larger, while the remaining two are considerably smaller. This reflects Louvain's tendency to optimize modularity, often leading to a few dominant clusters that maximize intra-community density and minimize inter-community connections.

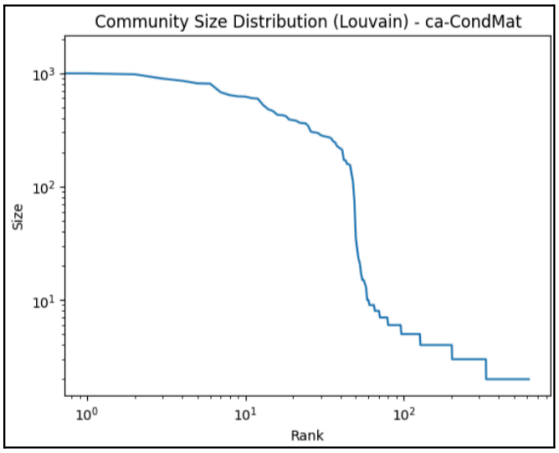
2. Jaccard Similarity Heatmap

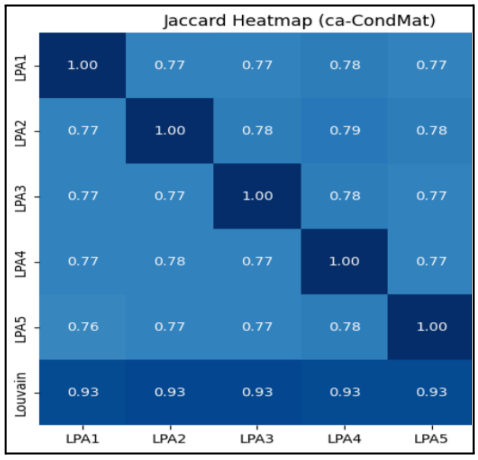
The Jaccard similarity heatmap compares the overlap between community detection results from five runs of the Label Propagation Algorithm (LPA) and a single Louvain partition:

- **Label Propagation (LPA):**
  The upper 5×5 block shows **high similarity values** (mostly between 0.80 and 1.00), indicating that LPA produces **consistent and repeatable community assignments** across different runs. This suggests stability in how LPA interprets the community structure of this network.

- **Louvain vs. LPA:**
  The bottom row and last column compare Louvain results with each LPA run. The Jaccard similarities range from **0.50 to 0.72**, showing **moderate agreement**. This suggests that while Louvain and LPA identify some common structure, they differ in key areas. Louvain likely groups nodes based on global modularity considerations, whereas LPA focuses more on local label propagation dynamics.

Co-Authorship Network





Louvian Q-Value: 0.7325

1. Community Size Distribution (Louvain)

The Louvain algorithm identifies a **broad range of community sizes** with a **long-tailed distribution**:

- A small number of communities are **very large** (with sizes exceeding 1000).

- The majority are **small to moderately sized**, forming a smooth decline over ranks.

- This pattern is typical for **real-world social networks**, where community structure often follows a heavy-tailed distribution due to hubs and clustering.

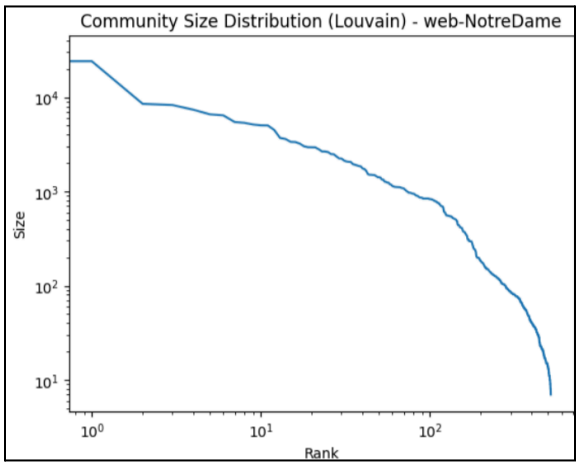
2. Jaccard Similarity Heatmap

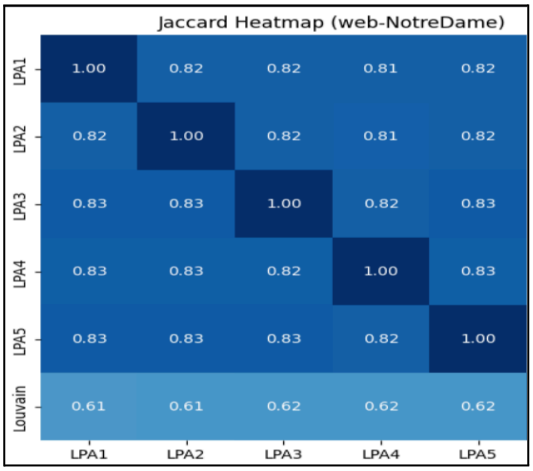
This heatmap shows the Jaccard similarity between different community detection outputs:

- **Label Propagation (LPA):**
  The 5×5 top-left block shows **consistent similarity scores** (~0.77 to 0.79), indicating **moderate internal stability** of LPA across runs. This reflects some variability in LPA outputs, possibly due to the size and complexity of the network.

- **Louvain vs. LPA:**
  The last row and column show **high similarity** values (0.93), indicating that Louvain produces a **similar community structure** compared to LPA.

WWW Network





Louvian Q-Value: 0.9372

1. Community Size Distribution (Louvain)

The Louvain community size distribution follows a **long-tailed pattern**, with:

- A few **very large communities** (sizes exceeding 10,000 nodes).

- A **gradual decline** across ranks, with a large number of small communities.

- This is typical for web graphs or hyperlink networks, where a few highly connected clusters (e.g., popular domains or hubs) dominate.

2. Jaccard Similarity Heatmap

The Jaccard similarity matrix highlights a sharp contrast between LPA runs and Louvain:

- **Label Propagation (LPA):**
  The top 5×5 section shows **very high internal consistency** (values ~0.81–0.83), indicating that LPA is **extremely stable** across runs for this network.

- **Louvain vs. LPA:**
  The bottom row and last column show **considerable similarity values (~0.61-0.62)** between Louvain and all LPA runs. This indicates **overlap** in community assignments, suggesting the two methods interpret the structure of this web graph **in a similar manner**.

Deliverable 3: *Robustness of graphs to noise*

1. **Detect Baseline Communities with Label Propagation**.

2. **Introduce Noise via Edge Removal**
   Two types of perturbations are applied at various noise levels (fractions of edges to remove):

   - **Random Removal ("remove")**: simply delete a random set of all edges.

   - **Targeted Removal ("targeted_remove")**: remove edges that run between the baseline communities, i.e. the edges most important to holding communities apart.

3. **Re‑detect Communities and Measure Changes**
   For each noise level and each perturbation mode, the algorithm:

   - Re‑runs LPA on the perturbed graph.

   - Computes the new modularity.

   - Compares the new community split back to the original using:

     1. **Jaccard Index**—the fraction of node‑pairs that stay together in both partitions;

     2. **F_same**—a symmetric overlap score based on matching communities.

   - Repeats this several times to average out randomness.

4. **Summarize and Plot Results**

    - It aggregates mean and standard-deviation of modularity and stability scores across repeats.

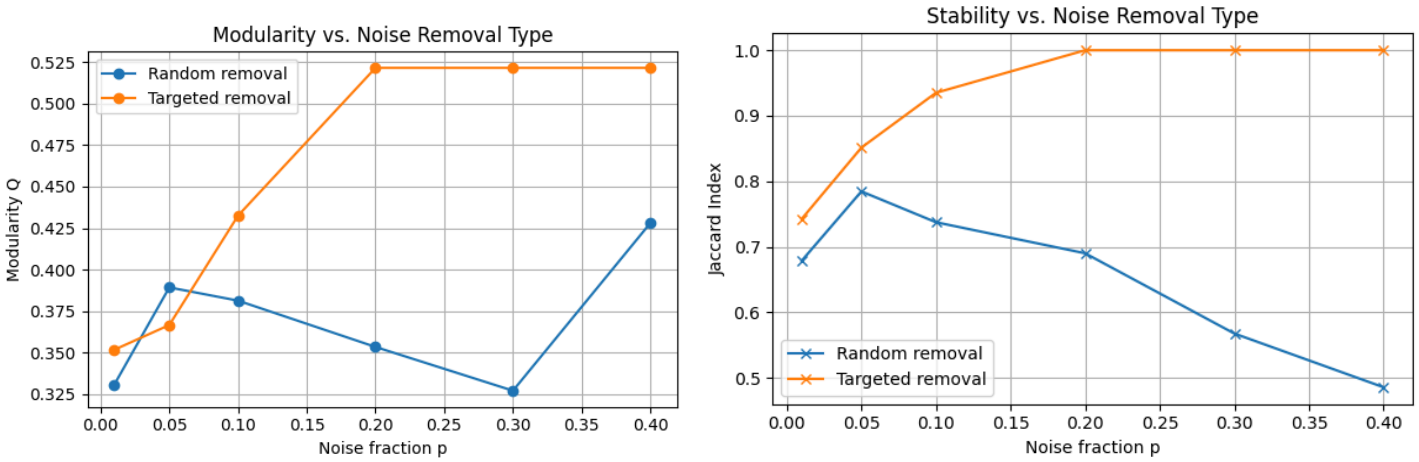
    - Finally, it produces two plots:

        1. **Modularity vs. Noise**: shows how community-quality degrades as more edges are removed, comparing random vs. targeted removal.

        2. **Stability vs. Noise**: shows how similar the detected communities remain to the baseline under each noise regime.

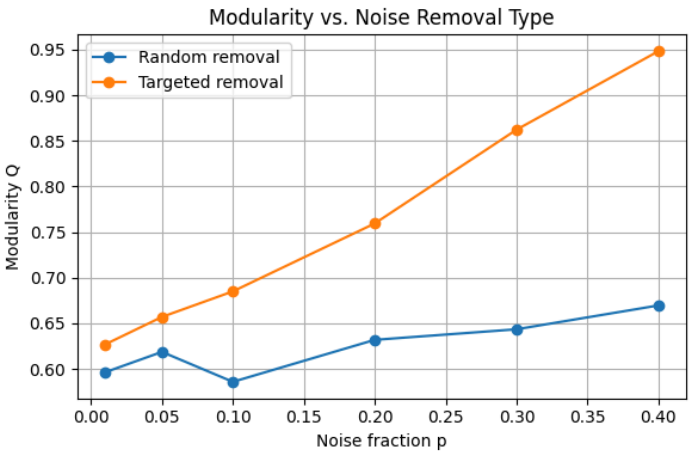
Results:

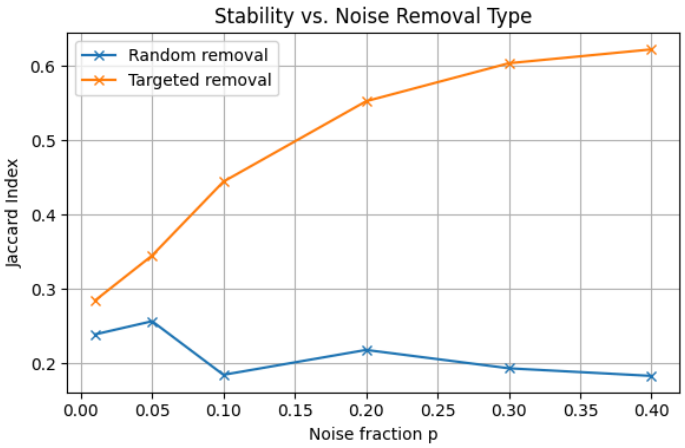
- Zachary



Targeted removals deliver **perfect stability** (Jaccard → 1.0 by 20 % noise) and lift modularity (0.35→0.52). Random removals give only a brief uptick at 5 % then **degrade** split quality and repeatability..

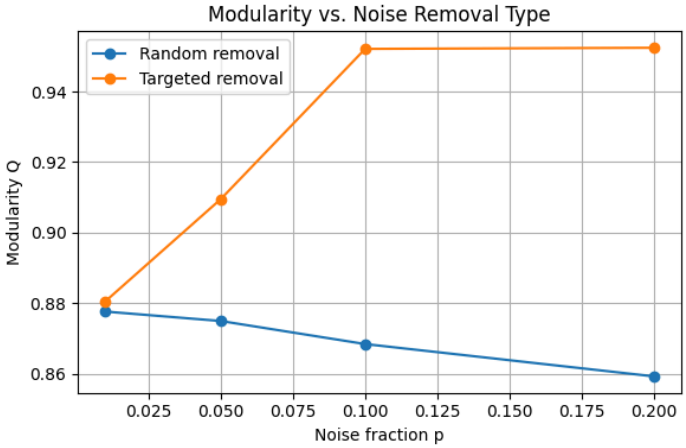
- Co- authorship

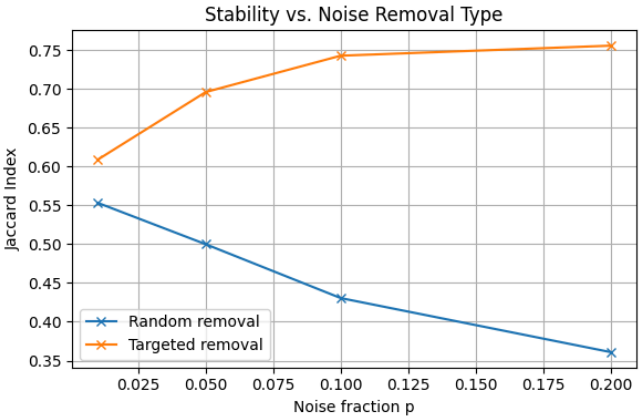




Targeted removal of inter-community edges sharply **raises modularity** (~0.63→0.95) and **stabilizes** clusters (Jaccard ~0.28→0.62). Random edge loss barely improves quality (~0.60→0.67) and yields **highly unstable** partitions (Jaccard falls to ~0.18).

- WWW





-

Removing boundary edges quickly boosts modularity (0.88→0.95 by 10 % noise) and increases reproducibility (Jaccard ~0.61→0.75). Random deletions erode both quality (0.88→0.86) and stability (0.55→0.36).

Deliverable 4: *GAE + K-Means*

1. Graph Auto-Encoder: learning a low-dimensional representation

- **Goal:** turn each node in your graph into a vector in $\mathbb{R}^d$ in such a way that nodes with similar connectivity patterns end up close together.

- **How (conceptually):**

    1. A two-layer graph convolutional network (GCN) encodes each node's one-hot feature into a hidden space.

    2. A decoder tries to reconstruct the original adjacency (i.e., predict which pairs of nodes were connected) from those embeddings.

    3. The encoder is trained so that reconstruction error is minimized.

- **Result:** an embedding matrix Z where row i is the d-dimensional "signature" of node i, capturing both local and global graph structure.

---

2. K-Means: a first pass at community discovery

- **Goal:** group nodes (now represented by vectors) into k clusters, where nodes in the same cluster have similar embeddings.

- **How:**

    1. Automatically choose the "best" k by evaluating silhouette scores for k = 2, 3,… (here 2 and 3).

    2. Run K-Means several times (with different random seeds) to gauge stability.

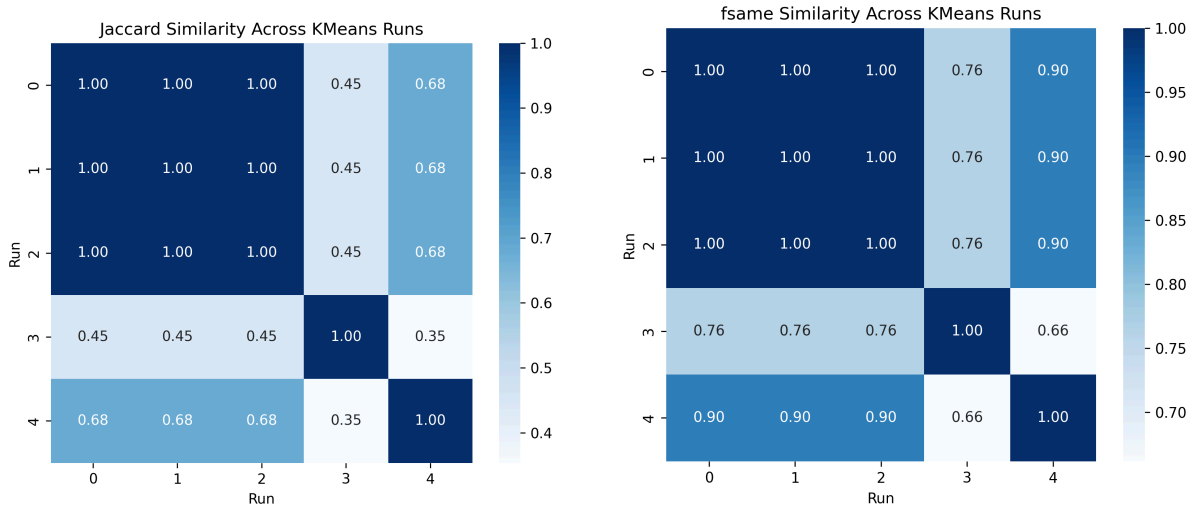
    3. Compare each pair of runs using metrics like Jaccard index or "fsame" (which measures overlap in community assignments).

- **Why:** the embedding gives you a geometry in which you can apply a classic clustering algorithm—this often finds more meaningful "blobs" than clustering on raw adjacency or hand-crafted features.

Results:

Zachary:

Modularity: 0.3991
Number of communities: 3



Runs 0–2: Perfect agreement (Jaccard = 1.00, fsame = 1.00) – they all found the exact same two-community split.

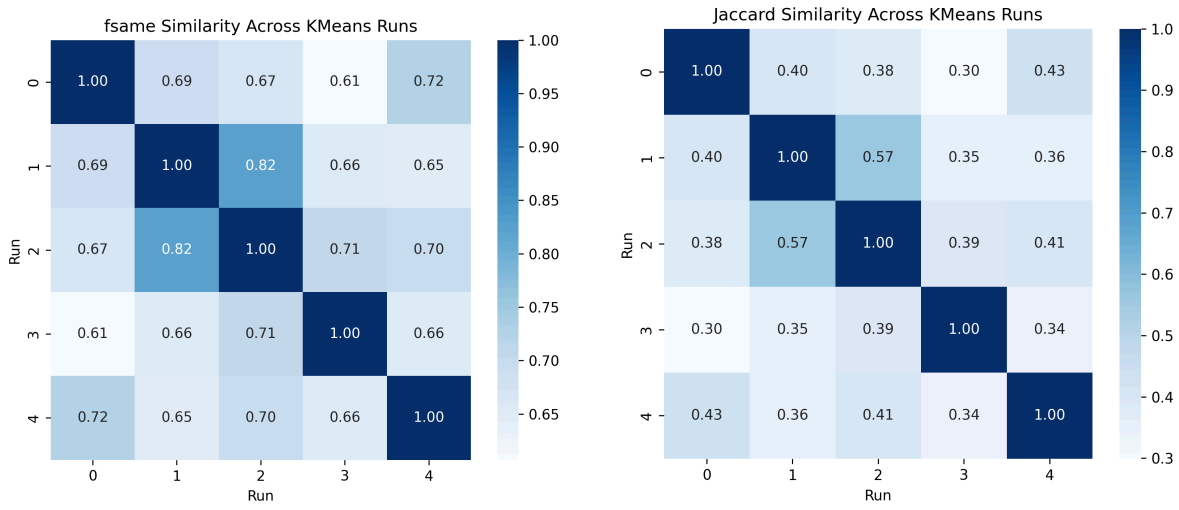
Run 4: Minor variation (Jaccard ≈ 0.68, fsame ≈ 0.90) – most nodes stay in their groups, but a few borderline nodes swapped.

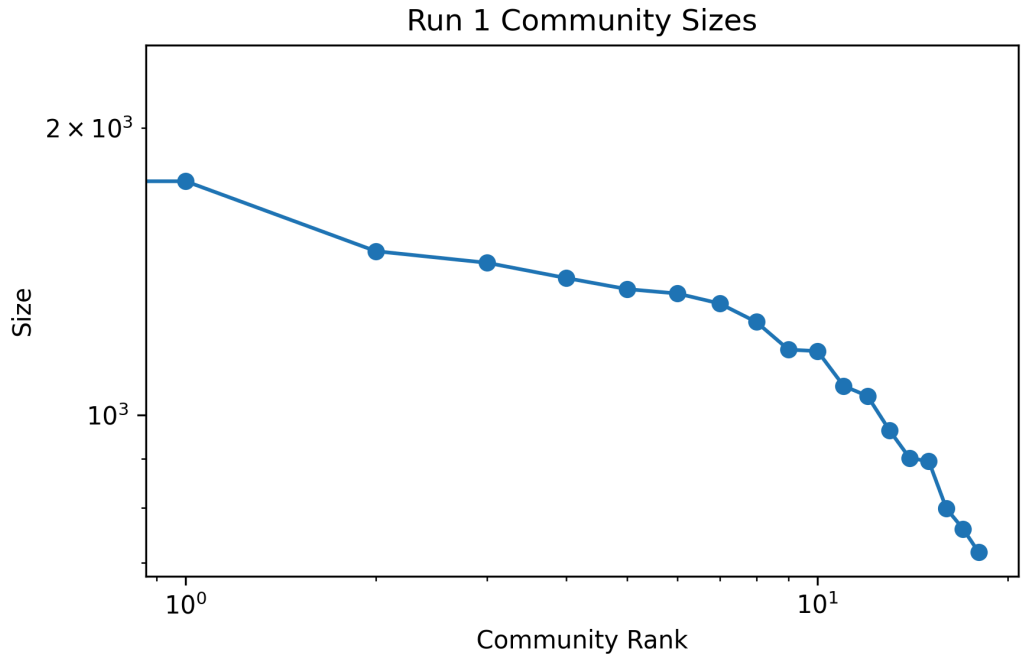
Run 3: Substantially different (Jaccard ≈ 0.45, fsame ≈ 0.76) – still two clusters of similar size, but many nodes moved across the divide.

Between Modes (runs 3 vs 4): Low overlap (Jaccard = 0.35, fsame = 0.66), showing the two alternate partitions disagree on a majority of node-pairs.

Co- authorship:

Modularity: 0.6015
Communities : 19





Low Jaccard Agreement (≈0.30–0.57):
None of the five KMeans runs matched each other very closely: only 30–57 % of node-pairs that sit together in one run do so in another. That points to **high sensitivity** to initialization—multiple local minima in the embedding space.

Moderate fsame Scores (≈0.61–0.82):
While exact pair-membership fluctuates, fsame's higher values (~0.6–0.8) tell us the **overall community sizes** stay roughly consistent across runs. In other words, we repeatedly see a few big clusters and many small ones, but which exact authors land where shifts.

Run 1 Community-Size Curve:
The log–log plot shows a **heavy-tailed distribution**:

- **A handful of large communities** (the leftmost points), likely corresponding to major subfields or prolific author groups.
- **A long tail of small communities** and singletons, reflecting niche collaborations or isolated pairs.

Deliverable 5: *Temporal Community Tracking in Networks*

Temporal Louvain + Community Evolution Tracking
Goal: Identify and track how communities of nodes evolve over time in two datasets- a dynamic email communication network (Email-EU-Core-Temporal) and a college's private social network (College_Msg).

How is it done:

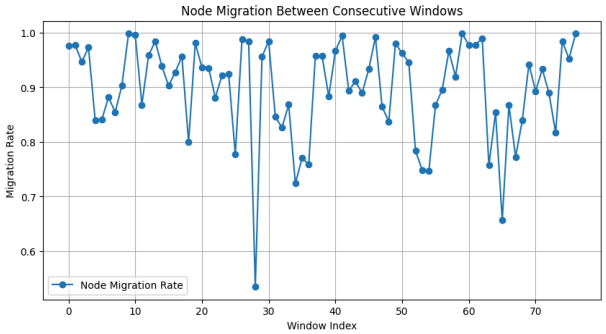
- The dataset is split into **fixed time windows** (7-day intervals), treating each window as a static graph snapshot.
- In each window, **Louvain community detection** is applied to detect communities based on modularity optimisation.
- A **TemporalClustering** object tracks communities across windows, allowing measurement of structural evolution.

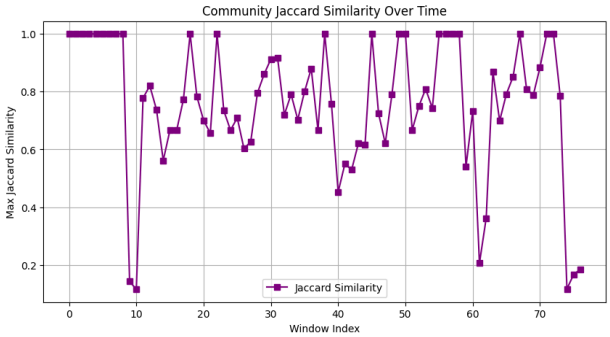
- Between each pair of consecutive time windows:
    - **Node Migration Rate** quantifies how many nodes switched communities.
    - **Jaccard Similarity** captures structural overlap between clusters.
    - The **number of communities** is tracked to observe merging/splitting trends.

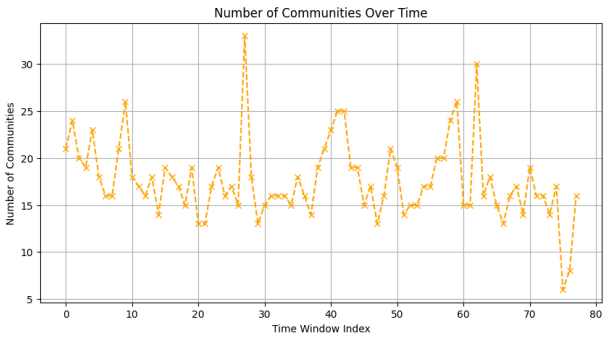
Result:
A time-indexed sequence of community assignments capturing dynamic interaction patterns, visualised through:

- **Node migration plots**
- **Jaccard similarity curves**
- **Community count trends**

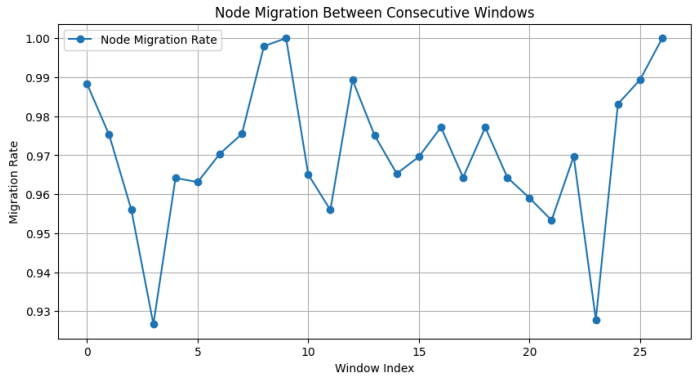
1. **Email-EU-Core-Temporal Dataset**

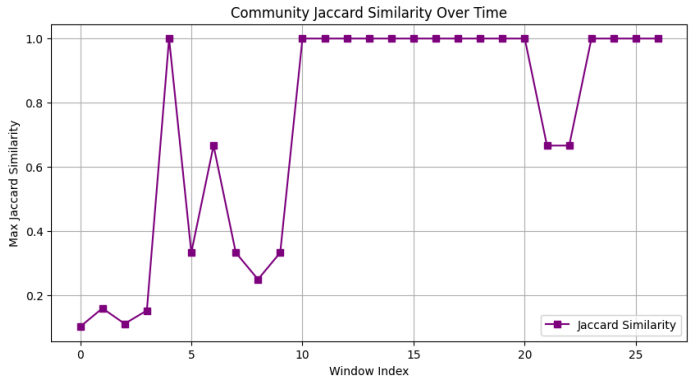


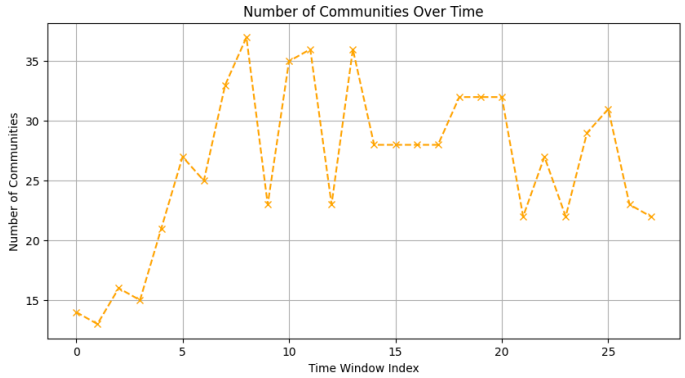




2. College-Msg Dataset







These highlight structural shifts and stability in the network over time.

Graph/Networks used

1) Zachary's Karate Club: Nodes = 34 Links = 78
2) Co-Authorship Network: Nodes = 23,123 Links = 93, 497
3) WWW Network: Nodes = 3,25,729 Links = 14,97,134
4) CollegeMsg: Nodes=1,899 Links = 59,835
5) Email-EU-Core Temporal: Nodes=986 Links=3,32,334