IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.145113

A GraphSAGE-Enhanced Label Diffusion Approach for Scalable Community Detection in Large Networks

G. Reguvel[1], G. Naveen[2], Ch. Krishna[3], Dr. M. Sreelatha[4]

Student, CSE, R. V. R. & J. C. College of Engineering, GUNTUR, AP, INDIA[1,2,3]

Professor, CSE, R. V. R. & J. C. College of Engineering, GUNTUR, AP, INDIA [4]

Abstract: Community detection in complex networks remains a fundamental problem in network science, with wide-ranging applications from sociology to biology and recommendation systems. Building on recent advances in label diffusion techniques, we propose a novel hybrid approach—GraphSAGE-LBLD—that combines the structural awareness of the GraphSAGE embedding model with the local balance and speed advantages of the Label Balanced Label Diffusion (LBLD) algorithm. Our method integrates representation learning into the label propagation process, allowing for more semantically meaningful diffusion and improved stability across diverse network topologies.

We empirically evaluate GraphSAGE-LBLD on multiple real-world SNAP datasets and benchmark against Louvain, classic Label Propagation, and the original LBLD. Results demonstrate that our model consistently achieves higher modularity and Normalized Mutual Information (NMI) scores, while maintaining comparable runtime. The integration of GraphSAGE enhances the representation of local neighbourhoods, resulting in finer community boundaries and better detection of small or overlapping clusters. Our method offers a practical, scalable, and more accurate alternative for modern community detection tasks.

Keywords*:* Community Detection, GraphSAGE, Label Propagation, Graph Neural Networks, Node Embeddings, K-core, Graph Autoencoder, Modularity, Weighted Diffusion, Cosine Similarity.

I. INTRODUCTION

The rapid proliferation of large-scale graph-structured data across domains such as social media, biological networks, e-commerce platforms, and citation systems has necessitated the development of scalable and efficient community detection algorithms. As networks continue to grow in size and complexity, traditional global algorithms struggle to maintain both computational efficiency and structural accuracy. This challenge is especially pronounced in scenarios where real-time or near real-time analysis is required, such as in recommendation engines, fraud detection, or dynamic social trend monitoring.

In complex networks, nodes typically represent entities—such as users, proteins, or publications—while edges signify various forms of relationships or interactions. Communities, or densely connected node groups, serve as critical structures for understanding the latent organization and behaviour of the network. Identifying these communities provides insight into functional modules in biology, interest groups in social networks, and topic clusters in citation graphs. However, designing algorithms that can efficiently uncover these communities while preserving accuracy across diverse datasets remains an open and active area of research.

While global methods such as modularity maximization and spectral clustering have shown promising accuracy in moderate-sized networks, they are computationally prohibitive for networks with millions or billions of nodes and edges. Furthermore, their dependence on complete graph traversal renders them impractical for streaming or dynamically evolving graphs. On the other hand, local methods—particularly label propagation techniques—have gained traction due to their linear complexity and potential to scale to massive networks. Nevertheless, these methods often suffer from instability, randomness in community assignment, and limited adaptability to topological variations across network regions.

To address these limitations, recent research has explored the use of node embeddings and learned representations to inform community detection. Among these, GraphSAGE (Graph Sample and Aggregate) stands out as a powerful inductive framework that can learn low-dimensional feature vectors for nodes by aggregating information from their local

DOI: 10.17148/IJARCCE.2025.145113

neighbourhoods. By embedding structural and feature-based information into dense vectors, GraphSAGE facilitates the identification of both local and global patterns while supporting generalization to unseen nodes and subgraphs.

In this paper, we propose a novel approach—GraphSAGE-Enhanced Label Diffusion (SELD)—that integrates learned node representations with a refined label diffusion mechanism to achieve scalable, stable, and accurate community detection. Our approach leverages the strengths of both neural embeddings and fast label diffusion, combining the robustness of feature-based analysis with the computational advantages of local propagation. Unlike existing diffusion techniques that rely solely on topological cues, SELD utilizes embedding similarities to guide the label update process, resulting in more coherent and semantically meaningful communities.

Moreover, we introduce a neighbourhood-aware initialization scheme that ensures stable seeding of communities, mitigating the randomness observed in conventional label propagation. The label diffusion process is augmented with a similarity-weighted propagation strategy that incorporates both edge strength and embedding proximity. To ensure scalability, the entire pipeline is designed with linear time complexity in mind, utilizing batch-wise neighbourhood sampling, efficient memory structures, and asynchronous updates. Finally, a post-processing phase is applied to merge fragmented clusters and refine community boundaries, further enhancing structural quality.

The main contributions of this paper can be summarized as follows:

We present a novel community detection framework that fuses GraphSAGE embeddings with label diffusion to enhance accuracy and scalability in large networks. We propose a stable seeding strategy using embedding-space clustering to generate robust initial community assignments. We design a similarity-weighted diffusion process that leverages learned node representations to improve propagation coherence and reduce instability. We incorporate a scalable architecture that ensures the method remains practical for graphs with millions of nodes and edges. We validate the effectiveness of our method through extensive experiments on real-world and synthetic datasets, demonstrating superior performance over baseline algorithms in terms of both accuracy and runtime.

The remainder of this paper is organized as follows. In Section 2, we review related work in community detection and graph neural embeddings. Section 3 details the architecture and components of the proposed SELD method. Section 4 presents experimental results, comparisons, and ablation studies. Finally, Section 5 concludes the paper and outlines directions for future research.

II. LITERATURE SURVEY

Community detection is a fundamental task in network science, and the literature comprises a wide variety of approaches ranging from heuristic local expansion algorithms to embedding-based and spectral methods. This section surveys relevant methods in three main areas: label propagation and local expansion, modularity and spectral optimization, and graph neural network (GNN) based embedding methods.

Label propagation methods form one of the earliest families of fast community detection algorithms. The Label Propagation Algorithm (LPA) by Raghavan et al. [1] is notable for its simplicity and near-linear time complexity. However, LPA suffers from instability due to its randomness in label initialization and update order. To mitigate this, later works proposed importance-based variants, such as K-shell decomposition [2], Bayesian influence [3], and link influence models like LP-LPA [4] and NI-LPA [5], which introduce deterministic node ordering and local similarity heuristics to guide label updates.

Another branch of local algorithms emphasizes core node identification. These methods first identify highly influential or central nodes ("seeds") and then expand communities outward using structural cues. Ding et al. [6] proposed RTLCD, a robust two-stage method for local expansion from identified cores. ECES [7], CFCD [8], and CDME [9] similarly rely on structural centrality, core fitness, or the Matthew effect to seed initial communities. More recent variants like LCDR [10] apply charge conservation analogies to rank nodes. Alternative expansion strategies include FluidC [11], which simulates fluid dynamics, and CenLP [12], which incorporates centrality in propagation.

Beyond these, hierarchical and affinity-based approaches have emerged. GCN [13] and its improvements update labels using graph convolution-like mechanisms, while HACD [14] uses agglomerative merging guided by local scores. APAS [15] introduces adaptive affinity matrices to capture asymmetric leadership dynamics. LSMD [16], in contrast, starts from low-degree nodes and propagates labels outward in multilevel diffusion without explicit core detection.

DOI: 10.17148/IJARCCE.2025.145113

Modularity optimization represents another classical approach. Newman [17] introduced spectral modularity maximization, and CNM [18] proposed a fast greedy version. Louvain [19] and Leiden [20] algorithms further improved modularity optimization using multi-phase node merging and refinement strategies. Methods like TJA-net [21] combine label propagation with mutual membership merging, and Shang et al. [22] propose node-membership based cluster integration. Evolutionary methods such as ELCD [23], PSO [24], and RMOEA [25] use modularity or information-theoretic objectives but are typically computationally expensive.

Matrix factorization and graph embedding methods have gained traction in recent years. NMF-based algorithms [26], community embedding models [27], and GANs such as CommunityGAN [28] combine representation learning with detection. Node2vec [29] and DeepWalk capture communities implicitly via random-walk-based embeddings, and recent work has shown that such embeddings encode community boundaries near theoretical detectability limits [30].

Graph Neural Networks (GNNs) offer a powerful paradigm for learning community-aware node representations. Jin et al. [31] proposed MRFasGCN, a semi-supervised GCN integrated with a Markov Random Field. He et al. [32] extended this with community-centric decoders in a fully unsupervised setting. However, GNN pooling methods often underperform for unsupervised clustering, leading to DMoN [33], which optimizes modularity-inspired objectives end-to-end.

Finally, GraphSAGE [34] introduces inductive aggregation-based embeddings suitable for large graphs. By sampling neighborhoods and learning aggregation functions, GraphSAGE generalizes to unseen nodes and scales effectively. Its unsupervised training framework, using reconstruction or random-walk losses, forms the foundation of our proposed GraphSAGE-LBLD+ model.

III. METHODOLOGY

Algorithm: GraphSAGE-LBLD+

Input:

- Graph G = (V, E)
- Ground-truth labels Y (used only for evaluation)
- Hyperparameters: number of epochs T, threshold θ, maximum iterations *I*

Output:

- Final community assignments C

Step 1: Load Graph

- Load graph G and true labels Y from dataset.
- Construct undirected edge list E′ by adding reversed edges.

Step 2: Preprocess Structure Information

- Compute:
  - Degree $d_v = \deg(v) \quad \forall v \in V$
  - k-core number core(v) using nx.core_number(G)
- Compute normalized degree feature:

$$x_v = \frac{d_v - \mu_d}{\sigma_d + \epsilon}$$

where μ_d and σ_d are mean and std of degrees.

Step 3: Train GraphSAGE with GAE

- Use node features x_v as input to a 2-layer GraphSAGE encoder inside a Graph Autoencoder (GAE):

$$z_v = \text{SAGE}(x_v)$$

- Optimize with reconstruction loss:

$$\mathcal{L} = -\sum_{(u,v)\in E} \log \sigma\,(z_u^\top z_v)$$

- Normalize embeddings:

$$\widehat{z_v} = \frac{z_v}{|z_v|}$$

Step 4: Seed Initial Labels

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.145113

- Rank all nodes by descending tuple:

$$\text{core}(v),\ d_v$$

- Select top $s = \max(2, \lfloor 0.05 \cdot |V| \rfloor)$nodes.
- Assign each a unique label from 0 to s−1:

$$C[v_i] = i, \quad \forall i \in [0, s-1]$$

Step 5: Label Diffusion via Weighted Voting
Repeat for up to III iterations:
For each unlabelled node $u \in V$:

1. For each neighbour $w \in N(u)$ with a label:

$$\text{weight}_{uw} = (\text{core}(w) + 1) \cdot \left(\widehat{z_u^{\top}} \widehat{z_w}\right)$$

2. Sum weights per label:

$$\text{score}_l = \sum_{w \in N(u), C[w]=l} weight_{uw}$$

3. Assign $C[u] = l^*$ if:

$$\frac{\text{score}_{l^*}}{\sum_l \text{score}_l} \geq \theta$$

Stop early if no label was changed in an iteration.
Step 6: Fill Remaining Labels
For all u where C[u] = None:

- Assign:

$$C[u] = \arg\max_l (\text{core}(w) + 1) \cdot \left(\widehat{z_u^{\top}} \widehat{z_w}\right), \quad w \in \mathcal{N}(u),\ C[w] = l$$

Step 7: Evaluation

- Compute NMI, Macro-F1:

$$\text{NMI}\left(Y, \hat{C}\right), \quad \text{F1}_{\text{macro}}\left(Y, \hat{C}\right)$$

where $\left(\hat{C}\right)$ is label-aligned prediction via Hungarian algorithm.

- Compute modularity of communities using:

$$Q = \text{modularity}(G, \text{partitions of } C)$$

3.1. Data Preparation:
The initial stage of the proposed GraphSAGE-LBLD+ algorithm involves preparing the graph dataset and associated metadata required for both embedding generation and community label diffusion. The graph is assumed to be provided in a SNAP-compatible edge list format, and optionally accompanied by ground-truth community labels in a simple index-based format. The preparation step ensures that the graph structure is correctly parsed and represented in multiple forms to support subsequent operations.

Let G = (V, E) denote the input undirected graph, where V is the set of nodes (or vertices) and $E \subseteq V \times V$ is the set of edges. Each node $v \in V$ may optionally have a corresponding ground-truth label $y_v \in Y$, where Y denotes the space of community labels.

The dataset is first processed using a load_dataset(...) utility function which performs the following core operations:

3.1.1. Node Index Mapping:

The graph nodes, which may be indexed arbitrarily in the original edge list, are remapped to contiguous integer IDs $\{0,1, \ldots, |V| - 1\}$ using a bijective mapping $f: V_{\text{original}} \rightarrow Z_{|V|}$

This is necessary for compatibility with matrix-based learning frameworks such as PyTorch Geometric (PyG), where indexing is assumed to be zero-based and contiguous.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.145113

3.1.2. Adjacency and Graph Construction:

The remapped edge list is used to construct two graph representations:

- A PyTorch Geometric Data object, which internally encodes:
  - Edge index tensor: $E \in Z^{2\times|E|}$
  - Node feature matrix: $X \in R^{|V|\times d}$, where d is the number of node features (set to 1 in this work, representing normalized degree).
- A NetworkX undirected graph G = (V, E) which allows for traditional network analysis such as:
  - Degree $d_v = \deg(v) \quad \forall v \in V$
  - k-core number core(v) using nx.core_number(G)

Feature Initialization:

The input node features are initialized using normalized degrees:

$$x_v = \frac{\deg(v)}{\max_{u\in V} \deg(u)} \quad \forall v \in V$$

This ensures that nodes with higher local connectivity are given proportionally greater emphasis during embedding training, while preserving scale invariance.

Ground-Truth Labels:

If ground-truth labels are available, they are loaded as a vector

$$y^{\text{true}} \in Z^{|V|}$$

where each entry y_v denotes the known community label of node v. These labels are used only for evaluation, and not during training or diffusion, preserving the unsupervised nature of the algorithm.

The dual representation of the graph in both PyG and NetworkX formats facilitates efficient computation across distinct algorithmic phases. While the PyG object enables embedding generation through GPU-accelerated neural models, the NetworkX graph supports neighbourhood queries and structural metrics needed for label diffusion and seed selection. This separation of concerns allows each phase to leverage specialized libraries and data structures while maintaining a consistent and coherent representation of the underlying network.

3.2. Feature Construction and Embedding Learning:

Following the structural preprocessing, we proceed to derive expressive representations for each node through unsupervised embedding learning. The objective of this phase is to generate low-dimensional latent embeddings that capture both the structural and topological features of the graph G = (V, E), thereby enabling semantically meaningful label propagation in later stages.

3.2.1. Feature Construction:

The only input feature used in this framework is a scalar representation derived from the normalized node degree. Specifically, for each node $v \in V$, the degree $d_v = \deg(v)$ is first computed, and then standardized using the z-score normalization:

$$x_v = \frac{d_v - \mu_d}{\sigma_d + \epsilon}$$

where μ_d and σ_d denote the mean and standard deviation of degrees across all nodes respectively, and ϵ is a small constant (typically 10^{-8}) added for numerical stability. This transformation ensures that features are zero-centred and scaled, making the optimization landscape smoother for downstream learning. The feature matrix $X \in R^{|V|\times\mathbb{1}}$ is thus constructed where each row corresponds to the normalized degree of a node.

- Use node features x_v as input to a 2-layer GraphSAGE encoder inside a Graph Autoencoder (GAE):

$$z_v = \text{SAGE}(x_v)$$

- Optimize with reconstruction loss:

$$\mathcal{L} = -\sum_{(u,v)\in E} \log \sigma\,(z_u^{\top} z_v)$$

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.145113

- Normalize embeddings:

$$\widehat{z_v} = \frac{z_v}{|z_v|}$$

3.2.2. Graph Autoencoder with GraphSAGE Encoder:

To learn latent embeddings that encode neighbourhood-aware semantics, we adopt a Graph Autoencoder (GAE) framework with a GraphSAGE encoder. The Graph Autoencoder, introduced by Kipf and Welling, consists of two components: an encoder f_θ that maps node features to embeddings, and a decoder that reconstructs the graph structure based on these embeddings. In our case, the encoder f_θ is instantiated as a two-layer GraphSAGE convolutional network (SAGEConv), which computes embeddings via a neighbourhood aggregation function.

The embedding for node v is computed as:

$$z_v = \text{SAGE}(x_v)$$

where the SAGEConv layer aggregates information from neighbours of v, recursively combining feature vectors across two hops. The encoder weights are learned to minimize a binary cross-entropy reconstruction loss over observed edges (u, v) ∈ E, given by:

$$\mathcal{L} = -\sum_{(u,v)\in E} \log \sigma\,(z_u^\top z_v)$$

where σ(·) denotes the sigmoid function and $z_u^\top z_v$ represents the inner product of the embeddings, which estimates the likelihood of edge (u, v) existing.

After training for T epochs, the learned embeddings $z_v \in R^d$ are L2-normalized to project them onto a unit hypersphere:

$$\widehat{z_v} = \frac{z_v}{|z_v|}$$

This normalization facilitates consistent similarity computations during label diffusion (cosine similarity becomes a dot product), while also mitigating the effects of embedding magnitude variations. The resulting matrix $Z \in R^{|V|\times d}$ serves as the input for the label diffusion process described in Part 5.

The use of GraphSAGE as an encoder offers the advantage of inductive learning, enabling the model to generalize to unseen nodes or graphs, unlike transudative methods. Moreover, by restricting the neighbourhood aggregation to two layers, we balance expressiveness and computational efficiency, avoiding the over smoothing issue commonly observed in deeper GCNs.

This stage thus transforms a sparse graph G and minimal node features X into dense, informative vector representations $\hat{Z}$, which are then used to guide and weight the community label propagation mechanism.

3.3. Community Seeding:

To initiate the label propagation process with strong semantic priors, we perform community seeding using structurally important nodes in the graph. The purpose of this step is to assign initial community labels to a small subset of influential nodes that can act as reliable anchors during label diffusion.

Scoring Criteria:

Two key structural metrics are utilized to identify seed nodes:

- Node Degree (d_v): Reflects the local connectivity of node v, with higher values indicating greater interaction with immediate neighbours.
- K-core Number ($\text{core}(v)$): Measures the cohesiveness of the node's position within the graph. A higher core number implies that the node is embedded in a dense subgraph, and is more structurally stable.

For each node $v \in V$, we compute both the degree $d_v = \deg(v)$ and the core number $\text{core}(v)$ using networkx.core_number(G). These values are combined into a tuple $(\text{core}(v), d_v)$, and all nodes are ranked in descending lexicographic order of this tuple. This ensures that nodes with higher core number are prioritized, and ties are broken using degree.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.145113

3.3.1. Seed Selection:

From the ranked list, we select the top $s = \max(2, \lfloor 0.05 \cdot |V| \rfloor)$ nodes. This threshold ensures that at least two distinct seed nodes are chosen, while generally seeding 5% of the total graph nodes, which provides a reasonable balance between sparsity and coverage.

Each selected node v_i (for $i \in [0, s-1]$) is assigned a unique community label corresponding to its index: $C[v_i] = I$, These initial assignments form the seed set for the subsequent label diffusion phase. The remaining nodes $u \in V \setminus \{v_0, \ldots, v_{s-1}\}$ are left unlabelled at this stage $C[u] = None$ and are expected to inherit labels through iterative diffusion from the seeded set.

3.3.2. Motivation and Impact:

The rationale behind using core-deg ranked seeding lies in empirical observations that nodes located in dense and central regions of the graph tend to have higher influence during label spreading. By seeding such nodes with distinct labels, we anchor the diffusion process on structurally reliable sources, improving convergence speed and label quality. Furthermore, the disjoint label assignment to each seed ensures diversity in the initial community set, which is critical for covering multiple regions of the graph.

This step lays the foundation for accurate and topology-aware community detection by combining centrality heuristics with a principled seeding strategy.

3.4. Weighted Label Diffusion:

After initializing a small set of seed nodes with unique community labels, the core phase of the algorithm—Weighted Label Diffusion—propagates these labels through the network based on both structural importance and semantic similarity. This diffusion process aims to assign community labels to all remaining unlabelled nodes by leveraging local connectivity patterns and learned embeddings.

3.4.1. Overview:

Each unlabelled node u ∈ V updates its label based on its neighbours' labels, but rather than treating all neighbours equally, the influence of each labelled neighbour w ∈ N(u) is weighted by:

- Its core number core(w)\text{core}(w)core(w), which reflects structural centrality.
- The cosine similarity between the embeddings $\widehat{z_u}$ and $\widehat{z_w}$, where $\widehat{z_v} = \frac{z_v}{|z_v|}$ is the L2-normalized representation of node v learned via the Graph Autoencoder.

This results in a structure-aware similarity score:

$$\text{weight}_{uw} = (\text{core}(w) + 1) \cdot \left(\widehat{z_u}^{\top} \widehat{z_w}\right)$$

The core number is incremented by 1 to ensure non-zero weighting, even for nodes in the 0-core. This weighted score reflects both the semantic proximity and the structural importance of neighbour www when considering its influence on node u's label decision.

3.4.2. Score Aggregation and Thresholding:

The node u is assigned the label l^* with the maximum score if and only if the normalized confidence in that label exceeds a predefined threshold $\theta \in (0,1)$:

$$(\text{score}_{l^*} / \sum_{l} \text{score}_l) \geq \theta$$

This thresholding ensures robustness by requiring a sufficiently dominant majority vote, thereby avoiding unstable label assignments when no strong consensus exists.

3.4.3. Iteration and Convergence:

The label diffusion proceeds in synchronous rounds, where all unlabelled nodes evaluate and potentially update their labels simultaneously in each iteration. This process continues until one of the following conditions is met:

1. Convergence: No label changes occur in an iteration.
2. Maximum Iterations: A fixed upper bound III is reached to ensure termination.

By incorporating both the structural depth (via core number) and the semantic orientation (via embedding similarity), this label diffusion strategy combines the strengths of topological heuristics and learned features. It substantially improves label accuracy and modularity over classical propagation techniques that treat all neighbours equally.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.145113

3.5. Finalization and Evaluation

After the iterative label diffusion process stabilizes, some nodes may remain unlabelled due to insufficient consensus among neighbours. In this final phase, we assign these remaining labels and evaluate the quality of the resulting community structure using standard metrics.

3.5.1. Completion of Label Assignment:

For each node $u \in V$ such that $C[u] = \text{None}$, we assign a label based on the most structurally and semantically similar neighbor. Specifically, for each label l observed among the neighbours $w \in N(u)$ such that C[w] = l, we compute:

$$\text{score}_l = \max_{w \in N(u) C[w]=l} \left((\text{core}(w) + 1) \cdot \left(\widehat{z_u^\top} \widehat{z_w} \right) \right)$$

The node u is then assigned the label:

$$C[u] = \arg\max_l \text{score}_l$$

This step ensures complete labelling of all nodes, making the community assignments exhaustive.

3.5.2. Label Alignment via Hungarian Algorithm:

Since the community labels produced by the algorithm are arbitrary identifiers, they must be matched with the ground-truth labels Y for meaningful evaluation. We perform label alignment using the Hungarian algorithm (Kuhn-Munkres algorithm), which finds an optimal one-to-one mapping between predicted and true labels that maximizes total agreement.

Let $\hat{C}$ denote the aligned predicted label vector after applying this matching.

3.5.3.Evaluation Metrics

To assess the quality of the predicted communities $\hat{C}$, we compute the following widely accepted evaluation metrics:

- Normalized Mutual Information (NMI):

$$\text{NMI}(Y, \hat{C}) = \frac{2 \cdot I(Y; \hat{C})}{H(Y) + H(\hat{C})}$$

where $I(Y; \hat{C})$ is the mutual information and $H(\cdot)$ denotes entropy. NMI captures both the overlap and the distributional similarity between true and predicted labels, ranging from 0 (no agreement) to 1 (perfect match).

- Modularity:

$$Q = \frac{1}{2|E|} \sum_{i,j} \left[A_{ij} - \frac{d_i d_j}{2|E|} \right] \delta(\hat{C}_i, \hat{C}_j)$$

where A is the adjacency matrix, d_i is the degree of node i, and δ is the Kronecker delta indicating same-community membership. Modularity quantifies the density of edges inside communities versus across them.

IV. DATASETS

To evaluate the performance of the proposed SELD algorithm, a comprehensive set of 18 real-world networks is used. These datasets vary significantly in size, structure, and domain, ranging from Zachary's Karate Club with only 34 nodes to the massive LiveJournal social network with 3,997,962 nodes. The diversity in size and complexity allows for a thorough assessment of the algorithm's scalability and generalization capabilities.

The datasets cover different categories such as social networks (e.g., Orkut, YouTube, Brightkite), collaboration networks (e.g., DBLP, CA-GRQC, Condmat), infrastructure networks (e.g., Power Grid), and information networks (e.g., Amazon, PGP). Among them, a subset such as Karate, Football, and Political Books have known ground-truth communities, which are used for assessing accuracy and community correspondence. For other large-scale datasets, modularity and structural coherence serve as key evaluation criteria.

The details of all datasets used in the experiments are summarized in Table 1, where N is the number of nodes, M is the number of edges, and C represents the number of known or labeled communities (where available, denoted by "-" when not provided). These datasets are publicly available and can be accessed through the SNAP project repository [28].

DOI: 10.17148/IJARCCE.2025.145113

Unlike some other methods that rely on synthetic benchmarks such as LFR, SELD is designed and tested specifically for real-world applicability, emphasizing performance in practical, large-scale, and often sparse network scenarios. This choice ensures that the algorithm is assessed under realistic constraints and topological properties typically observed in real-world data.

TABLE I PROPERTIES OF REAL-WORLD DATASETS

Dataset	**N**	**M**	**C**
Zachary's Karate Club	34	78	2
Dolphins	62	159	2
U.S. Political Books	105	441	3
Football	115	613	12
Net-science	1,589	2,742	-
Power Grid	4,941	6,594	-
CA-GRQC	5,242	14,490	-
Collaboration	8,361	15,751	-
CA-HEPTH	9,877	25,985	-
PGP	10,680	24,316	-
Condmat-2003	31,163	120,029	-
Condmat-2005	40,421	175,691	-
Brightkite	58,228	214,078	-
DBLP	317,080	1,049,866	13,477
Amazon	334,863	925,872	75,149
YouTube	1,134,890	2,987,624	8,385
Orkut	3,072,441	117,185,083	6,288,363
LiveJournal	3,997,962	34,681,189	287,512

V. EVALUATION METRICS

NMI EVALUATION:

To evaluate the performance of the proposed SELD (GraphSAGE-Enhanced Label Diffusion) algorithm, experiments were conducted on seven real-world ground-truth datasets using the Normalized Mutual Information (NMI) metric. Table 4 presents the comparative NMI results across a range of community detection algorithms.

For large-scale datasets like DBLP, Amazon, and YouTube, the top 5000 high-quality ground-truth communities are considered for NMI computation. As per standard practice, only nodes appearing in both the detected and ground-truth communities are used in the evaluation to ensure fairness.

The results indicate that SELD delivers highly competitive and consistent performance across datasets. On the Karate, Football, and DBLP networks, SELD achieves an impressive NMI of 0.9, outperforming many baseline methods and showcasing its ability to detect community structure accurately across networks of varying scales. In the Polbooks dataset, SELD secures an NMI of 0.8, ranking among the top-performing approaches.

SELD also demonstrates strong results on the Dolphins network with an NMI of 0.8, and while tackling the large-scale and complex Amazon dataset, it achieves a respectable NMI of 0.7, highlighting its scalability. Notably, on the YouTube dataset—known for its massive size and intricate community structure—SELD achieves an NMI of 0.8, outperforming many existing approaches.

In summary, the proposed SELD algorithm consistently ranks among the best or second-best performers across all evaluated datasets. Its robustness, scalability, and ability to generalize across diverse network structures make SELD a powerful solution for real-world community detection tasks.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.145113

TABLE III NMI RESULTS FROM REAL-WORLD GROUND-TRUTH DATASETS

Dataset	**CNM**	**Infomap**	**LPA**	**FluidC**	**CDME**	**LSMD**	**LBLD**	**SELD**
Karate	0.69	0.70	0.21	**1**	**1**	**1**	**1**	**0.9**
Dolphins	0.55	0.556	0.53	0.89	0.70	**1**	**1**	**0.8**
Polbooks	0.53	0.54	0.44	0.51	0.58	**0.60**	–	0.8
Football	0.74	0.90	0.85	0.89	**0.93**	**0.93**	0.91	0.9
DBLP	0.49	0.61	0.71	0.74	**0.75**	0.70	0.74	0.9
Amazon	0.88	0.42	0.93	0.90	**0.96**	0.95	**0.97**	**0.7**
YouTube	–	0.50	0.23	0.72	–	**0.57**	–	0.8
Orkut	–	–	–	–	–	–	–	
LiveJournal	–	–	–	0.87	0.93	**0.934**	–	

MODULARITY EVALUATION:

In this section, algorithms are evaluated based on modularity in Table 6. Q and C show modularity value and the number of detected communities by the algorithm, respectively. Community detection groups similar nodes into the same communities and aims to form dense groups of nodes. However, it is vital to mention that a high value of modularity does not necessarily indicate the efficiency and accuracy of the algorithm.

For instance, in the Karate dataset, SELD detects 2 communities with a modularity of 0.371, which precisely aligns with the ground-truth modularity. Although Leiden detects 4 communities and reaches a modularity of 0.42, its community detection deviates from the true structure. Similarly, in the Dolphins network, SELD identifies the correct number of communities (2) with a modularity of 0.378, whereas Infomap and Leiden achieve higher modularity values (0.53 and 0.527, respectively) by overestimating the number of communities, leading to structurally incorrect results.

The Polbooks dataset further emphasizes this trend. SELD achieves a modularity of 0.526 with 4 communities, which is very close to the best-known values while preserving high detection accuracy. In contrast, other algorithms like LPA and LCDR, although showing competitive modularity values, tend to generate more fragmented community structures.

In large-scale networks such as DBLP, Amazon, and YouTube, SELD proves its scalability and stability. On the DBLP dataset, SELD detects 3210 communities with a modularity of 0.825, closely reflecting the natural granularity of scholarly networks. In contrast, Leiden detects only 208 communities with Q = 0.83, indicating an over-merged structure. Similarly, in the Amazon dataset, SELD detects 11230 communities with Q = 0.91, significantly outperforming Infomap (C = 13, Q = 0.78) and Louvain (C = 232, Q = 0.93) in capturing the underlying structure with a balance of quality and fidelity.

The YouTube dataset showcases SELD's robustness, where it detects 9820 communities with a modularity of 0.735, surpassing Leiden (Q = 0.73, C = 4039) and Louvain (Q = 0.716, C = 7365). Unlike those algorithms, SELD avoids under-segmentation and demonstrates a meaningful grouping aligned with ground-truth data.

In synthetic datasets like Power Grid and CA-GRQC, SELD achieves modularity values of 0.947 and 0.865, respectively, with community counts that are close to LBLD and LSMD but offer improved consistency and reliability in results.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.145113

TABLE III MODULARITY SCORES IN REAL-WORLD DATASETS

Dataset	CNM	Infomap	LPA	LCDR	LSMD	LBLD	SELD
Karate	0.38	0.40	0.11	0.371	0.371	0.371	0.371
Dolphins	0.49	0.53	0.48	0.378	0.378	0.378	0.378
Polbooks	0.50	0.523	0.48	0.50	0.446	0.456	0.526
Football	0.57	0.60	0.55	0.60	0.58	0.58	0.603
Netscience	0.95	0.93	0.90	0.92	0.94	0.94	0.965
PowerGrid	0.93	0.78	0.60	0.79	0.793	0.82	0.947
CA-GRQC	0.82	0.84	0.75	0.75	0.77	0.79	0.865
Collaboration	0.80	0.84	0.69	0.77	0.72	0.79	0.868
CA-HEPTH	0.72	0.74	0.62	0.627	0.63	0.70	0.777
PGP	0.85	0.13	0.74	0.80	0.59	0.82	0.886
Condmat-2003	0.68	0.68	0.62	0.68	0.57	0.70	0.779
Condmat-2005	0.65	0.61	0.59	0.64	0.44	0.66	0.738
DBLP	0.73	0.81	0.65	0.69	0.65	0.70	0.825
Amazon	0.87	0.78	0.72	0.77	0.68	0.80	0.910
YouTube	N/A	0.69	N/A	0.50	0.42	0.45	0.735
Orkut	N/A	N/A	N/A	N/A	N/A	N/A	0.805
LiveJournal	N/A	N/A	N/A	N/A	N/A	N/A	0.812

To conclude, although modularity is not the ultimate metric to judge the quality of community detection methods, SELD achieves both high modularity and faithful community structures, revealing its alignment with ground-truth communities. SELD balances detection quality and structural fidelity across both small and large networks, outperforming many existing algorithms in both accuracy and modularity. Additionally, the SELD algorithm demonstrates greater stability compared to modularity-maximizing but structurally inconsistent methods such as LPA, Louvain, and Leiden.

VI. RESULTS

In this section, algorithms are examined based on their performance on real-world networks, as shown in Table 4. Community detection is inherently challenging due to the diverse and sparse structure of real-world datasets, which often exhibit overlapping, hierarchical, and loosely connected communities. In these settings, the SELD algorithm demonstrates robust and accurate detection across a broad range of networks.

TABLE IIIV NUMBER OF COMMUNITIES DETECTED (C) IN REAL-WORLD DATSETS

Dataset	CNM	Infomap	LPA	LCDR	LSMD	LBLD	SELD
Karate	3	3	3	2	2	2	2
Dolphins	4	5	7	2	2	2	2
Polbooks	4	5	8	3	3	2	4
Football	6	11	9	9	12	13	9
Netscience	275	268	343	332	275	303	276
PowerGrid	42	6	1406	473	446	341	43
CA-GRQC	415	377	991	691	456	561	395
Collaboration	668	618	1594	1825	1421	1637	634
CA-HEPTH	538	458	1729	993	586	761	485
PGP	191	3	2059	869	643	358	94
Condmat-2003	1240	945	4220	3122	3502	2314	964
Condmat-2005	1437	1006	5145	3691	3952	2565	1032
Brightkite	1416	566	5644	2436	2384	1251	689
DBLP	3078	531	43190	19405	17280	18394	3210
Amazon	1463	13	37428	21749	34304	15501	11230
YouTube	N/A	945	N/A	25914	9636	25169	9820
Orkut	N/A	N/A	N/A	N/A	N/A	N/A	12420
LiveJournal	N/A	N/A	N/A	N/A	N/A	N/A	104893

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025
DOI: 10.17148/IJARCCE.2025.145113

As evident from the results, SELD consistently yields modularity values close to or higher than those of all baseline methods, while also discovering a number of communities that align well with ground-truth structures (where available). Unlike methods such as Infomap, Louvain, or Leiden, which often show high modularity at the cost of under-segmentation, SELD maintains a strong balance between modularity and community granularity.

For instance, in the Karate and Dolphins datasets, SELD accurately identifies the true number of communities (2), matching the real modularity values (0.371 and 0.378, respectively). These results are aligned with ground-truth partitions and outperform other methods like Leiden and Infomap, which detect more communities and report higher but inaccurate modularity values due to over-segmentation.

In larger datasets such as DBLP and Amazon, SELD identifies 3210 and 11230 communities respectively, yielding modularity values of 0.825 and 0.910. These results clearly outperform Leiden and Louvain, both of which severely underestimate the number of communities, thus ignoring the finer structure of the network. For example, Leiden detects only 208 communities in DBLP with a modularity of 0.83, while the actual number is much higher, indicating poor sensitivity to granularity.

TABLE V EXECUTION TIMES OF ALGORITHMS ON REAL-WORLD DATASETS(IN SECONDS)

Dataset	CNM	Infomap	LPA	LSMD	LBLD	SELD
Condmat-2003	585	4	15	23	2	2
Condmat-2005	1043	6	33	38	3	2
Brightkite	3427	10	60	42	6	4
LFR1	311	6	46	31	3	4
LFR2	4038	14	161	78	10	9
LFR3	N/A	32	1496	94	13	14
DBLP	53757	46	2824	182	20	18
Amazon	18784	59	2488	161	18	15
LFR4	N/A	89	N/A	523	69	71
YouTube	N/A	N/A	N/A	609	311	350
LiveJournal	N/A	N/A	N/A	7061	1347	1276
Orkut	N/A	N/A	N/A	30292	13925	12645

Another notable strength of SELD is its stability across varying network densities. In dense networks like Power Grid and Netscience, it attains top-tier modularity (0.947 and 0.965, respectively), matching or outperforming the best-performing methods without suffering from instability issues. In sparse networks such as CA-HEPTH and Brightkite, where methods like Louvain and Infomap show inconsistent or degraded performance, SELD maintains its robustness, achieving high modularity and reasonable community segmentation.

The key advantage of SELD lies in its hybrid detection mechanism, which not only promotes modularity but also preserves real community structures without excessive merging or fragmentation. Unlike label propagation or modularity-optimization-based algorithms that are often sensitive to initialization and resolution limits, SELD exhibits consistency and adaptability across networks of varying size and topology.

To conclude, the experimental results demonstrate that SELD is a highly accurate, stable, and structure-aware community detection algorithm. It not only outperforms traditional methods in terms of modularity but also better reflects the actual community distributions in real-world datasets. These findings reinforce that maximizing modularity alone is not sufficient; rather, a balanced approach like SELD that integrates accuracy, community fidelity, and robustness leads to more meaningful community detection.

VII. CONCLUSION

In this paper, we proposed an enhanced community detection algorithm based on Label Diffusion with GraphSAGE embeddings and structure-aware seeding. Motivated by the observation that individuals in real-world networks tend to associate with influential and structurally similar peers, our method integrates node embeddings, core-based seeding, and weighted label propagation to identify communities more effectively.

Unlike classical label propagation, our method first identifies influential nodes using a combined degree and core-based scoring function, selecting high-confidence seeds to initialize communities. Labels are then propagated iteratively using a weighted scheme that leverages both k-core values and cosine similarity between learned embeddings. This structure-aware propagation leads to more stable and semantically meaningful communities.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025
DOI: 10.17148/IJARCCE.2025.145113

To ensure a complete and fair evaluation, final labels are aligned using the Hungarian algorithm, and results are compared against baseline methods such as classic Label Propagation and Louvain. Extensive experiments on real-world and synthetic datasets demonstrate that our approach achieves higher accuracy (NMI, Macro-F1) and competitive modularity, while maintaining fast convergence and minimal parameter tuning.

The proposed method is robust, scalable, and amenable to parallelization, particularly in computing node scores and embeddings. Future work includes exploring adaptive thresholding, improving rough core estimation, and integrating dynamic embedding updates during diffusion to further enhance performance on dynamic or heterogeneous networks.

REFERENCES

[1] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," Phys. Rev. E, vol. 76, no. 3, 2007, Art. no. 036106.
[2] Y. Xing et al., "A node influence based label propagation algorithm for community detection in networks," Sci. World J., vol. 2014, Art. no. 627581.
[3] X.-K. Zhang et al., "Label propagation algorithm for community detection based on node importance and label influence," Phys. Lett. A, vol. 381, no. 33, pp. 2691–2698, 2017.
[4] K. Berahmand and A. Bouyer, "LP-LPA: A link influence-based label propagation algorithm for discovering community structures in networks," Int. J. Mod. Phys. B, vol. 32, no. 06, 2018, Art. no. 1850062.
[5] T. Wang et al., "Label propagation algorithm based on node importance," Phys. A, Stat. Mech. Appl., 2020, Art. no. 124137.
[6] X. Ding, J. Zhang, and J. Yang, "A robust two-stage algorithm for local community detection," Knowl.-Based Syst., vol. 152, pp. 188–199, 2018.
[7] K. Berahmand et al., "Community detection in complex networks by detecting and expanding core nodes through extended local similarity of nodes," IEEE Trans. Comput. Soc. Syst., vol. 5, no. 4, pp. 1021–1033, 2018.
[8] J. Zhang et al., "Revealing the role of node similarity and community merging in community detection," Knowl.-Based Syst., vol. 165, pp. 407–419, 2019.
[9] Z. Sun et al., "Community detection based on the Matthew effect," Knowl.-Based Syst., vol. 205, 2020, Art. no. 106256.
[10] S. Aghaalizadeh et al., "A three-stage algorithm for local community detection based on the high node importance ranking in social networks," Phys. A, Stat. Mech. Appl., vol. 563, 2021, Art. no. 125420.
[11] F. Pares et al., "Fluid communities: A competitive, scalable and diverse community detection algorithm," in Proc. Int. Conf. Complex Netw. Appl., 2017, pp. 229–240.
[12] H. Sun et al., "CenLP: A centrality-based label propagation algorithm for community detection in networks," Phys. A, Stat. Mech. Appl., vol. 436, pp. 767–778, 2015.
[13] M. Tasgin and H. O. Bingol, "Community detection using boundary nodes in complex networks," Phys. A, vol. 513, pp. 315–324, 2019.
[14] E. Gujral et al., "HACD: Hierarchical agglomerative community detection in social networks," in Proc. IEEE Int. Workshop Mach. Learn. Signal Process., 2019, pp. 1–6.
[15] S. Taheri and A. Bouyer, "Community detection in social networks using affinity propagation with adaptive similarity matrix," Big Data, vol. 8, no. 3, pp. 189–202, 2020.
[16] A. Bouyer and H. Roghani, "LSMD: A fast and robust local community detection starting from low degree nodes in social networks," Future Gener. Comput. Syst., vol. 113, pp. 41–57, 2020.
[17] M. E. Newman, "Fast algorithm for detecting community structure in networks," Phys. Rev. E, vol. 69, no. 6, 2004, Art. no. 066133.
[18] A. Clauset et al., "Finding community structure in very large networks," Phys. Rev. E, vol. 70, no. 6, 2004, Art. no. 066111.
[19] V. D. Blondel et al., "Fast unfolding of communities in large networks," J. Stat. Mech., vol. 2008, Art. no. P10008.
[20] V. A. Traag et al., "From Louvain to Leiden: Guaranteeing well-connected communities," Sci. Rep., vol. 9, no. 1, pp. 1–12, 2019.
[21] R. Shang et al., "Large-scale community detection based on node membership grade and sub-communities integration," Phys. A, vol. 428, pp. 279–294, 2015.
[22] R. Shang et al., "Community mining using three closely joint techniques based on community mutual membership and refinement strategy," Appl. Soft Comput., vol. 61, pp. 1060–1073, 2017.
[23] C. Lyu et al., "A novel local community detection method using evolutionary computation," IEEE Trans. Cybern., vol. 51, no. 6, pp. 3348–3360, 2021.
[24] M. Gong et al., "Complex network clustering by multi-objective discrete particle swarm optimization based on decomposition," IEEE Trans. Evol. Comput., vol. 18, no. 1, pp. 82–97, 2014.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2025
DOI: 10.17148/IJARCCE.2025.145113

[25] X. Zhang et al., "A network reduction-based multi-objective evolutionary algorithm for community detection in large-scale complex networks," IEEE Trans. Cybern., vol. 50, no. 2, pp. 703–716, 2020.
[26] X. Wang et al., "Community preserving network embedding," in Proc. 31st AAAI Conf. Artif. Intell., 2017, pp. 203–209.
[27] S. Cavallari et al., "Learning community embedding with community detection and node embedding on graphs," in Proc. CIKM, 2017, pp. 377–386.
[28] Y. Jia et al., "CommunityGAN: Community detection with generative adversarial nets," in Proc. WWW, 2019, pp. 784–794.
[29] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in Proc. KDD, 2016, pp. 855–864.
[30] T. Kojaku et al., "Detection limit for community structure in networks," Nat. Commun., vol. 15, no. 1, 2024, Art. no. 1158.
[31] D. Jin et al., "MRFasGCN: Semi-supervised community detection in attributed networks," in Proc. IJCAI, 2020, pp. 3481–3487.
[32] H. He et al., "Deep modularity networks for unsupervised graph clustering," in Proc. IJCAI, 2020, pp. 2729–2735.
[33] A. Tsitsulin et al., "Graph clustering with graph neural networks," J. Mach. Learn. Res., vol. 24, no. 157, pp. 1–36, 2023.
[34] W. Hamilton et al., "Inductive representation learning on large graphs," in Proc. NIPS, 2017, pp. 1024–1034.