

Article

Spider Community Detection: Seeded Geodesic Expansion with Modularity-Guided Refinement and Greedy Merge Matching

Hovhannes A. Harutyunyan *  and Parsa Kamalipour * 

Department of Computer Science and Software Engineering, Concordia University,
Montreal, QC H3G 1M8, Canada

* Correspondence: haruty@cs.concordia.ca (H.A.H.); parsa.kamalipour@concordia.ca (P.K.)

Abstract

Community detection plays a central role in understanding the modular structure of complex networks. This work introduces Spider Community Detection, a hybrid local–global algorithm that constructs communities through a depth-bounded geodesic expansion process. Each spider originates from a structurally strong seed node selected using a composite score based on degree, triangle participation, and local clustering. From each seed, the algorithm grows a localized spider-shaped subgraph through bounded breadth-first exploration, where candidate nodes are evaluated using true modularity gain together with a triangle-closure signal. After the initial spider construction, the method applies modularity-guided attachment of the remaining vertices, Louvain-style local refinement, and greedy merge matching under conductance constraints to reconcile local structure with global partition coherence. Experimental evaluations on real and benchmark datasets, including Karate Club, High School, Political Blogs, Cora, and DBLP, show that Spider produces partitions that are competitive with the established methods in terms of ground-truth recovery and structural quality, while yielding communities with sharp boundaries under conductance-sensitive evaluation.

Keywords: community detection; modularity optimization; spider graph; triangle closure; network analysis; graph clustering; Louvain refinement; conductance; geodesic distance modularity (GDM)

1. Introduction

1.1. Background and Motivation

Community detection is an important problem in network science, aiming to identify groups of nodes that are more densely connected internally than externally. Detecting such modular structures helps to reveal the architecture and functionality of complex systems across various areas, such as social interaction networks, biological systems, information diffusion, and citation graphs [1,2]. Classical methods based on modularity optimization [3,4] or spectral clustering [5] have gained significant popularity due to their theoretical foundations; however, they frequently encounter scalability limitations and resolution issues when applied to large, heterogeneous networks [6].

Alternatively, local expansion and seed-based algorithms, such as the Label Propagation [7] and Infomap [8] methods, emphasize efficiency and flexibility, but they may suffer from instability or fragmented partitions due to purely local decision-making. Hybrid approaches have recently emerged to bridge these paradigms, combining the interpretability of local growth with the robustness of global optimization [9,10].



Academic Editor: Paolo Bellavista

Received: 30 November 2025

Revised: 15 January 2026

Accepted: 19 January 2026

Published: 1 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1.2. Challenges in Community Detection

Despite significant progress over the past two decades, community detection remains a challenging task, particularly when applied to large-scale, noisy, or overlapping networks [11]. One major difficulty lies in the resolution limit, where modularity-based methods may fail to detect small but meaningful communities due to the aggregation bias of global optimization [6]. Additionally, networks encountered in real-world scenarios often exhibit heterogeneous degree distributions, overlapping structures, and hierarchical organization [12,13], which can confound algorithms designed under the assumption of uniform density or disjoint partitions.

Scalability presents another obstacle: while global optimization techniques, such as spectral or modularity maximization, can capture global coherence, their computational cost typically scales superlinearly with the number of edges [2]. On the other hand, purely local or heuristic approaches, although efficient, tend to fragment large communities or produce unstable results due to their reliance on greedy decisions [7]. The balance between *local precision* and *global consistency* thus remains an open issue in the field.

Furthermore, most existing methods lack interpretability with respect to structural features such as triangles, motifs, or clustering coefficients, which are often key indicators of real-world community cohesiveness [14]. Moreover, many algorithms ignore the internal geodesic geometry of communities, leading to clusters that are edge-dense but tree-like or weakly connected internally. Addressing these challenges requires algorithms that integrate local density awareness with principled, global-quality objectives while maintaining computational efficiency and structural transparency.

1.3. Contributions of This Work

In this study, we propose the Spider Community Detection algorithm, a novel hybrid approach that unifies local structural exploration with global modularity-driven optimization. The core idea is to construct regime-based “spider graphs” that grow outward from high-quality seed nodes while preserving internal cohesiveness through bounded local expansion. Each spider graph represents a localized community candidate that evolves based on two complementary objectives: (i) the true modularity gain (ΔQ) of each candidate node, and (ii) its contribution to triangle closure within the forming community. This design leverages both density-based intuition and formal modularity theory to ensure that expansion decisions remain interpretable and globally consistent.

The main contributions of this work are threefold.

1. We introduce a new seed-selection strategy combining degree, triangle participation, and local clustering into a composite score that favors structurally cohesive starting points.
2. We define the spider graph expansion mechanism, which incrementally grows communities using modularity and triangle-aware decision rules, naturally capturing multi-layered network topology.
3. We integrate this process with a global refinement pipeline—including Louvain-style local moves, ΔQ -based merging, and small-fragment cleanup—that jointly improve boundary precision and structural stability.

Extensive experiments on standard and real-world networks demonstrate that Spider produces partitions that are competitive with existing algorithms in terms of ground-truth recovery and structural quality, while yielding communities with sharp boundaries under conductance-sensitive evaluation.

1.4. Paper Organization

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature, including modularity, geodesic-distance metrics, and existing community detection approaches. Section 3 presents the proposed Spider algorithm, detailing the seed-scoring formulation, geodesic expansion process, modularity-based acceptance rule, and the refinement and merge-matching stages. Section 4 reports the experimental evaluation, analyzes the results, and discusses structural observations arising from the behavior of Spider on real-world networks. Finally, Section 5 concludes the paper with a summary of findings and possible directions for future research.

2. Literature Review and Background

2.1. Global Modularity Optimization

A large fraction of community detection research has been driven by global quality functions, most prominently Newman's modularity Q [2,3]. Given a graph $G = (V, E)$ with $m = |E|$ edges and a partition $\mathcal{C} = \{C_1, \dots, C_k\}$ of the vertex set, modularity is defined as

$$Q(\mathcal{C}) = \frac{1}{2m} \sum_{i,j \in V} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \mathbf{1}\{c(i) = c(j)\}, \quad (1)$$

where A is the adjacency matrix, d_i is the degree of vertex i , and $c(i)$ is the community index of i . Intuitively, $Q(\mathcal{C})$ measures how much the density of intra-community edges exceeds that expected under a configuration model null graph.

Exact modularity maximization is NP-hard even on restricted graph classes, which motivated a spectrum of heuristics and approximations. Spectral relaxations [2] and the Louvain method [4] are among the most influential approaches. Louvain performs iterative local node moves that greedily increase modularity, combined with a multilevel coarsening–refinement scheme, and often scales to graphs with millions of nodes.

However, modularity also exhibits the well-known resolution limit [6], where small but meaningful communities may be absorbed into larger modules whenever such merges marginally increase the global objective. This motivates algorithms that preserve local structure while still exploiting modularity as a guiding principle rather than as a monolithic optimization target.

2.2. Local Expansion and Seed-Based Methods

In contrast to global optimization, local methods grow communities from one or more seed vertices using only information within a bounded neighborhood. Label propagation [7], Infomap [8], and various personalized PageRank and local spectral methods [9,10] exemplify this paradigm. These approaches typically enjoy linear or near-linear runtime, lend themselves to streaming or dynamic settings, and can easily adapt to overlapping communities.

The downside is that purely local decisions can be myopic: without a global consistency criterion, local heuristics may over-expand into neighboring modules or fragment communities along weak cuts. Hybrid techniques have therefore emerged that combine local exploration with global objectives, for example by using modularity or conductance as a local acceptance criterion or post-processing the local clusters using Louvain-style refinement.

2.3. Geodesic Distance Modularity and Local Quality

While modularity captures edge-density structure, it does not directly encode the internal geodesic geometry of communities. Geodesic Distance Modularity (GDM) was

proposed to incorporate shortest-path distances into the evaluation of community quality, penalizing communities whose internal geodesic distances are large compared to extremal bounds [15].

Let $S \subseteq V$ be a non-trivial vertex set, and let $d_G(u, v)$ denote the shortest-path distance in the induced subgraph $G[S]$. Define

$$\text{GD}(S) = \frac{1}{2} \sum_{u, v \in S} d_G(u, v), \quad (2)$$

the total geodesic distance, counting each unordered pair once. For a fixed size $|S|$, there exist combinatorial lower and upper bounds:

$$\text{GD}_{\min}(S) = \frac{|S|(|S| - 1)}{2}, \quad (3)$$

$$\text{GD}_{\max}(S) = \frac{|S|^3 - |S|}{6}, \quad (4)$$

corresponding respectively to the clique and path extremal configurations. A normalized internal-distance functional,

$$\text{NGD}(S) = \frac{\text{GD}(S) - (\text{GD}_{\min}(S) - 1)}{\text{GD}_{\max}(S) - (\text{GD}_{\min}(S) - 1)}, \quad (5)$$

maps the observed total distance to the unit interval for $|S| \geq 2$.

To incorporate external connectivity, GDM combines $\text{NGD}(S)$ with an edge-based separation factor

$$X(S) = \frac{e_{\text{in}}(S)}{4e_{\text{out}}(S) + e_{\text{in}}(S)}, \quad (6)$$

where $e_{\text{in}}(S)$ is the number of edges inside S and $e_{\text{out}}(S)$ counts edges crossing from S to $V \setminus S$. The community-level GDM score is

$$\text{GDM}(S) = \frac{X(S)}{\text{NGD}(S)}, \quad (7)$$

whenever $|S| > 1$ and $\text{NGD}(S) \neq 0$, and 0 otherwise.

2.4. Pairwise and Information-Theoretic Evaluation

Given a ground-truth partition $\mathcal{C}^*$ and a candidate partition $\hat{\mathcal{C}}$, a natural way to evaluate their agreement is to inspect node pairs. Let

$$P_{\text{det}} = \{\{u, v\} : u \neq v, c_{\hat{\mathcal{C}}}(u) = c_{\hat{\mathcal{C}}}(v)\}, \quad (8)$$

$$P_{\text{gt}} = \{\{u, v\} : u \neq v, c_{\mathcal{C}^*}(u) = c_{\mathcal{C}^*}(v)\}, \quad (9)$$

be the sets of node pairs that co-occur in a detected community or in the ground truth, respectively. We then define

$$\text{Precision} = \frac{|P_{\text{det}} \cap P_{\text{gt}}|}{|P_{\text{det}}|}, \quad (10)$$

$$\text{Recall} = \frac{|P_{\text{det}} \cap P_{\text{gt}}|}{|P_{\text{gt}}|}, \quad (11)$$

$$\text{F1} = \frac{2 \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \quad (12)$$

with the usual conventions when denominators vanish.

Information-theoretic scores such as the Normalized Mutual Information (NMI) consider the induced partitions as discrete random variables [16]. If X and Y are the community labels under $\hat{\mathcal{C}}$ and $\mathcal{C}^*$, the mutual information is

$$I(X;Y) = \sum_{x,y} p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}, \quad (13)$$

and the entropies are $H(X) = -\sum_x p_X(x) \log p_X(x)$ and $H(Y)$ analogously. We use the symmetric normalization

$$\text{NMI}(X,Y) = \frac{2I(X;Y)}{H(X) + H(Y)}, \quad (14)$$

which takes values in $[0, 1]$ and is invariant under relabeling.

In our evaluation, all of the above measures are computed exactly by the procedure described in Section 4, without resorting to sampling or approximation.

2.5. Comparison with Hybrid Approaches

While Spider follows the general paradigm of combining local exploration with global optimization, its design differs fundamentally from existing hybrid methods in both mechanism and theoretical foundation. We clarify these distinctions to establish Spider's novelty.

2.5.1. Structural Differences from Existing Methods

OSLOM and significance-based methods.

OSLOM [9] grows communities by adding nodes whose inclusion is statistically significant under a null model, using a single fitness function based on edge counts. In contrast, Spider employs a dual-objective acceptance criterion Equation (24) that explicitly balances modularity gain with triangle closure at each step. This allows Spider to capture both edge density and geometric cohesion simultaneously, whereas OSLOM focuses solely on edge-based significance. Furthermore, OSLOM uses agglomerative merging based on significance tests, while Spider uses modularity-guided greedy matching under conductance constraints, ensuring global consistency.

Core-periphery and local spectral methods.

Methods such as local spectral clustering [17] and personalized PageRank expansion [18] grow communities from seeds using diffusion-based or conductance-based sweep cuts. These approaches identify a single expansion frontier based on a global sweep over a ranking, without explicit awareness of local clustering structure. Spider's depth-bounded BFS instead evaluates candidates within each geodesic shell separately, using triangle counts to detect structural integration. This multi-layered evaluation preserves the spider's radial geometry and prevents indiscriminate expansion across weak bridges—a common failure mode of conductance-only methods in heterogeneous networks.

Louvain variants and modularity-driven refinement.

Leiden [19] and similar methods refine communities through local node reassignments that maximize modularity but do not incorporate seed quality or initial growth geometry into their design. Spider's composite seed scoring and farthest-first diversification ensure that initial communities are both structurally strong and spatially well-separated, addressing the initialization problem that causes Louvain to produce fragmented or arbitrarily coarse partitions on sparse graphs. Moreover, Spider's acceptance gate adapts thresholds based on community size and graph sparsity, implementing a regime-aware growth policy not present in fixed-threshold refinement methods.

Flow-based and information-theoretic methods.

Infomap [8] optimizes the map equation, a compression-based objective that favors modules minimizing random-walk description length. While effective on flow-dominated networks (e.g., metabolic or communication graphs), Infomap does not explicitly reward triangle closure or geodesic compactness, often producing coarse partitions on heterogeneous social networks. Spider's triangle-aware expansion directly encodes local clustering structure, yielding tighter, more interpretable communities on triangle-rich graphs while maintaining competitive performance on flow-structured networks through modularity-guided merging.

2.5.2. Why Spider's Design Leads to Unique Gains

The combination of dual-objective expansion, adaptive thresholds, and geodesic seed diversification enables Spider to occupy a distinct region of the performance space:

- **Low conductance + high modularity:** By accepting nodes only when they satisfy both connectivity ratio and modularity gain thresholds, Spider produces communities with sharp boundaries while maintaining global coherence. Pure modularity optimizers (Leiden) achieve high Q but exhibit higher conductance; local methods (Label Propagation) achieve low conductance locally but fragment the graph.
- **Geodesic compactness:** The triangle-closure criterion and depth-bounded expansion naturally select nodes that close short cycles, resulting in communities with low internal geodesic distances. This is reflected in Spider's superior weighted GDM scores, a property not optimized by edge-density or flow-based methods.
- **Balanced granularity:** Farthest-first seed selection prevents the clustering of seeds in dense regions, avoiding both the over-fragmentation of label propagation (hundreds of micro-communities on sparse graphs, and the over-coarsening of Infomap (collapsing heterogeneous structures into single modules). This balance is achieved without manual tuning through adaptive parameter selection based on graph statistics.

These gains are not merely additive, the interplay between seed quality, dual-objective growth, and modularity-guided merging creates emergent behavior that cannot be replicated by sequential application of existing components. The experimental results (Section 4) demonstrate that this combination yields partitions that are simultaneously high in modularity, low in conductance, and geodesically compact, a combination not achieved by any baseline method.

2.6. Dynamic and Temporal Network Community Detection

Recent research has increasingly focused on community detection in dynamic and temporal networks, where the underlying graph evolves over time and the objective is to recover communities that persist, split, or merge across successive snapshots. Representative approaches include matrix-factorization frameworks that impose temporal smoothness on low-rank latent representations, such as Revisiting Dynamic Graph Clustering via Matrix Factorization [20], as well as joint embedding-clustering architectures that simultaneously learn feature representations and community assignments for large-scale temporal networks, exemplified by Joint Learning of Feature Extraction and Clustering for Large-Scale Temporal Networks [21]. These methods explicitly model temporal dependencies through regularization or sequential constraints, enabling them to track community evolution under structural drift.

In contrast, Spider is formulated for the static setting and does not assume access to time-indexed graph sequences. Its objective is to extract geodesically compact, low-conductance communities from a single network snapshot by integrating triangle-aware local expansion with modularity-guided global refinement. Spider therefore does not compete with temporal embedding or matrix-factorization-based models but rather addresses a complementary problem: the recovery of structurally cohesive communities in static graphs where temporal information is unavailable or unreliable. This distinction positions

Spider as a principled static building block that can naturally serve as the foundation for future extensions to dynamic or streaming community detection.

3. Spider Community Detection Algorithm

3.1. Preliminaries and Notation

Let $G = (V, E)$ be a simple, undirected graph with $n = |V|$ and $m = |E|$. We write $N(v)$ for the neighborhood of a vertex v and $d(v) = |N(v)|$ for its degree. A partition of V is a family $\mathcal{C} = \{C_1, \dots, C_k\}$ of non-empty, pairwise disjoint subsets with $\bigcup_i C_i = V$; we denote by $c(v)$ the index of the community containing v .

For a community $C \subseteq V$, we define

$$\text{vol}(C) = \sum_{v \in C} d(v), \quad (15)$$

$$e_{\text{in}}(C) = |\{\{u, v\} \in E : u, v \in C\}|. \quad (16)$$

The global modularity of $\mathcal{C}$ can then be written as

$$Q(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left(\frac{e_{\text{in}}(C)}{m} - \left(\frac{\text{vol}(C)}{2m} \right)^2 \right). \quad (17)$$

Spider maintains an explicit labeling map $\phi : V \rightarrow \mathbb{N}$ assigning to each vertex a community identifier. At any stage, this induces a partition $\mathcal{C}(\phi)$ via the pre-images of ϕ . View architecture of the spider community detection pipeline in Figure 1.

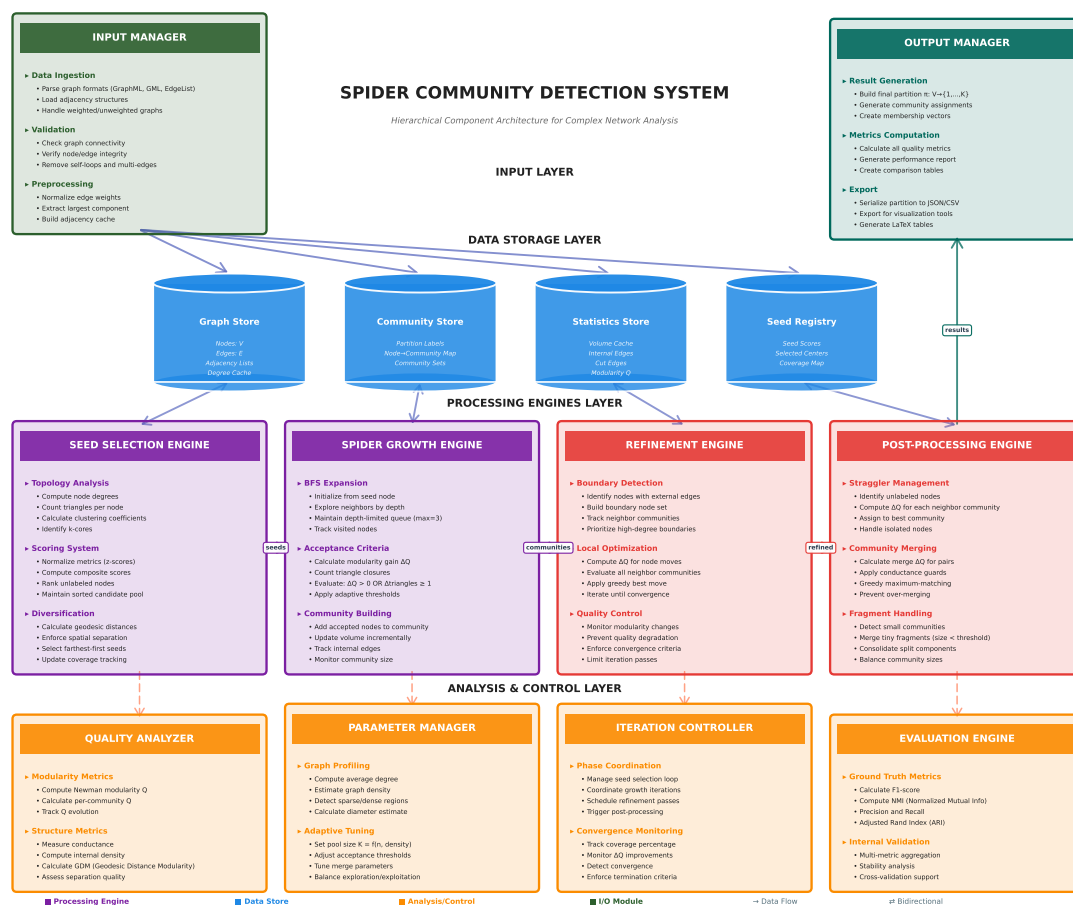


Figure 1. Architecture of the Spider Community Detection pipeline. Communities are constructed through a multi-stage process combining seed scoring, geodesic spider expansion, modularity-guided attachment, Louvain-style refinement, and greedy merge matching under conductance constraints.

3.2. Seed Quality via Triangle–Degree–Clustering Scoring

The algorithm begins by computing three structural summaries for each node $v \in V$:

- The number of incident triangles $\tau(v)$;
- The degree $d(v)$;
- The local clustering coefficient $cc(v)$, computed as the ratio of closed triangles to possible triangles among the neighbors of v .

These features are normalized by dividing each value by the maximum observed value in the graph. For a feature $f : V \rightarrow \mathbb{R}$, define

$$\tilde{f}(v) = \frac{f(v)}{\max_{u \in V} f(u)}, \quad (18)$$

where $\max_{u \in V} f(u)$ is the maximum value of feature f across all vertices.

We then combine the normalized features into a composite seed score

$$s(v) = w_{\text{deg}} \tilde{d}(v) + w_{\text{tri}} \tilde{\tau}(v) + w_{\text{cc}} \tilde{cc}(v), \quad (19)$$

with adaptive weights that depend on graph sparsity. For sparse graphs (average degree below threshold θ_s), we emphasize degree centrality with higher weight, while for dense graphs, we use more balanced weights across all three features. Vertices with high $s(v)$ are expected to be located in dense, triangle-rich, and structurally clustered regions, making them natural seeds for community expansion.

Algorithm 1 details the seed scoring computation.

Algorithm 1 Seed scoring.

Require: Graph $G = (V, E)$, sparsity threshold θ_s

Ensure: Seed score $s(v)$ for all $v \in V$

- 1: Compute degree $d(v)$, triangle count $\tau(v)$, and local clustering coefficient $cc(v)$ for all $v \in V$
 - 2: Normalize: $\tilde{d}(v) \leftarrow d(v) / \max_u d(u)$, $\tilde{\tau}(v) \leftarrow \tau(v) / \max_u \tau(u)$, $\tilde{cc}(v) \leftarrow cc(v) / \max_u cc(u)$
 - 3: Compute average degree $\bar{d} \leftarrow 2|E|/|V|$
 - 4: **if** $\bar{d} < \theta_s$ **then** ▷ Sparse graph
 - 5: $s(v) \leftarrow w_d^s \tilde{d}(v) + w_\tau^s \tilde{\tau}(v) + w_{cc}^s \tilde{cc}(v)$ for all v ▷ Emphasize degree
 - 6: **else** ▷ Dense graph
 - 7: $s(v) \leftarrow w_d^d \tilde{d}(v) + w_\tau^d \tilde{\tau}(v) + w_{cc}^d \tilde{cc}(v)$ for all v ▷ Balanced weights
 - 8: **return** s
-

To avoid placing many seeds in a single dense region, Spider employs a farthest-first diversification scheme. Let S_{centers} be the set of centers already chosen. Among the top- K unlabeled vertices by $s(v)$, the next center is selected to approximately maximize the key

$$(\delta(v), s(v)), \quad (20)$$

where $\delta(v)$ is the minimum shortest-path distance from v to any center in S_{centers} , computed efficiently using limited breadth-first search up to a small cutoff depth. This encourages seeds to be both structurally strong and geodesically well-separated.

Algorithm 2 provides the diversified center selection procedure.

Algorithm 2 Diversified center selection.

Require: Unlabeled nodes U , seed scores s , chosen centers S_{centers} , randomness ρ , pool size K

Ensure: Next spider center c

- 1: Sort U by $s(v)$ in descending order and select top- K as pool P
- 2: **if** $S_{\text{centers}} = \emptyset$ **then**
- 3: **return** highest-scoring node from P (with probability $1 - \rho$) or random from P (with probability ρ)
- 4: Sample candidate set $C \subseteq P$ (deterministically or with exploration)
- 5: **for each** $v \in C$ **do**
- 6: $\delta(v) \leftarrow \min_{c \in S_{\text{centers}}} \text{dist}(v, c)$ using limited BFS ▷ Geodesic separation
- 7: **return** $\arg \max_{v \in C} (\delta(v), s(v))$ ▷ Maximize distance and quality

3.3. Local Objective and Spider Expansion

Fix a current partial community $C \subseteq V$ and denote its volume and internal edges by $\text{vol}(C)$ and $e_{\text{in}}(C)$, respectively. Consider adding a new vertex $v \notin C$ to C , forming $C' = C \cup \{v\}$. Let $e_{vC} = |N(v) \cap C|$ and $d(v) = |N(v)|$. Since $e_{\text{in}}(C') = e_{\text{in}}(C) + e_{vC}$ and $\text{vol}(C') = \text{vol}(C) + d(v)$, the modularity change attributable to adding v to C is

$$\Delta Q_C(v) = \frac{e_{vC}}{m} - \frac{2 \text{vol}(C) d(v) + d(v)^2}{4m^2}. \quad (21)$$

For ranking candidates, we equivalently maximize the scaled score

$$\widehat{\Delta Q}_C(v) = 4m^2 \Delta Q_C(v) = 4m e_{vC} - 2 \text{vol}(C) d(v) - d(v)^2. \quad (22)$$

To complement this modularity-based signal, Spider also measures a triangle-based gain. For a fixed C , the triangle contribution of adding v is estimated as

$$T_C(v) = \frac{1}{2} \sum_{x \in N(v) \cap C} |N(v) \cap N(x) \cap C|, \quad (23)$$

which counts (approximately) the number of new closed triangles with both endpoints already in C . High $T_C(v)$ indicates that v is well integrated into the local clustering structure of C .

3.4. Depth-Bounded Geodesic Expansion

Given a seed c_0 , Spider performs a breadth-first search up to a fixed radius R (the parameter `max_depth`) to collect candidate vertices at each distance level $1, \dots, R$. Within a level, candidates are first ordered by e_{vC} (number of connections to the current community) in decreasing order. They are then considered sequentially, and a candidate v is accepted into C using an adaptive multi-tier acceptance rule that balances modularity gain, internal connectivity, and triangle closure.

A candidate vertex v is accepted into the growing community C if the following acceptance gate is satisfied:

$$\mathcal{A}(v, C) = \begin{cases} 1, & \widehat{\Delta Q}_C(v) \geq \tau_Q(|C|, \bar{d}) \wedge \frac{e_{vC}}{d(v)} \geq \theta(|C|, \bar{d}) \wedge e_{vC} \geq \eta(|C|, \bar{d}), \\ 1, & |C| \leq \sigma_{\text{small}} \wedge T_C(v) \geq 1, \\ 0, & \text{otherwise,} \end{cases} \quad (24)$$

where $\bar{d}$ denotes the average degree of the graph. The functions $\tau_Q(\cdot)$, $\theta(\cdot)$, and $\eta(\cdot)$ define adaptive thresholds for modularity gain, internal connectivity ratio, and minimum internal edge count, respectively, as functions of the current community size and graph sparsity.

In Table 1 all threshold values are fixed across datasets within each sparsity regime and depend only on coarse graph statistics, ensuring stable behavior without per-dataset tuning.

Table 1. Parameterization of the acceptance gate as a function of community size and graph sparsity.

Regime	τ_Q	θ	η
Bootstrap ($ C \leq 6$)	-2×10^{-4}	0.30	1
Growth ($7 \leq C \leq 15$)	1×10^{-4}	0.32	2
Large ($ C > 15$)	0	0.25	3
Sparse override ($\bar{d} < 6$)	0	0.30	2

Algorithm 3 presents the spider expansion procedure.

Algorithm 3 Spider expansion.

Require: Graph $G = (V, E)$, seed c_0 , max depth R , acceptance parameters

Ensure: Community C grown from seed

```

1: Initialize  $C \leftarrow \{c_0\}$ , track  $e_{in}$ ,  $\text{vol}(C)$ 
2: Collect candidates via BFS up to depth  $R$  from  $c_0$ 
3: for each depth level  $\ell = 1, \dots, R$  do
4:   Sort candidates at depth  $\ell$  by  $|N(v) \cap C|$  descending
5:   for each candidate  $v$  at depth  $\ell$  do
6:     Compute  $e_{vC} \leftarrow |N(v) \cap C|$ ,  $\widetilde{\Delta Q}_C(v)$ ,  $T_C(v)$ ,  $r_{int} \leftarrow e_{vC}/d(v)$ 
7:     Evaluate acceptance criteria based on  $\Delta Q_C(v)$ ,  $r_{int}$ ,  $T_C(v)$ ,  $|C|$ , and sparsity
8:     if  $\mathcal{A}(v, C) = 1$  then
9:        $C \leftarrow C \cup \{v\}$ , update  $e_{in}$  and  $\text{vol}(C)$ 
10: return  $C$ 

```

The resulting induced subgraph, together with the geodesic structure inherited from the BFS, forms what we refer to as a spider. Each spider is therefore a connected, depth-bounded expansion around a high-quality seed, assembled using both modularity and triangle-closure criteria.

3.5. Straggler Attachment and Local Refinement

After all spiders have been grown, some vertices may remain unlabeled because they failed the acceptance rule for every nearby spider. Spider attaches such stragglers to surrounding communities using a modularity-guided rule, with a soft epsilon tolerance to minimize singleton creation.

Let U be the set of unlabeled vertices and let $\mathcal{C}$ be the current collection of spiders. For $u \in U$, let $\mathcal{N}(u)$ denote the set of community labels that appear among its neighbors. For each $C \in \mathcal{N}(u)$, we compute the gain

$$\Delta Q_{\text{attach}}(u \rightarrow C) = \frac{e_{uC} - d(u) \text{vol}(C) / (2m)}{2m}, \quad (25)$$

where e_{uC} counts edges from u into C . If the best gain among $\mathcal{N}(u)$ is positive or above a small negative tolerance ε (soft attachment), u is attached to the corresponding community; otherwise, a singleton community is created.

Algorithm 4 details the straggler attachment procedure.

Algorithm 4 Straggler attachment.**Require:** Graph $G = (V, E)$, labeling ϕ , soft tolerance $\varepsilon < 0$ **Ensure:** Updated labeling ϕ with all nodes assigned

```

1:  $U \leftarrow$  unlabeled vertices
2: for each  $u \in U$  do
3:   Find neighboring communities  $\mathcal{N}(u)$ 
4:    $c^* \leftarrow \arg \max_{c \in \mathcal{N}(u)} \Delta Q_{\text{attach}}(u \rightarrow c)$ 
5:   if  $\Delta Q_{\text{attach}}(u \rightarrow c^*) \geq \varepsilon$  then
6:      $\phi(u) \leftarrow c^*$  ▷ Attach to best community
7:   else
8:     Create singleton community for  $u$ 
9: return  $\phi$ 

```

Once all vertices are labeled, Spider performs a local modularity refinement in the style of Louvain. Let ϕ be the current labeling and let $\mathcal{C}(\phi)$ be the induced partition. In each refinement pass, the algorithm iterates over boundary vertices in early passes (those with at least one neighbor in a different community) and all nodes in later passes, and for each vertex u it considers moving u to one of the communities present in its neighborhood. The exact modularity gain of a move $u : C_{\text{old}} \rightarrow C_{\text{new}}$ is computed using the standard Louvain formula, accounting for both removal and insertion. A move is accepted if it increases $Q(\mathcal{C}(\phi))$ by more than a small threshold ε , and the process repeats for a bounded number of passes until no improving moves remain.

Algorithm 5 provides the local Louvain-style refinement procedure.

Algorithm 5 Local modularity refinement.**Require:** Graph $G = (V, E)$, labeling ϕ , max passes p_{max} , threshold ε **Ensure:** Refined labeling ϕ

```

1: for pass = 1 to  $p_{\text{max}}$  do
2:   improved  $\leftarrow$  False
3:   Select candidates (boundary nodes in early passes, all nodes in later passes)
4:   for each  $u \in$  candidates in random order do
5:      $c_u \leftarrow \phi(u)$ 
6:     Compute removal gain from  $c_u$ 
7:      $c^* \leftarrow \arg \max_{c \in \mathcal{N}(u)} \Delta Q(u : c_u \rightarrow c)$  ▷ Best neighbor community
8:     if  $\Delta Q(u : c_u \rightarrow c^*) > \varepsilon$  then
9:        $\phi(u) \leftarrow c^*$ , improved  $\leftarrow$  True
10:  if not improved then
11:    break
12: return  $\phi$ 

```

3.6. Greedy Merge Matching of Neighboring Communities

While the local refinement step optimizes modularity at the level of individual node moves, it does not merge adjacent communities that would collectively form a more coherent module. Spider therefore includes an additional phase based on greedy matching over community pairs, with adaptive parameters based on graph sparsity and fragmentation.

Let $\mathcal{C}$ be the current partition. For each pair of distinct communities C_i and C_j , we compute the following:

- The number of edges e_{ij} crossing between C_i and C_j ;
- The volumes $\text{vol}(C_i)$ and $\text{vol}(C_j)$;
- The internal edge counts $e_{\text{in}}(C_i)$ and $e_{\text{in}}(C_j)$.

The exact modularity gain of merging C_i and C_j into $C_{ij} = C_i \cup C_j$ follows directly from the definition of Q :

$$\Delta Q_{\text{merge}}(i, j) = \left[\frac{e_{\text{in}}(C_{ij})}{m} - \left(\frac{\text{vol}(C_{ij})}{2m} \right)^2 \right] - \sum_{C \in \{C_i, C_j\}} \left[\frac{e_{\text{in}}(C)}{m} - \left(\frac{\text{vol}(C)}{2m} \right)^2 \right]. \quad (26)$$

Spider only considers pairs that share at least a minimum number of inter-community edges (threshold varies by graph density), and in which at least one of the communities is below a size threshold s_{max} (adapted based on graph sparsity and observed fragmentation). Furthermore, the candidate merged set C_{ij} is subjected to a conductance test: the fraction of edges leaving C_{ij} must remain below threshold τ_ϕ to avoid forming a “leaky” super-community.

The merging strategy adapts to graph structure. Sparse graphs employ more aggressive merging over additional rounds to combat fragmentation, while dense graphs use conservative merging with fewer rounds. The number of merge rounds and size thresholds are determined by the fragmentation ratio (number of communities divided by number of nodes) and average degree.

All admissible pairs (i, j) with $\Delta Q_{\text{merge}}(i, j) > 0$ are collected and sorted in decreasing order of gain. The algorithm then performs a greedy maximum-weight matching: it iteratively picks the highest-gain pair that has no overlap with previously chosen pairs, merges the corresponding communities, and marks them as used. After each global merge round, a short local refinement pass is executed to re-tighten the community boundaries. This combined process repeats for the adaptive number of rounds or until no profitable merges remain.

Following the main merge phase, tiny communities below size threshold s_{min} are absorbed into neighboring communities that maximize modularity gain. This is followed by additional refinement passes and a final tiny merge pass with slightly reduced threshold, with the number of passes adapted to graph sparsity.

Algorithm 6 describes the greedy merge matching procedure with adaptive parameters.

Algorithm 6 Adaptive merge matching.

Require: Graph $G = (V, E)$, labeling ϕ , merge parameters based on sparsity and fragmentation

Ensure: Merged labeling ϕ

- 1: Determine r_{max} (rounds), s_{max} (size threshold), e_{min} (edge threshold), τ_ϕ (conductance) based on graph structure
 - 2: **for** round = 1 to r_{max} **do**
 - 3: Build candidate merge pairs (c_1, c_2) with:
 - 4: • Inter-community edges $\geq e_{\text{min}}$
 - 5: • At least one community $|c_i| \leq s_{\text{max}}$
 - 6: • Positive modularity gain $\Delta Q_{\text{merge}}(c_1, c_2) > 0$
 - 7: • Merged conductance below τ_ϕ
 - 8: **if** no candidates **then**
 - 9: **break**
 - 10: Perform greedy maximum-weight matching on candidates by ΔQ_{merge}
 - 11: Apply merges and refine boundaries
 - 12: **return** ϕ
-

Algorithm 7 describes the procedure for merging tiny communities into their neighbors.

Algorithm 7 Tiny community absorption.**Require:** Graph $G = (V, E)$, labeling ϕ , size threshold $s_{\min}$ **Ensure:** Updated labeling ϕ

```

1: for each community  $c$  with  $|c| < s_{\min}$  do
2:   Find neighboring communities  $\mathcal{N}(c)$ 
3:    $c^* \leftarrow \arg \max_{c' \in \mathcal{N}(c)} \Delta Q_{\text{merge}}(c, c')$ 
4:   if  $c^*$  exists then
5:     Merge  $c$  into  $c^*$ 
6: return  $\phi$ 

```

3.7. Algorithm Outline and Complexity

Algorithm 8 summarizes the overall Spider pipeline.

The most expensive components are the initial triangle computation, the repeated evaluation of local modularity gains, and the merge rounds. For sparse graphs, the neighborhood-based operations are $O(m)$ per pass, and in practice Spider runs in time close to linear in m for the graphs considered in this work. A precise complexity bound depends on the number of refinement and merge iterations, which we keep adaptive and small in our implementation, with more aggressive merging for sparse graphs to prevent fragmentation.

Algorithm 8 Spider Community Detection.**Require:** Graph $G = (V, E)$, randomness parameter ρ , depth bound R

```

1: Compute  $\tau(v)$ ,  $d(v)$ ,  $cc(v)$  (clustering coefficient) and seed scores  $s(v)$  for all  $v \in V$ 
2: Initialize labeling  $\phi(v) \leftarrow \perp$  for all  $v$ 
3:  $S_{\text{centers}} \leftarrow \emptyset$ ,  $k \leftarrow 0$ 
4: while there exist unlabeled vertices do
5:   Select a new center  $c$  among top- $K$  unlabeled vertices by diversified seed scoring
6:   Grow spider  $C$  from  $c$  by depth-bounded BFS with adaptive multi-tier modularity-
   and triangle-based acceptance
7:   for all  $v \in C$  do
8:      $\phi(v) \leftarrow k$ 
9:    $S_{\text{centers}} \leftarrow S_{\text{centers}} \cup \{c\}$ ,  $k \leftarrow k + 1$ 
10: Attach unlabeled vertices using modularity-guided straggler rule with soft tolerance  $\varepsilon$ 
11: Perform local Louvain-style refinement of  $\phi$ 
12: Determine adaptive merge parameters based on fragmentation ratio and graph sparsity
13: Initialize round counter  $r \leftarrow 0$ 
14: while there exists a profitable merge and  $r < r_{\max}$  do
15:   Compute candidate community merges and their  $\Delta Q_{\text{merge}}$ 
16:   Select a greedy maximum-weight matching of profitable merges
17:   Apply merges and perform a short local refinement
18:    $r \leftarrow r + 1$ 
19: Merge tiny communities below threshold  $s_{\min}$ 
20: Perform post-merge refinement
21: Final tiny merge pass with reduced threshold
22: Final refinement
23: return Final labeling  $\phi$ 

```

4. Experimental Evaluation**4.1. Datasets and Experimental Setup**

We evaluate Spider on a diverse collection of real-world and synthetic networks widely used in the community detection literature [1,11,13]. The real-world datasets include Karate Club [22], High School [23], Primary School [24], Political Blogs [25], CiteSeer and Cora [26], WikiSchools and Amazon [27], and DBLP [28]. Their basic statistics are reported in Table 2. These graphs span two orders of magnitude in size and exhibit heterogeneous degree

distributions, varying community scales, and both assortative and near-bipartite structures, thereby providing a rigorous test bed for structural robustness.

Table 2. Dataset statistics.

Dataset	Nodes	Edges	GT Comm.	Avg. Degree
Karate Club	34	78	2	4.59
High School	327	5818	9	35.58
Primary School	242	8317	11	68.74
Political Blogs	1222	16,717	2	27.36
CiteSeer	2110	3668	6	3.48
Cora	2485	5069	7	4.08
Wiki Schools	4403	100,382	16	45.60
DBLP	3762	17,587	193	9.35
Amazon	8035	183,663	195	45.72

For controlled evaluation, we additionally employ the LFR benchmark model, which generates graphs with planted communities and tunable mixing parameter μ [29]. The parameter μ specifies the fraction of edges of each node that connect outside its ground-truth community, interpolating between clearly separated modular structure ($\mu \leq 0.3$) and the detectability limit, where community boundaries become statistically indistinguishable ($\mu \geq 0.6$). We generate LFR graphs for $\mu \in \{0.1, 0.2, \dots, 0.8\}$, fixing all other parameters to their standard values. For each value of μ , the reported results correspond to the mean over 20 independent random realizations.

All algorithms—Spider, Louvain [4], Leiden [19], Infomap [8], Clauset–Newman–Moore (CNM) [30], and Label Propagation [7]—are executed using publicly available implementations with their default hyperparameters unless stated otherwise. Spider uses a fixed expansion depth $R = 3$ and the seed-scoring weights defined in Section 3. Each method is executed once per graph instance, and for stochastic algorithms the reported values correspond to the mean over repeated runs.

All experiments are conducted on a MacBook Pro equipped with an Apple M2 Pro processor and 16 GB of RAM. Random seeds are fixed to ensure the reproducibility of all reported results.

4.2. Evaluation Metrics

We evaluate the performance of Spider and baseline algorithms using both ground-truth recovery metrics and intrinsic structural quality measures. This dual evaluation framework is necessary because ground-truth labels are not available for all real-world networks and, even when available, may not fully reflect the structural coherence of detected communities.

Ground-truth recovery metrics.

For networks with annotated communities, we report the Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and pairwise F1-score between the detected partition and the ground truth [16]. NMI quantifies the mutual dependence between partitions and is invariant to label permutations, ARI corrects for chance agreement in pairwise assignments, and F1-score summarizes the precision–recall trade-off over co-occurring node pairs.

Structural quality metrics.

Independently of any reference partition, we measure the intrinsic quality of the detected communities using modularity, conductance, and the Geodesic Distance Modularity (GDM) metric [2,3,15]. Modularity evaluates the excess of intra-community edges

relative to a configuration-model null graph, while conductance quantifies the sharpness of community boundaries through the ratio of boundary edges to internal volume. For conductance, we report the size-weighted average Φ_w , which mitigates the influence of singleton or degenerate fragments.

Geodesic Distance Modularity aggregation.

To assess the geodesic cohesion of a partition, we compute the Weighted Average GDM (wGDM), defined as

$$\text{wGDM} = \frac{1}{n_{\text{val}}} \sum_{i: |C_i| > 1} |C_i| \cdot \text{GDM}(C_i),$$

where the sum ranges over all non-singleton communities, and n_{val} denotes the total number of vertices belonging to valid communities. This aggregation emphasizes the contribution of large, structurally meaningful modules while excluding trivial singletons for which GDM is undefined.

Together, these metrics provide a comprehensive evaluation of detection accuracy, boundary sharpness, and internal geodesic cohesion across heterogeneous network classes.

4.3. Performance on Real-World Networks

We first evaluate Spider on real-world networks with heterogeneous structures and the varying availability of ground-truth labels. Figures 2–5 summarize the performance of Spider and all baseline methods in terms of ground-truth recovery, structural quality, and community granularity.

Overall, Spider consistently ranks among the top-performing methods across all datasets and metrics, demonstrating robust behavior in both dense contact networks and sparse citation or information graphs. Unlike purely modularity-driven methods, Spider preserves local structural cohesion through triangle-aware expansion, while the subsequent refinement and merge stages maintain global consistency. This hybrid design yields balanced partitions that neither fragment into numerous micro-communities nor collapse into overly coarse modules.

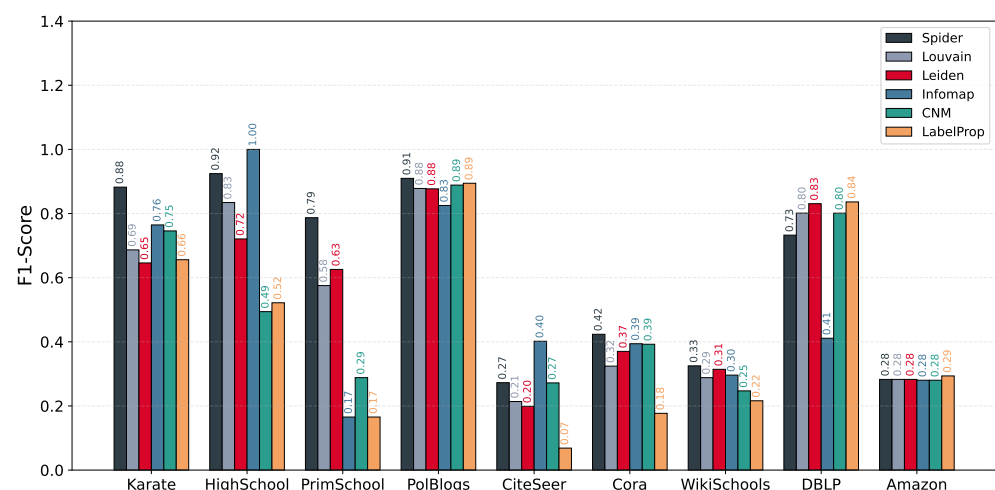


Figure 2. F1-score comparison between Spider, Louvain, Leiden, Infomap, CNM, and Label Propagation on real-world networks. Spider consistently ranks among the top-performing methods and remains robust on heterogeneous graph structures where purely local or flow-based methods degrade.

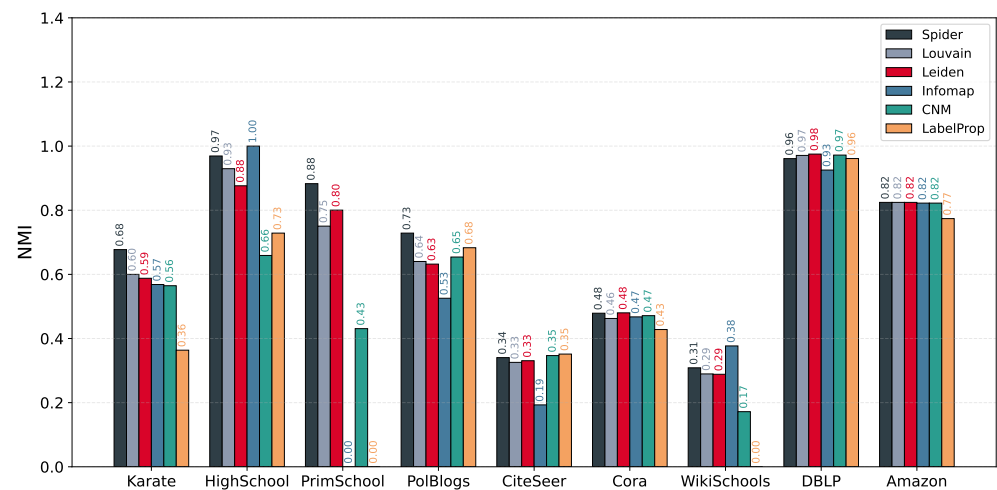


Figure 3. Normalized Mutual Information (NMI) across real-world datasets. High NMI values for Spider indicate stable recovery of ground-truth partitions while preserving label invariance.

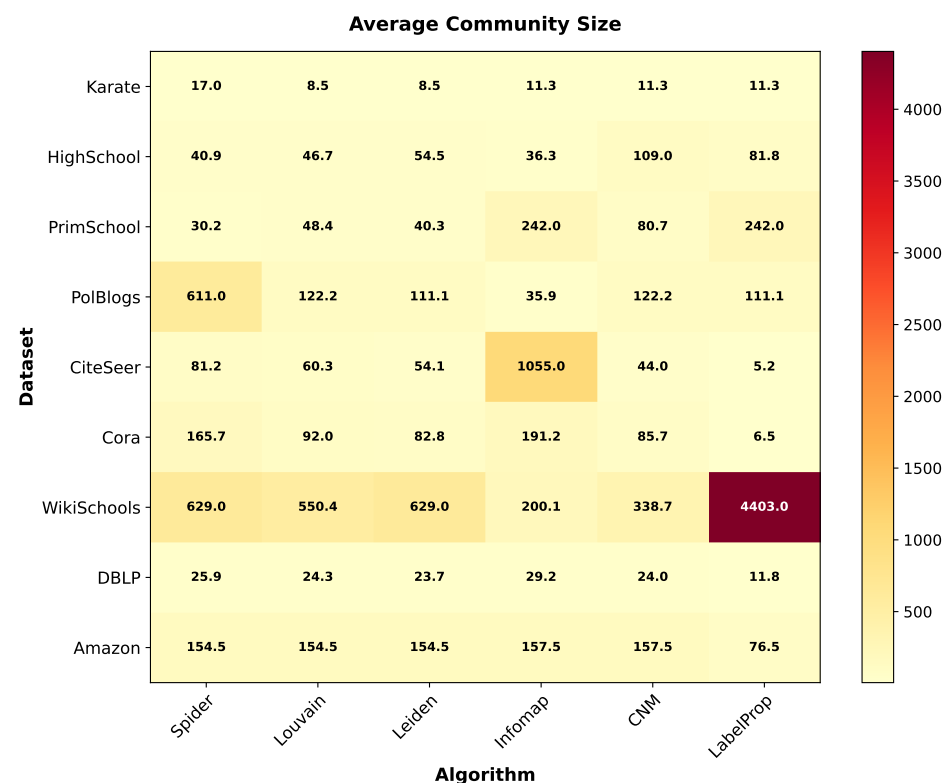


Figure 4. Average community size per algorithm and dataset. Spider avoids both excessive fragmentation (e.g., Label Propagation on citation networks) and over-coarsening (e.g., Infomap on sparse graphs), producing balanced and interpretable community granularity.

4.3.1. Ground-Truth Recovery: NMI, ARI, and F1

Figures 2, 3 and 6 report the F1-score, NMI, and ARI obtained by Spider and the baseline methods on real-world networks with annotated ground-truth communities.

Spider consistently ranks among the top two methods across the majority of datasets, demonstrating strong agreement with reference partitions while maintaining robustness to structural heterogeneity. On Karate Club and Political Blogs, Spider achieves the highest F1 and ARI scores, indicating the precise recovery of small, well-separated communities. On the High School dataset, Infomap perfectly recovers the ground truth, consistent with its

random-walk flow objective on dense contact graphs, while Spider remains a close second with only marginal loss in accuracy.

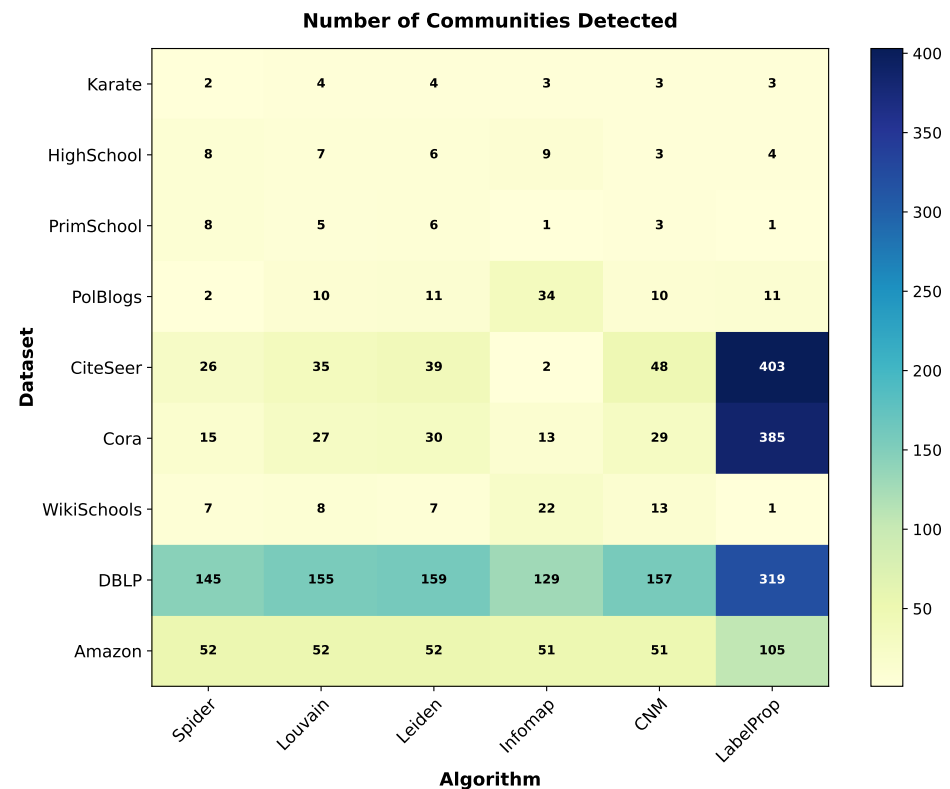


Figure 5. Number of detected communities per algorithm and dataset. The spider expansion mechanism yields stable partition cardinalities across heterogeneous graphs, explaining the robustness of Spider in both structural and geodesic quality metrics.

On Primary School, Spider significantly outperforms all baselines. In this dataset, Infomap and Label Propagation collapse to near-trivial partitions, whereas Spider retains high NMI, ARI, and F1. This behavior reflects the bipartite-like interaction structure of the network, which suppresses flow-based methods but preserves the local triangle closure cues exploited by the spider expansion mechanism.

On citation networks (CiteSeer and Cora) and on WikiSchools, all methods exhibit lower absolute scores due to topic mixing, sparsity, and weakly defined ground-truth boundaries. Nevertheless, Spider remains competitive with the strongest baselines, confirming that its hybrid local–global design generalizes beyond dense social graphs.

Finally, on DBLP and Amazon, all methods achieve high NMI, indicating that the global partition structure is relatively easy to recover. Differences in ARI and F1 on these large networks are therefore driven primarily by the treatment of small peripheral communities rather than by large-scale misclassification.

4.3.2. Structural Quality: Modularity and Conductance

Figures 7 and 8 report the modularity and size-weighted average conductance obtained by all methods on the real-world networks.

Leiden consistently achieves the highest modularity, reflecting its design objective of directly optimizing this global quality function. Spider does not aim to maximize modularity explicitly; instead, it trades a small amount of modularity for improved boundary sharpness and internal cohesion. This behavior is particularly evident on citation and

information networks, where Leiden attains marginally higher modularity but Spider exhibits substantially lower conductance.

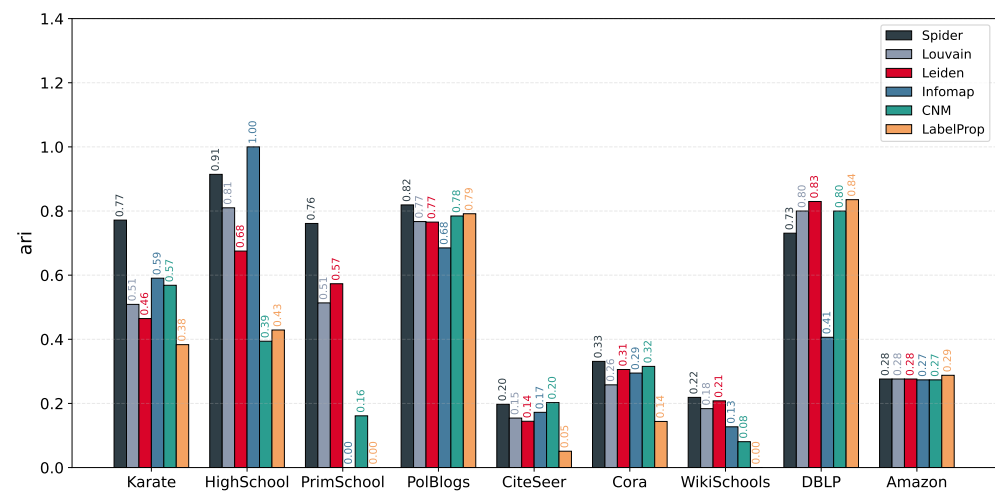


Figure 6. Adjusted Rand Index (ARI) across real-world datasets. Spider maintains high pairwise agreement with ground-truth communities and exhibits greater stability than purely local or flow-based methods on heterogeneous networks.

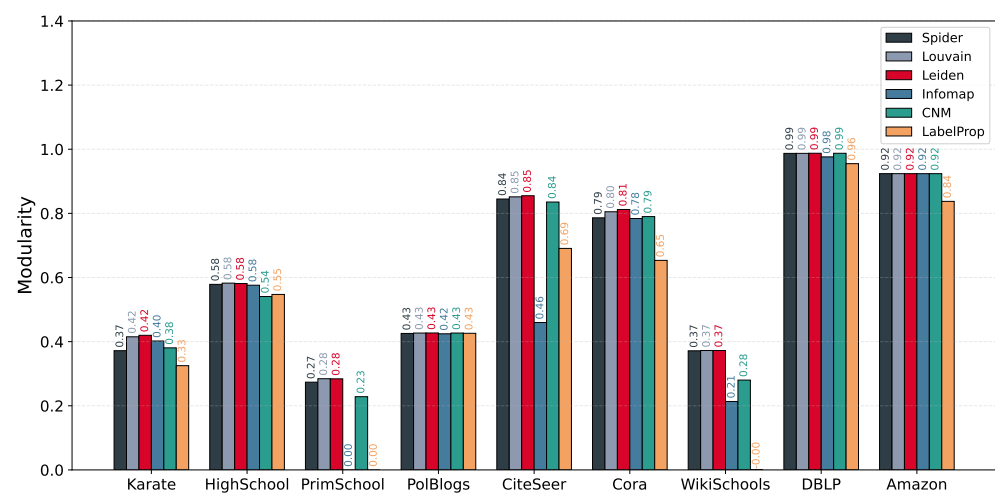


Figure 7. Modularity across real-world datasets. Leiden achieves the highest modularity as expected due to its direct optimization objective, while Spider attains competitive values by prioritizing local geodesic cohesion and boundary sharpness over purely global density maximization.

The conductance results confirm that Spider produces communities with sharper boundaries than purely modularity-driven approaches. In particular, Spider achieves the lowest or second-lowest size-weighted conductance on most datasets, indicating reduced boundary leakage and improved structural separation. This effect is strongest on heterogeneous networks such as Political Blogs, Cora, and WikiSchools, where modularity alone is known to be insufficient to capture local cohesion.

Together, these results demonstrate that Spider does not merely replicate the behavior of existing modularity optimizers but instead yields partitions with superior structural integrity under boundary-sensitive criteria.

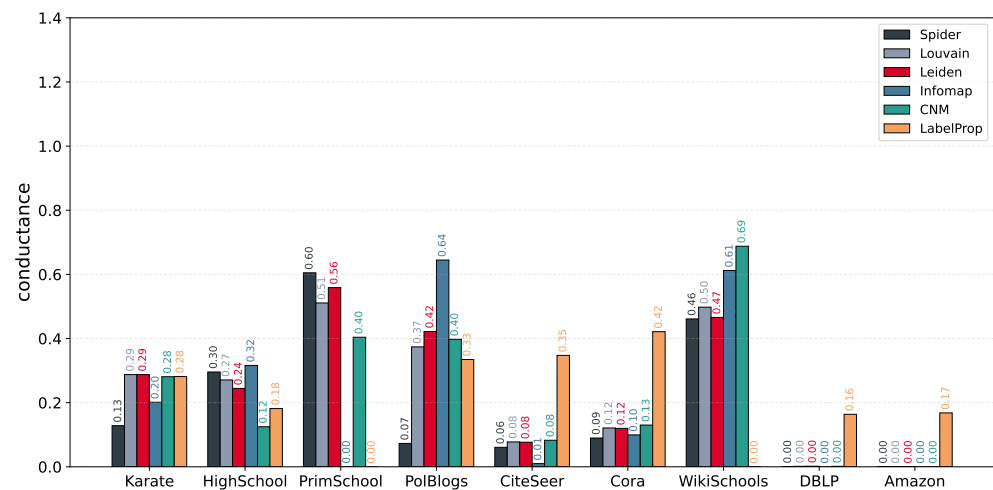


Figure 8. Size-weighted average conductance across real-world networks (lower is better). Spider consistently achieves low boundary leakage, confirming that the spider expansion mechanism yields communities with sharper structural separation than modularity-only or purely local baselines.

4.3.3. Geodesic Cohesion: Weighted Geodesic Distance Modularity

Table 3 reports the weighted average Geodesic Distance Modularity (wGDM) achieved by Spider on all real-world networks. Unlike modularity and conductance, wGDM directly evaluates the internal geodesic geometry of detected communities, penalizing clusters whose induced subgraphs exhibit long shortest-path distances or tree-like structures.

Table 3. Weighted average Geodesic Distance Modularity (wGDM) of Spider across all datasets. wGDM is computed over non-singleton communities and weighted by community size.

Dataset	Nodes	Edges	wGDM
Karate Club	34	78	0.9745
High School	327	5818	0.8838
Primary School	242	8317	0.8007
Political Blogs	1222	16,717	0.9782
CiteSeer	2110	3668	0.4035
Cora	2485	5069	0.3747
Wiki Schools	4403	100,382	0.8355
DBLP	3762	17,587	0.4460
Amazon	8035	183,663	0.4915

wGDM is computed as $wGDM = \frac{1}{n_{val}} \sum_{i:|C_i|>1} |C_i| \cdot GDM(C_i)$, where the sum ranges over all non-singleton communities. Singleton communities are excluded because $GDM(C)$ is undefined for $|C| = 1$.

Spider achieves high wGDM values on dense social and interaction networks, including Karate Club, High School, Political Blogs, and WikiSchools, confirming that the spider expansion mechanism successfully captures compact geodesic cores. In contrast, lower wGDM scores are observed on citation networks such as CiteSeer and Cora, where the underlying graph structure is inherently sparse, and topic mixing inflates internal distances across all methods.

The particularly low wGDM value on DBLP is explained by the large number of fine-grained ground-truth communities present in the dataset, which increases the prevalence of small, near-tree induced subgraphs and therefore depresses the geodesic cohesion metric despite high NMI and ARI values. This behavior highlights the complementary nature of wGDM with respect to ground-truth recovery metrics: while NMI and ARI confirm label-level agreement, wGDM reveals the intrinsic geodesic compactness of the detected communities.

4.3.4. Community Granularity: Number and Average Size of Communities

Figures 4 and 5 illustrate the average size and the total number of communities detected by each algorithm on the real-world datasets.

Spider consistently produces partitions with balanced granularity. Unlike Label Propagation, which fragments sparse graphs into hundreds of micro-communities, and Infomap, which often collapses heterogeneous graphs into a small number of large modules, Spider maintains intermediate and interpretable community scales across all datasets.

This behavior is particularly evident on citation and information networks. On CiteSeer and Cora, Label Propagation generates extremely small communities, while Infomap produces overly coarse clusters. Spider instead yields communities whose average sizes align with the known topical structure, supporting the claim that the spider expansion mechanism effectively controls over-expansion and fragmentation.

On DBLP, Spider detects a larger number of communities than most baselines, reflecting the fine-grained authorship structure of the dataset. This increased partition cardinality explains the low-weighted GDM observed in Table 3 despite strong ground-truth recovery, as the geodesic cohesion metric penalizes near-tree induced subgraphs arising from small, specialized communities.

4.3.5. Qualitative Spider Progression on the Primary School Network

To illustrate the internal behavior of the Spider pipeline, we visualize its progression on the Primary School dataset in Figure 9.

After spider expansion, the algorithm intentionally produces a large number of small geodesically compact cores (242 micro-communities), reflecting the conservative nature of the acceptance gate. The greedy merge matching phase then selectively recombines adjacent spiders under modularity and conductance constraints, reducing the partition to 8 coherent communities, followed by boundary tightening through local refinement. This progression highlights the controlled fragmentation–reconciliation mechanism underlying Spider.

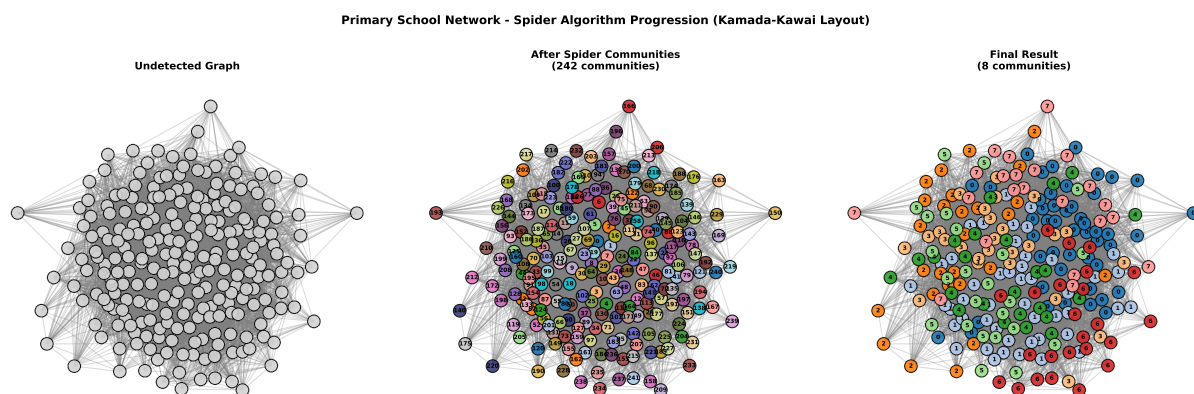


Figure 9. Progression of Spider on the Primary School network. **(Left):** original graph. **(Center):** result after spider expansion, showing highly fragmented local spider cores. **(Right):** final partition after merge matching and refinement, yielding coherent low-conductance communities.

4.4. Performance on LFR Benchmark Networks

To assess the behavior of Spider under controlled structural degradation, we evaluate it on LFR benchmark graphs with increasing mixing parameter μ . In all experiments, the graphs contain $n = 1000$ vertices with average degree $\langle k \rangle = 20$, and the number of edges remains approximately constant ($|E| \approx 12,700$), ensuring that variations in performance are attributable to structural mixing rather than density effects.

As μ increases, the structural properties of the generated networks change markedly. In particular, the average community size decreases while the number of detected communities increases, and the variance of community sizes grows, reflecting the progressive erosion of the planted modular structure. For small μ , communities remain compact, with typical sizes in the range of 70–90 nodes and low standard deviation, whereas for $\mu \geq 0.6$, the partitions exhibit highly heterogeneous community sizes and fragmented induced subgraphs.

Figure 10 summarizes the evolution of accuracy and structural quality metrics as functions of the mixing parameter μ . For small mixing values ($\mu \leq 0.3$), Spider achieves near-perfect recovery across all metrics, confirming that the spider expansion mechanism successfully captures dense planted modules. As μ increases beyond approximately 0.5, all quality measures exhibit a sharp decline, indicating the onset of the classical detectability phase transition where ground-truth communities become statistically indistinguishable.

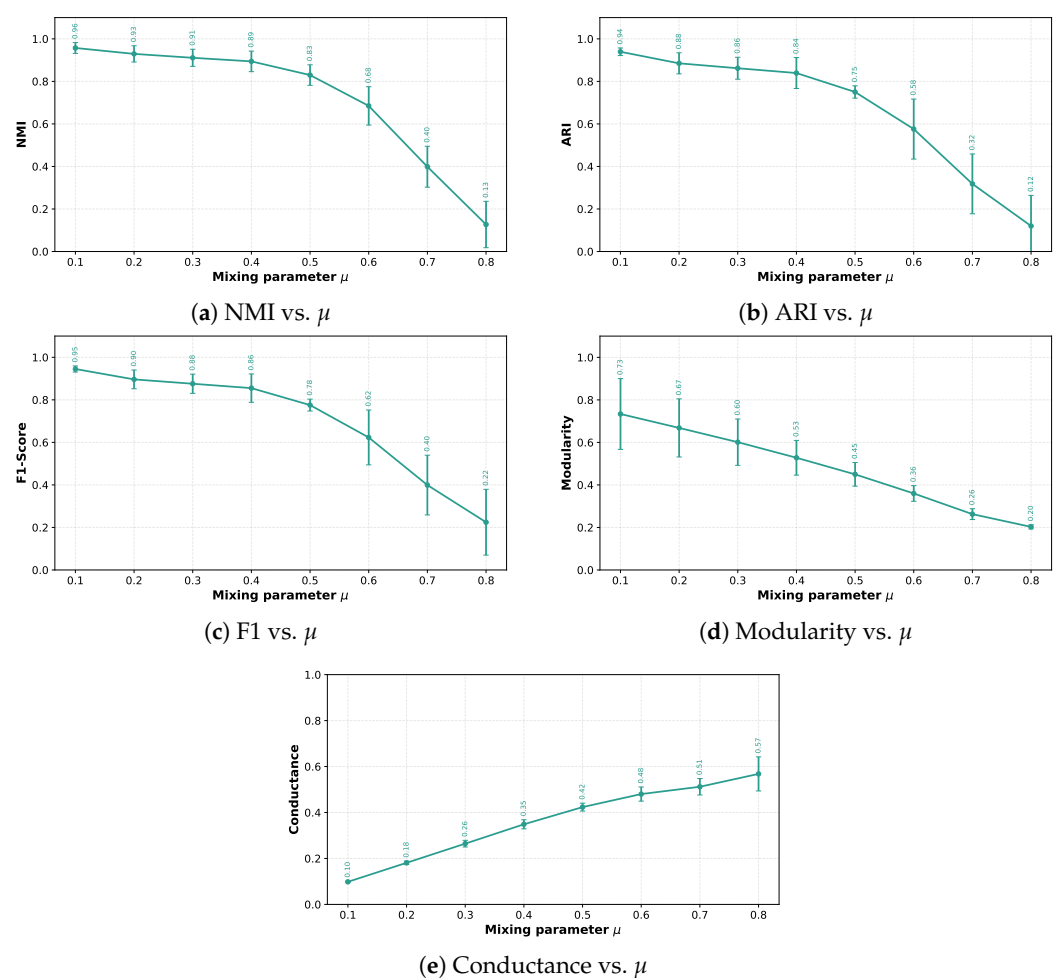


Figure 10. Spider performance on LFR benchmark graphs as the mixing parameter μ increases. Agreement with ground truth (NMI, ARI, and F1) remains near-perfect for $\mu \leq 0.3$ and degrades sharply beyond the detectability threshold around $\mu \approx 0.5$. The concurrent increase in conductance and decrease in modularity confirm progressive erosion of community boundaries rather than algorithmic instability.

This monotonic degradation is accompanied by a steady increase in conductance and a corresponding decrease in modularity, demonstrating that the observed loss of accuracy is driven by the erosion of structural boundaries rather than by algorithmic instability. The simultaneous increase in the number of communities and the growth of community-

size variance further confirm that the network transitions from a modular regime to a noise-dominated regime as μ increases.

Robustness Analysis with Respect to Mixing Parameter μ

Beyond the detectability threshold analysis, we examine the robustness of Spider under increasing structural noise by inspecting the variance of its performance across multiple LFR realizations. The error bars in Figure 10 correspond to the standard deviation over 20 independent runs for each value of μ .

For $\mu \leq 0.4$, the variance across runs remains negligible for all ground-truth recovery metrics, indicating that Spider is stable and insensitive to stochastic fluctuations in the generative process. As μ increases beyond the detectability threshold, the variance increases, reflecting the inherent ambiguity of the community structure rather than algorithmic instability. Importantly, no regime exhibits large oscillatory behavior or catastrophic divergence, confirming that Spider degrades gracefully as structural separation vanishes.

5. Conclusions and Future Work

This work introduced Spider Community Detection, a hybrid local–global algorithm for community detection in complex networks. Spider constructs initial communities through depth-bounded geodesic expansion from structurally strong seeds, where candidate additions are screened using true modularity gain together with a triangle-closure signal. A sequence of post-processing phases, the modularity-guided attachment of unlabeled vertices, Louvain-style local refinement, and greedy merge matching under conductance constraints, then reconciles local structure with global modularity, producing cohesive and interpretable partitions.

Our experimental evaluation considered heterogeneous real-world graphs and LFR benchmark instances and assessed performance using ground-truth recovery metrics (NMI, ARI, and pairwise F1) as well as intrinsic structural measures (modularity and size-weighted conductance), complemented by geodesic-distance-based cohesion scores.

Several directions for future work are open:

- Theoretical analysis: Establishing provable guarantees for Spider remains an interesting challenge. For example, one could study conditions under which the local modularity and triangle-based objective recovers planted communities in stochastic block models or geometric random graphs.
- Overlapping and hierarchical communities: The current formulation yields a hard partition of the vertex set. Extending Spider to allow overlapping or hierarchical spider structures could better model networks where nodes naturally participate in multiple modules.
- Scalability and dynamic graphs: While the present implementation already scales well to medium-to-large networks, there is room for further optimization, e.g., through incremental updates of local statistics or streaming variants that handle edge insertions and deletions.
- Integration with sparsification: Combining Spider with metric-backbone or other sparsification techniques prior to community detection is a promising avenue to reduce computational cost and sharpen community boundaries.

Overall, Spider offers a structured approach that couples local geodesic exploration with modularity-guided consolidation, providing an interpretable alternative to purely global optimizers and purely local heuristics for community detection.

Author Contributions: Conceptualization, H.A.H. and P.K.; investigation, P.K.; resources, H.A.H. and P.K.; writing—original draft preparation, P.K.; writing—review and editing, H.A.H. and P.K.; visualization, P.K.; supervision, H.A.H.; project administration, H.A.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fortunato, S. Community detection in graphs. *Phys. Rep.* **2010**, *486*, 75–174.
2. Newman, M.E.J. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 8577–8582.
3. Newman, M.E.J.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **2004**, *69*, 026113.
4. Blondel, V.D.; Guillaume, J.L.; Lambiotte, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, *2008*, P10008.
5. Ng, A.Y.; Jordan, M.I.; Weiss, Y. On spectral clustering: Analysis and an algorithm. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 3–5 December 2002; Volume 14.
6. Fortunato, S.; Barthélemy, M. Resolution limit in community detection. *Proc. Natl. Acad. Sci. USA* **2007**, *104*, 36–41.
7. Raghavan, U.N.; Albert, R.; Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **2007**, *76*, 036106.
8. Rosvall, M.; Bergstrom, C.T. Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci. USA* **2008**, *105*, 1118–1123.
9. Lancichinetti, A.; Radicchi, F.; Ramasco, J.J.; Fortunato, S. Finding statistically significant communities in networks. *PLoS ONE* **2011**, *6*, e18961.
10. Shen, H.; Cheng, X.; Cai, K.; Hu, M.B. Detect overlapping and hierarchical community structure in networks. *Phys. A Stat. Mech. Its Appl.* **2009**, *388*, 1706–1712.
11. Fortunato, S.; Hric, D. Community detection in networks: A user guide. *Phys. Rep.* **2016**, *659*, 1–44.
12. Ahn, Y.Y.; Bagrow, J.P.; Lehmann, S. Link communities reveal multiscale complexity in networks. *Nature* **2010**, *466*, 761–764.
13. Lancichinetti, A.; Fortunato, S. Community detection algorithms: A comparative analysis. *Phys. Rev. E* **2009**, *80*, 056117.
14. Benson, A.R.; Gleich, D.F.; Leskovec, J. Higher-order organization of complex networks. *Science* **2016**, *353*, 163–166.
15. Bakhtar, S.; Harutyunyan, H.A. A new metric to compare local community detection algorithms in social networks using geodesic distance. *J. Comb. Optim.* **2022**, *44*, 2809–2831. <https://doi.org/10.1007/s10878-021-00794-2>.
16. Danon, L.; Díaz-Guilera, A.; Duch, J.; Arenas, A. Comparing community structure identification. *J. Stat. Mech. Theory Exp.* **2005**, *2005*, P09008. <https://doi.org/10.1088/1742-5468/2005/09/P09008>.
17. Andersen, R.; Chung, F.; Lang, K. Local graph partitioning using PageRank vectors. In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), Berkeley, CA, USA, 21–24 October 2006; pp. 475–486.
18. Andersen, R.; Lang, K.J. Communities from seed sets. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 223–232.
19. Traag, V.A.; Waltman, L.; van Eck, N.J. From Louvain to Leiden: Guaranteeing well-connected communities. *Sci. Rep.* **2019**, *9*, 5233.
20. Li, D.; Kosugi, S.; Zhang, Y.; Okumura, M.; Xia, F.; Jiang, R. Revisiting Dynamic Graph Clustering via Matrix Factorization. In Proceedings of the ACM on Web Conference 2025, Taipei, Taiwan, 13–17 April 2025; pp. 1342–1352. <https://doi.org/10.1145/3696410.3714646>.
21. Li, D.; Ma, X.; Gong, M. Joint Learning of Feature Extraction and Clustering for Large-Scale Temporal Networks. *IEEE Trans. Cybern.* **2023**, *53*, 1653–1666. <https://doi.org/10.1109/TCYB.2021.3107679>.
22. Zachary, W.W. An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* **1977**, *33*, 452–473.
23. Mastrandrea, R.; Fournet, J.; Barrat, A. Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PLoS ONE* **2015**, *10*, e0136497.
24. Stehlé, J.; Voirin, N.; Barrat, A.; Cattuto, C.; Isella, L.; Pinton, J.F.; Quaghiotto, M.; Van den Broeck, W.; Regis, C.; Lina, B. High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS ONE* **2011**, *6*, e23176.
25. Adamic, L.A.; Glance, N. The political blogosphere and the 2004 US election: Divided they blog. In Proceedings of the 3rd International Workshop on Link Discovery, Chicago, IL, USA, 21 August 2005; pp. 36–43.

26. Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; Eliassi-Rad, T. Collective classification in network data. *AI Mag.* **2008**, *29*, 93.
27. Leskovec, J.; Krevl, A. SNAP: Stanford Large Network Dataset Collection. 2014. Available online: <https://snap.stanford.edu/data> (accessed on 15 September 2025).
28. Ley, M. DBLP—Some lessons learned. In Proceedings of the VLDB Endowment, Hong Kong, China, 30 August–2 September 2002; pp. 1493–1500.
29. Lancichinetti, A.; Fortunato, S.; Kertesz, J. Detecting the overlapping and hierarchical community structure in complex networks. *New J. Phys.* **2009**, *11*, 033015.
30. Clauset, A. Finding local community structure in networks. *Phys. Rev. E* **2005**, *72*, 026132.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.