Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.087484

ARTICLE

Relation-Aware Adaptive Anchor Spectral Clustering for Large-Scale Attributed and Heterogeneous Graphs

Xiangmin Liu[1,2], **Weizhi Xiong**[1,*], **Wei Zhang**[1], **Taojian Luo**[1], **Xiu Yao**[1], **Jian Hu**[1] and **Peng Peng**[2,*]

[1]School of Information Engineering, Gannan College of Science and Technology, Ganzhou, China
[2]College of Computer Science and Electronic Engineering, Hunan University, Changsha, China

*Corresponding Authors: Weizhi Xiong. Email: mrxiongwz@foxmail.com; Peng Peng. Email: hnu16pp@hnu.edu.cn

Received: 17 June 2026; Accepted: 17 August 2026; Published: 22 October 2026

ABSTRACT: Spectral clustering (SC) effectively captures non-convex and graph-structured cluster patterns, but its application to large-scale attributed and heterogeneous graphs is hindered by the high costs of similarity graph construction and Laplacian eigendecomposition, as well as by insufficient use of relation semantics. To address these challenges, we propose Relation-aware Adaptive Anchor Spectral Clustering (RAASC). RAASC first identifies relation-aware local components in relation-specific subgraphs and allocates anchor budgets according to component difficulty. It then constructs an adaptive, load-aware sample-anchor graph by jointly considering node uncertainty, relation compatibility, and anchor-load balance. Using the resulting sparse sample-anchor matrices, RAASC builds a compact relation-aware reduced anchor graph and performs spectral learning, anchor-level discretization, and label recovery in the anchor space. This design avoids full similarity graph construction and reduces the scale of eigendecomposition while preserving heterogeneous structural information. Experiments on DBLP, IMDB, and OGBN-MAG show that RAASC achieves the highest mean Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI) among the representative clustering and spectral clustering baselines. Efficiency and scalability results further demonstrate that RAASC maintains practical runtime and memory usage on large-scale heterogeneous graphs and achieves a 3.72-fold speedup using 16 workers.

KEYWORDS: Spectral clustering; anchor-based clustering; heterogeneous graphs; distributed clustering; large-scale graph mining; reduced graph learning

1 Introduction

Spectral clustering (SC) [1] is a classical graph-based clustering paradigm that converts data clustering into a graph partitioning problem. In this framework, data samples are represented as graph nodes, pairwise similarities are encoded as edge weights, and the eigenvectors of the graph Laplacian are used to derive a low-dimensional spectral embedding for partitioning. Compared with conventional methods such as k-means [2], which often rely on convex cluster assumptions in Euclidean space, spectral clustering is more capable of capturing non-convex structures, manifold patterns, and complex graph relationships, and has therefore been widely applied to image segmentation, community detection, text clustering, and bioinformatics [3]. However, real-world graph data are increasingly represented as large-scale attributed and heterogeneous graphs, such as citation graphs, academic networks, social networks, and knowledge graphs. In such graphs, node attributes provide semantic similarity, node types reflect entity-level heterogeneity, and

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

relation types encode rich structural semantics. Directly collapsing different relations into a single homogeneous graph may obscure relation-specific structures, while constructing a full $n \times n$ similarity matrix and performing Laplacian eigen-decomposition lead to prohibitive time and memory costs. Therefore, scalable spectral clustering on attributed and heterogeneous graphs requires not only reducing the computational cost of graph construction and eigen-decomposition, but also preserving relation-specific structural semantics during graph reduction.

To improve scalability, representative large-scale spectral clustering methods mainly reduce the original node-level graph through approximation. The Nyström method approximates the full affinity matrix by sampling a small subset of columns [4], while Large-scale Spectral Clustering (LSC) constructs a landmark-based sparse representation to avoid full similarity matrix construction [5]. Divide-and-conquer Spectral Clustering (DnC-SC) further improves landmark selection through a divide-and-conquer strategy and approximate nearest-landmark search [6], and Robust and Efficient Spectral k-Means (RESKM) provides a general acceleration framework for anchor-based spectral clustering [7]. These methods substantially reduce the cost of spectral decomposition, but their anchors or landmarks are usually selected by global sampling, geometric distances, or partition-based heuristics. As a result, relation types, node attributes, and relation-specific local connectivity patterns are not explicitly used to guide anchor generation and sample-anchor assignment. Moreover, most anchor-based methods connect each sample to a fixed number of anchors, which may introduce unreliable links around ambiguous boundaries or density-imbalanced regions. Parallel and distributed spectral clustering methods improve efficiency through parallel similarity construction, distributed eigensolvers, and communication reduction [8,9], but they mainly focus on computational parallelism rather than relation-aware reduced graph quality. Therefore, scalable spectral clustering for heterogeneous graphs should not only reduce graph size, but also preserve relation-aware local structures and adaptive sample-anchor connectivity.

Beyond scalable approximation methods, attributed multi-relational spectral clustering has also been studied. SpectralMix jointly embeds graph structures, relation types, and categorical node attributes into a unified low-dimensional space, showing that attributes and relation-specific structures should not be collapsed into a single homogeneous graph [10]. However, SpectralMix mainly focuses on joint embedding with categorical attributes and still relies on subsequent k-means clustering on the learned embeddings. It does not explicitly address adaptive anchor construction, load-aware sample-anchor assignment, reduced anchor-level spectral learning, or distributed scalability for large-scale heterogeneous graphs. Moreover, its dependence on the conventional "embedding + k-means" pipeline indicates that relation-aware embedding alone is insufficient to eliminate discretization loss or support efficient out-of-sample prediction. This issue is related to another line of work on direct and inductive spectral clustering. Methods such as Direct Spectral Clustering (DSC) [11], Unified One-step Multi-view Spectral Clustering (UOMvSC) [12], and Neural Normalized Cut (NeuNcut) [13] have been proposed to reduce the discretization loss of the conventional "embedding + k-means" pipeline or to improve out-of-sample prediction. Nevertheless, most existing one-step or inductive methods are not specifically designed for large-scale attributed and heterogeneous graphs. They often still depend on dense affinity graphs or centralized optimization, and insufficient attention has been paid to relation-aware anchor construction, reduced anchor-level graph learning, and communication constraints in distributed settings.

Another relevant line of research is deep graph clustering based on graph neural networks, graph autoencoders, and contrastive representation learning. A recent representative method is Multiscale Contrastive Variational Graph Autoencoder (MCVGAE) [14], which integrates a variational graph autoencoder with cluster-level and graph-level contrastive learning. By jointly refining node representations and clustering-oriented graph structures, MCVGAE alleviates feature randomness, feature drift, posterior

collapse, and the mismatch between inference and generative models, and achieves strong clustering performance on attributed graphs. However, MCVGAE is formulated for a node-level adjacency matrix and learns clusters through iterative encoder-decoder training. It does not explicitly preserve typed relations through relation-specific anchor construction, nor does it reduce heterogeneous graphs to a compact anchor-level spectral problem.

Despite these advances, existing methods do not jointly address relation-aware anchor construction, adaptive sample-anchor connectivity, compact spectral learning with direct discretization, and distributed or inductive inference at scale. To fill this gap, we propose Relation-aware Adaptive Anchor Spectral Clustering (RAASC) for large-scale attributed and heterogeneous graph clustering. The main contributions are summarized as follows:

- Relation-aware adaptive anchor construction. RAASC generates anchors from relation-aware local components and allocates anchor budgets according to component difficulty, improving the representation of dense, complex, and boundary-sensitive regions.
- Adaptive load-aware sample-anchor assignment. RAASC determines node-specific anchor connections based on local density, boundary risk, and assignment ambiguity, while using load control to reduce unreliable and imbalanced assignments.
- Relation-aware reduced spectral learning. RAASC constructs a compact reduced anchor graph from relation-specific sample-anchor matrices and performs spectral learning in the anchor space, avoiding full similarity graph construction and large-scale Laplacian eigen-decomposition.
- Anchor-level discretization and inductive label recovery. RAASC learns discrete labels on the anchor graph and propagates them to original nodes through the sparse sample-anchor matrix; new nodes can be labeled by anchor assignment without recomputing spectral decomposition.

The remainder of this paper is organized as follows. Section 2 reviews spectral clustering and anchor-based spectral approximation. Section 3 presents the proposed RAASC framework and its complexity analysis. Section 4 evaluates its clustering effectiveness, efficiency, scalability, robustness, and inductive capability. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Spectral Clustering

Spectral clustering formulates clustering as a graph partitioning problem. Given a target node set $V^* = \{v_i\}_{i=1}^n$, a similarity graph is first constructed, where each node corresponds to one data object and the edge weight measures pairwise similarity. Let $W \in \mathbb{R}^{n\times n}$ denote the similarity matrix, where W_{ij} represents the similarity between v_i and v_j. The corresponding degree matrix is defined as $D = \mathrm{diag}\left(W\mathbf{1}_n\right)$, where $\mathbf{1}_n$ is an n-dimensional all-one vector. The normalized graph Laplacian is given by $L = I_n - D^{-1/2}WD^{-1/2}$ where I_n is the $n \times n$ identity matrix. Spectral clustering obtains a low-dimensional embedding by solving the following trace minimization problem $\min_{U^TU=I_k} Tr\left(U^TLU\right)$, where $U \in \mathbb{R}^{n\times k}$ is the spectral embedding matrix, and k is the number of clusters. The optimal U is given by the k eigenvectors associated with the smallest eigenvalues of L. After obtaining U, conventional spectral clustering usually applies k-means to the rows of U to obtain the final discrete clustering labels.

2.2 Anchor-Based Spectral Approximation

Anchor-based spectral approximation is commonly used to improve the scalability of spectral clustering. Instead of constructing the full similarity matrix $W \in \mathbb{R}^{n\times n}$, a small set of anchors is selected to represent the original target nodes. Let $A = \{a_1, a_2, \ldots, a_m\}$, $m \ll n$ denote the anchor set, where m is the number

of anchors. Each target node $v_i \in V^*$ is connected only to a small number of anchors, producing a sparse sample-anchor matrix $B \in \mathbb{R}^{n\times m}$ where B_{ij} measures the assignment weight between node v_i and anchor a_j. Usually, $B_{ij} = 0$ if a_j is not selected as one of the neighboring anchors of v_i The row-wise normalized assignment satisfies $\sum_{j=1}^{m} B_{ij} = 1$. Based on B, the original node-level similarity graph can be approximated by a much smaller anchor-level graph. A common reduced anchor graph can be written as $S_A = B^T D_B^{-1} B$ where $D_B = \mathrm{diag}\left(B\mathbf{1}_m\right) + \varepsilon I_n$ is the node-side degree matrix, and ε is a small positive constant used to avoid zero diagonal entries. The degree matrix of the reduced anchor graph is $D_A = \mathrm{diag}\left(S_A\mathbf{1}_m\right)$ and the corresponding normalized anchor-level Laplacian is $L_A = I_m - D_A^{-1/2} S_A D_A^{-1/2}$. Since $L_A \in \mathbb{R}^{m\times m}$ and $m \ll n$, spectral decomposition can be performed on the reduced anchor graph rather than on the full node-level graph. In this way, the main computational burden is shifted from the full similarity matrix W to the sparse sample-anchor matrix B and the compact anchor graph S_A.

3 Proposed Method

3.1 Overview of RAASC

RAASC extends anchor-based spectral approximation to large-scale attributed heterogeneous graphs by integrating relation-aware anchor construction, adaptive sample-anchor assignment, reduced spectral learning, and inductive label recovery. Fig. 1 illustrates the data flow and intermediate representations of RAASC, whereas Algorithm 1 summarizes the corresponding computational procedure. Given an attributed heterogeneous graph, relation-aware local components are first identified to guide anchor generation. Different from uniform anchor allocation strategies, the anchor budget is adaptively determined according to the structural difficulty of local components, and representative anchors are selected by jointly considering local density, boundary characteristics, and coverage information. Based on the obtained anchors, an adaptive load-aware sample-anchor graph is constructed, where the number of connected anchors is adjusted according to node uncertainty and the assignment process is regularized by relation compatibility and anchor load balance. Finally, a compact relation-aware reduced anchor graph is constructed from sparse sample-anchor matrices, and spectral learning is performed in the anchor space rather than the original node space. The learned anchor-level labels are subsequently propagated to original nodes, enabling efficient clustering and inductive inference for newly arriving nodes. Given an attributed heterogeneous graph $G = (V, E, X, \mathcal{R})$, RAASC aims to perform scalable clustering on the target node set $V^* \subseteq V$. Algorithm 1 summarizes the complete procedure.

Algorithm 1: Training and clustering procedure of RAASC

Input: Attributed heterogeneous graph $G = (V, E, X, \varphi, \psi)$, target node set V^*, number of clusters k, global anchor budget m.

Output: Cluster labels $\hat{y}$ for target nodes.

Construct relation-specific subgraphs $\{G_r\}_{r\in\mathcal{R}}$
Identify relation-aware local components $\{C_r\}_{r\in\mathcal{R}}$ in each G_r
Calculate component difficulty scores and allocate anchor budgets m_C for retained components
Select relation-aware anchors A from each retained component using density, boundary, and coverage-aware scores
For each target node v_i:
    Construct candidate anchor set Ω_i
    Estimate node uncertainty U_i and adaptive neighborhood size K_i
    Compute relation-compatible scores and update load-aware scores
    Select top-K_i anchors and construct sample-anchor matrix B

(Continued)

Algorithm 1 (continued)

End for
Construct relation-specific matrices $\{B_r\}_{r\in\mathcal{R}}$ according to relation support scores
Build the relation-aware reduced anchor graph S_A and Laplacian L_A
Perform distributed spectral decomposition on L_A to obtain U_A
Generate anchor labels Y_A using anchor-level discretization
Recover node-level soft labels: P = BY_A
Assign final cluster labels: $\hat{y}_i = argmax_c\, P_{ic}$
Return $\hat{y}$

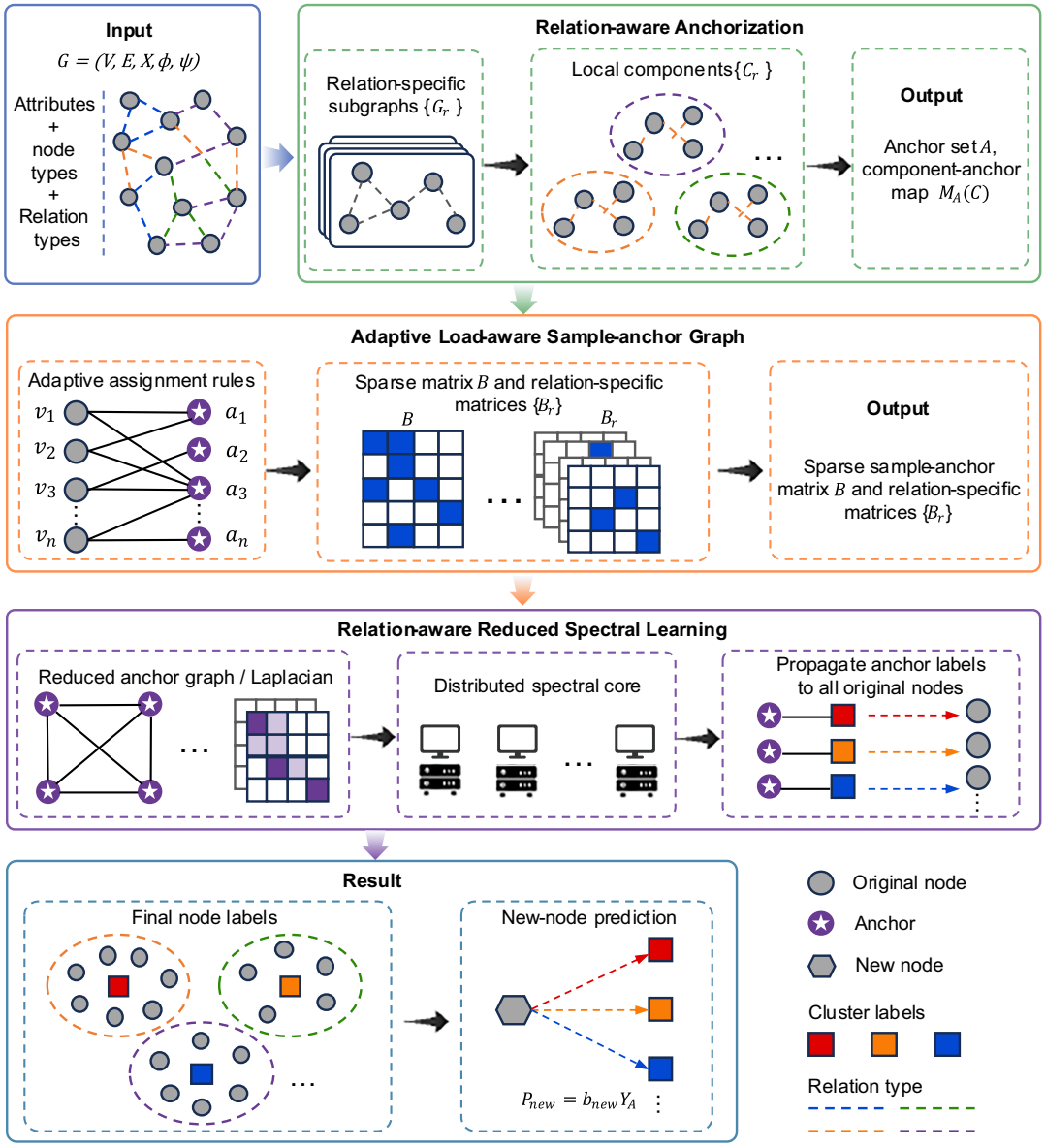


Figure 1: Overall workflow of RAASC.

3.2 Relation-Aware Anchorization

Relation-aware anchorization aims to generate a compact yet structure-preserving anchor set from the input attributed or heterogeneous graph. Instead of treating the input graph as a single homogeneous structure, RAASC identifies relation-aware local components in relation-specific subgraphs and uses them as basic units for anchor allocation and anchor candidate selection. The anchors are subsequently generated by considering the structural characteristics of these components, including their representation difficulty, internal variation, and boundary complexity.

3.2.1 Relation-Aware Local Components

For each relation type r, a relation-specific subgraph is constructed by retaining only edges associated with r. Formally, the relation-specific subgraph is defined as $G_r = (V_r, E_r), \quad E_r = \{(u, v) \in E \mid \psi(u, v) = r\}$ where V_r denotes the nodes involved in relation type r. Since a node may participate in multiple relation types, it can appear in multiple relation-specific subgraphs. For each relation type $r \in \mathcal{R}$, the weakly connected components of G_r are denoted as $\mathcal{C}_r = \{C_{r,1}, C_{r,2}, \ldots, C_{r,s_r}\}$, where s_r is the number of local components under relation type r. The complete component set used for anchorization is then given by $\mathcal{C} = \bigcup_{r \in \mathcal{R}} \mathcal{C}_r$.

Definition 1: *(Relation-aware Local Component) Given a relation-specific subgraph G_r, a relation-aware local component $C_{r,j}$ is defined as a weakly connected component in G_r. For any two nodes $u, v \in C_{r,j}$, there exists a weakly connected path between them using only edges of relation type r. This condition can be written as $u, v \in C_{r,j}, \exists \rho_r(u, v) \subseteq E_r$,where $\rho_r(u, v)$ denotes a weakly connected path between u and v in the relation-specific subgraph G_r.*

It should be noted that relation-aware local components are not regarded as final clusters. Instead, they serve as structural units for anchor budget allocation and anchor candidate generation. A node may participate in multiple relation-specific components if it is involved in multiple relation types. Therefore, for each node v, its component membership set is defined as $\Gamma(v) = \{C \in \mathcal{C} \mid v \in C\}$. When clustering is performed on a target node type, only target-type nodes are considered as anchor candidates, while auxiliary-type nodes are retained for preserving relation-aware connectivity. Let $V^* \subseteq V$ denote the target node set. The anchor candidate set associated with component C is defined as $\mathcal{A}(C) = C \cap V^*$. If all node types are involved in clustering, then $V^* = V$ and $\mathcal{A}(C) = C$.

3.2.2 Component Difficulty and Anchor Budget

After relation-aware local components have been obtained, the global anchor budget is allocated according to the difficulty of each component. Uniform allocation may assign excessive anchors to simple regions while insufficiently representing complex regions. Therefore, the difficulty of each component is evaluated from three aspects: component size, internal structural variation, and boundary complexity.

Let $\mathcal{C}$ be the set of relation-aware local components defined in Section 3.2.1. Since some components may contain no target nodes or may be too small to provide reliable anchor candidates, RAASC first constructs a retained component set $\mathcal{C}^+ = \{C \in \mathcal{C} \mid \mathcal{A}(C) \neq \varnothing, |\mathcal{A}(C)| \geq n_{\min}\}$, where $\mathcal{A}(C)$ denotes the anchor candidate set in component $\mathcal{C}$, and $n_{\min}$ is the minimum candidate size threshold. Components excluded by the threshold $n_{\min}$ do not receive separate anchor budgets. Their nodes and edges are nevertheless retained in the relation-specific subgraphs and may obtain candidate anchors from adjacent retained components during sample-anchor assignment. For each retained component $C \in \mathcal{C}^+$, the number of candidate nodes is defined as $n(C) = |\mathcal{A}(C)|$. The normalized component size is defined as $\hat{n}(C) = \frac{n(C)}{\sum_{C' \in \mathcal{C}^+} n(C')}$. Since component difficulty is determined by both attribute diversity and relation-specific

structural diversity, node representations combining semantic and structural information are constructed. Specifically, the structural sketch of node v_i is defined as the logarithmic degree profile over different relation types $q_i = \left[\log\left(1 + d_i^{(r_1)}\right), \ldots, \log\left(1 + d_i^{(r_{|\mathcal{R}|})}\right)\right]^T$ where $d_i^{(r)}$ denotes the degree of node v_i under relation type r. The logarithmic transformation is adopted to reduce the influence of high-degree nodes. The structural sketch preserves relation-specific connectivity patterns and avoids collapsing different relations into a single degree value. The attribute vector and structural sketch are normalized separately, and the final representation is constructed as $z_i = \left[\overline{x}_i; \overline{q}_i\right]$ where $\overline{x}_i$ and $\overline{q}_i$ denote the normalized attribute and structural representations, respectively. Based on the node representation z_i, the internal variation of component C is measured by the average distance between nodes and the component centroid $\mu_C = \frac{1}{n(C)} \sum_{v_i \in C} z_i$, $Var(C) = \frac{1}{n(C)} \sum_{v_i \in C} \|z_i - \mu_C\|_2^2$. A larger variation indicates that more anchors are required to represent diverse structures within the component. The normalized internal variation is

$$\widehat{\mathrm{Var}}(C) = \frac{\mathrm{Var}(C)}{\sum_{C' \in \mathcal{C}^+} \mathrm{Var}\,(C') + \varepsilon} \tag{1}$$

Besides internal variation, components with complicated boundaries require additional anchors to preserve transition structures. Let $E_{\mathrm{in}}(C)$ and $E_{\mathrm{out}}(C)$ denote internal edges and boundary-crossing edges, respectively. The boundary complexity is defined as $\mathrm{Boundary}(C) = \frac{|E_{\mathrm{out}}(C)|}{|E_{\mathrm{in}}(C)| + |E_{\mathrm{out}}(C)| + \epsilon}$. A higher boundary complexity indicates stronger interactions between the component and surrounding regions. Its normalized form is

$$\widehat{\mathrm{Boundary}}(C) = \frac{\mathrm{Boundary}(C)}{\sum_{C' \in \mathcal{C}^+} \mathrm{Boundary}\,(C') + \epsilon} \tag{2}$$

The three factors are normalized and integrated to obtain the component difficulty score $\Delta(C) = \alpha_n \hat{n}(C) + \alpha_v \widehat{\mathrm{Var}}(C) + \alpha_b \widehat{\mathrm{Boundary}}(C) + \varepsilon$ where $\alpha_n, \alpha_v, \alpha_b \geq 0$ and $\alpha_n + \alpha_v + \alpha_b = 1$. A larger $\Delta(C)$ indicates that the component is more difficult to represent and therefore requires more anchors. Based on the difficulty score, the initial anchor budget assigned to component C is calculated as

$$m_C^{(0)} = \max\left(1, \left\lfloor m \frac{\Delta(C)}{\sum_{C' \in \mathcal{C}^+} \Delta\,(C')} \right\rfloor\right) \tag{3}$$

Since the floor operation may cause the total number of allocated anchors to deviate from the predefined budget m, a budget correction procedure is applied to ensure that the final allocation satisfies $\sum_{C \in \mathcal{C}^+} m_C = m$. If the total budget is smaller than m, the remaining anchors are assigned to components with the largest fractional residuals of their continuous budgets. If the total exceeds m, anchors are removed from components with budgets larger than one, starting from those with smaller difficulty scores. This procedure continues until the corrected budgets m_C sum to m. The feasibility of the corrected anchor allocation is theoretically guaranteed, and the proof is provided in Lemma 1.

Lemma 1: *(Anchor Budget Feasibility) Given the retained component set $\mathcal{C}^+$ and the total anchor budget m, if $m \geq |\mathcal{C}^+|$, then the budget correction step produces a feasible allocation $\{m_C \mid C \in \mathcal{C}^+\}$ satisfying $\sum_{C \in \mathcal{C}^+} m_C = m, \quad m_C \geq 1$.*

Proof: The initial allocation guarantees $m_C^{(0)} \geq 1$ for each retained component due to the lower-bound operation. If the total initial budget is smaller than m, the correction step only increases the budgets of selected components, and therefore the lower-bound constraint is preserved. If the total initial budget is larger than m, anchors are removed only from components whose current budget is larger than one,

so no component budget can become smaller than one. Since $m \geq |\mathcal{C}^+|$, at least one feasible allocation exists in which each retained component receives one anchor. By repeatedly adding or removing anchors until the total budget equals m, the corrected allocation satisfies both the total budget constraint and the component-wise lower-bound constraint. □

3.2.3 Core-Boundary-Coverage Anchor Selection

A single criterion is insufficient for anchor selection in heterogeneous graphs because dense regions, ambiguous boundaries, and sparsely represented regions require different types of representatives. To obtain a more balanced representation, a core-boundary-coverage strategy is adopted. Specifically, core anchors are used to represent dense and stable local regions, boundary anchors are used to preserve ambiguous transition areas between components, and coverage anchors are used to prevent sparse or under-represented regions from being ignored.

Local density is used to identify representative nodes in stable internal regions. To estimate the local density, the pairwise similarity between neighboring nodes is first calculated based on their representation distance. For each candidate node $v_i \in \mathcal{A}(C)$, its neighborhood is constructed by retrieving the κ-nearest target nodes in the representation space z_i. The similarity between neighboring nodes is measured by a Gaussian kernel $w_{ij} = \exp\left(-\frac{|z_i - z_j|_2^2}{2\sigma_z^2}\right)$ where σ_z is determined according to the median distance among local neighbors. The local density of v_i within component C is then calculated as $\mathrm{Den}_C(v_i) = \sum_{v_j \in N_\kappa(v_i) \cap \mathcal{A}(C)} w_{ij}$. To make the density term comparable across nodes in the same component, it is normalized as $\widehat{\mathrm{Den}}_C(v_i) = \frac{\mathrm{Den}_C(v_i)}{\max_{v_l \in \mathcal{A}(C)} \mathrm{Den}_C(v_l) + \epsilon}$. Nodes with higher density are more likely to represent stable local structures and therefore receive higher priority as core anchors.

Density-based selection alone may overlook transition regions between different components. To preserve ambiguous structures, boundary risk is introduced to measure whether a node is close to relation-aware component boundaries. It is defined as $\mathrm{BR}_C(v_i) = \frac{\sum_{v_j \in N_\kappa(v_i)} \mathbf{1}(v_j \notin C) w_{ij}}{\sum_{v_j \in N_\kappa(v_i)} w_{ij} + \epsilon}$. A larger boundary risk indicates that the node has stronger interactions with neighboring components and is therefore valuable for preserving structural transitions.

In addition to core and boundary information, coverage is considered to avoid selecting anchors that are concentrated in limited regions. During the greedy selection process, candidates that are far from already selected anchors are preferred. Let $A_C^{(t)}$ denote the anchor set already selected from component C after t iterations, $A_C^{(t)} \neq \varnothing$, the coverage score of candidate node v_i is defined as

$$Cov_C\left(v_i; A_C^{(t)}\right) = \frac{\min_{a_l \in A_C^{(t)}} \|z_i - z_l\|_2}{\max_{v_q \in A(C)} \min_{a_l \in A_C^{(t)}} \|z_q - z_l\|_2 + \epsilon} \tag{4}$$

where z_i and z_l are the node representations of v_i and anchor a_l, respectively. This term assigns a higher score to candidates that are far from the already selected anchors, thereby improving the coverage of the anchor set. If no anchor has been selected yet, the coverage score is initialized by the distance to the component centroid

$$\mathrm{Cov}_C(v_i; \varnothing) = \frac{\|z_i - \mu_C\|_2}{\max_{v_q \in \mathcal{A}(C)} \|z_q - \mu_C\|_2 + \epsilon} \tag{5}$$

where μ_C is the centroid of component C, as defined in Section 3.2.2.

The final anchor selection score, combining the above three terms, is

$$\mathrm{Score}_C\left(v_i; A_C^{(t)}\right) = \beta_{\mathrm{den}}\widehat{\mathrm{Den}}_C\left(v_i\right) + \beta_{\mathrm{bd}}\mathrm{BR}_C\left(v_i\right) + \beta_{\mathrm{cov}}\mathrm{Cov}_C\left(v_i; A_C^{(t)}\right) \tag{6}$$

where the balancing coefficients satisfy $\beta_{\mathrm{den}} + \beta_{\mathrm{bd}} + \beta_{\mathrm{cov}} = 1, \beta_{\mathrm{den}}, \beta_{\mathrm{bd}}, \beta_{\mathrm{cov}} \geq 0$. At each iteration, the candidate with the highest score is selected as the next anchor according to $a_C^{(t+1)} = argmax_{v_i \in \mathcal{A}(C)\setminus A_C^{(t)}} \mathrm{Score}_C\left(v_i; A_C^{(t)}\right)$. The component-level anchor set is then updated as $A_C^{(t+1)} = A_C^{(t)} \cup \left\{a_C^{(t+1)}\right\}$. This greedy selection process is repeated until $|A_C| = m_C$. After the process terminates, the selected anchor set of component C is denoted by A_C, and the component-anchor mapping is recorded as $\mathcal{M}_A(C) = A_C$. The final anchor set generated from all retained components is $A = \bigcup_{C \in \mathcal{C}^+} \mathcal{M}_A(C)$. If the same node is selected by multiple relation-aware components, RAASC keeps it as a single anchor and records its multiple component memberships. If duplicate removal causes the total number of anchors to be smaller than the global budget m, additional anchors are selected from the remaining candidates with the largest component-level scores.

3.3 Adaptive Load-Aware Sample-Anchor Graph

Nodes located in dense and stable local regions can usually be represented by only a few nearby anchors, whereas nodes located near relation-aware boundaries or in low-density regions may require more anchors to avoid unreliable assignment. Therefore, RAASC constructs an adaptive load-aware sample-anchor graph, where the number of connected anchors is determined according to node uncertainty and the assignment process is further regularized by relation compatibility and anchor load.

3.3.1 Adaptive Neighborhood Sizes

Since anchors have been selected according to relation-aware local components, nodes are first connected to anchors belonging to their associated components. For node v_i, the preliminary candidate anchor set is obtained from anchors selected in its associated relation-aware components, as

$$\Omega_i^{(0)} = \bigcup_{C \in \Gamma(v_i)} \mathcal{M}_A(C) \tag{7}$$

where $\Gamma\left(v_i\right)$ is the component membership set of v_i, and $\mathcal{M}_A(C)$ denotes the anchors selected from component C. Therefore, adjacent components are identified according to the overlap between component neighborhoods and the local κ-nearest neighbors of v_i. The adjacent component set of node v_i is defined as $\left\{C' \in \mathcal{C}^+ \setminus \Gamma\left(v_i\right) | C' \cap N_\kappa\left(v_i\right) \neq \varnothing\right\}$. Thus, a retained component is adjacent to v_i if it contains at least one node in the κ-nearest-neighbor set of v_i. The boundary-anchor candidate set is constructed as $\mathrm{KNN}\, K \max\left(v_i; \left(\bigcup_{C' \in \mathrm{Adj}(v_i)} \mathcal{M}_A\left(C'\right)\right) \setminus \Omega_i^{(0)}\right)$ where $\left(\mathrm{KNN}\, K \max\left(v_i, \cdot\right)\right)$ retrieves at most $K_{\max}$ nearest anchors according to Euclidean distance in the node-representation space. The complete candidate anchor set is then $\Omega_i^{(0)} \cup \Omega_i^{\mathrm{bd}}$. If $\mathrm{Adj}\left(v_i\right) = \varnothing$, then $\Omega_i^{\mathrm{bd}} = \varnothing$. If $|\Omega_i| < K_{\min}$, the candidate set is supplemented with globally nearest anchors from $A \setminus \Omega_i$ until at least $K_{\min}$ candidates are available.

Since nodes with different structural characteristics require different numbers of anchors, the assignment uncertainty of v_i is estimated by local density, boundary risk, and anchor ambiguity. The local density is defined as $\rho_i = \max_{C \in \Gamma(v_i)} \widehat{\mathrm{Den}}_C\left(v_i\right)$, where $\widehat{\mathrm{Den}}_C\left(v_i\right)$ is the normalized component-wise density defined in Section 3.2.3. The boundary risk is $\mathrm{BR}\left(v_i\right) = \max_{C \in \Gamma(v_i)} \mathrm{BR}_C\left(v_i\right)$. Let $d_i^{(1)}$ and $d_i^{(2)}$ be the smallest and second smallest distances between v_i and anchors in $\Omega_i^{(0)}$, respectively. The anchor ambiguity is defined as

$\mathrm{AA}(v_i) = \mathrm{clip}\left(1 - \frac{d_i^{(2)} - d_i^{(1)}}{d_i^{(2)} + \epsilon}, 0, 1\right)$. A larger ambiguity value indicates that the nearest anchors have similar distances to v_i, making the assignment less certain.

Definition 2: *(Adaptive Neighborhood Size) Given the* ρ_i, $\mathrm{BR}(v_i)$, *and* $\mathrm{AA}(v_i)$, *the uncertainty score of node* v_i *is defined as* $U_i = \eta_\rho(1-\rho_i) + \eta_b \mathrm{BR}(v_i) + \eta_a \mathrm{AA}(v_i)$, *where* $\eta_\rho + \eta_b + \eta_a = 1, \eta_\rho, \eta_b, \eta_a \geq 0$. *The preliminary adaptive neighborhood size is denoted by* $\overline{K}_i = \mathrm{clip}\left(K_{\min} + \lfloor (K_{\max} - K_{\min}) U_i \rfloor, K_{\min}, K_{\max}\right)$. *Considering the number of available candidate anchors, the neighborhood size actually used by RAASC is* $K_i = \min\left\{\overline{K}_i, |\Omega_i|\right\}$. *Here,* $K_{\min}$ *and* $K_{\max}$ *are the minimum and maximum numbers of anchors allowed for each node. Thus, dense internal nodes with low uncertainty tend to connect to fewer anchors, while boundary or ambiguous nodes are assigned more anchors to reduce unreliable hard assignments.*

Proposition 1: *(Sparsity Bound) For the sample-anchor matrix* B *constructed by RAASC, if each node* v_i *is connected to* K_i *anchors and* $K_{\min} \leq K_i \leq K_{\max}$, *then* $nK_{\min} \leq \mathrm{nnz}(B) = \sum_{i=1}^{n} K_i \leq nK_{\max}$.

Proof: Each node v_i contributes exactly K_i nonzero entries to the sample-anchor matrix B. Since $K_{\min} \leq K_i \leq K_{\max}$ holds for every node, summing over all n nodes gives $\sum_{i=1}^{n} K_{\min} \leq \sum_{i=1}^{n} K_i \leq \sum_{i=1}^{n} K_{\max}$, which is equivalent to $nK_{\min} \leq \mathrm{nnz}(B) \leq nK_{\max}$. Thus, the number of nonzero entries in B is linearly bounded by the number of nodes, and the constructed sample-anchor graph remains sparse. □

3.3.2 Relation-Compatible Anchor Scoring

For each candidate anchor $a_j \in \Omega_i$, RAASC computes a relation-compatible score by jointly considering attribute similarity, structural similarity, relation compatibility, and cross-component reliability. The attribute similarity is defined as

$$\mathrm{sim}_f(v_i, a_j) = \exp\left(-\frac{\|x_i - x_{a_j}\|_2^2}{\sigma_f^2}\right) \tag{8}$$

where x_i and x_{a_j} denote the attribute vectors of node v_i and anchor a_j, respectively. Suppose q_i and q_{a_j} are the relation-wise logarithmic degree profiles defined in Section 3.2.2, then structural similarity can be obtained from

$$\mathrm{sim}_s(v_i, a_j) = \exp\left(-\frac{\|q_i - q_{a_j}\|_2^2}{\sigma_s^2}\right) \tag{9}$$

Relation-aware compatibility is measured by the overlap between the relation-aware local-component membership sets of node v_i and anchor a_j, as follows:

$$\mathrm{sim}_r(v_i, a_j) = \frac{|\Gamma(v_i) \cap \Gamma(a_j)|}{|\Gamma(v_i) \cup \Gamma(a_j)| + \epsilon} \tag{10}$$

A larger $\mathrm{sim}_r(v_i, a_j)$ indicates that v_i and a_j are supported by more shared relation-aware local components. In addition, a cross-component penalty, as shown in (11), is introduced to prevent internal nodes from making unreliable assignments to anchors outside their own components.

$$\mathrm{CP}(v_i, a_j) = (1 - \mathrm{BR}(v_i)) \cdot \mathbf{1}\left(\Gamma(v_i) \cap \Gamma(a_j) = \varnothing\right) \tag{11}$$

here, $\mathrm{BR}(v_i)$ is the boundary risk. Cross-component allocation is heavily penalized when v_i is an internal node with a low boundary risk. Conversely, the penalty is reduced when v_i is close to a relation-aware boundary, so that necessary cross-component anchors can still be taken into account.

Based on these terms, the relation-compatible score between node v_i and anchor a_j is defined as

$$s_{ij} = \lambda_f \mathrm{sim}_f\left(v_i, a_j\right) + \lambda_s \mathrm{sim}_s\left(v_i, a_j\right) + \lambda_r \mathrm{sim}_r\left(v_i, a_j\right) - \lambda_c \mathrm{CP}\left(v_i, a_j\right) \tag{12}$$

where $\lambda_f, \lambda_s, \lambda_r, \lambda_c \geq 0$ are balancing parameters. The four terms are assigned equal importance in the implementation $\lambda_f = \lambda_s = \lambda_r = \lambda_c = 0.25$. Thus, s_{ij} jointly measures attribute similarity, structural similarity, relation compatibility, and cross-component reliability without using ground-truth labels. However, using s_{ij} alone may over-assign nodes to a few high-density anchors. Therefore, RAASC further introduces a load-aware penalty to balance anchor coverage during assignment.

Target nodes are processed in ascending node-ID order. After node v_i is assigned, the loads of its selected anchors are updated immediately. If multiple anchors have identical load-aware scores, the anchor with the smaller ID is selected first. Let $\mathcal{V}^{(t)}$ denote the set of nodes that have already been processed before the current assignment step t. The current load of anchor a_j is defined as $\mathrm{load}^{(t)}\left(a_j\right) = \sum_{v_i \in \mathcal{V}^{(t)}} \mathbf{1}\left(a_j \in A_i\right)$, where A_i is the selected anchor set of node v_i, and $\mathbf{1}(.)$ is the indicator function. The average anchor load at step t is computed by $\overline{\mathrm{load}}^{(t)} = \frac{1}{m}\sum_{j=1}^{m} \mathrm{load}^{(t)}\left(a_j\right)$, where m is the total number of anchors. Based on the current load, the load penalty of anchor a_j is defined as $\mathrm{LP}^{(t)}\left(a_j\right) = \max\left(0, \frac{\mathrm{load}^{(t)}\left(a_j\right)}{\overline{\mathrm{load}}^{(t)} + \epsilon} - 1\right)$, where ε is a small positive constant used to avoid division by zero. If the load of a_j is not larger than the average load, then $\mathrm{LP}^{(t)}\left(a_j\right) = 0$, which means that no penalty is imposed. If a_j is overloaded, then $\mathrm{LP}^{(t)}\left(a_j\right) > 0$, and the penalty increases with its relative overload degree. The load-aware score is then adjusted as $\tilde{s}_{ij}^{(t)} = s_{ij} - \mu \mathrm{LP}^{(t)}\left(a_j\right)$, where $\mu \geq 0$ controls the strength of load regularization. A larger μ encourages more balanced anchor assignments, while a smaller μ makes the assignment rely more on the original relation-compatible score s_{ij}.

Lemma 2: *(Load-aware Preference) For two candidate anchors a_j and a_l, if they have the same relation-compatible score for node v_i, i.e., $s_{ij} = s_{il}$ and their current loads satisfy $\mathrm{load}^{(t)}\left(a_j\right) < \mathrm{load}^{(t)}\left(a_l\right)$, then the load-aware scores satisfy $\tilde{s}_{ij}^{(t)} \geq \tilde{s}_{il}^{(t)}$ provided that the load penalty is monotonically nondecreasing with respect to anchor load.*

Proof: According to the definition of $\mathrm{LP}^{(t)}\left(a_j\right)$, the load penalty is a nondecreasing function of $\mathrm{load}^{(t)}\left(a_j\right)$. Therefore, $\mathrm{load}^{(t)}\left(a_j\right) < \mathrm{load}^{(t)}\left(a_l\right)$ implies $\mathrm{LP}^{(t)}\left(a_j\right) \leq \mathrm{LP}^{(t)}\left(a_l\right)$. By the definition of the load-aware score, we have $\tilde{s}_{ij}^{(t)} = s_{ij} - \mu \mathrm{LP}^{(t)}\left(a_j\right)$ and $\tilde{s}_{il}^{(t)} = s_{il} - \mu \mathrm{LP}^{(t)}\left(a_l\right)$. Since $s_{ij} = s_{il}$, the difference between the two load-aware scores is $\tilde{s}_{ij}^{(t)} - \tilde{s}_{il}^{(t)} = \mu\left(\mathrm{LP}^{(t)}\left(a_l\right) - \mathrm{LP}^{(t)}\left(a_j\right)\right)$. Because $\mu \geq 0$ and $\mathrm{LP}^{(t)}\left(a_l\right) \geq \mathrm{LP}^{(t)}\left(a_j\right)$, it follows that $\tilde{s}_{ij}^{(t)} - \tilde{s}_{il}^{(t)} \geq 0$. Thus, $\tilde{s}_{ij}^{(t)} \geq \tilde{s}_{il}^{(t)}$. □

3.3.3 Sparse Sample-Anchor Matrix

For each node v_i, the top-K_i anchors $A_i = \mathrm{TopK}_{K_i}\left(\left\{\tilde{s}_{ij}^{(t)} \mid a_j \in \Omega_i\right\}\right)$ are selected from its candidate anchor set Ω_i according to the load-aware scores, where $A_i \subseteq A$ and $|A_i| = K_i$. Let σ_i be the local scaling parameter for node v_i. The sample-anchor weight B_{ij} is computed by a normalized exponential function as

$$B_{ij} = \begin{cases} \dfrac{\exp\left(\tilde{s}_{ij}^{(t)}/\sigma_i\right)}{\sum_{a_l \in A_i} \exp\left(\tilde{s}_{il}^{(t)}/\sigma_i\right)}, & a_j \in A_i, \\ 0, & a_j \notin A_i. \end{cases} \tag{13}$$

where $\sigma_i > 0$ is a fixed temperature parameter controlling the sharpness of the soft assignment. Since each node is only connected to K_i anchors, B is a sparse matrix with row-wise normalized weights. Therefore, for each node v_i, the following condition holds $\sum_{j=1}^{m} B_{ij} = 1$. This means that each row of B can be interpreted as a soft assignment distribution from one node to its selected anchors. Compared with hard assignment, this soft weighting strategy preserves the relative importance of multiple anchors, which is especially useful for nodes located near relation-aware boundaries.

For relation-aware reduced graph construction in Section 3.4, relation-specific sample-anchor matrices can be derived from B. Let $\{C \in \mathcal{C}_r \mid v \in C\}$ denote the component membership set of node v under relation type r. For each selected node-anchor connection v_i and a_j, its support under relation type r is measured by $\frac{|\Gamma_r(v_i) \cap \Gamma_r(a_j)|}{|\Gamma_r(v_i) \cup \Gamma_r(a_j)| + \varepsilon}$. The relation type associated with the connection is then determined as $\underset{r \in \mathcal{R}}{argmax}\, h_{ij}^{(r)}$. If multiple relation types obtain the same maximum support score, the relation type with the smaller relation ID is selected. The entries of the relation-specific sample-anchor matrix B_r are consequently defined as

$$B_r = \begin{cases} B_{ij}, & r = r_{ij}^{*}, \\ 0, & otherwise. \end{cases} \tag{14}$$

Under this assignment rule, each nonzero sample-anchor connection is associated with exactly one relation type, and the following consistency condition holds $\sum_{r \in \mathcal{R}} B_r = B$. The resulting sparse matrix B is the final output of the adaptive load-aware assignment stage and is used for node-level label recovery. The relation-specific matrices $\{B_r\}_{r \in \mathcal{R}}$ are subsequently used to construct the relation-aware reduced anchor graph.

3.4 Relation-Aware Reduced Spectral Learning

3.4.1 Relation-Aware Reduced Spectral Core

RAASC builds a reduced graph over anchors rather than directly performing spectral learning on the original node-level graph. For each relation type $r \in \mathcal{R}$, let $B_r \in \mathbb{R}^{n \times m}$ denote the relation-specific sample-anchor matrix defined in Section 3.3.3. The corresponding node-side degree matrix is defined as $D_r = \mathrm{diag}\left(B_r \mathbf{1}_m\right) + \epsilon I$, where $\mathbf{1}_m$ is an m-dimensional all-one vector, I is the identity matrix, and ε is a small positive constant used to avoid zero diagonal entries. The relation-specific anchor graph is computed by $S_A^{(r)} = B_r^T D_r^{-1} B_r$. Then, all relation-specific anchor graphs are fused into a relation-aware reduced anchor graph $S_A = \sum_{r \in \mathcal{R}} \omega_r B_r^T D_r^{-1} B_r = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} B_r^T D_r^{-1} B_r$ where $\omega_r \geq 0$ is the weight of relation type r. Because no prior preference among relation types is assumed, the relation-aware reduced anchor graph is the average of all relation-specific anchor graphs. The degree matrix of the reduced anchor graph is defined as $D_A = \mathrm{diag}\left(S_A \mathbf{1}_m\right)$, and the normalized anchor-level Laplacian is given by $L_A = I - D_A^{-1/2} S_A D_A^{-1/2}$, where $L_A \in \mathbb{R}^{m \times m}$.

Lemma 3: *(Positive Semidefiniteness) If S_A is symmetric and nonnegative, then the normalized anchor Laplacian L_A is positive semidefinite.*

Proof: For any vector $f \in \mathbb{R}^m$, let $g = D_A^{-1/2} f$. Then, $f^T L_A f = f^T \left(I - D_A^{-1/2} S_A D_A^{-1/2}\right) f$, which can be rewritten as $f^T L_A f = g^T \left(D_A - S_A\right) g$. Since S_A is symmetric and nonnegative, the quadratic form of the graph Laplacian satisfies $g^T \left(D_A - S_A\right) g = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} S_{A,ij} \left(g_i - g_j\right)^2 \geq 0$. Therefore, $f^T L_A f \geq 0$, for any $f \in \mathbb{R}^m$, which proves that L_A is positive semidefinite. □

The spectral embedding of anchors is obtained by solving the following constrained trace minimization problem $\min_{U_A^T U_A = I_k} \mathrm{Tr}\left(U_A^T L_A U_A\right)$, where $U_A \in \mathbb{R}^{m \times k}$ denotes the anchor-level spectral embedding, and

k is the number of clusters. Equivalently, U_A can be obtained from the k smallest eigenvectors of L_A: $L_A U_A = U_A \Lambda_A$, where $\Lambda_A \in \mathbb{R}^{k\times k}$ is the diagonal matrix of the selected eigenvalues. Specifically, anchors are sorted by global anchor ID and divided into P_w contiguous, non-overlapping blocks, one per worker. Each worker stores the rows and columns of S_A and L_A associated with its local anchor block. The k eigenvectors corresponding to the smallest eigenvalues of L_A are computed using a distributed block LOBPCG eigensolver. The initial block vectors are sampled from a standard Gaussian distribution using the random seed of the corresponding experimental run and are orthonormalized before the first iteration. The solver terminates when the relative residual norm falls below the convergence tolerance ε_e, or when the number of iterations reaches the maximum value T_e. The numerical settings of the eigensolver are described in Section 4.1.3. The set of cross-worker anchor edges is defined as $E_A^{\text{cut}} = \left\{\left(a_i, a_j\right) | a_i \in A^{(p)},\ a_j \in A^{(q)}, p \neq q\right\}$, satisfying $S_{A,ij} \neq 0$. In each iteration, only the embedding vectors associated with cross-worker anchor edges need to be exchanged.

Theorem 1: *(Communication Cost Bound) Assume that the distributed eigensolver runs for T_e iterations, the embedding dimension is k, and the reduced anchor graph is distributed across P_w workers. Then the communication cost of the distributed spectral core is bounded by $O\left(T_e\left(k|E_A^{\text{cut}}| + k^2 \log P_w\right)\right)$, where $|E_A^{\text{cut}}|$ is the number of cross-worker anchor edges.*

Proof: In each eigensolver iteration, sparse matrix-vector multiplication requires exchanging k-dimen sional embedding vectors along the cross-worker anchor edges. Therefore, the communication caused by boundary vector exchange is $O\left(k|E_A^{\text{cut}}|\right)$. In addition, orthogonalization and small projected eigenproblem updates require collective synchronization among workers. The communication cost of this step can be written as $O\left(k^2 \log P_w\right)$. Thus, the communication cost of one iteration is $O\left(k|E_A^{\text{cut}}| + k^2 \log P_w\right)$. After T_e iterations, the total communication cost is $O\left(T_e\left(k|E_A^{\text{cut}}| + k^2 \log P_w\right)\right)$. □

3.4.2 Anchor-Level One-Step Discretization

Let $Y_A \in \{0,1\}^{m\times k}$ denote the anchor-level discrete label matrix, where each row corresponds to one anchor and each column corresponds to one cluster. The one-hot assignment constraint is given by $Y_A \mathbf{1}_k = \mathbf{1}_m$, where $\mathbf{1}_k$ and $\mathbf{1}_m$ are all-one vectors with dimensions k and m, respectively. To account for cluster-size normalization, the normalized discrete indicator matrix is defined as $F_A = Y_A\left(Y_A^T Y_A\right)^{-1/2}$. Then the anchor-level discrete objective is formulated as $\min_{Y_A \in \mathrm{Y}} \mathcal{J}\left(Y_A\right) = \mathrm{Tr}\left(F_A^T L_A F_A\right)$ where $\mathcal{Y} = \left\{Y_A \in \{0,1\}^{m\times k} \mid Y_A \mathbf{1}_k = \mathbf{1}_m,\ Y_A^T \mathbf{1}_m > 0\right\}$. Here, $Y_A^T \mathbf{1}_m > 0$ ensures that each cluster contains at least one anchor. This objective directly minimizes the normalized cut on the relation-aware reduced anchor graph.

The initial anchor labels are obtained by applying k-means to the rows of U_A. Starting from this initialization, anchors are visited in ascending anchor-ID order and their labels are refined sequentially according to the reduced normalized-cut objective. For anchor a_i, let $Y_A^{(i\to c)}$ denote the label matrix obtained by assigning anchor a_i to cluster c while keeping other anchor labels fixed. The label of a_i is updated by $c_i = argmin_{c\in\{1,\ldots,k\}} \mathcal{J}\left(Y_A^{(i\to c)}\right)$ and the i-th row of Y_A is set as $Y_A\left(i,:\right) = e_{c_i}^T$ where e_{c_i} is the one-hot vector of cluster c_i. One complete visit to all anchors is regarded as a refinement sweep. A candidate label change is accepted only if it strictly decreases the objective; otherwise, including the case of equal objective values, the current label is retained. The theoretical refinement procedure continues until a complete sweep accepts no label change. In the practical implementation, the procedure is additionally terminated when the relative objective decrease is smaller than 10^{-5} or when 20 sweeps have been completed. These practical stopping criteria limit the computational cost without affecting the monotonicity of the accepted updates, while the local-optimality guarantee in Theorem 2 applies to the uncapped refinement procedure terminated by a

no-change sweep. This refinement step directly updates the discrete anchor labels according to the reduced graph objective, rather than treating label assignment as an independent k-means post-processing stage. The ablation variant RAASC-kmeans uses the same k-means initialization on U_A, but omits the subsequent discrete refinement.

Theorem 2: *(Finite Termination and Local Optimality of Anchor-level Discretization) Suppose that a candidate anchor-label change is accepted only when it strictly decreases $\mathcal{J}(Y_A)$, while the current label is retained in the case of equal objective values. Then the objective sequence generated by the refinement procedure is monotonically non-increasing. Moreover, if the refinement is continued until a complete sweep accepts no label change, it terminates after finitely many accepted updates at a one-anchor local discrete optimum.*

Proof: Let $Y_A^{(T)}$ denote the anchor-level label matrix after the t-th accepted update. During each refinement step, the label of one anchor is considered while all other anchor labels remain fixed. By the acceptance rule, every accepted label change satisfies $\mathcal{J}\left(Y_A^{(t+1)}\right) < \mathcal{J}\left(Y_A^{(t)}\right)$. Therefore, the objective is strictly decreasing at every accepted update and remains unchanged when a candidate update is rejected. Because each of the m anchors can take one of k labels, the feasible label set contains at most k^m assignments. Since every accepted update strictly decreases the objective, no previously visited label assignment can be revisited. Consequently, only finitely many label changes can be accepted. When a complete sweep accepts no label change, changing any single anchor to any alternative label cannot decrease the objective; otherwise, the corresponding minimizing candidate would be accepted. Hence, for every anchor i and every candidate label c, $\mathcal{J}(Y_A) \le \mathcal{J}\left(Y_A^{(i\to c)}\right)$ where $Y_A^{(i\to c)}$ denotes the label matrix obtained by assigning anchor i to cluster c while keeping all other labels fixed. Therefore, the final labeling is a one-anchor local discrete optimum. □

3.4.3 Label Propagation and Inductive Inference

Since each row of B represents the soft assignment distribution from a node to its selected anchors, the label information learned on the reduced anchor graph can be propagated back to the original nodes by $\mathrm{P} = \mathrm{BY}_A$, where $P \in \mathbb{R}^{n\times k}$ denotes the node-level soft label matrix, and $Y_A \in \mathbb{R}^{m\times k}$ denotes the anchor-level label matrix. The final label of node v_i is assigned according to the largest entry in the i-th row of P: $\hat{y}_i = argmax_c P_{ic}$. This propagation step avoids running k-means on all original nodes. Instead, node labels are recovered by sparse matrix multiplication, whose cost depends on the number of nonzero entries in B. Since B is constructed with adaptive neighborhood size K_i, the label propagation cost is $O(\mathrm{nnz}(B)k) = O\left(k\sum_{i=1}^{n} K_i\right)$. Based on Proposition 1, this cost is upper-bounded by $O(nK_{\max}k)$. In addition to transductive label recovery, RAASC naturally supports inductive inference for newly arriving nodes. For a new node v_{new}, the global spectral decomposition does not need to be recomputed. Instead, RAASC only constructs its sparse anchor-assignment vector $b_{\mathrm{new}} \in \mathbb{R}^{1\times m}$ using the same adaptive and relation-compatible assignment strategy described in Section 3.3. The soft label distribution of v_{new} is then obtained by $p_{\mathrm{new}} = b_{\mathrm{new}} Y_A$ and its predicted label is given by $\hat{y}_{\mathrm{new}} = argmax_c \left(p_{\mathrm{new}}\right)_c$.

Proposition 2: *(Inductive Inference Cost) For a new node v_{new} connected to K_{new} anchors, its label distribution can be inferred in $O(K_{\mathrm{new}}k)$ time without recomputing the spectral decomposition.*

Proof: The anchor-assignment vector b_{new} has only K_{new} nonzero entries. Therefore, computing $p_{\mathrm{new}} = b_{\mathrm{new}} Y_A$ only requires multiplying these K_{new} nonzero weights with the corresponding k-dimensional anchor label vectors in Y_A. Thus, the computational cost is $O(K_{\mathrm{new}}k)$. Since Y_A has already been learned on the reduced anchor graph, no eigen-decomposition on either the full graph or the reduced graph is required for the new node. □

3.5 Complexity Analysis

For complexity analysis, let n, d, m, k, and $|E|$ denote the numbers of target nodes, feature dimensions, anchors, clusters, and relation-specific edges, respectively. The RAASC-specific sparsity parameters ℓ, b, and s denote the average numbers of stored local neighbors, examined candidates during local-neighbor and anchor retrieval, and retained sample-anchor connections per node, respectively, where $s \leq \min b, K_{\max}$. Let $e_A = \mathrm{nnz}(S_A)$ denote the number of nonzero edges in the reduced anchor graph, I_e and I_d the numbers of eigensolver iterations and anchor-level discretization sweeps, and p the number of workers. In practice, $s, \ell \ll m \ll n$, while b depends on the relation-specific component restriction and candidate-retrieval implementation.

- *Relation-aware Anchorization:* Constructing relation-specific subgraphs and identifying their weakly connected components requires $O(|E| + n|\mathcal{R}|)$ time. Computing local structural statistics and selecting anchors requires approximately $O(nbd)$ time, where b is the average number of candidates examined in local-neighbor and anchor-selection operations. The corresponding storage cost is $O(|E| + nd + n\ell)$.
- *Adaptive Sample-anchor Construction:* For each target node, RAASC evaluates at most b candidate anchors and retains $s \leq K_{\max}$ assignments. Therefore, candidate scoring and adaptive assignment require $O(nbd)$ time, while the sparse sample-anchor matrices contain $O(ns)$ nonzero entries and require $O(ns)$ memory.
- *Reduced Spectral Learning:* Because each node is connected to at most s anchors, accumulating node-wise contributions to the reduced anchor graph requires $O(ns^2)$ time. Let $e_A \leq m^2$ denote the number of nonzero edges in the resulting anchor graph. A sparse block LOBPCG iteration requires $O(e_A k + mk^2)$ operations. Thus, the distributed spectral core requires approximately $O(I_e(e_A k + mk^2)/p)$ computation time, and $O(e_A + mk)$ memory.
- *Label Recovery:* With incremental objective updates, anchor-level discretization requires $O(I_d mk)$ time, while propagating anchor labels through the sparse sample-anchor matrix requires $O(nsk)$ time. Node labels can be recovered in a streaming or mini-batch manner, requiring $O(k)$ temporary memory per node and $O(\mathrm{n})$ storage for the final hard labels, without materializing the complete $\mathrm{n} \times k$ soft-label matrix.

Combining the above stages, the dominant time complexity of RAASC is $O(|E| + n|\mathcal{R}| + nbd + ns^2 + I_e(e_A k + mk^2)/p + I_d mk + nsk)$, and its dominant memory complexity is $O(|E| + nd + n(\ell + s) + e_A + mk)$. When the candidate size b, neighborhood sizes ℓ and s, cluster number k, and anchor budget m are bounded or grow much more slowly than n, the data-dependent part of RAASC scales approximately linearly with $n + |E|$. In the worst case, exhaustive anchor search gives $b = m$, and a dense reduced graph gives $e_A = m^2$. Even in this case, the eigenproblem is defined over m anchors rather than n original nodes, where $m \ll n$.

To place these results in context, Table A1 compares RAASC with the principal competing methods. Conventional SC constructs an $n \times n$ affinity matrix and solves a node-level eigenproblem, resulting in quadratic memory and cubic eigen-decomposition costs. Similar to LSC, DnC-SC, and RESKM, RAASC transfers the spectral problem from the original n-node graph to a much smaller m-anchor graph. Relative to lightweight anchor approximations, RAASC introduces $O(|E| + nbd)$ relation-aware preprocessing to preserve typed local structures and relation-compatible sample-anchor assignments. Nevertheless, the resulting sample-anchor matrices contain only $O(ns)$ nonzero entries, and the spectral cost depends on m and e_A rather than directly on n. SpectralMix also avoids a dense node-level eigen-decomposition, but its iterative optimization repeatedly processes graph neighborhoods, relation structures, and attribute categories. Therefore, the theoretical advantage of RAASC is best understood as a trade-off: it combines

relation-aware representation with sparse anchor-level computation, rather than claiming a uniformly lower asymptotic order than every anchor-based method.

4 Experimental Evaluation

4.1 Experimental Setup

4.1.1 Datasets

To evaluate RAASC under both medium-scale and large-scale heterogeneous graph settings, three datasets are used: DBLP [15], IMDB [16], and OGBN-MAG [17]. DBLP and IMDB are adopted from the PyG/MAGNN processed versions and are used as medium-scale heterogeneous graph benchmarks, with authors and movies as target nodes, respectively. OGBN-MAG is adopted from the Open Graph Benchmark and is used to evaluate scalability on large-scale heterogeneous graphs, where papers are treated as target nodes. The statistics of the datasets are summarized in Table 1.

Table 1: Dataset statistics.

Dataset	Version	Target Type	Target Nodes	Total Nodes	Relation Types	Classes
DBLP	PyG/MAGNN	Author	4057	26,128	6 Directed relations	4
IMDB	PyG/MAGNN	Movie	4278	11,616	4 Directed relations	3
OGBN-MAG	OGB	Paper	736,389	1,939,743	4 Relation types	349

Detailed relation-specific component statistics are reported in Tables A2 and A3. After applying $n_{\min} = 10$, the number of retained components ranges from 12 to 1430 across relations, and the relation-wise anchor budgets sum to 406, 428, and 5000 on DBLP, IMDB, and OGBN-MAG, respectively. Moreover, 68.3%–88.6% of target nodes belong to at least two retained relation-specific components. These long-tailed and overlapping component structures motivate the proposed component-aware anchor allocation strategy.

4.1.2 Compared Methods and Evaluation Protocol

RAASC is compared with representative methods from three categories. The first category contains basic clustering methods, including spectral clustering (SC) [1] and k-means [2]. The second category contains large-scale spectral clustering methods, including LSC [5], DnC-SC [6], and RESKM [7], which represent landmark-based, divide-and-conquer-based, and framework-based accelerated spectral clustering methods, respectively. The third category contains attributed multi-relational spectral clustering, represented by SpectralMix [10]. Because MCVGAE [14] requires a homogeneous node-level adjacency matrix, its controlled comparison with RAASC is presented separately in Section 4.3.

Clustering quality is evaluated using accuracy (ACC), normalized mutual information (NMI), and adjusted Rand index (ARI). ACC is calculated after aligning predicted and ground-truth labels using the Hungarian algorithm. Computational efficiency is evaluated using total runtime and peak memory consumption. Unless otherwise specified, the cluster number (k) is set to the number of ground-truth classes. Each experiment is independently repeated five times with different random seeds, and the mean and standard deviation are reported.

To ensure comparability, all methods are evaluated using the same target nodes, cluster numbers, and evaluation labels for each dataset. The same target-node attributes are provided to all methods that support attributed inputs, whereas methods that do not accept attributes use their required graph or similarity representation. Except for the number of clusters, method-specific parameters are not selected using ACC,

NMI, or ARI. All methods are evaluated under the same five-run protocol and on the same CPU cluster described in Section 4.1.3. MCVGAE is evaluated separately in Section 4.3 because it requires a homogeneous node-level adjacency matrix; in that controlled comparison, MCVGAE and RAASC use exactly the same adjacency matrix, node features, cluster number, labels, and random seeds.

4.1.3 Implementation and Parameter Settings

RAASC was implemented in Python 3.10. Spark 3.5 with GraphX was used to preprocess each input graph, construct relation-specific subgraphs, and identify relation-aware local components, while OpenMPI 4.1 was used for distributed spectral learning on the reduced anchor graph. NumPy 1.26, SciPy 1.12, and scikit-learn 1.4 were used for numerical computation and evaluation. PyTorch 2.2 and PyTorch Geometric 2.5 were used only for feature preprocessing and data handling and were not involved in the main spectral clustering procedure.

All experiments were conducted on a CPU cluster consisting of four identical physical servers. Each server was equipped with two Intel Xeon Gold 5218 processors, with 16 physical cores per socket and 32 physical cores in total at 2.30 GHz, 256 GB DDR4 RAM, and a 2 TB NVMe SSD. The servers were interconnected through a 10-Gigabit Ethernet network. Each worker was allocated eight physical CPU cores and 32 GB of memory, including a 28 GB Spark executor heap and 4 GB of memory overhead. At most four workers were deployed on each physical server. Consequently, the experiments with 1, 2, and 4 workers were executed on one server, while those with 8 and 16 workers used two and four servers, respectively. The reported runtime and memory results were obtained under CPU-only execution. Accordingly, the reported speedup and parallel efficiency are measured with respect to the number of eight-core workers rather than the number of individual CPU cores.

Except for the cluster number k, all parameters are selected without using node labels. The anchor budget is set to 10% of the number of target nodes, with a lower bound of 200 and an upper bound of 5000. The bandwidths $\sigma_z, \sigma_f, \sigma_s$ are automatically set to the median local node-node distance in the space, the median node-anchor attribute distance, and the median node-anchor structural distance, respectively. All balancing weights and scalar parameters are fixed across datasets whenever possible, while Gaussian-kernel bandwidths are automatically estimated using the median-distance heuristic. No parameter is selected using ACC, NMI, or ARI. Detailed settings are provided in Table A4.

4.2 Clustering Performance

Table 2 reports the clustering performance of RAASC and the compared methods on DBLP, IMDB, and OGBN-MAG under the unified experimental protocol described in Section 4.1. Conventional spectral clustering can be executed on DBLP and IMDB but fails on the large-scale OGBN-MAG dataset because of excessive memory consumption or runtime and is therefore marked as Out of Memory (OOM)/Time Limit Exceeded (TLE). On the three heterogeneous benchmarks in the main comparison, RAASC achieves the highest mean ACC, NMI, and ARI among the compared clustering and spectral clustering baselines. Compared with k-means and conventional SC, RAASC benefits from exploiting graph structures through relation-aware anchor construction and adaptive sample-anchor assignment. Compared with scalable spectral clustering methods, including LSC, DnC-SC, and RESKM, RAASC further improves clustering quality by preserving relation-specific local structures during anchor generation and reduced graph construction. SpectralMix is the strongest baseline because it also considers attributed multi-relational graph information. Compared with SpectralMix, RAASC obtains absolute mean improvements of 0.024–0.038 in ACC, 0.031–0.057 in NMI, and 0.029–0.061 in ARI. These mean improvements indicate the effectiveness of the proposed relation-aware adaptive anchor strategy on both medium-scale and large-scale heterogeneous graphs.

Table 2: Clustering performance comparison on DBLP, IMDB, and OGBN-MAG.

Dataset	Metric	k-Means	SC	LSC	DnC-SC	RESKM	SpectralMix	RAASC
DBLP	ACC	0.513 ± 0.024	0.563 ± 0.029	0.602 ± 0.025	0.623 ± 0.021	0.635 ± 0.017	0.681 ± 0.016	0.705 ± 0.018
	NMI	0.298 ± 0.020	0.350 ± 0.027	0.381 ± 0.019	0.414 ± 0.030	0.434 ± 0.039	0.479 ± 0.037	0.521 ± 0.015
	ARI	0.252 ± 0.021	0.312 ± 0.015	0.344 ± 0.026	0.376 ± 0.029	0.390 ± 0.030	0.422 ± 0.032	0.483 ± 0.022*
IMDB	ACC	0.395 ± 0.033	0.454 ± 0.030	0.472 ± 0.029	0.491 ± 0.032	0.501 ± 0.022	0.538 ± 0.021	0.569 ± 0.015*
	NMI	0.079 ± 0.012	0.108 ± 0.014	0.119 ± 0.015	0.153 ± 0.017	0.162 ± 0.017	0.211 ± 0.020	0.242 ± 0.009*
	ARI	0.058 ± 0.021	0.087 ± 0.015	0.108 ± 0.024	0.135 ± 0.036	0.140 ± 0.018	0.183 ± 0.024	0.212 ± 0.014
OGBN-MAG	ACC	0.096 ± 0.019	OOM/TLE	0.117 ± 0.018	0.137 ± 0.028	0.143 ± 0.034	0.156 ± 0.035	0.194 ± 0.020
	NMI	0.177 ± 0.028	OOM/TLE	0.205 ± 0.021	0.238 ± 0.020	0.256 ± 0.023	0.276 ± 0.024	0.333 ± 0.016*
	ARI	0.039 ± 0.015	OOM/TLE	0.046 ± 0.007	0.066 ± 0.011	0.065 ± 0.014	0.074 ± 0.032	0.115 ± 0.023*

Note: Results are reported as mean ± standard deviation over five independent runs. *denotes a statistically significant improvement of RAASC over SpectralMix according to a two-sided Welch's t-test ($p < 0.05$).

To assess the statistical reliability of the observed performance differences, two-sided Welch's t-tests were conducted between RAASC and SpectralMix, the strongest competing method in the main experiments. Welch's t-test was adopted because the runs of different algorithms were independently generated and equal variances were not assumed. The significance level was set to 0.05. The Welch's t-test results provide statistical support for part of these performance improvements. Specifically, RAASC significantly outperforms SpectralMix in terms of ARI on DBLP, ACC and NMI on IMDB, NMI and ARI on OGBN-MAG at the 0.05 significance level. For the remaining dataset-metric combinations, RAASC still obtains higher mean scores, although the differences do not reach statistical significance. Statistically significant improvements are marked with an asterisk in Table 2.

4.3 Comparison with Recent Deep Graph Clustering

To further compare RAASC with recent deep graph clustering methods, we include MCVGAE [14], a multiscale contrastive variational graph autoencoder for node clustering. MCVGAE is designed for attributed homogeneous graphs represented by a single node-level adjacency matrix, whereas the main RAASC experiments use heterogeneous graphs containing multiple node and relation types. Therefore, directly inserting MCVGAE into Table 2 would result in different structural inputs. For a controlled comparison, both MCVGAE and RAASC are evaluated on the same homogeneous DBLP author graph used by MCVGAE. The homogeneous DBLP benchmark contains 4057 author nodes, 7056 undirected edges, 334-dimensional node features, and four ground-truth classes. Both methods use exactly the same adjacency matrix, node features, number of clusters, evaluation labels, and random seeds. For MCVGAE, we follow the settings reported in the original paper: the confidence threshold is set to $\alpha = 0.31$, the graph-update interval is $M = 20$, the pretraining and training learning rates are 0.01 and 0.001, respectively, and the maximum number of training epochs is 500. The remaining parameters are set according to the authors' implementation. For RAASC, the graph is treated as a single-relation attributed graph, with $m = 406$, $n_{\min} = 10$, $\kappa = 10$, $K_{\min} = 2$, $K_{\max} = 10$, and $\mu = 0.2$. Each method is independently executed five times using the same random seeds, and the mean and standard deviation are reported.

As shown in Table 3, MCVGAE achieves a slightly higher ACC than RAASC by 0.011, whereas RAASC outperforms MCVGAE by 0.024 in NMI and 0.008 in ARI. The results indicate that MCVGAE provides slightly better pointwise label alignment, while RAASC produces a partition with stronger overall information consistency and pairwise clustering agreement. Notably, RAASC remains competitive even in this homogeneous single-relation setting, although it is primarily designed for attributed heterogeneous graphs.

Table 3: Comparison with MCVGAE on the homogeneous DBLP benchmark.

Method	ACC	NMI	ARI
MCVGAE	0.739 ± 0.008	0.389 ± 0.013	0.439 ± 0.010
RAASC	0.728 ± 0.011	0.413 ± 0.009	0.447 ± 0.008

4.4 Ablation Experiments

To examine the contribution of each component, five RAASC variants are constructed for ablation studies: RAASC-R removes relation-aware local components; RAASC-U uses uniform component-level anchor allocation; RAASC-F replaces adaptive K_i with a fixed anchor number; RAASC-L removes anchor load control by setting $\mu = 0$; and RAASC-kmeans replaces anchor-level one-step discretization with k-means on U_A. All variants of the ablation experiments were evaluated using the same dataset, target nodes, and evaluation metrics as RAASC.

As shown in Table 4, the complete RAASC model achieves the highest mean ACC, NMI, and ARI across all datasets, confirming that the proposed components provide complementary benefits. Among the variants, RAASC-R produces the largest overall degradation, with absolute decreases ranging from 0.034 to 0.058 across the nine dataset-metric combinations. In particular, removing relation-aware local components reduces both NMI and ARI by 0.058 on DBLP and reduces NMI by 0.055 on OGBN-MAG. These relatively large decreases demonstrate that relation-specific local structures provide essential guidance for anchor generation, especially on graphs containing complex and overlapping relation patterns. RAASC-U and RAASC-F also consistently underperform the complete model, with absolute decreases of 0.019–0.032 and 0.021–0.033, respectively. These results support difficulty-aware anchor allocation and node-specific adaptive sample-anchor connectivity. RAASC-L produces smaller but consistent reductions of 0.013–0.022, indicating that anchor-load regularization improves the balance and stability of sample-anchor assignment. RAASC-kmeans is the closest variant to the complete model but still decreases performance by 0.010–0.017 across all datasets and metrics, confirming that anchor-level discrete refinement helps reduce the mismatch between continuous spectral embeddings and final discrete labels. Overall, the ablation results verify that all five model components contribute to the effectiveness and robustness of RAASC, with the identification and use of relation-aware local components providing the most substantial contribution.

4.5 Efficiency and Scalability

To evaluate computational efficiency, we compare RAASC with k-means, SC, LSC, DnC-SC, RESKM, and SpectralMix in terms of total runtime, peak memory usage, and distributed scalability. For OGBN-MAG, RAASC is evaluated with 4 workers by default unless otherwise specified.

Fig. 2a,b compares the runtime and peak memory usage of different methods on DBLP, IMDB, and OGBN-MAG. K-means has the lowest computational cost because it performs only vector-based clustering, but its clustering quality is limited. Conventional SC is feasible on medium-scale datasets but fails on OGBN-MAG, confirming the poor scalability of full spectral clustering. RAASC introduces relation-aware

anchorization and adaptive sample-anchor assignment, which brings moderate additional cost compared with lightweight anchor-based methods such as LSC and RESKM. Nevertheless, RAASC is faster than SC and SpectralMix on DBLP and IMDB. On OGBN-MAG, RAASC finishes in 627 s with 21.0 GB memory, achieving the lowest runtime among scalable graph-based baselines and requiring less memory than SpectralMix. These results indicate that RAASC provides a favorable trade-off between clustering quality, runtime, and memory usage.

Table 4: Ablation study of different components of RAASC.

Variant		RAASC-R	RAASC-U	RAASC-F	RAASC-L	RAASC-kmeans	RAASC
DBLP	ACC	0.658 ± 0.016	0.676 ± 0.023	0.681 ± 0.030	0.690 ± 0.020	0.693 ± 0.015	0.705 ± 0.018
	NMI	0.463 ± 0.018	0.489 ± 0.010	0.496 ± 0.015	0.505 ± 0.031	0.508 ± 0.029	0.521 ± 0.015
	ARI	0.425 ± 0.016	0.451 ± 0.012	0.456 ± 0.011	0.467 ± 0.025	0.469 ± 0.022	0.483 ± 0.022
IMDB	ACC	0.520 ± 0.027	0.541 ± 0.021	0.539 ± 0.013	0.550 ± 0.008	0.554 ± 0.027	0.569 ± 0.015
	NMI	0.188 ± 0.021	0.211 ± 0.028	0.210 ± 0.025	0.221 ± 0.023	0.225 ± 0.021	0.242 ± 0.009
	ARI	0.168 ± 0.024	0.189 ± 0.022	0.187 ± 0.027	0.197 ± 0.010	0.201 ± 0.028	0.212 ± 0.014
OGBN-MAG	ACC	0.160 ± 0.010	0.175 ± 0.024	0.173 ± 0.011	0.181 ± 0.018	0.184 ± 0.029	0.194 ± 0.020
	NMI	0.278 ± 0.019	0.304 ± 0.015	0.300 ± 0.028	0.311 ± 0.022	0.316 ± 0.023	0.333 ± 0.016
	ARI	0.080 ± 0.020	0.094 ± 0.022	0.091 ± 0.019	0.098 ± 0.011	0.103 ± 0.028	0.115 ± 0.023

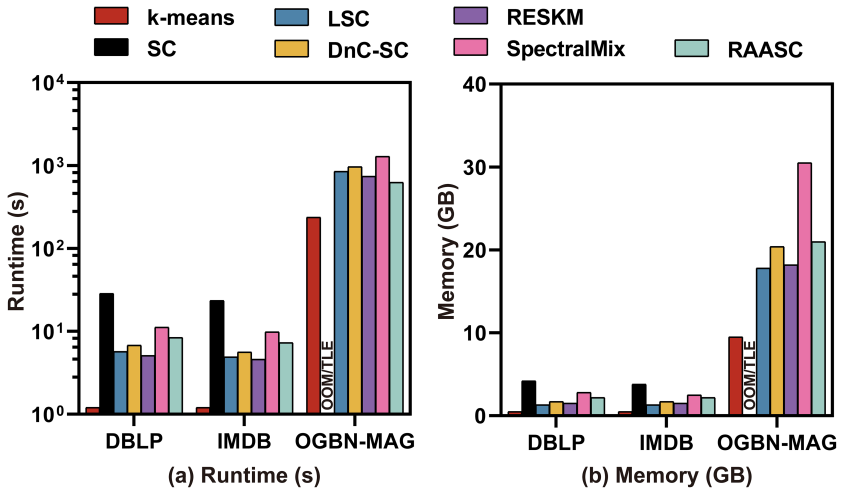


Figure 2: Runtime and peak memory usage of different methods.

Table 5 further reports the runtime breakdown of RAASC. The main computational costs come from sample-anchor construction and the spectral core. On OGBN-MAG, these two stages take 218 and 154 s, respectively. This is expected because sample-anchor construction involves candidate anchor retrieval, adaptive neighborhood assignment, and load-aware scoring over a large heterogeneous graph, while the spectral core still requires iterative learning on the reduced anchor graph. In contrast, reduced graph construction and label propagation are relatively efficient. In particular, label propagation takes only 43 s on OGBN-MAG because it only requires sparse multiplication between the sample-anchor matrix and the anchor-level label matrix.

Table 5: Runtime breakdown of RAASC.

Dataset	Anchorization	Sample-Anchor Construction	Reduced Graph Construction	Spectral Core	Label Propagation	Total
DBLP	1.7	2.5	0.8	2.7	0.7	8.4
IMDB	1.3	2.1	0.7	2.6	0.6	7.3
OGBN-MAG	145	218	67	154	43	627

To further quantify distributed scalability, let T_p denote the end-to-end wall-clock time of RAASC using p workers. The worker-level speedup and parallel efficiency are calculated as $S_p = T_1/T_p$ and $E_p = S_p/p \times 100\%$, respectively. Communication and synchronization time includes embedding-vector exchanges along cross-worker anchor edges and MPI collective operations in the distributed spectral core. For each run, the maximum accumulated communication and synchronization time across workers is used to represent the critical communication path. Its ratio to the total runtime is calculated as $R_p^{\mathrm{comm}} = T_p^{\mathrm{comm}}/T_p \times 100\%$. Here, p denotes the number of eight-core workers rather than the number of individual CPU cores. Each configuration was executed five times, and the average results are reported in Table A5.

Fig. 3 reports the worker-level scalability of RAASC on OGBN-MAG. The configurations with one to four workers were executed within a single physical server, whereas the configurations with eight and sixteen workers were distributed over two and four servers, respectively. Increasing the worker count from 1 to 16 reduces the end-to-end runtime from 1452 to 390 s, corresponding to a speedup of 3.72. As detailed in Table A5, the worker-level parallel efficiency decreases from 100.0% to 23.3%, while the communication and synchronization ratio increases from 0.0% to 53.8%. Most of the runtime reduction is obtained within four workers. Beyond four workers, cross-server embedding-vector exchanges along cross-worker anchor edges, collective synchronization in the distributed eigensolver, and the sequential fraction of the overall pipeline become increasingly prominent. Moreover, because the reduced spectral problem contains only 5000 anchors, the computation assigned to each worker decreases more rapidly than the communication and synchronization requirements as additional workers are introduced. The 349-dimensional spectral embedding and the nonuniform distributions of relation-specific components and cross-worker anchor edges may further cause communication overhead and workload imbalance. These factors explain the observed sublinear speedup and diminishing parallel efficiency beyond four workers.

4.6 Robustness Analysis

To evaluate the robustness of RAASC with respect to its main parameters, we investigate the anchor ratio m/n, the maximum neighborhood size $K_{\max}$, and the load-penalty strength μ on DBLP and IMDB. The anchor ratio is varied over $\{5\%, 10\%, 15\%, 20\%, 25\%\}$, $K_{\max}$ over $\{4, 6, 8, 10, 12\}$, and μ over $\{0, 0.1, 0.2, 0.5, 1.0\}$. When one parameter is varied, all other parameters are fixed at their default values reported in Table A4. Each parameter setting is independently executed five times using the same evaluation protocol as the main experiments, and the mean and standard deviation of ACC, NMI, and ARI are reported.

As shown in Fig. 4, RAASC exhibits stable clustering performance over moderate ranges of the three parameters. Increasing the anchor ratio from 5% to 10% generally improves ACC, NMI, and ARI because additional anchors provide better coverage of relation-specific local structures. Further increasing the anchor ratio results in only limited improvements or small fluctuations, while increasing the size of the reduced graph. This observation supports the use of a 10% anchor ratio as an accuracy-efficiency trade-off rather than a dataset-specific optimum. For $K_{\max}$, the clustering performance generally improves as the value increases

from 4 to 8 or 10 and then remains stable or decreases slightly. A small $K_{\max}$ may provide insufficient anchor support for boundary and ambiguous nodes, whereas a large value may introduce less reliable sample-anchor connections. Regarding μ, disabling load control with $\mu = 0$ leads to lower performance, while moderate values between 0.1 and 0.2 provide a favorable balance between relation-compatible assignment and anchor-load regularization. Excessively large values may over-penalize highly representative anchors. Overall, the adopted defaults, $m/n = 10\%$, $K_{\max} = 10$, and $\mu = 0.2$, lie within stable performance regions on both datasets.

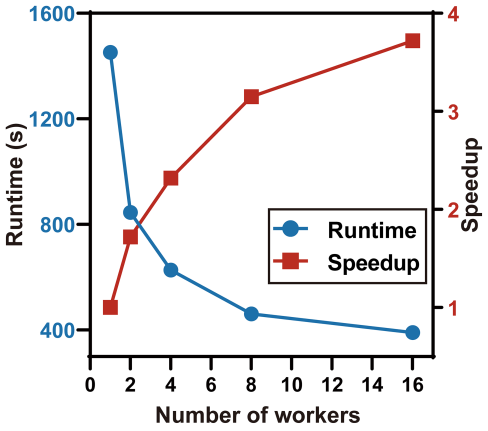


Figure 3: Worker-level scalability of RAASC on OGBN-MAG.

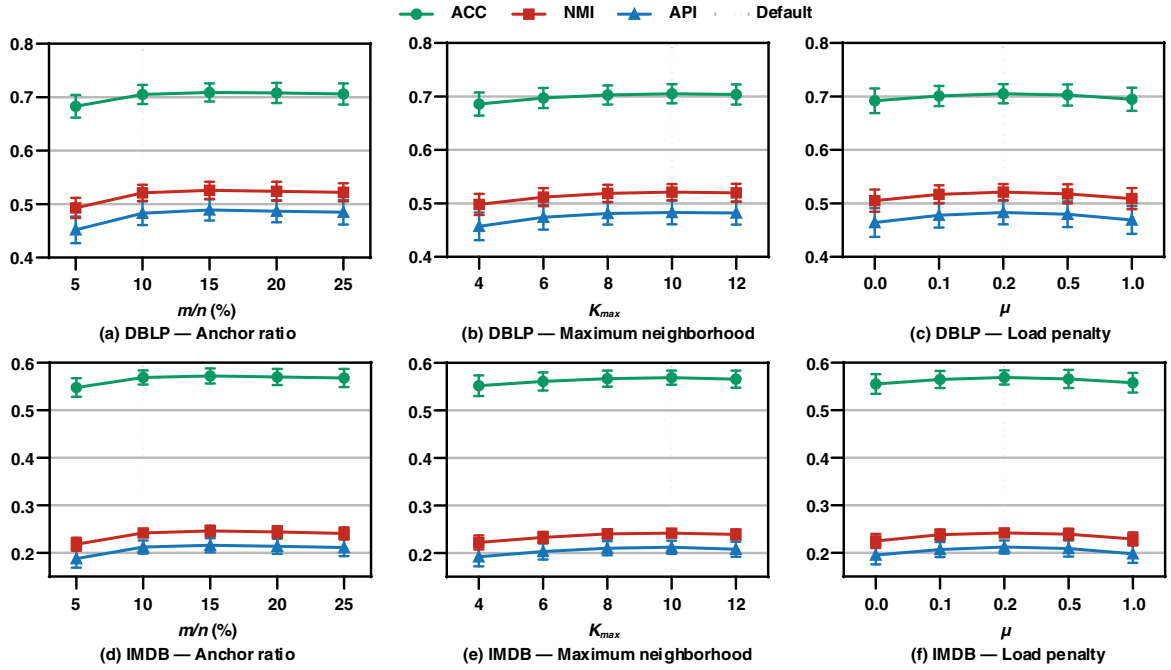


Figure 4: Parameter sensitivity of RAASC on DBLP and IMDB.

In addition to parameter sensitivity, we examine whether RAASC is affected by the arbitrary numerical ordering of nodes. Five random permutations of the node indices were generated for each dataset, with the feature matrix, labels, and relation-specific edge indices consistently remapped to preserve the underlying graph. As reported in Table A6, randomly permuting the node indices produces only minor variations in

ACC, NMI, and ARI compared with the original ordering, while the corresponding standard deviations remain comparable to those in the main experiments. This indicates that the identification of relation-aware local components and the adaptive anchor assignment procedure in RAASC are not materially affected by the storage order or numerical indices of the nodes.

4.7 Inductive New-Node Evaluation

To evaluate the inductive capability of RAASC, we conducted a held-out new-node experiment on DBLP and IMDB. For each dataset, 80% of the target nodes were randomly selected for model construction, while the remaining 20% were treated as newly arriving nodes. The held-out nodes and all their incident edges were removed before identifying relation-aware local components, generating anchors, performing reduced spectral learning, and conducting anchor-level discretization. The splits were generated without using ground-truth labels. After training, the anchor set, anchor labels, kernel bandwidths, and training-stage anchor loads were fixed. For each held-out node, only its attributes and relation-specific edges to the existing training graph were provided. Candidate anchors were retrieved from the relation-specific components associated with its observed neighbors. If no relation-supported candidate was available, the globally nearest anchors were used as a fallback. The node was then assigned through the learned sample-anchor model, without regenerating anchors or recomputing the spectral decomposition. Edges between held-out nodes were not used. All hyperparameters followed the default settings in Table A4, and the anchor budget was determined only from the training target nodes.

Table 6 compares inductive inference with full retraining, in which the held-out nodes were added to the graph and the complete RAASC pipeline was rerun. Both settings were evaluated on the same held-out nodes. Results are reported as mean and standard deviation over five random 80/20 splits. As shown in Table 6, inductive inference achieves clustering performance close to that obtained by full retraining. On DBLP, the absolute decreases in ACC, NMI, and ARI are 0.011, 0.009, and 0.014, respectively, while the corresponding decreases on IMDB are 0.013, 0.011, and 0.014. Meanwhile, inductive inference reduces the processing time from 8.5 to 0.7 s on DBLP and from 7.4 to 0.6 s on IMDB, corresponding to approximately 12.1× and 12.3× acceleration, respectively. These results demonstrate that the learned relation-aware anchors can generalize to previously unseen nodes and enable efficient new-node assignment without recomputing the reduced spectral decomposition.

Table 6: Inductive clustering performance and processing time of RAASC on held-out target nodes.

Dataset	Mode	ACC	NMI	ARI	Processing Time (s)
DBLP	Full retraining	0.699 ± 0.019	0.516 ± 0.016	0.476 ± 0.023	8.5
	Inductive inference	0.688 ± 0.021	0.507 ± 0.018	0.462 ± 0.025	0.7
IMDB	Full retraining	0.562 ± 0.017	0.238 ± 0.012	0.205 ± 0.016	7.4
	Inductive inference	0.549 ± 0.019	0.227 ± 0.014	0.191 ± 0.018	0.6

5 Conclusion and Future Work

In this paper, we proposed RAASC, a relation-aware adaptive anchor spectral clustering framework for large-scale attributed and heterogeneous graphs. RAASC constructs relation-aware anchors, builds an adaptive load-aware sample-anchor graph, and performs spectral learning on a reduced anchor graph to avoid full graph eigen-decomposition. On the three heterogeneous benchmarks in the main comparison, RAASC achieves the highest mean ACC, NMI, and ARI among the compared clustering and spectral

clustering baselines. Compared with SpectralMix, the strongest attributed multi-relational baseline in the main experiments, RAASC obtains absolute improvements of 0.024–0.038 in ACC, 0.031–0.057 in NMI, and 0.029–0.061 in ARI. On OGBN-MAG, RAASC completes clustering in 627 s using 21.0 GB of peak memory with four workers and achieves a speedup of 3.72 with sixteen workers. Moreover, the held-out new-node experiments show that inductive inference provides approximately 12.1× and 12.3× acceleration on DBLP and IMDB, respectively, while incurring only small absolute performance decreases of 0.009–0.014. Ablation and parameter-sensitivity results further verify the effectiveness and robustness of the principal components. Nevertheless, RAASC still relies on informative relation types and several balancing parameters. In future work, we plan to investigate automatic parameter selection, dynamic heterogeneous graph clustering, and automatic cluster number estimation within the reduced spectral learning framework.

Acknowledgement: None.

Funding Statement: This research was funded by Science and Technology Research Project of Jiangxi Provincial Department of Education, China (Nos. GJJ2203614 and GJJ2503509).

Author Contributions: Xiangmin Liu developed the methodology, implemented the code, and drafted the manuscript. Weizhi Xiong prepared the figures and tables and contributed to reviewing and editing the manuscript. Wei Zhang performed the experimental analysis. Taojian Luo and Xiu Yao contributed to data collection and organization. Jian Hu and Peng Peng supervised the research. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: All data that support the findings of this study are included within the article.

Ethics Approval: Not applicable. This study did not involve human participants or animals.

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

Appendix A

Table A1: Theoretical complexity comparison of RAASC and the principal competing methods.

Method	Dominant Time Complexity	Dominant Memory Complexity
SC	$O\left(n^2d + n^3\right)$	$O\left(n^2\right)$
LSC	$O\left(I_L nmd + nm^2 + m^3\right)$	$O\left(nm + m^2\right)$
DnC-SC	$O\left(n\left(\alpha d + s^2 + sk + sd + k^2 I_K\right) + m^3 + m^2\left(d + s\right)\right)$	$O\left(ns + m^2 + nk\right)$
RESKM	$O\left(C_{\mathrm{AS}} + C_{\mathrm{AG}} + m^3 + I_K mk^2 + nsk\right)$	$O\left(ns + m^2 + mk\right)$
SpectralMix	$O\left(I_{\mathrm{S}}\left(\lvert E\rvert h + nh^2 + nCh\right) + I_{\mathrm{KM}} nkh\right)$	$O\left(\lvert E\rvert + \mathrm{nnz}(A) + nh + Ch + kh\right)$
RAASC	$O\left(\lvert E\rvert + nbd + ns^2 + I_e\left(e_A k + mk^2\right)/p + I_d mk + nsk\right)$	$O\left(\lvert E\rvert + nd + n\left(\ell + s\right) + e_A + mk\right)$

Note: n, d, m, and k denote the numbers of target nodes, feature dimensions, anchors, and clusters, respectively. s is the number of retained sample-anchor connections, b is the average candidate set size examined by RAASC. For DnC-SC, α is the selection-rate parameter that bounds the number of subsets generated in each divide-and-conquer step. For SpectralMix, h is the embedding dimension, C is the total number of attribute categories, and $\mathrm{nnz}(A)$ denotes the number of nonzero entries in the attribute matrix A. For RAASC, ℓ is the average number of retained relation-specific component memberships per target node, e_A is the number of nonzero edges in the reduced anchor graph, and p is the number of workers. I_L denotes the number of landmark-selection iterations in LSC; I_K denotes the number of k-means iterations used by DnC-SC and RESKM; I_S and I_{KM} denote the numbers of SpectralMix optimization and subsequent k-means iterations, respectively; and I_e and I_d denote the numbers of eigensolver iterations and anchor-level discretization sweeps in RAASC, respectively. For RESKM, C_{AS} and C_{AG} denote the anchor-selection and anchor-graph construction costs, which depend on the Nyström or sparse-representation variant integrated into the framework. Input-data storage is included when it is explicitly required by the corresponding method.

Table A2: Relation-wise weakly connected component statistics and anchor allocation.

Dataset	Relation	WCCs	All-Node Size Median [IQR]; Max	Target-Candidate Size Median [IQR]; Max	Retained	Anchors
DBLP	r_1	74	28 [8–93]; 18,140	12 [3–41]; 3810	21	92
	r_2	74	28 [8–93]; 18,140	12 [3–41]; 3810	21	92
	r_3	138	15 [4–57]; 12,450	8 [2–28]; 2760	30	63
	r_4	138	15 [4–57]; 12,450	8 [2–28]; 2760	30	63
	r_5	29	220 [45–814]; 13,980	43 [9–167]; 3520	12	48
	r_6	29	220 [45–814]; 13,980	43 [9–167]; 3520	12	48
IMDB	r_1	286	13 [4–38]; 10,240	7 [2–19]; 3980	46	122
	r_2	286	13 [4–38]; 10,240	7 [2–19]; 3980	46	122
	r_3	1096	4 [2–9]; 6180	2 [1–4]; 1460	22	92
	r_4	1096	4 [2–9]; 6180	2 [1–4]; 1460	22	92
OGBN-MAG	r_1	1832	7 [2–31]; 1,020,000	3 [1–12]; 420,000	610	1350
	r_2	4986	3 [1–9]; 1,450,000	2 [1–5]; 690,000	1430	1700
	r_3	742	18 [4–76]; 1,310,000	9 [2–39]; 710,000	520	1250
	r_4	1260	6 [2–21]; 870,000	4 [1–14]; 510,000	440	700

Table A3: Cross-relation component overlap and final anchor statistics.

Dataset	Average Memberships	Multi-Membership Ratio	Maximum Memberships	Final Anchors
DBLP	3.18	88.6%	6	406
IMDB	2.67	79.4%	4	428
OGBN-MAG	2.12	68.3%	4	5000

Note: "WCCs" denotes the number of weakly connected components in each relation-specific subgraph. Component-size distributions are reported as median [interquartile range]; maximum. Target-candidate size counts only target-type nodes eligible for anchor selection. A component is retained when its target-candidate size satisfies $s_C \geq n_{\min} = 10$. "Anchors" denotes the sum of the component-level anchor budgets associated with each relation. Components are disjoint within the same relation, and overlap is therefore measured across retained components from different relation types.

Table A4: Default parameter settings of RAASC.

Category	Description	Parameter	DBLP	IMDB	OGBN-MAG
Clustering	Number of clusters	k	4	3	349
Anchorization	Total anchor budget	m	406	428	5000
Component filtering	Minimum candidate size	$n_{\min}$	10	10	10

(Continued)

Table A4 (continued)

Category	Description	Parameter	DBLP	IMDB	OGBN-MAG
Local neighborhood	Number of local neighbors	κ	10	10	10
Adaptive assignment	Minimum anchors per node	$K_{\min}$	2	2	2
Adaptive assignment	Maximum anchors per node	$K_{\max}$	10	10	10
Component difficulty	Size, variation, and boundary weights	$\alpha_n, \alpha_v, \alpha_b$	1/3	1/3	1/3
Anchor selection	Density, boundary, and coverage weights	$\beta_{\text{den}}, \beta_{\text{bd}}, \beta_{\text{cov}}$	1/3	1/3	1/3
Node uncertainty	Density, boundary, and ambiguity weights	$\eta_\rho, \eta_b, \eta_a$	1/3	1/3	1/3
Anchor scoring	Attribute, structure, relation, and penalty weights	$\lambda_f, \lambda_s, \lambda_r, \lambda_c$	0.25	0.25	0.25
Load control	Load-penalty strength	μ	0.2	0.2	0.2
Soft assignment	Softmax temperature	σ_i	0.2	0.2	0.2
Numerical stability	Stabilization constant	ϵ	10^{-8}	10^{-8}	10^{-8}
Relation fusion	Relation-specific graph weight	ω_r	1/6	1/4	1/4
Eigensolver	Maximum solver iterations	T_e	200	200	200
Eigensolver	Solver tolerance	ε_e	10^{-6}	10^{-6}	10^{-6}
Discretization	Maximum refinement sweeps	T_d	20	20	20
Discretization	Refinement tolerance	ε_d	10^{-5}	10^{-5}	10^{-5}
Distributed setting	Default number of workers	P_w	1	1	4

Table A5: Worker-level parallel efficiency and communication and synchronization overhead of RAASC on OGBN-MAG.

Workers	Physical Nodes	Runtime (s)	Speedup	Efficiency (%)	Comm. and Sync. Time (s)	Comm. and Sync. Ratio (%)
1	1	1452	1.00	100.0	0	0.0
2	1	845	1.72	85.9	60	7.1
4	1	627	2.32	57.9	115	18.3
8	2	461	3.15	39.4	175	38.0
16	4	390	3.72	23.3	210	53.8

Note: Runtime denotes the end-to-end wall-clock time. Speedup and worker-level parallel efficiency are calculated as $S_p = T_1/T_p$ and $E_p = S_p/p \times 100\%$, respectively. Communication and synchronization time includes embedding-vector exchanges along cross-worker anchor edges and MPI collective operations in the distributed spectral core. For each run, the maximum accumulated time across workers is used to represent the critical communication path. All values are averaged over five runs.

Table A6: Robustness of RAASC to random node-order permutations.

Dataset	Order	ACC	NMI	ARI
DBLP	ID order	0.705 ± 0.018	0.521 ± 0.015	0.483 ± 0.022
	Random permutation	0.702 ± 0.017	0.518 ± 0.016	0.480 ± 0.020
IMDB	ID order	0.569 ± 0.015	0.242 ± 0.009	0.212 ± 0.014
	Random permutation	0.567 ± 0.016	0.240 ± 0.011	0.210 ± 0.013
OGBN-MAG	ID order	0.194 ± 0.020	0.333 ± 0.016	0.115 ± 0.023
	Random permutation	0.192 ± 0.019	0.329 ± 0.018	0.113 ± 0.021

References

1. Ng AY, Jordan MI, Weiss Y. On spectral clustering: analysis and an algorithm. Adv Neural Inf Process Syst. 2001;14:1–8.
2. Hartigan JA, Wong MA. Algorithm AS 136: a K-means clustering algorithm. J R Stat Soc Ser C. 1979;28(1):100. doi:10.2307/2346830.
3. Shi J, Malik J. Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell. 2000;22(8):888–905. doi:10.1109/34.868688.
4. Fowlkes C, Belongie S, Chung F, Malik J. Spectral grouping using the nystrom method. IEEE Trans Pattern Anal Machine Intell. 2004;26(2):214–25. doi:10.1109/tpami.2004.1262185.
5. Cai D, Chen X. Large scale spectral clustering via landmark-based sparse representation. IEEE Trans Cybern. 2015;45(8):1669–80. doi:10.1109/TCYB.2014.2358564.
6. Li H, Ye X, Imakura A, Sakurai T. Divide-and-conquer based large-scale spectral clustering. Neurocomputing. 2022;501:664–78. doi:10.1016/j.neucom.2022.06.006.
7. Yang G, Deng S, Chen X, Chen C, Yang Y, Gong Z, et al. RESKM: a general framework to accelerate large-scale spectral clustering. Pattern Recognit. 2023;137(8):109275. doi:10.1016/j.patcog.2022.109275.
8. Huo Z, Mei G, Casolla G, Giampaolo F. Designing an efficient parallel spectral clustering algorithm on multi-core processors in *Julia*. J Parallel Distrib Comput. 2020;138(4):211–21. doi:10.1016/j.jpdc.2020.01.003.
9. Yan D, Wang Y, Wang J, Wu G, Wang H. Fast communication-efficient spectral clustering over distributed data. IEEE Trans Big Data. 2021;7(1):158–68. doi:10.1109/TBDATA.2019.2907985.

10. Sadikaj Y, Velaj Y, Behzadi S, Plant C. Spectral clustering of attributed multi-relational graphs. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining; 2021 Aug 14–18; Virtual Event. New York, NY, USA: Association for Computing Machinery (ACM); 2021. p. 1431–40.
11. Nie F, Liu C, Wang R, Li X. A novel and effective method to directly solve spectral clustering. IEEE Trans Pattern Anal Mach Intell. 2024;46(12):10863–75. doi:10.1109/TPAMI.2024.3447287.
12. Tang C, Li Z, Wang J, Liu X, Zhang W, Zhu E. Unified one-step multi-view spectral clustering. IEEE Trans Knowl Data Eng. 2023;35(6):6449–60. doi:10.1109/tkde.2022.3172687.
13. He W, Zhang S, Li CG, Qi X, Xiao R, Guo J. Neural normalized cut: a differential and generalizable approach for spectral clustering. Pattern Recognit. 2025;164(4):111545. doi:10.1016/j.patcog.2025.111545.
14. Aghayengejeh NP, Balafar MA, Tanha J, Khasmakhi NN, Minaee S. Multiscale contrastive learning for node clustering based on variational graph auto-encoder. IEEE Trans Neural Netw Learn Syst. 2026;37(2):646–60. doi:10.1109/TNNLS.2025.3610024.
15. Luo D, Ni J, Wang S, Bian Y, Yu X, Zhang X. Deep multi-graph clustering via attentive cross-graph association. In: Proceedings of the 13th International Conference on Web Search and Data Mining; 2020 Feb 3–7; Houston, TX, USA. New York, NY, USA: Association for Computing Machinery (ACM); 2020. p. 393–401.
16. Park C, Kim D, Han J, Yu H. Unsupervised attributed multiplex network embedding. In: Proceedings of the AAAI Conference on Artificial Intelligence; 2020 Feb 7–12; New York, NY, USA. Palo Alto, CA, USA: AAAI Press; 2020. p. 5371–8.
17. Hu W, Fey M, Zitnik M, Dong Y, Ren H, Liu B, et al. Open graph benchmark: datasets for machine learning on graphs. Adv Neural Inf Process Syst. 2020;33:22118–33.