Can Large Language Models Tackle Graph Partitioning?

Yiheng Wu, Ningchao Ge, Yanmin Li, Liwei Qian, Mengna Zhu, Haoyu Yang, Haiwen Chen, Jibing Wu*
Laboratory for Big Data and Decision,
National University of Defense Technology
{wuyh, geningchao, yanminli}@nudt.edu.cn
{qianliwei17, zhumengna16, yanghaoyu20}@nudt.edu.cn
{chenhaiwen13, wujibing}@nudt.edu.cn

Abstract

Large language models (LLMs) demonstrate remarkable capabilities in understanding complex tasks and have achieved commendable performance in graph-related tasks, such as node classification, link prediction, and subgraph classification. These tasks primarily depend on the local reasoning capabilities of the graph structure. However, research has yet to address the graph partitioning task that requires global perception abilities. Our preliminary findings reveal that vanilla LLMs can only handle graph partitioning on extremely small-scale graphs. To overcome this limitation, we propose a three-phase pipeline to empower LLMs for large-scale graph partitioning: coarsening, reasoning, and refining. The coarsening phase reduces graph complexity. The reasoning phase captures both global and local patterns to generate a coarse partition. The refining phase ensures topological consistency by projecting the coarse-grained partitioning results back to the original graph structure. Extensive experiments demonstrate that our framework enables LLMs to perform graph partitioning across varying graph scales, validating both the effectiveness of LLMs for partitioning tasks and the practical utility of our proposed methodology.

1 Introduction

The exponential growth and increasing complexity of graph-structured data, prevalent from social networks to biological systems, present significant analytical challenges (Hüffner et al., 2014; Ni et al., 2022; Ayall et al., 2022). Addressing these challenges often necessitates graph partitioning, a foundational problem in graph computing with broad applications in distributed databases, social network architecture, transportation optimization, circuit design, and power grid management (Cong et al., 1996; Tafreshian and Masoud, 2020; Çatalyürek et al., 2023). The goal is to divide a graph into disjoint subsets where nodes within each subset are strongly connected internally, while minimizing inter-subgraph connections. This is typically pursued under constraints on subgraph size or load balance.

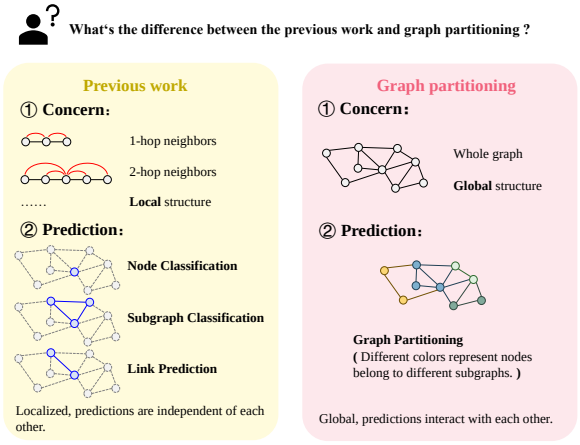

Figure 1: Differences between the graph partitioning task and previous work: (a) Previous work focuses on local answers (in blue), where the surrounding graph data (in grey) may contain the required structural and feature information. (b) Graph partitioning task divides the whole graph into disjoint subgraphs based on the whole graph data.

Large Language Models (LLMs) have demonstrated remarkable reasoning and generalization capabilities in natural language processing tasks (Joshi et al., 2025), prompting research efforts to extend these advantages to graph-structured data processing (Jin et al., 2024). However, a fundamental challenge arises from the inherent mismatch between the sequential processing architecture of LLMs and the non-Euclidean, often highly interconnected, nature of graphs. Unlike text or other sequential data for which LLMs are primarily designed, graphs lack a canonical node ordering, exhibit vastly variable neighborhood structures, and their complex topological properties (including ar-

*Corresponding author

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 15711–15731
November 4-9, 2025

bitrary connectivity and potential long-range dependencies) are not readily digestible by the standard input formats and local context mechanisms typical of LLMs. This intrinsic difficulty in representing and reasoning about global graph topology with sequence-based models has shaped the trajectory of initial research.

Therefore, early studies (He et al., 2024a; Wu et al., 2024; Chen et al., 2024b) primarily leveraged the textual parsing capabilities of LLMs to enhance Graph Neural Networks (GNNs) in handling node attributes, while subsequent work (Zhang et al., 2024; Huang et al., 2024b; Dai et al., 2025) has begun exploring LLMs' perception mechanisms for graph topological structures. These investigations predominantly focus on tasks such as node classification, subgraph classification, and link prediction, which can often be accomplished by analyzing local features of limited nodes or small-scale subgraph information (Liu et al., 2024a; Chen et al., 2024a; Huang et al., 2024a; Li et al., 2024b).

The aforementioned focus of existing LLM-based graph analyses on localized reasoning starkly contrasts with the requirements of tasks demanding a holistic understanding of network topology. Graph partitioning is precisely such a challenge, fundamentally necessitating modeling macroscopic patterns across the entire graph, rather than relying on isolated local features. As illustrated in Figure 1, the objective in graph partitioning (b) is to divide the whole graph based on its overall structure, differing significantly from tasks solvable with more localized information (a). This critical discrepancy between the prevalent local-task-oriented methodologies and the global perception essential for graph partitioning is precisely what motivates our investigation into adapting LLMs for this challenge.

To bridge the gap between the local reasoning capabilities of current LLMs and the global perception required for effective graph partitioning (Li et al., 2024a), we propose a three-stage pipeline: *coarsening*, *reasoning*, and *refining*.

This pipeline unfolds as follows: First, to directly address LLM limitations such as context truncation and the "lost-in-the-middle" issue (Liu et al., 2024b; Firooz et al., 2025), we hierarchically coarsen the original graph into a compact representation that preserves its essential structural properties. Second, this coarsened graph is then encoded into complementary textual sequences—capturing direct neighborhood relations alongside integrated centrality metrics and local patterns. These carefully crafted representations translate the graph's structural information into a format more amenable to LLMs, enabling them to reason about local and global patterns within this abstracted yet structurally informative space. Finally, the initial partitioning derived by the LLM from the coarsened representation is systematically refined. This involves projecting the reasoned divisions back onto the original large-scale graph, thereby translating the model's abstract understanding into a practical, full-graph partition while ensuring topological consistency.

To validate our approach, we construct multi-scale graph partitioning datasets based on Cora and conduct comprehensive experiments. Our findings reveal that while vanilla LLMs struggle with large-scale graphs due to inherent limitations in processing extensive, non-sequential structures (including context length constraints and the 'lost-in-the-middle' phenomenon), our framework markedly empowers them to perceive global graph structures and perform effective partitioning.

Our contributions are summarized as follows:

- We conduct the first systematic investigation into the capabilities of LLMs as native structural reasoners for the graph partitioning task, identifying their potential on extremely small-scale graphs and pinpointing the critical bottlenecks that hinder their application to larger instances.
- To overcome the limitations of LLMs in processing large-scale graphs, we introduce a novel three-stage pipeline, including coarsening, reasoning, and refining.
- Through comprehensive experiments on graphs of varying scales, we demonstrate the feasibility of employing LLMs for graph partitioning and validate the effectiveness of our proposed three-stage pipeline, particularly showcasing its ability to empower LLMs to tackle large-scale instances where they would otherwise struggle or fail.

2 Related Work

2.1 Graph Partitioning

Conventional graph partitioning methodologies formulate the problem as NP-hard discrete optimization tasks, which are then solved through heuristics or approximations (Karypis et al., 1997; Karypis

and Kumar, 1999; Andersen et al., 2006). However, these methodologies are constrained by handcrafted objectives, limited node feature utilization, and scenario-specific applicability.

Neural methodologies have enabled the effective harnessing of node attributes and achieved flexible adaptation to heterogeneous partitioning requirements. MGAE (Wang et al., 2017) leverages node attributes for graph clustering. GAP (Nazi et al., 2019) develops a continuous relaxation of the normalized cut formulation into a differentiable loss function. DMoN (Tsitsulin et al., 2023) optimizes cluster assignments via modularity maximization. Deep-MinCut (Duong et al., 2023) jointly optimizes node embeddings through spectral relaxation of the mincut objective. MPNN (Jung and Keuper, 2023) reformulates integer linear programming constraints into polynomial loss formulations. DGCluster (Bhowmick et al., 2024) addresses undetermined partition counts via modularity maximization in attributed graphs. NeuroCUT (Shah et al., 2024) decouples parameter space from partition counts through reinforcement learning. Nevertheless, these neural approaches necessitate model retraining for distinct graph instances, thereby constraining their cross-graph generalization capabilities.

2.2 LLMs for Graph

The integration of LLMs into graph analytics has undergone a transformative progression, reflecting both technological advancements and persistent limitations. Early frameworks such as Graph-ToolFormer (Zhang, 2023) positioned LLMs as linguistic intermediaries, translating natural language queries like "partition this graph into communities" into predefined algorithmic calls such as k-means. While bridging language and graph processing, these methods reduced LLMs to task dispatchers, outsourcing structural computation to external solvers and ignoring their intrinsic capacity for topological understanding.

Subsequent research diverged into two streams: one enhancing GNNs via LLMs' textual capabilities (Chen et al., 2024b; He et al., 2024b; Chen et al., 2025), and another exploring LLMs' standalone structural reasoning (Huang et al., 2024b; Guo et al., 2024; Yu et al., 2025). The former, including LLMs-as-Enhancers (Liu et al., 2023; Xia et al., 2024; Wei et al., 2024) and LLMs-as-Predictors (Ye et al., 2024; Tang et al., 2024a,b), synergized text and structure by refining node attributes or encoding graphs into prompts. However, such hybrid methods remain constrained to text-attributed graphs (TAGs) and localized tasks (e.g., node classification), with no validation on large graphs or global structural tasks like graph partitioning. The latter stream explored the standalone structural reasoning of LLMs (Chai et al., 2023; Wang et al., 2023; Agrawal et al., 2025), uncovering their nascent ability to perform local tasks such as connectivity checks, but remained confined to microscopic analyzes.

Notably, even Graph-ToolFormer (Zhang, 2023), the sole prior work involving partitioning, merely invoked k-means through linguistic commands, treating LLMs as passive translators rather than active structural interpreters. This highlights a fundamental gap: existing paradigms either disregard structural semantics or restrict reasoning to local patterns.

Our work represents the first effort to harness LLMs as native structural reasoners for graph partitioning. Unlike prior studies that prioritize text-grounded tasks or localized substructures, we explicitly guide LLMs to interpret macroscopic graph semantics through natural language instructions.

3 Preliminary

Graph. A *graph* is defined as $G = (V, E, w)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $E \subseteq \big\{\{v_i, v_j\} \mid v_i, v_j \in V\big\}$ is the set of edges, and $w : E \to \mathbb{R}^+$ assigns a non-negative weight $w(\{v_i, v_j\})$ to each edge. If G is unweighted, $w(\{v_i, v_j\}) = 1$ holds for all $\{v_i, v_j\} \in E$.

Graph Partitioning. For $G = (V, E, w)$, the *graph partitioning* operation decomposes G into k disjoint subgraphs $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$, such that the total weight of edges in E whose incident nodes belong to different subsets is minimized. Each subgraph $S_i = (V_i, E_i)$ satisfies:

$$\bigcup_{i=1}^{k} V_i = V, \tag{1}$$

$$V_i \cap V_j = \emptyset, \quad \forall i \neq j \in [1, k], \tag{2}$$

where $[1, k] \equiv \{1, 2, \ldots, k\}$ denotes the partition index set, and $E_i = \{\{v_p, v_q\} \in E \mid v_p, v_q \in V_i\}$.

Cut. The *cut* quantifies the total weight of edges removed to form disjoint subgraphs $\mathcal{S}$. For a subgraph S_i and its complement $\overline{S_i} = V \setminus S_i$, the cut

is defined as:

$$\text{cut}(S_i, \overline{S_i}) = \sum_{\substack{(v_p,v_q)\in E \\ v_p\in S_i, v_q\in \overline{S_i}}} w(\{v_p, v_q\}). \quad (3)$$

The total edge cut cost for all partitions is:

$$\text{cut}(\mathcal{S}) = \sum_{i=1}^{k} \text{cut}(S_i, \overline{S_i}). \quad (4)$$

Prompts. A *prompt* is a structured instruction mechanism that programs LLMs by augmenting raw input text with task-specific rules, contextual guidelines, and output constraints. It defines the interaction protocol between the user and the LLM, explicitly specifying the form and content of desired outputs while implicitly shaping the model's reasoning process.

4 Methodology

In this section, we provide a detailed implementation of the proposed three-stage pipeline, the architecture of which is illustrated in Figure 2.

4.1 Coarsening

To mitigate the input length limitations and "lost-in-the-middle" challenges of LLMs in processing large-scale graphs, we propose Algorithm 1.

The algorithm initializes the original node set $\mathcal{V}_0 = V$ and edge set $\mathcal{E}_0 = E$ (Line 1). In the Heavy-Edge Matching (HEM) stage, it iteratively merges node pairs connected by the maximum-weight edge $\{u^*, v^*\}$ (Line 4), replacing them with a supernode $\{u^*, v^*\}$ and updating $\mathcal{V}_t$ (Line 5). For each supernode pair $\{p, q\}$, the aggregated edge weights $\widetilde{w}(\{p, q\})$ are computed by summing original edge weights between their constituent nodes, and the edge set $\mathcal{E}_t$ is dynamically updated (Lines 6-9). This process terminates when $|\mathcal{V}_t| - m \leq \delta$, where δ is a preset tolerance threshold.

If the coarsened graph size $|\mathcal{V}_t|$ does not match the target m, spectral clustering clusters supernodes $\mathcal{V}_t$ into m supernodes $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ via the normalized Laplacian matrix $\mathcal{L}$ (Lines 14-15), and updates $\mathcal{V}_t$ and $\mathcal{E}_t$ to reflect the coarsened graph (Lines 16-17). The final coarsened graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ preserves key topological features through edge weight accumulation. The overall time complexity is $O(T|E| + m|\mathcal{V}_t|^2)$, dominated by HEM iterations ($O(T|E|)$) and spectral clustering ($O(m|\mathcal{V}_t|^2)$).

Algorithm 1 Hierarchical Graph Coarsening

Input: Graph $G = (V, E, w)$, target size m, tolerance threshold δ
Output: Coarsened graph $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{w})$

1: Initialize $\mathcal{V}_0 \leftarrow V$, $\mathcal{E}_0 \leftarrow E$, $t \leftarrow 0$
2: // HEM:
3: **while** $|\mathcal{V}_t| > m + \delta$ **do**
4: $\quad \{u^*, v^*\} \leftarrow \arg\max_{\{u,v\}\in\mathcal{E}_t} w(\{u, v\})$
5: $\quad \mathcal{V}_{t+1} \leftarrow \mathcal{V}_t \setminus \{u^*, v^*\} \cup \{\{u^*, v^*\}\}$
6: $\quad \mathcal{E}_{t+1}, \widetilde{w} \leftarrow \text{MergeEdges}(\mathcal{E}_t, \{u^*, v^*\}, w)$
7: $\quad$ **for** each $\{p, q\} \in \mathcal{V}_t \times \mathcal{V}_t$ **do**
8: $\quad\quad \widetilde{w}(\{p, q\}) \leftarrow \sum_{\substack{u\in p, v\in q \\ \{u,v\}\in\mathcal{E}_t}} w(\{u, v\})$
9: $\quad$ **end for**
10: $\quad t \leftarrow t + 1, T \leftarrow t$
11: **end while**
12: // Spectral Clustering:
13: **if** $|\mathcal{V}_t| \neq m$ **then**
14: $\quad \mathcal{L} \leftarrow \text{BuildLaplacian}(\mathcal{V}_t, \mathcal{E}_t, \widetilde{w})$
15: $\quad \{\mathcal{C}_1, \ldots, \mathcal{C}_m\} \leftarrow \text{SpectralCluster}(\mathcal{L}, m)$
16: $\quad \mathcal{V}_t \leftarrow \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$
17: $\quad \mathcal{E}_t, \widetilde{w} \leftarrow \text{MergeEdgesAll}(\mathcal{E}_t, \mathcal{V}_t, \widetilde{w})$
18: **end if**
19: $\widetilde{V} \leftarrow \mathcal{V}_t$, $\widetilde{E} \leftarrow \mathcal{E}_t$, $\widetilde{w} \leftarrow \widetilde{w}$
20: **return** $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{w})$

4.2 Reasoning

In order to execute graph partitioning tasks with precision, it is imperative to furnish both local and global information of the graph. This is done to facilitate LLMs in comprehending the graph's structural information in its entirety. To facilitate the expeditious transfer and deployment of diverse LLMs across a range of graph datasets, we employ prompt engineering techniques to enhance capabilities of LLMs in inference.

Conventional prompts for graph data have historically concentrated on acquiring local feature information based on a limited number of nodes, a consequence of the inherent characteristics of the task. Consequently, we have devised novel prompts to facilitate the comprehension of the graph's structural characteristics at an aggregate level by models. Specifically, two types of prompts have been designed: firstly, 1-hop neighbor sequences, which describe the basic structural information of the coarse graph; and secondly, enhanced feature sequences, which describe both local and global feature information of the graph. Through this approach, we not only provide local information about the graph, but also supplement it with global

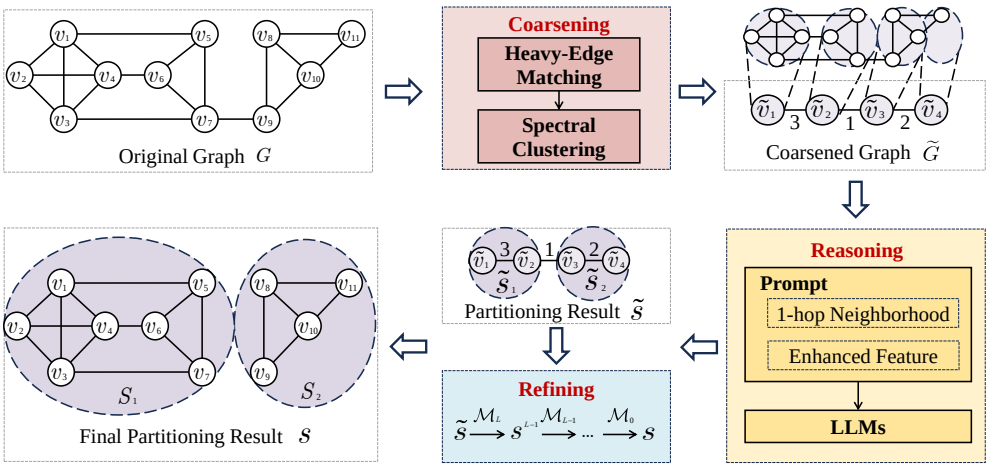

Figure 2: The overall structure of our proposed pipeline. Firstly, in the **coarsening** stage, we extract the key features of the graph and construct a coarsened graph. Secondly, in the **reasoning** stage, the coarsened graph is transformed into a textual serialization that can be understood by LLMs. Based on this serialization, we construct prompts to induce the LLMs to perform reasoning about the partitioning. Finally, in the **refining** stage, we employ a layer-by-layer mapping strategy to project the partitioning results back onto the original graph.

information, enabling large models to comprehensively understand and process the graph structure. The prompts are shown in Appendix A.

4.2.1 1-hop Neighborhood Sequences

In contrast to conventional methodologies that exclusively encompass neighbor node information, our neighbor sequence integrates both node information and the corresponding edge weights. Specifically, for a node v_i, its neighbor sequence is composed of pairs of 1-hop neighbors and their associated edge weights:

- **Neighbor nodes**: $v_j, v_l, \ldots, v_r$;
- **Edge weights**: $w_{ij}, w_{il}, \ldots, w_{ir}$.

This formulation provides a richer description of the topological structure by incorporating both node identities and edge weights, enabling the model to better understand the graph's overall structural characteristics.

4.2.2 Enhanced Feature Sequences

To further enrich the model's understanding of the graph, we introduce enhanced feature sequences. These sequences include both local neighborhood features and global centrality features. Below are the formulas used to compute these features.

Global Centrality Features. We define four node centrality metrics at network scale:

$$C_d(v_i) = \frac{\deg(v_i)}{n-1}, \tag{5}$$

$$C_b(v_i) = \sum_{s \neq v_i \neq t} \frac{\sigma_{st}(v_i)}{\sigma_{st}}, \tag{6}$$

$$C_c(v_i) = \frac{n-1}{\sum_{j \neq i} d(v_i, v_j)}, \tag{7}$$

$$C_e(v_i) = \frac{1}{\lambda} \sum_{j \in N(v_i)} w_{ij} C_e(v_j), \tag{8}$$

where $\deg(v_i)$ is node degree, σ_{st} counts shortest paths between nodes s and t, $d(v_i, v_j)$ denotes weighted shortest-path distance, and λ is the largest adjacency eigenvalue. C_d (degree centrality) measures direct connection density, C_b (betweenness centrality) detects bridge nodes between communities, C_c (closeness centrality) evaluates global reachability efficiency, and C_e (eigenvector centrality) captures influence in global structures.

Local Neighborhood Features. Five statistics characterize local connectivity:

$$\phi_1(v_i) = |N_1(v_i)|, \tag{9}$$

$$\phi_2(v_i) = |N_2(v_i) \setminus N_1(v_i)|, \tag{10}$$

$$\phi_3(v_i) = \frac{1}{\phi_1} \sum \deg(v_j), \tag{11}$$

$$\phi_4(v_i) = \text{median}(\deg(v_j)), \tag{12}$$

$$\phi_5(v_i) = \sqrt{\frac{1}{\phi_1} \sum (\deg(v_j) - \phi_3)^2}, \tag{13}$$

where $N_k(v_i)$ denotes k-hop neighbors, and $v_j \in N_1(v_i)$. ϕ_1 (number of 1-hop neighbors)

and ϕ_2 (number of 2-hop neighbors) quantify direct/indirect connectivity, ϕ_3-ϕ_5 (mean/median/std of neighbor degrees) reveal neighborhood quality and consistency.

Normalization. All features undergo Z-score normalization to avoid magnitude differences.

4.3 Refining

The refining phase maps coarse partitions back to the original graph through index-guided reconstruction. During coarsening (Section 4.1), each layer l maintains a mapping table $\mathcal{M}_l$ linking supernodes to their subnodes at layer $l-1$. Labels propagate from the coarsest layer L through $\mathcal{M}_L \to \cdots \to \mathcal{M}_0$.

Example For a 3-layer hierarchy ($L = 2$):

- **Layer 0**: $\{v_1, \ldots, v_6\}$
- **Layer 1**: $S_1^{(1)}=\{v_1, v_2\},\ S_2^{(1)}=\{v_3, v_4\},$ $S_3^{(1)}=\{v_5\},\ S_4^{(1)}=\{v_6\}$
- **Layer 2**: $S_1^{(2)}=\{S_1^{(1)}, S_3^{(1)}\},\ S_2^{(2)}=\{S_2^{(1)}, S_4^{(1)}\}$

refining proceeds as:

- Layer 2 partition $\{S_1^{(2)}, S_2^{(2)}\}$
- $\xrightarrow{\mathcal{M}_2}$ Layer 1: $\{S_1^{(1)}, S_3^{(1)}\}, \{S_2^{(1)}, S_4^{(1)}\}$
- $\xrightarrow{\mathcal{M}_1}$ Layer 0: $\{v_1, v_2, v_5\}, \{v_3, v_4, v_6\}$

This index-guided reconstruction preserves constraints with $O(L)$ complexity. This hierarchical mapping preserves balance and connectivity constraints through index-guided reconstruction, ensuring bijective correspondence between coarse and fine layers with $O(L)$ complexity.

5 Experiments

5.1 Experimental Setup

This section provides a comprehensive overview of the experimental configuration, including the **models and baselines**, **parameter settings**, **datasets**, and **evaluation metrics**.

Models and Baselines. To explore whether LLMs possess inherent capabilities for graph partitioning, we selected several baseline methods for comparison. These include a simple baseline, **k-means** (McQueen, 1967), as well as established graph partitioning approaches: **METIS** (Karypis and Kumar, 1998), a standard partitioner; **GAP** (Dwivedi et al., 2023), a GNN-based algorithm; and **MinCutPool** (Bianchi et al., 2020), another GNN-based approach. We also evaluated three LLMs directly on the original graph data: **Qwen2.5-Plus**, **Qwen2.5-Max**, and **DeepSeek-R1**. Prompts for these models were constructed from the graph data using the strategy described in Section 4.2. To further validate the effectiveness of our proposed framework, we applied our full three-stage pipeline (coarsening, reasoning, and refining) to Qwen2.5-Max and DeepSeek-R1. The resulting variants are denoted as **Ours-Qwen** and **Ours-DeepSeek**, respectively, in subsequent evaluations.

Parameter Settings. The experiments were conducted using the APIs provided by Qwen (Bai et al., 2023) and DeepSeek (Team, 2024), with their default parameter configurations. Specifically, DeepSeek-R1 supports a context length of up to 64K tokens, with a maximum output length of 8K tokens. Qwen2.5-Max has a maximum context length of 32K tokens and a generation length of 8K tokens. Qwen2.5-Plus offers enhanced contextual processing capabilities, supporting up to 132K input tokens and 8.2K output tokens. These parameter details help clarify the capabilities and limitations of each model. We set $m = 50$ clusters and $\delta = 1$ to speed up inference while maintaining consistent supernode counts. We evaluate performance across $k = 2$, 5, and 10 partitions.

Datasets. We conducted experiments on multiple graph datasets to evaluate both the limitations of direct LLM-based graph processing and the effectiveness of our proposed framework. As primary testbed, we used the Cora citation network (Sen et al., 2008), focusing on its largest connected component, which contains 2,485 nodes (papers) and 5,069 edges (citations). To systematically study the impact of structural and length-related challenges, specifically the 'lost-in-the-middle' phenomenon and input truncation, we constructed six Cora subgraph datasets using hierarchical random walk sampling, ranging from 50 to 2,485 nodes.

To further assess the scalability and practical utility of our method, we also experimented with three larger real-world graphs that exceed the direct processing capacity of vanilla LLMs such as Qwen2.5-Plus, Qwen2.5-Max, and DeepSeek-R1: **Facebook** (McAuley and Leskovec, 2012), a social network with high average degree and dense connectivity; **Anaheim** (McNally et al., 1999), a transportation network with sparse connectivity and very low clustering; and **FB15k-237** (Toutanova and Chen, 2015), a large-scale knowledge graph with complex relational semantics. These graphs enable

comparison against well-established partitioning methods like METIS, GAP, and MinCutPool. Note that **Cora-full** is included in both the subgraph series and the extended benchmark set.

Table 1 provides structural statistics for all datasets, including node count ($|V|$), edge count ($|E|$), average degree (AD), and clustering coefficient (CC). For each Cora subgraph size, we report the best result from five independent runs. All datasets, along with sampling trajectories and encoding protocols, will be publicly released to ensure reproducibility.

Dataset	$\|V\|$	$\|E\|$	**AD**	**CC**
Cora-50	50	85	3.40	0.384
Cora-100	100	164	3.27	0.205
Cora-200	200	333	3.32	0.232
Cora-500	500	893	3.57	0.220
Cora-1000	1,000	2,033	4.06	0.233
Cora-full	2,485	5,069	4.08	0.238
Facebook	4,039	88,234	21.85	0.606
Anaheim	416	914	2.20	0.001
FB15k-237	13,680	99,717	19.49	0.169

Table 1: Structural statistics of the experimental datasets. Note: $|V|$ = Node count, $|E|$ = Edge count, AD = Average Degree, CC = Clustering Coefficient.

Evaluation Metrics. We evaluate the quality of graph partitioning using four widely adopted objectives from Shah et al., 2024: k-MinCut, Normalized Cut, Balanced Cut, and Sparsest Cut. These metrics assess partitioning quality from distinct perspectives, including (1) the number of inter-partition edges, (2) connection balance, (3) subgraph size balance, and (4) node distribution sparsity. Lower values of these metrics indicate better partitioning performance. Detailed formulations of these objectives are provided in Appendix B.

5.2 Overall Results

We evaluate the graph partitioning performance of three vanilla LLMs (Qwen2.5-Plus, Qwen2.5-Max, DeepSeek-R1) and our proposed three-stage pipeline (Ours-Qwen, Ours-DeepSeek) across all datasets, which range from small-scale Cora subgraphs to large real-world graphs. The evaluation follows a two-phase strategy: (1) *Feasibility Phase*: We assess whether LLMs can perform graph partitioning at all by comparing them against the simple baseline **k-means** on small graphs; (2) *Competitiveness Phase*: We evaluate whether our framework can scale effectively and compete with, or even surpass, both classical graph partitioning algorithms (such as **METIS**) and learning-based approaches (including **GAP** and **MinCutPool**) on large real-world graphs.

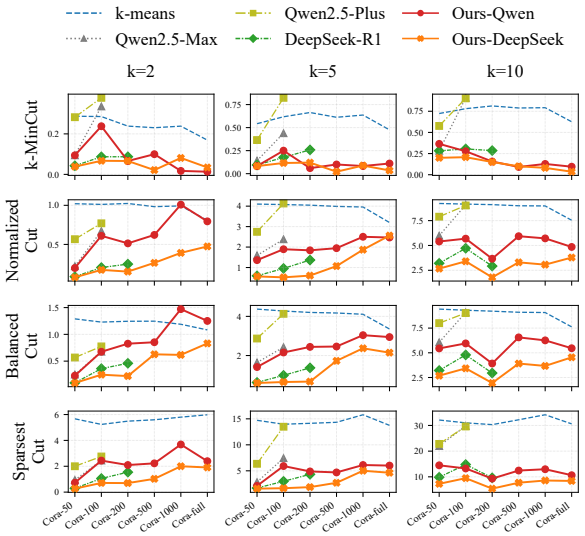

Figure 3: Performance across Cora subgraphs (Lower values indicate better performance).

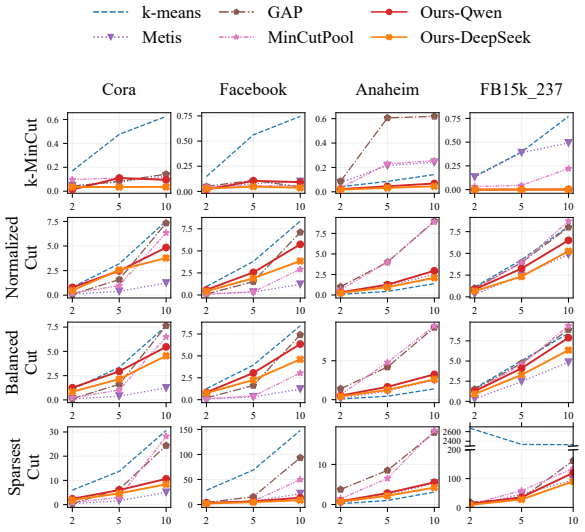

Figure 4: Comparison with baselines on large-scale graphs across multiple partitioning objectives (Lower values indicate better performance).

The results are visualized in two complementary figures, with detailed tabular data provided in Appendix D: Figure 3 illustrates performance trends across graphs of increasing scale (from Cora-50 to Cora-full), highlighting the limitations of direct LLM processing and the consistent gains achieved by our pipeline, especially when compared to **k-means**. Figure 4 compares our method against classical baselines (**METIS** and **k-means**) and learning-based approaches (**GAP** and **MinCut-**

Pool) on large-scale graphs including Facebook, Anaheim, and FB15k-237. The results reveal nuanced trade-offs across different optimization objectives and graph domains.

Based on these results, we derive the following key observations:

(1) LLMs demonstrate viability for graph partitioning, but only in ultra-small-scale scenarios. DeepSeek-R1 can perform graph partitioning tasks on graphs with no more than 200 nodes, while Qwen2.5-Plus and Qwen2.5-Max handle graphs containing up to 100 nodes. Within these limits, LLMs achieve partitioning quality comparable to or better than k-means on minimal datasets. Case studies in Appendix C.1 confirm LLMs' inherent structural reasoning capabilities, validating their potential while also highlighting their severe scalability constraints.

(2) Our three-stage pipeline significantly enhances LLM-based graph partitioning. As revealed in Appendix C.2, by introducing a coarsening stage that filters redundant information and preserves critical structural features, our framework enables LLMs to process graphs far beyond their native context limits. As shown in Figure 3, Ours-Qwen and Ours-DeepSeek maintain stable, high-quality partitioning even at the Cora-full scale (2,485 nodes), where direct LLM input completely fails. This structured refining mechanism is key to unlocking LLMs' potential for graph-structured tasks.

(3) Our method not only scales effectively, but also competes with and often surpasses established graph partitioning algorithms. As shown in Figure 4, our pipeline consistently outperforms learning-based baselines (GAP and MinCutPool) across all datasets and metrics, demonstrating a superior ability to learn effective partitioning strategies from data. Compared to the classical METIS algorithm, our method achieves competitive or superior performance on **k-MinCut** and **Sparsest Cut** objectives. This is particularly evident on the Anaheim (transportation) and FB15k-237 (knowledge graph) datasets, indicating that our data-driven approach can discover high-quality solutions that complement or even surpass handcrafted heuristics, especially for non-standard objectives.

(4) The framework exhibits strong cross-domain generalization. Our method demonstrates robust performance across social networks (Facebook), infrastructure graphs (Anaheim), and semantic knowledge graphs (FB15k-237). Each of these graph types exhibits vastly different structural properties, including density, clustering, and relational complexity. The consistent performance confirms that our pipeline is not tailored to any specific graph type. This generalization ability, combined with its scalability, positions our method as a flexible, LLM-powered alternative to traditional graph partitioning systems.

5.3 Evaluation of Coarsening Stage

The **number of supernodes** in the coarsened graph, generated during the coarsening stage, impacts graph partitioning performance. While more complex coarsened graphs may contain richer feature information beneficial for partitioning, the resulting complex context could also hinder performance.

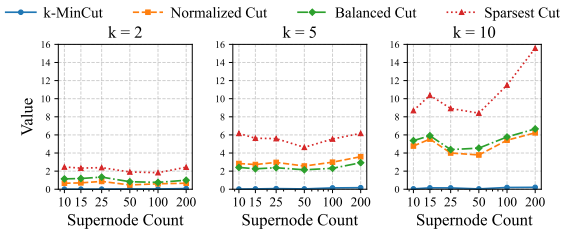

Figure 5: Performance comparison of different supernode counts (Lower values indicate better performance).

Figure 5 shows the performance of Ours-DeepSeek on the Cora-full dataset across varying supernode counts. Metrics such as Normalized Cut, Balanced Cut, and Sparsest Cut exhibit a clear U-shaped trend, with the lowest points predominantly occurring at and near the 50 supernodes. Similarly, k-MinCut shows a U-shaped trend when excluding extreme low supernode count settings (10, 15, 25), also achieving optimal performance at 50 supernodes. The strong performance of k-MinCut at very low supernode counts (e.g., 10 or 15) is attributed to HEM and spectral clustering, which preserve partition balance under high compression. At this time, the decision-making effect of large models is limited and close to random matching. However, as coarsened nodes increase, k-MinCut's performance aligns with other metrics, further supporting the choice of 50 as the optimal coarsening level. As the task difficulty increases with higher target partitions k, the U-shaped trends for all metrics become sharper, emphasizing the need for careful selection of coarsened node counts to balance structural fidelity and input complexity.

5.4 Evaluation of Reasoning Stage

To assess the role of structural and feature information in reasoning stage, we evaluate three **prompt strategies** on the k=10 graph partitioning task using the Cora-full dataset with **Ours-DeepSeek** as the baseline: (1) *only1hop* (1-hop neighborhood only), (2) *onlyEF* (enhanced features only), and (3) *mix* (combined input).

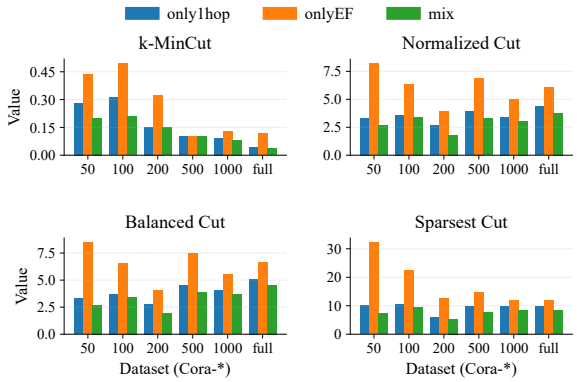

Figure 6: Performance comparison of different prompt methods across datasets (Lower values indicate better performance).

Results in Figure 6 show that *onlyEF* underperforms significantly across all metrics, underscoring the necessity of topological modeling. The *mix* strategy outperforms *only1hop*, particularly on larger graphs, demonstrating improved robustness and balance when semantic features supplement structural inputs. These findings confirm that structural information dominates LLM-based graph reasoning, while node features serve as a valuable auxiliary signal for refined partitioning. The case analysis in Appendix C.3 demonstrates the differences in the reasoning processes brought about by these three prompt strategies, further supporting this conclusion.

5.5 Evaluation of Refining Stage

The **size of graph** determines the mapping accuracy of the division results in the refining stage. To evaluate the effectiveness of the refining strategy, we compare the performance improvement of Our-DeepSeek against the baseline method DeepSeek-R1 on graph datasets of varying sizes (Cora50, Cora100, Cora200). As shown in Figure 7, the improvement achieved by Our-DeepSeek becomes more pronounced as the graph size increases, with particularly notable gains observed under the Normalized Cut and Balanced Cut metrics. Moreover, as observed in the experiments in Section 5.2, when the number of nodes exceeds 200, DeepSeek-R1 fails to proceed with the partitioning task, while Our-DeepSeek remains stable and effective, demonstrating superior scalability and practicality. In conclusion, the proposed three-stage pipeline effectively enhances the model's capability to handle large-scale graphs through the coarsening and refining mechanisms, achieving consistent performance gains across multiple partitioning criteria.

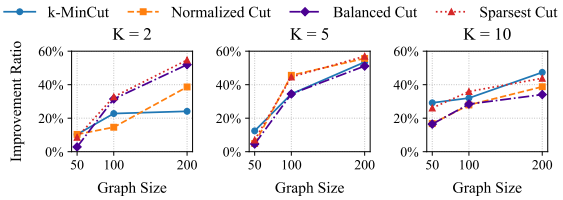

Figure 7: Improvement ratio in different graph size (Higher values indicate better performance).

6 Conclusion

This work investigates the application of LLMs to graph partitioning and proposes a three-stage pipeline comprising *coarsening*, *reasoning*, and *refining*. We found that LLMs can effectively perform graph partitioning tasks on small-scale graph data, and their performance is close to or even better than traditional methods such as k-means. However, as the size of the graph increases, LLMs are unable to complete the graph partitioning task. The three-stage pipeline we designed enables LLMs to scale up to larger scale graph data for partitioning tasks. Specifically, our method extracts key structural information through graph coarsening and integrates both structural and attribute features into sequence-based representations. This enables LLMs to indirectly perceive global graph topology and perform partitioning on graphs of arbitrary scale. The experimental results confirm the efficacy of our approach in datasets of different sizes.

Limitations and Future Works

We acknowledge the limitations of this work.

(1) The proposed three-stage pipeline mitigates the challenge of LLMs struggling to effectively perceive the global topology of large-scale graph data by constructing a coarsened graph. However, this approach introduces additional feature loss during the coarsening process, which limits the performance ceiling of our method. Furthermore, the number of supernodes in the coarsened graph significantly impacts the quality of graph partitioning. In future work, we aim to develop a more precise

and robust method for graph data transformation that minimizes information loss while preserving the global structure of the original graph.

(2) During the refining stage, to avoid interference from local optimization algorithms on the LLMs' partitioning results, we map the partitioning results from the coarsened graph back to the original graph. While this ensures that the LLMs' inherent partitioning logic is preserved, it also leads to relatively coarse-grained results when scaled back to the original graph's dimensions. To address this limitation, we plan to explore new task formulations in future work, enabling LLMs to iteratively refine their own partitioning results. This approach aims to achieve finer-grained and more accurate graph partitioning while maintaining the integrity of the original graph structure.

(3) While our experimental validation demonstrates effectiveness across graphs ranging from 50 to 13,680 nodes with approximately 50 supernodes, the current framework exhibits inherent scalability boundaries when addressing extreme-scale scenarios requiring significantly higher partition counts (e.g., 100-way partitioning) or processing graphs exceeding 100K nodes. The single-level coarsening approach, while effective within our tested range, may face computational and representational challenges when directly scaled to these extreme conditions. Future work will explore hierarchical decomposition strategies to extend the framework's applicability, including recursive partitioning approaches for high-partition-count tasks and multilevel graph processing paradigms for hyperscale graphs. These extensions would maintain the core advantages of our LLM-guided partitioning methodology while addressing the scalability constraints identified in this work.

(4) This work adopts a zero-shot prompting framework to fundamentally investigate the intrinsic capability of untrained LLMs in perceiving global graph structures—a core research objective that establishes the baseline reasoning capacity of foundation models for graph partitioning tasks. While this approach delivers strong generalizability across diverse graph topologies without task-specific adaptation, we recognize that integrating task-aware strategies such as supervised fine-tuning or chain-of-thought prompting holds significant promise for further enhancing performance in specialized application domains. Future research will productively explore synergistic frameworks that preserve the zero-shot foundation's computational efficiency and broad applicability while incorporating targeted adaptations, thereby creating a continuum of solutions spanning from general-purpose reasoning to domain-optimized partitioning methodologies.

Acknowledgments

This work was supported by the National Science Foundation of China under grant (72401288), National Funding Project for Postdoctoral Researchers of China under grant (GZC20233528), Hunan Provincial Natural Science Foundation of China under grant (2025JJ60463). DeepSeek-R1 was used to enhance the readability of some of the text and improve the language of this paper, after the content was first added manually. All material was then checked manually before submission.

References

Palaash Agrawal, Shavak Vasania, and Cheston Tan. 2025. Can llms perform structured graph reasoning tasks? In *Pattern Recognition*, pages 287–308, Cham. Springer Nature Switzerland.

Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486.

Tewodros Alemu Ayall, Huawen Liu, Changjun Zhou, Abegaz Mohammed Seid, Fantahun Bogale Gereme, Hayla Nahom Abishu, and Yasin Habtamu Yacob. 2022. Graph computing systems and partitioning techniques: A survey. *IEEE Access*, 10:118523–118550.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Aritra Bhowmick, Mert Kosan, Zexi Huang, Ambuj Singh, and Sourav Medya. 2024. Dgcluster: A neural framework for attributed graph clustering via modularity maximization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11069–11077.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2020. Spectral clustering with graph neural

networks for graph pooling. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 874–883. PMLR.

Ümit Çatalyürek, Karen Devine, Marcelo Faraj, Lars Gottesbüren, Tobias Heuer, Henning Meyerhenke, Peter Sanders, Sebastian Schlag, Christian Schulz, Daniel Seemaier, and Dorothea Wagner. 2023. More recent advances in (hyper)graph partitioning. *ACM Comput. Surv.*, 55(12).

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *CoRR*, abs/2310.05845.

Haibo Chen, Xin Wang, Zeyang Zhang, Haoyang Li, Weigao Wen, Ling Feng, and Wenwu Zhu. 2025. Curriculum GNN-LLM alignment for text-attributed graphs.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. 2024a. Exploring the potential of large language models (llms)in learning on graphs. *SIGKDD Explor. Newsl.*, 25(2):42–61.

Zhikai Chen, Haitao Mao, Jingzhe Liu, Yu Song, Bingheng Li, Wei Jin, Bahare Fatemi, Anton Tsitsulin, Bryan Perozzi, Hui Liu, and Jiliang Tang. 2024b. Text-space graph foundation models: Comprehensive benchmarks and new insights. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

J. Cong, W. Juan Labio, and N. Shivakumar. 1996. Multiway vlsi circuit partitioning based on dual net representation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(4):396–409.

Xinnan Dai, Haohao Qu, Yifei Shen, Bohang Zhang, Qihao Wen, Wenqi Fan, Dongsheng Li, Jiliang Tang, and Caihua Shan. 2025. How do large language models understand graph patterns? a benchmark for graph pattern comprehension. In *The Thirteenth International Conference on Learning Representations*.

Chi Thang Duong, Thanh Tam Nguyen, Trung-Dung Hoang, Hongzhi Yin, Matthias Weidlich, and Quoc Viet Hung Nguyen. 2023. Deep mincut: Learning node embeddings by detecting communities. *Pattern Recognition*, 134:109126.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2023. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48.

Hamed Firooz, Maziar Sanjabi, Wenlong Jiang, and Xiaoling Zhai. 2025. Lost-in-distance: Impact of contextual proximity on llm performance in graph tasks. *Preprint*, arXiv:2410.01985.

Zirui Guo, Lianghao Xia, Yanhua Yu, Yuling Wang, Zixuan Yang, Wei Wei, Liang Pang, Tat-Seng Chua, and Chao Huang. 2024. Graphedit: Large language models for graph structure learning. *CoRR*, abs/2402.15183.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024a. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *ICLR*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024b. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems*, volume 37, pages 132876–132907. Curran Associates, Inc.

Chao Huang, Xubin Ren, Jiabin Tang, Dawei Yin, and Nitesh Chawla. 2024a. Large language models for graphs: Progresses and directions. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 1284–1287, New York, NY, USA. Association for Computing Machinery.

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2024b. Can LLMs effectively leverage graph structural information through prompts, and why? *Transactions on Machine Learning Research*.

Falk Hüffner, Christian Komusiewicz, Adrian Liebtrau, and Rolf Niedermeier. 2014. Partitioning biological networks into highly connected clusters with maximum edge coverage. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(3):455–467.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8622–8642.

Aditya Joshi, Raj Dabre, Diptesh Kanojia, Zhuang Li, Haolan Zhan, Gholamreza Haffari, and Doris Dippold. 2025. Natural language processing for dialects of a language: A survey. *ACM Comput. Surv.*, 57(6).

Steffen Jung and Margret Keuper. 2023. Learning to solve minimum cost multicuts efficiently using edge-weighted graph convolutional neural networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 485–501, Cham. Springer International Publishing.

George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1997. Multilevel hypergraph partitioning: application in vlsi domain. In *Proceedings of the 34th Annual Design Automation Conference*, DAC '97, page 526–529, New York, NY, USA. Association for Computing Machinery.

George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392.

George Karypis and Vipin Kumar. 1999. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference*, DAC '99, page 343–348, New York, NY, USA. Association for Computing Machinery.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2024a. A survey of graph meets large language model: progress and future directions. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24.

Zhonghang Li, Lianghao Xia, Jiabin Tang, Yong Xu, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024b. Urbangpt: Spatio-temporal large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 5351–5362, New York, NY, USA. Association for Computing Machinery.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2023. One for all: Towards training one graph model for all classification tasks. *CoRR*, abs/2310.00149.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2024a. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Julian McAuley and Jure Leskovec. 2012. Learning to discover social circles in ego networks. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 539–547, Red Hook, NY, USA. Curran Associates Inc.

Mark G. McNally, S. P. Mattingly, J. E. Moore, H.-H. Hu, C. A. Maccarley, and R. Jayakrishnan. 1999. Evaluation of Anaheim adaptive control field operational test: Institutional issues. In *Transportation Research Record: Journal of the Transportation Research Board*, volume 1683, pages 1–8, Washington, DC, USA. Transportation Research Board. Presented at the 78th Annual Meeting of the Transportation Research Board, January 1999.

James B McQueen. 1967. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, pages 281–297.

Azade Nazi, Will Hang, Anna Goldie, Sujith Ravi, and Azalia Mirhoseini. 2019. Gap: Generalizable approximate graph partitioning framework. *arXiv preprint arXiv:1903.00614*.

Qiufen Ni, Jianxiong Guo, Weili Wu, Huan Wang, and Jigang Wu. 2022. Continuous influence-based community partition for social networks. *IEEE Transactions on Network Science and Engineering*, 9(3):1187–1197.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Mag.*, 29(3):93–106.

Rishi Shah, Krishnanshu Jain, Sahil Manchanda, Sourav Medya, and Sayan Ranu. 2024. Neurocut: A neural approach for robust graph partitioning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 2584–2595, New York, NY, USA. Association for Computing Machinery.

Amirmahdi Tafreshian and Neda Masoud. 2020. Trip-based graph partitioning in dynamic ridesharing. *Transportation Research Part C: Emerging Technologies*, 114:532–553.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024a. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 491–500, New York, NY, USA. Association for Computing Machinery.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024b. Higpt: Heterogeneous graph language model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 2842–2853, New York, NY, USA. Association for Computing Machinery.

DeepSeek-AI Team. 2024. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. Technical report, GitHub. First open-source RL-based reasoning model without SFT.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2023. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 24(127):1–21.

Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and*

Knowledge Management, CIKM '17, page 889–898, New York, NY, USA. Association for Computing Machinery.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems*, volume 36, pages 30840–30861. Curran Associates, Inc.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, page 806–815, New York, NY, USA. Association for Computing Machinery.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. 2024. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. In *Advances in Neural Information Processing Systems*, volume 37, pages 127129–127153. Curran Associates, Inc.

Lianghao Xia, Ben Kao, and Chao Huang. 2024. OpenGraph: Towards open graph foundation models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2365–2379, Miami, Florida, USA. Association for Computational Linguistics.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *EACL (Findings)*, pages 1955–1973.

Shuo Yu, Yingbo Wang, Ruolin Li, Guchun Liu, Yanming Shen, Shaoxiong Ji, Bowen Li, Fengling Han, Xiuzhen Zhang, and Feng Xia. 2025. Graph2text or graph2token: A perspective of large language models for graph learning. *Preprint*, arXiv:2501.01124.

Jiawei Zhang. 2023. Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt. *arXiv preprint arXiv:2304.11116*.

Yizhuo Zhang, Heng Wang, Shangbin Feng, Zhaoxuan Tan, Xiaochuang Han, Tianxing He, and Yulia Tsvetkov. 2024. Can LLM graph reasoning generalize beyond pattern memorization? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2289–2305, Miami, Florida, USA. Association for Computational Linguistics.

A Prompt Design

Our prompt is illustrated in Figure 8. The graph partitioning task to be performed is rigorously specified in the Task Description. The Adjacency List and Enhanced Node Features correspond to the 1-hop neighbor sequences constructed in Section 4.2.1 and the enhanced feature sequences developed in Section 4.2.2, respectively. Through the Output Format Instruction and Task components, we provide both exemplars and textual descriptions to guide the LLMs' output generation. As demonstrated in the Output, the LLMs successfully generate subgraph assignments for all nodes in the graph, accompanied by node-wise justifications for these assignments.

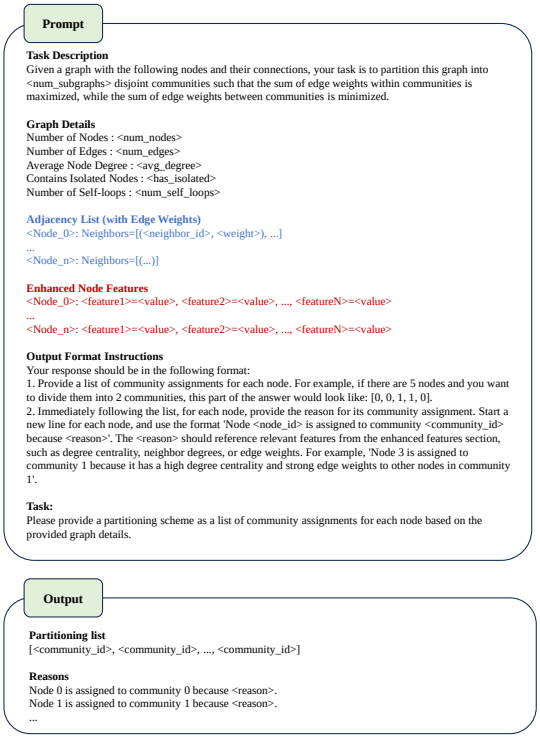

Figure 8: Illustration of how 1-hop neighborhood sequences and enhanced feature sequences are incorporated into prompts to guide large model inference.

B Evaluation Metrics

We use four widely adopted partitioning objectives from Shah et al., 2024 to evaluate the quality of graph partitioning: k-MinCut, Normalized Cut, Balanced Cut, and Sparsest Cut. These objectives assess the quality of graph partitioning from different perspectives, specifically focusing on (1) the number of cuts, (2) the balance of connections, (3) the balance of subgraph sizes, and (4) the sparsity of node distribution.

k-MinCut evaluates the number of edges cut between all subgraphs after partitioning.:

$$\text{k-MinCut}(\mathcal{S}) = \sum_{i=1}^{k} \frac{k \cdot \text{cut}(S_i, \bar{S}_i)}{\sum_{(v_p, v_q) \in E} e(v_p, v_q)}, \qquad (14)$$

where $\mathcal{S} = \{S_i\}$ follows Equation 1, and cut is from Equation 4. Smaller value indicates less edge loss.

Normalized Cut assesses whether the connection strength within and between subgraphs is balanced:

$$\begin{aligned} \text{Ncut}(\mathcal{S}) &= \sum_{i=1}^{k} \frac{\text{cut}(S_i, \bar{S}_i)}{\text{vol}(S_i)}, \\ \text{where } \text{vol}(S_i) &= \sum_{v_p \in S_i} \sum_{v_q \in V} e(v_p, v_q). \end{aligned} \tag{15}$$

Lower value indicates more balanced connections between partitions.

Balanced Cut focuses on the size of subgraphs:

$$\begin{aligned} \text{BalancedCut}(\mathcal{S}) = \sum_{l=1}^{k} \Bigg[& \frac{\text{cut}(S_i, \overline{S_i})}{\text{vol}(S_i)} \\ & + \frac{(|S_i| - |V|/k)^2}{|V|^2} \Bigg]. \end{aligned} \tag{16}$$

Smaller value indicates more balanced cluster sizes.

Sparsest Cut evaluates the condition of the subgraph with the lowest node density:

$$\begin{aligned} \text{SparseCut}(\mathcal{S}) &= \sum_{i=1}^{k} \phi(S_i, \bar{S}_i), \\ \text{where } \phi(S, \bar{S}) &= \frac{\text{cut}(S, \bar{S})}{\min(|S|, |\bar{S}|)}. \end{aligned} \tag{17}$$

Smaller value indicates a less extreme distribution of subgraphs.

C Case Studies

In this chapter, we employ case studies to investigate three pivotal issues: (1) Do Large Language Models (LLMs) possess the capability to perform graph partitioning? (2) Why are LLMs unable to directly execute graph partitioning tasks on large-scale graph data? (3) What influence does the construction of prompts based on varying information have on the reasoning processes of LLMs?

C.1 Do LLMs have the ability to perform graph partitioning?

As illustrated in Table 2, we provide several examples of successfully executed graph partitioning. It is evident that LLMs take into account the graph structure, node characteristics, and the allocation of nodes to subgraphs, assigning nodes from the input graph to the target set of nodes without overlap. These node sets, along with the connections between nodes within each set, form subgraphs.

Furthermore, the thinking process of Deepseek-R1 elucidates the typical logical process by which LLMs perform graph partitioning tasks, as shown in Table 3. Initially, key hub nodes (e.g., highly connected core nodes) and local dense substructures are identified through global structure analysis, and initial candidate communities are formed based on node features (degree centrality, betweenness centrality, eigenvector centrality) and connection patterns. Subsequently, a dynamic adjustment mechanism fuses or segments these candidate communities. This involves balancing the ownership of bridging nodes (based on connection strength and community closeness), adjusting subgraph sizes (merging fragmented clusters or splitting overly large clusters), and ultimately adapting the number of communities to meet the target partitioning requirements while optimizing modularity objectives. Throughout the process, the optimization of maximizing intra-community connection weights and minimizing cross-community connections guides the decision-making, combining topological features and numerical indicators for multi-level verification.

C.2 Why are LLMs unable to directly execute graph partitioning tasks on large-scale graph data?

Our analysis of the failure cases in Table 4 identifies two fundamental architectural constraints in transformer-based LLMs: *context truncation* and *the lost-in-the-middle phenomenon*. These limitations systematically corrupt graph partitioning outputs as graph size exceeds critical token thresholds.

Context Truncation. LLMs exhibit catastrophic failure modes when graph representations approach their fixed context windows. In Case 1 (200-node graph), Qwen2.5-Plus truncated community assignments at node 143, leaving incomplete outputs like `"Node 143: Assigned to community"` when output tokens saturated the output context limit. Case 2 (200-node graph) shows similar truncation patterns, detailed explanations terminate abruptly at node 162 with incomplete community IDs. Case 4 (1,000-node graph) and Case 5 (1,000-node graph) have incorrect output formats, which indicates that they did not obtain the output examples and format specifications at the end of the prompt.

Lost-in-the-Middle Phenomenon. The transformer's attention decay in long contexts induces systematic mid-sequence errors. Case 3 (2,485-node graph) exemplifies this: while claiming to generate a complete assignment list, the model abandoned detailed explanations after 10 nodes, resorting to the generic claim: "*This pattern continues...*". This is because it cannot obtain the detailed information of subsequent nodes, demonstrating soft degradation of mid-sequence focus. Case 4 (1,000-node graph) demonstrates this through invalid `Community 5` and duplicated node assignments (416/269), that the model forgets the number of target partitions and the assigned nodes. Case 5 (1,000-node graph) further confirms this spatial bias with persistent node duplication (72/763/237/613), forming artificial repeating patterns indicative of attention mechanism collapse.

Overall, due to model architecture limitations, LLMs encounter serious context truncation and the "lost-in-the-middle" issue when performing graph partitioning tasks on large-scale graphs, making it difficult to complete the task.

C.3 What influence does the construction of prompts based on varying information have on the reasoning processes of LLMs?

As shown in Table 5, we utilize the depth process of Deepseek-R1 to systematically compare the reasoning mechanisms and thinking differences of the three community detection strategies introduced in Section 5.4.

The **only1hop** approach strictly adheres to graph structural features, operating under the core hypothesis that local connection density determines community partitioning. During global structure analysis (Step 1), this method identifies hub nodes through adjacency matrices (e.g., Node 5 with 12 direct neighbors). The community formation process (Step 2) exclusively follows nodal radiation patterns, grouping Node 5 and its first-order neighbors (Nodes 1, 6, 19, etc.) into Community 0. The dynamic adjustment phase (Step 3) optimizes by minimizing cross-community edges, exemplified by reassigning Node 33 from Community 1 to Community 0 to eliminate its cross-community connection with Node 34.

In contrast, the **onlyEF** method relies entirely on nodal feature space analysis. Using feature centrality metrics (e.g., Eigenvector Centrality) as primary criteria, it establishes binary partitioning during initial community formation (Step 2) through feature thresholds: nodes with EigenCent ≥ 0 (e.g., Node 5: 3.5018) are assigned to Community 0, others to Community 1. This purely feature-driven strategy results in significantly imbalanced community sizes (14 vs. 36 nodes) and tolerates feature outliers during dynamic adjustments (e.g., retaining Node 36 in Community 0 despite near-zero EigenCent values).

The **mix** method implements multimodal reasoning through structure-feature synergy. During global analysis (Step 1), it computes both structural centrality (degree=12 for Node 5) and feature significance (EigenCent=3.5018), establishing dual validation criteria. The community formation stage (Step 2) employs decision tree logic requiring nodes to satisfy both structural connectivity ($\geq$ 2 intra-community links) and feature similarity (EigenCent difference <0.5) constraints. This hybrid strategy demonstrates superior performance in ambiguous cases like Node 24 allocation: despite weak structural connections to Community 0's Node 44, its feature dimension (Betweenness=2.6780) justifies placement in Community 1.

D Main Result Table

The main experimental results are shown in Table 6 and Table 7.

Method	Case
Qwen2.5-Plus	**Community Assignments**: `[0,0,...,0,0]` (50 nodes total) **Detailed Reasoning**: Node 0 is assigned to community 0 because it has a low degree centrality and only connects to node 21, which is in community 1. Node 1 is assigned to community 0 because it connects to nodes 5, 20, and 36, with node 5 being central in community 0. ⋮ Node 48 is assigned to community 0 because it connects to nodes 15 and 30, forming a dense subgraph within community 0. Node 49 is assigned to community 0 because it connects to nodes 13 and 27, forming a dense subgraph within community 0.
Qwen2.5-Max	**Community Assignments**: `[0,0,...,1,1]` (50 nodes total) **Detailed Reasoning**: Node 0 is assigned to community 0 because it has low degree centrality (-1.2247) and weak connections (only one neighbor, Node 21). Placing it in community 0 minimizes disruption to stronger clusters. Node 1 is assigned to community 0 because its neighbors (Nodes 5, 20, and 36) are predominantly in community 0, and its closeness centrality (-0.5151) suggests it is better grouped with nodes in community 0. ⋮ Node 48 is assigned to community 1 because its closeness centrality (0.0978) and connection to Node 15 support placement in community 1. Node 49 is assigned to community 1 because its closeness centrality (-1.2114) and connection to Node 13 support placement in community 1.
DeepSeek-R1	**Community Assignments**: `[0,0,...,1,0]` (50 nodes total) **Detailed Reasoning**: Node 0 is assigned to community 0 because it connects only to Node 21, which has moderate EigenCent and is grouped with other high-Betweenness nodes in community 1. However, Node 0's low DegreeCent (-1.2247) and isolation from hubs place it in community 0. Node 1 is assigned to community 0 because it connects to Node 5 (a central hub with EigenCent 3.5018) and shares neighbors with other community 0 nodes. ⋮ Node 48 is assigned to community 1 because it links to Node 15 and 30, but grouped with community 1's structure. Node 49 is assigned to community 0 due to its connection to Node 27 in community 0.

Table 2: Example of $k = 2$ LLM-based graph partitioning results on Cora-50. We demonstrate the first/last two nodes per method (intermediate nodes omitted with '...'). Each model uses structural and centrality features for community assignment to form subgraphs.

Step	Description	Example
1. Global Structure Analysis	Identify hub nodes and local dense substructures based on graph topology and centrality metrics (degree, betweenness, eigenvector).	Node 5 (hub): High degree (12 neighbors), high eigenvector centrality (3.5018) $\rightarrow$ core of community 0. Node 26: High betweenness (0.8988) $\rightarrow$ potential boundary node.
2. Initial Community Formation	Form initial candidate communities using connection patterns and node features.	Community 0: Nodes around Node 5 (e.g., 1, 6, 19, 23). Community 1: Nodes 2, 7, 26, 31, 32. Community 2: Nodes 3, 12, 18, 33, 34.
3. Dynamic Adjustment Mechanism	Fuse or split candidate communities; handle bridging nodes and optimize modularity.	Node 24 connects to both community 1 (Node 26) and community 4 (Node 44). Assigned to community 4 due to stronger connection to 44. Node 34 bridges community 2 and 0 $\rightarrow$ assigned to 2 for minimizing cross-community edges.
4. Final Assignment	Assign remaining nodes and finalize partitioning with size balancing and modularity maximization.	Node 0 connects to Node 21 (community 1), but low centrality $\rightarrow$ assigned to community 1. Node 48 connects to both community 3 and 4 $\rightarrow$ assigned to 4 based on majority.
5. Verification	Ensure all 50 nodes are assigned and that the final partition satisfies the target number of communities (5 in this case).	All nodes verified. Final assignment ensures no overlaps and optimized intra-community density.

Table 3: Step-by-step graph partitioning reasoning by DeepSeek-R1. Taking the graph partitioning task of dividing 5 subgraphs on Cora-50 as an example.

Case ID	Model & Task Description	Key Output Fragments (Direct Quotation)
Case 1	Model: Qwen2.5-Plus Task: 200-node graph → 2 subgraphs	[...] *Node 141: Assigned to community 1 because it has moderate degree centrality (0.9624) and connects to nodes with higher centrality.* *Node 142: Assigned to community 0 because it has very low degree centrality (-0.7665) and limited connections.* *Node 143: Assigned to community* [Node assignments truncated]
Case 2	Model: Qwen2.5-Max Task: 200-node subgraph → 10 subgraphs	*[...]* *Nodes 160–169:* *Node 160 is assigned to community 9 because it connects to nodes 56, 157, and 171, which exhibit high eigenvector centrality.* *Node 161 is assigned to community 0 because it connects to nodes 114 and 159, which form a cohesive subgroup.* *Node 162 is assigned to community* [Node assignments truncated]
Case 3	Model: Qwen2.5-Max Task: 2,485-node graph → 2 subgraphs	*[0, 0, 1, 1, 0, 1, 0, 0, 1, 1, ...] (full list of 2485 assignments)* [...] *Now, let's provide detailed reasoning for the first 10 nodes:* *Node 0:* [...] *Node 1:* [...] [...] *Node 9:* [...] *This pattern continues for all 2485 nodes, with each assignment based on the node's connectivity and the strength of its links within each potential community.*
Case 4	Model: DeepSeek-R1 Task: 1,000-node graph → 5 subgraphs	*Communities:* [...] *Community 5: [4, 16, 261, 416, 416]* *Community 4: [3, 257, 269, 14, 209, 104, 269, 349]* [Community assignments follow repeating patterns]
Case 5	Model: DeepSeek-R1 Task: 1,000-node graph → 2 subgraphs	*Community 1:* [...] *72, 763, 237, 613, 72, 763, 237, 613, 72, 763, 237, 613, 72, 763, 237, 613, 72, 763, 237, 613, 72, 763, 237, 613, 72, 763, 237, 613, ...* [Repeating sequence indicates lost context]

Table 4: Failure cases of LLM-based graph partitioning results: *Italicized text* represents verbatim model responses, [...] indicates non-critical content abridged by authors. **Context Truncation** (Cases 1–2: incomplete community assignments, Cases 4–5: incorrect output format), **Lost-in-the-Middle** (Cases 3: forgotten node information, Cases 4–5: repeated element artifacts)

Step	only1hop	onlyEF	mix
1. Global Structure Analysis	Focuses on adjacency relationships to identify hubs and local dense subgraphs.	Relies on node features (e.g., Eigenvector Centrality) to detect influential nodes.	Combines both adjacency structure and node features for a comprehensive view.
	Node 5 is identified as a central hub with many neighbors (e.g., Nodes 1,6,19,23).	*Node 5 has the highest Eigenvector Centrality (3.5018), indicating strong influence.*	*Node 5 is recognized as a key hub due to both high degree and Eigenvector Centrality.*
2. Initial Community Formation	Forms communities based on direct and indirect connections around hubs.	Groups nodes by feature similarity, especially Eigenvector Centrality thresholds.	Builds initial communities using both connection patterns and feature profiles.
	Community 0: Nodes around Node 5 (e.g., 1,6,19,23,25). *Community 1: Remaining nodes (e.g., 2,7,17,26).*	*Community 0: Nodes with positive Eigenvector Centrality (e.g., 1,5,6,19,23,25,34).* *Community 1: Nodes with low or negative Eigenvector Centrality.*	*Community 0: Dense cluster around Node 5 including its neighbors and high-feature nodes.* *Community 1: Other parts of the graph not connected to Node 5's cluster.*
3. Dynamic Adjustment Mechanism	Adjusts community boundaries to minimize cross-community edges.	Balances community sizes using feature thresholds and clustering logic.	Refines partitions by balancing connectivity and feature alignment.
	Node 33 is assigned to Community 0 to reduce inter-community edges. *Node 34 is kept in Community 0 due to its strong ties to Node 5.*	*Node 36 is included in Community 0 despite near-zero Eigenvector Centrality.* *High Betweenness nodes like Node 3 are placed in Community 1 due to low EigenCent.*	*Node 33 is evaluated based on both its connection to 34 and its link to Node 3.* *Node 24 is assigned to Community 1 based on its weaker links to Community 0.*
4. Final Assignment	Finalizes partitions to ensure maximum internal edge density.	Ensures feature consistency within each community.	Balances internal edge count and feature coherence across communities.
	Community 0: 18 nodes; Community 1: 32 nodes. *Only one inter-community edge between Nodes 3 and 33.*	*Community 0: 14 nodes; Community 1: 36 nodes.* *Communities differ significantly in size but maintain feature separation.*	*Community 0: 17 nodes; Community 1: 33 nodes.* *Inter-community edge minimized to one (between Nodes 3 and 33).*
5. Verification	Validates partition by counting intra- and inter-community edges.	Confirms feature-based consistency and logical groupings.	Ensures no overlaps and optimizes modularity and internal density.
	All nodes assigned; cross-edge minimized.	*Feature distribution aligns with community definitions.*	*Final assignment shows balanced modularity and clear structural boundaries.*

Table 5: Comparison of reasoning strategies across only1hop, onlyEF, and mix approaches for graph partitioning.

Dataset	Method	k=2				k=5				k=10			
		MinC	Norm	Bal	Spar	MinC	Norm	Bal	Spar	MinC	Norm	Bal	Spar
Cora-50	k-means	0.287	1.021	1.289	5.670	0.542	4.101	4.371	14.717	0.723	9.226	9.421	32.106
	Qwen2.5-Max	0.094	0.222	0.249	0.941	0.141	1.592	1.642	2.833	0.282	5.982	6.061	21.984
	Qwen2.5-Plus	0.282	0.567	0.568	2.000	0.365	2.745	2.870	6.353	0.577	7.894	8.001	22.816
	DeepSeek-R1	0.043	0.088	0.094	0.295	0.094	0.594	0.627	1.689	0.282	3.204	3.214	9.844
	Ours-Qwen	0.094	0.198	0.229	0.727	0.082	1.361	1.413	2.102	0.365	5.398	5.444	14.487
	Ours-DeepSeek	**0.038**	**0.079**	**0.092**	**0.269**	**0.082**	**0.565**	**0.598**	**1.571**	**0.200**	**2.668**	**2.685**	**7.278**
Cora-100	k-means	0.286	1.012	1.229	5.236	0.618	4.077	4.275	13.962	0.780	9.163	9.308	31.008
	Qwen2.5-Max	0.335	0.671	0.676	2.444	0.439	2.380	2.428	7.437	0.909	9.088	9.088	29.800
	Qwen2.5-Plus	0.378	0.770	0.775	2.756	0.823	4.131	4.131	13.481	0.902	9.026	9.026	29.600
	DeepSeek-R1	0.087	0.205	0.360	1.056	0.177	0.955	0.988	2.978	0.305	4.717	4.787	14.793
	Ours-Qwen	0.238	0.610	0.675	2.438	0.250	1.894	2.155	5.918	0.277	5.686	5.949	13.257
	Ours-DeepSeek	**0.067**	**0.175**	**0.247**	**0.710**	**0.116**	**0.520**	**0.648**	**1.650**	**0.207**	**3.402**	**3.427**	**9.479**
Cora-200	k-means	0.238	1.024	1.244	5.489	0.664	4.048	4.201	14.139	0.813	9.109	9.213	30.316
	Qwen2.5-Max	—	—	—	—	—	—	—	—	—	—	—	—
	Qwen2.5-Plus	—	—	—	—	—	—	—	—	—	—	—	—
	DeepSeek-R1	0.087	0.248	0.459	1.543	0.258	1.366	1.372	4.325	0.285	2.922	2.951	9.624
	Ours-Qwen	0.066	0.514	0.826	2.095	**0.060**	1.840	2.442	4.880	0.156	3.664	3.917	9.310
	Ours-DeepSeek	**0.066**	**0.152**	**0.221**	**0.698**	0.120	**0.606**	**0.671**	**1.861**	**0.150**	**1.789**	**1.944**	**5.407**
Cora-500	k-means	0.230	0.983	1.246	5.594	0.612	3.988	4.172	14.335	0.789	8.995	9.104	32.200
	Qwen2.5-Max	—	—	—	—	—	—	—	—	—	—	—	—
	Qwen2.5-Plus	—	—	—	—	—	—	—	—	—	—	—	—
	DeepSeek-R1	—	—	—	—	—	—	—	—	—	—	—	—
	Ours-Qwen	0.100	0.622	0.853	2.225	0.100	1.946	2.460	4.690	**0.092**	5.931	6.560	12.448
	Ours-DeepSeek	**0.021**	**0.264**	**0.627**	**1.027**	**0.021**	**1.070**	**1.729**	**2.697**	0.100	**3.300**	**3.912**	**7.710**
Cora-1000	k-means	0.238	0.993	1.188	5.798	0.636	3.959	4.108	15.786	0.792	8.978	9.074	34.157
	Qwen2.5-Max	—	—	—	—	—	—	—	—	—	—	—	—
	Qwen2.5-Plus	—	—	—	—	—	—	—	—	—	—	—	—
	DeepSeek-R1	—	—	—	—	—	—	—	—	—	—	—	—
	Ours-Qwen	**0.017**	1.009	1.471	3.684	**0.083**	2.502	3.041	6.116	0.127	5.713	6.258	12.958
	Ours-DeepSeek	0.081	**0.392**	**0.616**	**2.000**	0.088	**1.870**	**2.370**	**5.014**	**0.081**	**3.067**	**3.671**	**8.573**
Cora-full	k-means	0.169	0.808	1.083	5.983	0.476	3.199	3.355	13.746	0.626	7.538	7.624	30.612
	Qwen2.5-Max	—	—	—	—	—	—	—	—	—	—	—	—
	Qwen2.5-Plus	—	—	—	—	—	—	—	—	—	—	—	—
	DeepSeek-R1	—	—	—	—	—	—	—	—	—	—	—	—
	Ours-Qwen	**0.013**	0.795	1.251	2.393	0.110	**2.467**	2.943	6.005	0.095	4.849	5.462	10.630
	Ours-DeepSeek	0.034	**0.474**	**0.831**	**1.907**	**0.035**	2.563	**2.147**	**4.628**	**0.036**	**3.784**	**4.541**	**8.404**

Table 6: Main experimental results on Cora subgraphs. The symbol "—" indicates that the model is unable to perform the corresponding task. Lower values indicate better performance. The best results are highlighted in bold.

Dataset	Method	k=2				k=5				k=10			
		MinC	Norm	Bal	Spar	MinC	Norm	Bal	Spar	MinC	Norm	Bal	Spar
Cora-full	k-means	0.169	0.808	1.083	5.983	0.476	3.199	3.355	13.746	0.626	7.538	7.624	30.612
	Metis	0.046	0.092	**0.092**	**0.372**	0.078	**0.389**	**0.389**	**1.586**	0.124	**1.233**	**1.233**	**5.060**
	GAP	0.042	**0.089**	0.121	0.457	0.083	1.589	1.605	6.294	0.143	7.334	7.634	24.388
	MinCutPool	0.098	0.197	0.199	0.862	0.109	0.969	1.019	2.918	0.100	6.350	6.495	28.226
	Ours-Qwen	**0.013**	0.795	1.251	2.393	0.110	2.467	2.943	6.005	0.095	4.849	5.462	10.630
	Ours-DeepSeek	0.034	0.474	0.831	1.907	**0.035**	2.563	2.147	4.628	**0.036**	3.784	4.541	8.404
Facebook	k-means	0.148	0.959	1.257	28.400	0.562	3.800	3.952	68.925	0.746	8.373	8.434	147.889
	Metis	0.037	**0.077**	**0.077**	**1.600**	0.078	**0.320**	**0.320**	6.819	0.099	**1.210**	**1.210**	21.668
	GAP	0.050	0.107	0.207	3.923	0.110	1.509	1.620	15.526	0.045	7.102	7.407	93.549
	MinCutPool	0.050	0.118	0.124	2.432	0.055	0.368	0.410	6.740	0.058	2.912	3.039	49.794
	Ours-Qwen	**0.018**	0.585	0.832	3.150	0.105	2.556	3.051	6.302	0.092	5.740	6.350	13.526
	Ours-DeepSeek	0.028	0.423	0.752	1.957	**0.048**	1.900	2.257	**4.750**	**0.036**	3.852	4.600	**8.617**
Anaheim	k-means	0.042	0.083	0.086	**0.197**	0.085	0.431	0.443	0.999	0.141	1.386	1.389	3.102
	Metis	0.077	0.153	0.153	0.337	**0.216**	1.111	**1.111**	2.368	0.242	2.518	**2.518**	5.318
	GAP	0.087	1.002	1.406	3.756	0.607	4.002	4.187	8.520	0.619	9.003	9.277	17.951
	MinCutPool	0.032	0.634	0.635	1.467	0.229	4.007	4.787	6.500	0.256	8.917	9.487	18.465
	Ours-Qwen	0.022	0.325	0.525	0.815	0.045	1.254	1.645	2.847	0.068	2.952	3.250	5.535
	Ours-DeepSeek	**0.018**	**0.257**	**0.412**	0.723	0.032	**0.942**	1.253	**2.179**	**0.045**	**2.100**	2.600	**4.216**
FB15k_237	k-means	0.140	1.119	1.617	2671.357	0.397	4.261	5.033	2333.159	0.771	8.099	8.446	2327.503
	Metis	0.132	**0.293**	**0.293**	**5.137**	0.389	2.459	**2.459**	45.495	0.491	**4.852**	**4.853**	95.330
	GAP	**0.000**	1.000	1.500	18.000	0.001	3.951	4.742	31.345	**0.001**	8.001	8.900	162.167
	MinCutPool	0.031	0.868	1.260	10.557	0.045	3.932	4.596	57.999	0.221	8.713	9.369	135.367
	Ours-Qwen	**0.000**	0.850	1.257	12.345	**0.000**	3.257	4.102	35.018	**0.001**	6.500	7.889	119.946
	Ours-DeepSeek	**0.000**	0.653	0.907	8.524	**0.000**	**2.342**	3.286	**27.693**	**0.001**	5.252	6.345	**90.329**

Table 7: Main experimental results across different graph partitioning methods. Lower values indicate better performance. The best results are highlighted in bold.