

# See or Say Graphs: Agent-Driven Scalable Graph Understanding with Vision-Language Models

Shuo Han<sup>1,2\*</sup> Yukun Cao<sup>3,2\*</sup> Zezhong Ding<sup>2</sup>  
Zengyi Gao<sup>2</sup> S. Kevin Zhou<sup>1,4,5</sup> Xike Xie<sup>1,2†</sup>

<sup>1</sup>School of Biomedical Engineering, University of Science and Technology of China (USTC)

<sup>2</sup>Data Darkness Lab, MIRACLE Center, USTC

<sup>3</sup>School of Computer Science and Technology, Xidian University

<sup>4</sup>MIRACLE Center, Suzhou Institute for Advance Research, USTC

<sup>5</sup>Jiangsu Provincial Key Laboratory of Multimodal Digital Twin Technology

{shuo.han, zezhongding, gzy02}@mail.ustc.edu.cn

caoyukun@xidian.edu.cn

{skevinzhou, xkxie}@ustc.edu.cn

## Abstract

Vision-language models (VLMs) have shown promise in graph structure understanding, but remain limited by input-token constraints, facing scalability bottlenecks and lacking effective mechanisms to coordinate textual and visual modalities. To address these challenges, we propose GraphVista, a unified framework that enhances both scalability and modality coordination in graph structure understanding. For scalability, GraphVista organizes graph information hierarchically into a lightweight GraphRAG base, which retrieves only task-relevant textual descriptions and high-resolution visual subgraphs, compressing redundant context while preserving key reasoning elements. For modality coordination, GraphVista introduces a planning agent that decomposes and routes tasks to the most suitable modality—using the text modality for direct access to explicit graph properties and the visual modality for local graph structure reasoning grounded in explicit topology. Extensive experiments demonstrate that GraphVista scales to large graphs, up to  $200\times$  larger than those used in existing benchmarks, and consistently outperforms existing textual, visual, and fusion-based methods, achieving up to  $4.4\times$  quality improvement over the state-of-the-art baselines by fully exploiting the complementary strengths of both modalities.

## 1 Introduction

Recently, vision-language models (VLMs) have shown potential for general-purpose graph structure understanding (Tang et al., 2024a; Kong et al., 2025; Ding et al., 2025b), offering new paradigms for solving real-world problems naturally represented as graphs. A central research focus of this

\*Equal contributions

†Corresponding author

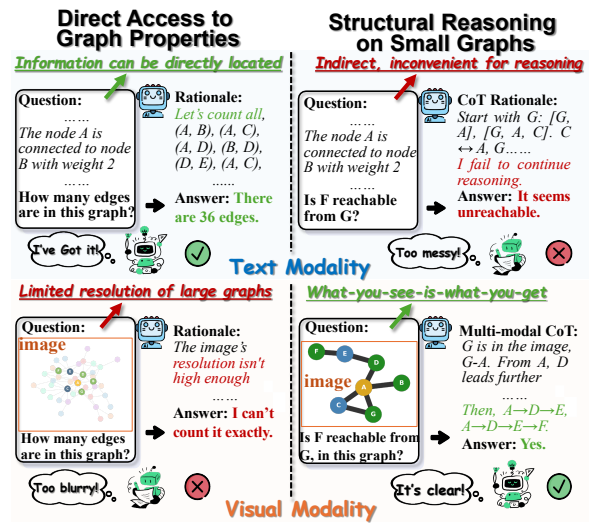


Figure 1: Comparison of textual and visual modalities across graph structure understanding tasks

emerging field is to enhance the foundational capability of VLMs to understand graph structures (Ren et al., 2024), particularly for large-scale graphs, without relying on external tools such as code execution or software systems (Ding et al., 2025a; Feng et al., 2025). To this end, existing studies have explored two primary graph input modalities for VLMs: textual graph descriptions<sup>1</sup>, which convert structural information into natural language for indirect memory and reasoning over node properties and relations (Jiang et al., 2023; Jin et al., 2024; Zhang et al., 2024a); and visual graph representations, which render nodes and edges as structured images to support direct structural perception (Li et al., 2024c; Zhu et al., 2025b). More recent ef-

<sup>1</sup>Most early studies (Xiong et al., 2024; Cao et al., 2025b; Guan et al., 2025; Peng et al., 2025) on text-modality-based graph understanding were developed on LLMs. As VLMs extend LLMs with visual capabilities, this paper presents a unified discussion from the perspective of graph structure understanding with VLMs.

forts further combine these modalities to improve structural understanding and complex reasoning on graphs (Wei et al., 2024).

However, despite these, current VLMs still lack a unified framework that systematically addresses the scalability challenges and the difficulties in coordinating textual and visual modalities inherent in graph structure understanding. Specifically:

**Scalability remains a fundamental limitation for both textual and visual graph inputs in VLMs.** Given their limited input token capacities, textual graph descriptions are constrained by sequence length, making it difficult to encode the full structure of large graphs (Chen et al., 2024a; Ding et al., 2024; Yuan et al., 2025a; Li et al., 2025). Visual graph inputs, on the other hand, are restricted by image resolution, resulting in the loss of structural details as graph size increases (Zhu et al., 2025b).

**In addition, VLMs face the lack of an effective mechanism to coordinate different input modalities for graph structure understanding,** which prevents them from fully leveraging the distinct strengths and complementary advantages of textual and visual modalities. As illustrated in Figure 1, under the token length limits of VLMs, the text modality is more effective for tasks that require direct statistical access to explicit graph properties, such as counting nodes, measuring degrees, and assessing connectivity. In these cases, the information can be directly located and reasoned over with minimal effort. By contrast, the visual modality often performs worse because the limited resolution of global visual graphs blurs structural details, causing VLMs to misidentify relevant elements and leading to performance degradation.

For tasks that require local graph structure reasoning<sup>2</sup>, such as shortest-path computation or cycle detection, high-resolution local visual subgraphs offer a clear advantage for VLMs. They provide a “what-you-see-is-what-you-get” view of the graph topology and naturally support structural reasoning. In contrast, textual descriptions are often lengthy and lack structural clarity, leaving VLMs with only indirect language cues and thus little structural grounding for effective complex graph reasoning.

Therefore, a natural conclusion is that fully leveraging VLMs for graph structure understanding re-

quires addressing these two challenges. This entails, on the one hand, preserving as much task-relevant graph information as possible within limited input tokens, and on the other hand, assigning tasks according to modality strengths: letting the text modality efficiently address graph property tasks, while leveraging the visual modality to directly support local structure reasoning tasks.

To this end, we propose **GraphVista**, an agent-driven unified framework designed to coordinate textual and visual modalities for robust graph understanding, ensuring the systematic robustness required to reliably tackle challenging graph tasks. For the scalability challenges, GraphVista is inspired by the retrieval-augmented generation (RAG) paradigm (Lewis et al., 2020). It organizes graph information hierarchically according to structural importance and stores multi-granularity textual descriptions in a lightweight GraphRAG base. For a given graph structure understanding task, GraphVista retrieves only the relevant information before input, thereby compressing redundant context and preserving key reasoning elements. When necessary, it can also generate high-resolution visual subgraphs from the retrieved local information to support complex reasoning tasks.

To enable effective coordination across modalities, GraphVista is built upon a planning agent that drives the overall workflow. It parses graph tasks, decomposes composite tasks if necessary, and routes them to the appropriate modality. The textual modality branch, enhanced with the RAG techniques, efficiently handles graph properties tasks by retrieving and reasoning over concise textual descriptions. In contrast, the visual modality branch employs the **Visual Graph Thoughts Agent**, which constructs a multimodal chain-of-thought called Visual Graph Thoughts over high-resolution visual subgraphs, enabling VLMs to perform stepwise visual reasoning grounded in explicit structural evidence for visual tasks.

The main contributions are as follows:

- We present the first systematic analysis of the scalability challenges and coordination gaps that hinder VLMs in graph structure understanding, providing both conceptual insights and empirical evidence.
- We propose **GraphVista**, the first unified framework for graph structure understanding with VLMs, designed around two objectives: improving scalability and enabling coordination across

---

<sup>2</sup>Tasks based on global graph structure require full-graph access coupled with fine-grained structural reasoning, and can be viewed as a composition of global information retrieval and local structural-reasoning tasks.

modalities. To this end, GraphVista integrates a set of innovative techniques, such as a *hierarchical GraphRAG base*, a *planning agent* for task routing, and a *multimodal graph thoughts agent* for complex reasoning.

- We present **Grena**, the first large-scale graph benchmark designed to support step-level evaluation of multimodal graph understanding.
- Extensive experiments across diverse graph tasks demonstrate the effectiveness of GraphVista, highlighting its ability to scale to large graphs and to exploit the complementary strengths of textual and visual modalities.

## 2 Related Work

### 2.1 Text Modality-based Graph Structure Understanding

Most existing studies convert graphs into textual descriptions to enable VLM-based graph structure understanding, primarily focusing on benchmark construction and capability analysis. Recent benchmarks evaluate VLMs on basic graph element comprehension (Wang et al., 2023; Guo et al., 2023; Wu et al., 2024) and graph-theoretic reasoning tasks (Chen et al., 2024a; Li et al., 2024b; Tang et al., 2024b; Luo et al., 2024; Yuan et al., 2025a), indicating that the research in this direction remains in an early exploratory phase. Methodologically, Ge et al. (2025); Wang et al. (2025) studied the influence of graph description order, while Cao et al. (2025b) identified the “Lost in the Middle” issue in graph sequence. Building upon these findings, Guan et al. (2025) analyzed VLM attention patterns over graph data, and Peng et al. (2025); Zhang et al. (2025) explored task transferability to broader reasoning domains. However, these strategies still face scalability bottlenecks and struggle with complex reasoning tasks.

Other studies address graph problems by leveraging external tools. Li et al. (2024a); Yuan et al. (2025b) employ predefined coding templates for algorithmic reasoning, while Perozzi et al. (2024) trains task-specific GNNs for graph processing. These methods depend on external modules rather than enhancing the intrinsic graph understanding of VLMs, and are thus orthogonal to our work.

### 2.2 Visual Modality-based Graph Structure Understanding

Visual modality-based graph structure understanding is still in its infancy. Li et al. (2024c), Zhu

et al. (2025b), and Zhao et al. (2025) benchmark VLMs on graph perception and reasoning, revealing difficulties in capturing structural information. Wei et al. (2024) further introduces a hybrid multimodal representation directly compatible with VLMs. However, these studies remain empirical and fail to address the scalability, structural understanding, and reasoning limitations inherent to vision-only representations.

## 3 Methodology

### 3.1 Overview

We introduce **GraphVista**, an agent-driven framework designed to address the challenges of scalability and modality coordination in graph structure understanding<sup>3</sup>, as shown in Figure 2.

GraphVista is coordinated by a planning agent that directs the entire workflow. Initially, the planning agent constructs a lightweight Hierarchical GraphRAG Base  $\mathcal{K}$  by partitioning the graph based on topological centrality. Upon receiving a question, the agent parses the request to identify the task type and key entities, decomposes composite tasks if necessary, and routes them to the appropriate reasoning modality. For text-modality tasks, the text modality-based branch is activated, employing GraphRAG-based retrieval to efficiently extract relevant context from  $\mathcal{K}$ . For visual-modality tasks, the task is delegated to the visual modality-based branch. In this branch, the Visual Graph Thoughts Agent extracts task-relevant visual subgraphs from  $\mathcal{K}$  and performs iterative multimodal reasoning grounded in explicit visual evidence.

The remainder of this section outlines our proposed framework. We begin with the construction of the hierarchical base  $\mathcal{K}$  in §3.2, followed by the planning agent-driven task routing mechanism in §3.3. Finally, we elaborate on the graph understanding modules specific to the visual and textual modalities in §3.4 and §3.5, respectively.

### 3.2 Hierarchical GraphRAG Base

Limited context windows often prevent VLMs from processing complete graphs, leading to the loss of key structural details. To address this, we propose constructing a local knowledge base  $\mathcal{K}$ .

<sup>3</sup>Aligned with prior studies (Wang et al., 2023; Chen et al., 2024a; Yuan et al., 2025a), our work focuses on pure graph structure understanding. The processing of semantic information is orthogonal to our method and can be seamlessly incorporated into GraphVista due to its modular design.

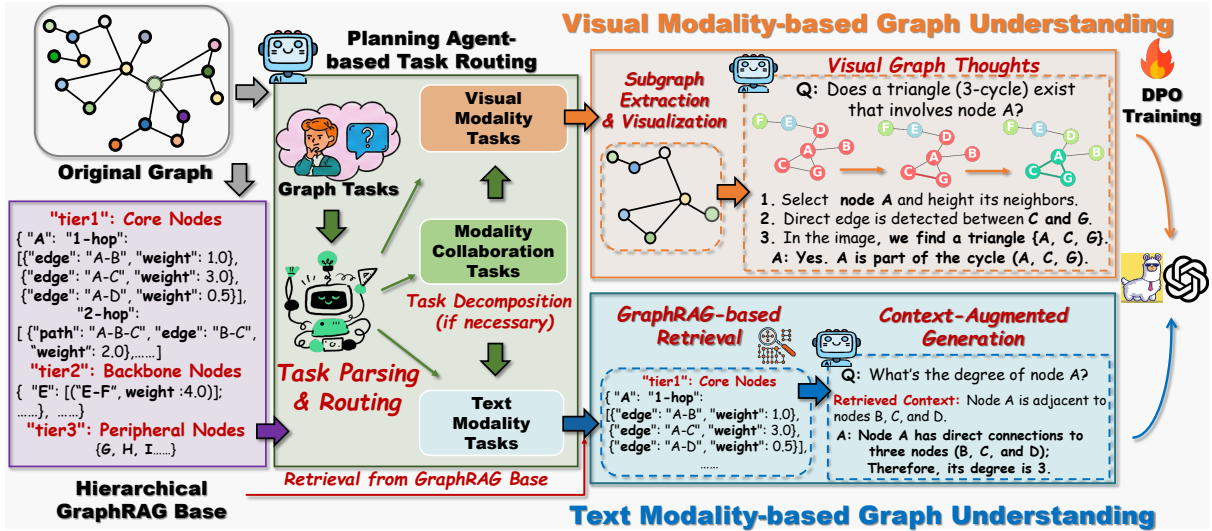


Figure 2: Framework of GraphVista

Diverging from conventional RAG pipelines (Guo et al., 2025; Jin et al., 2025) that rely on external knowledge sources, our approach constructs a compact multi-level structural representation directly from the graph and stores it in a lightweight GraphRAG Base, enabling scalable reasoning without external dependencies<sup>4</sup>.

Specifically, we partition the graph  $G = (V, E)$  with  $|V|$  nodes into three tiers based on the topological centrality of its nodes. Each tier uses a different storage granularity, significantly reducing storage needs while preserving the structural details essential for downstream tasks.

**Tier 1: Core Nodes.** This tier comprises nodes with the strongest structural influence, acting as hubs that govern information flow and global graph properties. We employ PageRank, which captures both connectivity and reachability, to identify the top  $K_1\%$  of nodes as core nodes. For each, we preserve the 2-hop neighborhood<sup>5</sup>  $\mathcal{N}_2(v)$  to explicitly retain fine-grained local topology.

**Tier 2: Backbone Nodes.** Situated along inter-community paths, these nodes are critical for maintaining global connectivity. We select them using

<sup>4</sup>For small graphs ( $|V| \leq 15$ ), we skip the storage phase and feed the entire graph directly into the VLMs.

<sup>5</sup>We limit the storage scope to a 2-hop radius to optimize the trade-off between structural coverage and computational cost. Extending to 3-hops (or beyond) leads to “neighbor explosion” (Barabási and Albert, 1999; Grabow et al., 2015), yet remote nodes contribute little to local structural reasoning (Li et al., 2018; Liben-Nowell and Kleinberg, 2003). We adopt this generalized strategy to ensure broad applicability; advanced GraphRAG storage optimizations are orthogonal to our contribution and can be seamlessly integrated.

Betweenness Centrality, prioritizing nodes that frequently bridge shortest paths between pairs. The subsequent  $K_2\%$  of nodes are retained; however, to balance representational fidelity with storage efficiency, we store only the 1-hop neighborhood  $\mathcal{N}_1(v)$  for each backbone node.

**Tier 3: Peripheral Nodes.** The remaining nodes are classified as peripheral, whose structural role is primarily defined through their links to higher-tier nodes. To minimize redundancy, we adopt a conditional storage policy: the 1-hop neighborhood  $\mathcal{N}_1(v)$  of a peripheral node is stored only if it is not connected to any Tier 1 or Tier 2 node. Otherwise, its structural information is implicitly covered by the neighborhoods of higher-tier nodes, ensuring maximal storage efficiency<sup>6</sup>.

$\mathcal{K}$  not only addresses scalability but also establishes the information backbone for subsequent multimodal collaborative reasoning. In the text modality-based branch, it enables efficient retrieval of relevant structured context, while in the visual modality-based branch, it guides the precise extraction of local visual subgraphs.

### 3.3 Planning Agent-based Task Routing

This module employs a planning agent to allocate each task to the optimal reasoning modality, either textual or visual.

**Task Parsing.** Given a natural language question  $Q$  over  $G$ , the planning agent employs semantic

<sup>6</sup>In practice,  $\mathcal{K}$  can be configured to utilize only a subset of tiers (e.g., only Tier 1) to adapt to storage or computational constraints, providing flexibility across application scenarios.

parsing to determine the task type  $T$  and extract relevant entities  $\mathcal{E}$ :

$$(T, \mathcal{E}) = f_{\text{parse}}(Q, G) \quad (1)$$

, where  $f_{\text{parse}}$  utilizes few-shot prompting<sup>7</sup> to infer task semantics, producing a task category  $T$  (e.g., neighbor identification) and a set of entity pairs  $\mathcal{E}$ . To ensure consistent downstream processing, queries involving a single entity (e.g., node degree) are normalized to  $(v_i, \emptyset)$  using a null placeholder.

**Task Routing and Execution Planning.** After parsing the task, the planning agent routes the problem to the appropriate processing module based on the task type  $T$  and formulates a detailed execution plan. Integrating task designs from existing graph understanding benchmarks, the planning agent routes tasks based on their dependence on structural reasoning, thereby aligning each task with the respective strengths of text and vision modalities<sup>8</sup>. Specifically, tasks that do not require structural reasoning are handled by the text modality, tasks requiring local structural reasoning are assigned to the visual modality, and tasks involving global graph structure are decomposed into stepwise text- and vision-based subtasks.

**Text-Modality Task Examples:**

“What is the total number of nodes in this graph?”; “What is the degree of node 15?”

**Visual-Modality Task Examples:**

“Find the shortest path between nodes A and B.”; “Whether node A is part of any triangle.”

**Modality-Collaborative Task Examples:**

“Calculate the diameter of the graph.”

**Decomposition:** (i) Identify peripheral nodes using text; (ii) Estimate pairwise distances through visual subgraph reasoning.”

**Text-Modality Tasks.** These tasks do not require structural reasoning over the graph. Instead, they rely on direct access to individual nodes or edges for basic statistical retrieval, such as counting the total number of nodes or querying the degree of

<sup>7</sup>The planning agent first matches the question against pre-defined task templates; if unmatched, it falls back to semantic parsing to infer the intent and choose the reasoning modality. Implementation details and a quantitative analysis of task classification error rates are provided in Appendix H.

<sup>8</sup>Composite tasks that can be decomposed into single-modality operations (e.g., determining which of two nodes shares more common neighbors with a given node) can be handled by the planning agent through sequential execution of the corresponding subtasks.

a given node. Accordingly, the planning agent directs these tasks to the GraphRAG agent module, providing a retrieval strategy for execution over  $\mathcal{K}$ .

**Visual-Modality Tasks.** These tasks involve reasoning over local graph structures rather than simple retrieval. They focus on topological relationships within a limited subgraph (e.g., shortest-path tasks). In such cases, visual subgraph representations provide a clearer and more efficient form for structural reasoning than textual descriptions. Consequently, the planning agent delegates such tasks to the visual graph thoughts agent, while decomposing the task into algorithm-guided substeps and constructing a high-level reasoning plan.

**Modality-Collaborative Tasks.** These tasks rely on global graph structure understanding that combines full-graph access with fine-grained structural analysis, which is often beyond a single modality. They are therefore formulated as a sequence of steps that alternate between global information access and local structural reasoning. The planning agent decomposes the task into sequential text- and visual-modality sub-tasks.

### 3.4 Visual Modality-based Graph Structure Understanding

This module leverages visual-modality graph representations to handle visual-modality tasks that require fine-grained topological understanding and multi-step logical inference. To address scalability, we extract a task-relevant  $k$ -hop subgraph  $G' = (V', E')$  centered on key entities  $\mathcal{E}$ , with at most  $N_{\text{max}}$  nodes (as detailed in **Appendices B and C**).  $G'$  is visualized as  $G_{\text{image}} = f_{\text{viz}}(G')$  via a task-driven strategy that adapts layouts to the task and simplifies dense subgraphs for visual clarity, ensuring that essential structural information remains perceivable to the VLM.

#### 3.4.1 Visual Graph Thoughts

Existing text-based CoT methods often suffer from *pseudo-visual reasoning*, in which visual information gradually fades as the reasoning chain lengthens (Chen et al., 2024b; Zhang et al., 2024b; Zou et al., 2024; Cheng et al., 2025; Xiong et al., 2025; Duan et al., 2026b). To overcome this limitation in multimodal reasoning, we introduce Visual Graph Thoughts.

After obtaining the visual subgraph  $G_{\text{image}}$ , the visual reasoning agent  $\mathcal{M}_{\text{VRA}}$ , instantiated from a VLM, performs iterative multimodal reasoning

under the guidance of a high-level plan  $\Pi = (\pi_1, \pi_2, \dots, \pi_n)$  generated by the planning agent.

We formalize this process as a state-transition system. At step  $t$  ( $1 \leq t \leq n$ ), the reasoning state  $S_t$  is defined by the current visual representation  $G_{\text{image}}^{(t-1)}$ , the accumulated reasoning history  $H_{t-1}$ , and the current plan instruction  $\pi_t$ . Conditioned on  $S_t$ ,  $\mathcal{M}_{\text{VRA}}$  produces an intermediate reasoning output  $o_t$  and an action  $a_t$ :

$$(o_t, a_t) = \mathcal{M}_{\text{VRA}}(S_t) = \mathcal{M}_{\text{VRA}}(G_{\text{image}}^{(t-1)}, H_{t-1}, \pi_t). \quad (2)$$

If  $a_t$  invokes  $f_{\text{viz}}$  (e.g., node highlighting), the visual representation is updated as  $G_{\text{image}}^{(t)} = f_{\text{viz}}(G_{\text{image}}^{(t-1)}, a_t)$ ; otherwise,  $G_{\text{image}}^{(t)} = G_{\text{image}}^{(t-1)}$ . This closed-loop process continues until all plan steps are executed. Finally,  $\mathcal{M}_{\text{VRA}}$  aggregates the reasoning trace  $H_n$  to produce the final answer.

### 3.4.2 Aligning Visual Graph Thoughts with Process-level DPO

To further improve the multimodal reasoning capability of  $\mathcal{M}_{\text{VRA}}$  by promoting the generation of reliable visual graph thoughts, we adopt the process-level Direct Preference Optimization (DPO) training strategy (Rafailov et al., 2023), which models the full multimodal reasoning trajectory and provides stable, step-level supervision for graph inference. Advanced reinforcement learning methods (Hu et al., 2025; Dinucu-Jianu et al., 2025; Yu et al., 2025; Duan et al., 2026a) are orthogonal to our framework and can be naturally integrated. We adopt the standard process-level DPO formulation to ensure generality and training stability.

#### 1. Construction of Process Preference Dataset.

We construct a process preference dataset  $\mathcal{D}$ , where each sample is a triplet  $(\mathbf{x}, y_w, y_l)$ . Here,  $\mathbf{x}$  denotes the multimodal input,  $y_w$  represents the ‘‘Chosen’’ path (preferred reasoning trajectory), and  $y_l$  represents the ‘‘Rejected’’ path (non-preferred trajectory). The construction details are in **Appendix A**.

**‘‘Chosen’’ Path ( $y_w$ ).** We construct standardized reasoning templates aligned with the logical structure of each graph task. Based on these templates, we generate ground-truth reasoning steps and answers using large-scale synthetic graphs (e.g., ER and BA graphs (Li et al., 2005)) and validate their correctness through graph analysis libraries.

**‘‘Rejected’’ Path ( $y_l$ ).** To help the VLM distinguish between preferred and non-preferred paths, we construct negative samples  $y_l$  by modifying  $y_w$ . Specifically, we extend error construction strategies from existing benchmarks (Chen et al., 2024a; Yuan et al., 2025a) to multimodal reasoning tasks. The error categories are summarized in **Table 5**.

**2. DPO Training Process.** To align the generation of visual graph thoughts with ground-truth multimodal reasoning trajectories, we fine-tune  $\mathcal{M}_{\text{VRA}}$  using DPO on  $\mathcal{D}$ . For stability, we use a reference policy  $\pi_{\text{ref}}$ , defined as a copy of  $\mathcal{M}_{\text{VRA}}$  that has been supervised to fine-tune in  $\mathcal{D}$  and kept fixed during DPO training. The DPO objective minimizes the following loss function:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(\mathbf{x}, y_w, y_l) \sim \mathcal{D}} \left[ \log \sigma \left( \beta \log \frac{r_{\theta}(y_w | \mathbf{x})}{r_{\theta}(y_l | \mathbf{x})} \right) \right] \quad (3)$$

, where the policy ratio is defined as  $r_{\theta}(y | \mathbf{x}) = \pi_{\theta}(y | \mathbf{x}) / \pi_{\text{ref}}(y | \mathbf{x})$ , and  $\beta$  controls the regularization strength. Maximizing the log-probability margin between preferred ( $y_w$ ) and rejected ( $y_l$ ) trajectories effectively suppresses predefined reasoning errors. The resulting policy generates visual graph thoughts that are logically coherent and faithfully grounded in visual evidence. Training details are available in **Appendices E and F**.

### 3.5 Text Modality-based Graph Structure Understanding

For text-modality tasks, we adopt GraphRAG-based retrieval for targeted information extraction. Note that existing RAG variants and optimizations (Deng et al., 2023; Lee et al., 2025; Gao et al., 2025; Deng et al., 2025; Cao et al., 2025a; Chang et al., 2025) are orthogonal to GraphVista and can be easily integrated to improve retrieval quality. To maintain generality, we employ the standard RAG formulation in this work.

Given the parsed tuple  $(Q, \mathcal{E})$ , the relevant context  $\mathcal{C}$  is extracted from  $\mathcal{K}$  by aggregating the structural descriptions of entities in  $\mathcal{E}$ . The VLM then generates the answer conditioned on both  $Q$  and  $\mathcal{C}$ .

## 4 Evaluation

### 4.1 Experimental Setup

**Baselines.** We compare GraphVista with state-of-the-art baselines across three categories: **(a) Text-based methods**, which reason exclusively over tex-

Table 1: Performance comparison on Grena and GraphSQA Benchmark. **Red** and **Blue** highlight the best and second-best results, respectively.

Method	Model	Grena				GraphSQA		
		Text	Visual	Collab.	Overall	Text	Visual	Overall
Text-only	GLM-4.1V-9B	0.0352	0.0055	0.0110	0.0217	0.2532	0.1670	0.1929
	InternVL3-9B	0.0452	0.0180	0.0303	0.0345	0.4788	0.2498	0.3185
	Qwen2.5-VL-7B	0.0558	0.0345	0.0337	0.0450	0.2340	0.1122	0.1487
	Qwen3-8B	0.0403	0.0367	0.0096	0.0321	0.1351	0.1352	0.1352
	Gemma-3-12B	0.1246	0.0591	0.0337	0.0858	0.5917	0.3245	0.4047
	GPT-5-mini	0.1451	0.0933	0.0515	0.1093	0.6464	0.6082	0.6197
GraphPRM (DPO)	Qwen2.5-VL-7B (DPO)	0.0589	0.0410	0.0314	0.0477	0.7198	0.3455	0.4578
	Qwen3-8B (DPO)	0.0446	0.0379	0.0068	0.0339	0.3629	0.2357	0.2739
	Gemma-3-12B (DPO)	0.1281	0.0590	0.0434	0.0899	0.6235	0.3455	0.4289
GraphInsight	GLM-4.1V-9B	0.1951	0.2655	0.2172	0.2189	0.4313	0.3273	0.3585
	InternVL3-9B	0.2507	0.2915	0.2088	0.2515	0.7564	0.2315	0.3890
	Qwen2.5-VL-7B	0.2011	0.1078	0.0976	0.1520	0.5769	0.2293	0.3336
	Gemma-3-12B	0.2469	0.2357	0.1131	0.2123	0.7372	0.2478	0.3946
	GPT-5-mini	0.3031	0.3429	0.2843	0.3091	0.7211	0.6446	0.6676
GraphToken	Qwen2.5-VL-7B	0.0141	0.0982	0.0396	0.0423	0.0388	0.1770	0.1355
	Qwen3-8B	0.0052	0.0665	0.0089	0.0222	0.1742	0.1508	0.1578
	Gemma-3-12B	0.0191	0.1064	0.0475	0.0488	0.0566	0.1091	0.0933
GITA	GLM-4.1V-9B	0.0665	0.2152	0.1835	0.1333	0.3397	0.1562	0.2112
	InternVL3-9B	0.0714	0.1045	0.1546	0.0998	0.3910	0.1755	0.2401
	Qwen2.5-VL-7B	0.1099	0.1799	0.2271	0.1561	0.3205	0.1596	0.2079
	Gemma-3-12B	0.1218	0.1591	0.2344	0.1583	0.2692	0.1739	0.2025
GraphVista (Ours)	GLM-4.1V-9B	0.9352	0.3540	0.2559	0.6214	0.7954	0.6657	0.7046
	InternVL3-9B	0.9357	0.3701	0.3449	0.6469	0.7970	0.6886	0.7211
	Qwen2.5-VL-7B (DPO)	0.9363	0.4309	0.4478	0.6876	0.7973	0.6806	0.7156
	Gemma-3-12B (DPO)	0.9361	0.4431	0.6018	0.7272	0.7991	0.6815	0.7168
	GPT-5-mini	<b>0.9406</b>	<b>0.4526</b>	<b>0.6241</b>	<b>0.7372</b>	<b>0.8497</b>	<b>0.7967</b>	<b>0.8126</b>

Table 2: Ablation study of reasoning strategies in GraphVista. Best results within each VLM group are bolded, and second-best results are underlined.

Strategy	Text	Visual	Collab.	Overall
<i>GLM-4.1V-9B</i>				
CoT (Baseline)		0.1376	0.1667	0.5433
Visual Graph Thoughts (Text)	0.9352	0.2734	0.1902	0.5846
Visual Graph Thoughts (Visual)		<b>0.3540</b>	<b>0.2559</b>	<b>0.6214</b>
<i>InternVL3-9B</i>				
CoT (Baseline)		0.1422	0.1332	0.5368
Visual Graph Thoughts (Text)	0.9357	0.2879	0.2511	0.6031
Visual Graph Thoughts (Visual)		<b>0.3701</b>	<b>0.3449</b>	<b>0.6469</b>
<i>Qwen2.5-VL-7B (DPO)</i>				
CoT (Baseline)		0.1691	0.1684	0.5525
Visual Graph Thoughts (Text)	0.9363	0.3629	0.3355	0.6431
Visual Graph Thoughts (Visual)		<b>0.4309</b>	<b>0.4478</b>	<b>0.6876</b>
<i>Gemma-3-12B (DPO)</i>				
CoT (Baseline)		0.1676	0.1703	0.5525
Visual Graph Thoughts (Text)	0.9361	0.3551	0.4200	0.6610
Visual Graph Thoughts (Visual)		<b>0.4431</b>	<b>0.6018</b>	<b>0.7272</b>

tual descriptions, including the standard Text-only baseline, GraphPRM (Peng et al., 2025) (trained via DPO), and GraphInsight (Cao et al., 2025b). **(b) GNN-based methods**, such as GraphToken (Perozzi et al., 2024), which encode structural information into latent representations<sup>9</sup>. **(c) Hybrid methods**, represented by GITA (Wei et al., 2024), which integrate both visual and textual modalities.

**Benchmarks and Evaluation Metrics.** Current graph understanding benchmarks suffer from re-

<sup>9</sup>These methods focus on graph encoding rather than VLM intrinsic understanding. We include them for completeness.

Table 3: Ablation comparing frozen base models (“Frozen”) and DPO-fine-tuned models (“DPO”). **Red** and **Blue** denote the best and second-best results.

Model	Strategy	Text	Visual	Collab.	Overall
Qwen2.5-VL-7B	Frozen		0.3951	0.3589	0.6571
	DPO	0.9363	0.4309	0.4478	0.6876
Gemma-3-12B	Frozen		0.4005	0.5387	0.7010
	DPO	0.9361	<b>0.4431</b>	<b>0.6018</b>	<b>0.7272</b>

stricted scale and topological diversity, with insufficient support for multi-step visual reasoning. We introduce **Grena**, a large-scale benchmark designed to evaluate VLMs across 20 distinct task types, covering the scope of existing baselines (Wang et al., 2023; Yuan et al., 2025a; Zhu et al., 2025b). Grena spans a wide range of graph scales (up to 2,050 nodes) and topologies. Further details are provided in **Appendix A**.

We use Grena to evaluate graph structure understanding performance on large-scale graphs ( $|V| \in [50, 2050]$ ), covering a total of 22,800 tasks. Complementarily, we use the GraphSQA benchmark (Cao et al., 2025b) for small-graph structure understanding, with  $|V|$  in the range of  $[15, 50]$ <sup>10</sup>. Across all tasks and benchmarks, we use **Accuracy**

<sup>10</sup>As this small-graph benchmark lacks modality-collaborative tasks, we do not separate local and global tasks. Instead, we directly adopt its original classification, referring to the two categories as “Text” and “Visual”.

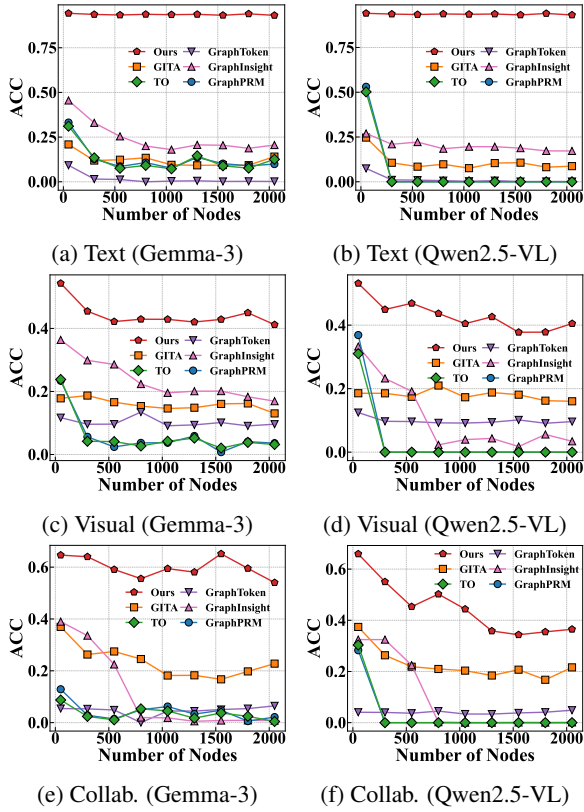


Figure 3: Analysis on Graphs with Different  $|V|$

(ACC) as the evaluation metric. Methods requiring DPO are trained on  $\mathcal{D}$ , as detailed in **Appendix E**.

**Models.** We instantiate all methods using recent open-source, competitive VLMs, including Gemma-3-12B (Kamath et al., 2025), Qwen2.5-VL-7B (Bai et al., 2025), InternVL3-9B (Zhu et al., 2025a), and GLM-4.1V-9B-Thinking (Hong et al., 2025), a reasoning-oriented VLM. For text-based methods, we include more capable LLMs such as Qwen3-8B (Yang et al., 2025) to ensure a fairer comparison. For training-free methods, we include GPT-5-mini to assess generality.

## 4.2 Performance

As shown in Table 1, GraphVista consistently outperforms all baselines on both benchmarks.

**Text-modality Tasks.** For text-modality tasks, GraphVista achieves an accuracy of 0.936 on the Grena benchmark, surpassing the strongest baseline, GraphInsight (0.247), by  $3.8\times$ . Notably, GraphVista also demonstrates strong generalization on the small-scale GraphSQA benchmark, consistently outperforming baselines across various VLMs. As shown in Figures 3a and 3b, our method maintains stable performance on Grena up to 2,050 nodes, whereas baseline methods suffer significant

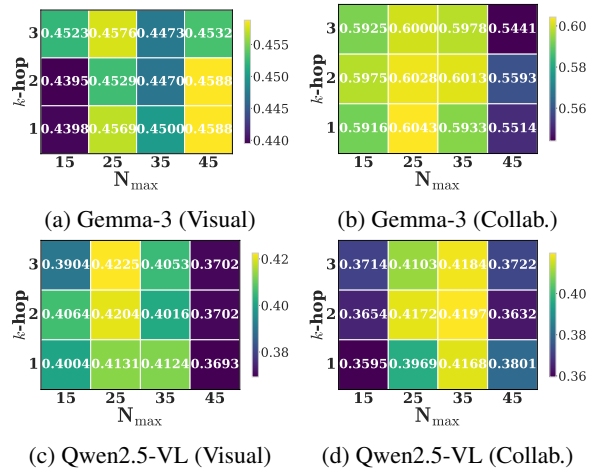


Figure 4: Hyperparameter analysis on  $k$  and  $N_{max}$  degradation (e.g., GraphInsight with Qwen2.5-VL drops from 0.270 to 0.172).

## Visual and Modality-Collaborative Tasks.

GraphVista shows robust performance on structural reasoning tasks. On visual-modality tasks, it achieves an average accuracy of 0.443 with Gemma-3, outperforming GITA and GraphToken by  $2.4\times$  and  $4.3\times$ , respectively. Similarly, for modality-collaborative tasks, GraphVista reaches 0.602, surpassing the best baseline (GITA, 0.234) by  $2.6\times$ . As shown in Figures 3c–3f, GraphVista scales robustly to 2,050 nodes, whereas baselines degrade to near-zero as graph size increases.

## 4.3 Ablation Study

**Reasoning Strategies.** We validate the efficacy of the Visual Graph Thoughts mechanism by comparing it against the standard Chain-of-Thought (CoT) baseline and its text-based variant. As detailed in Table 2, our approach consistently outperforms these text-dependent methods [cite: 791, 940], notably achieving  $1.3\times$  and  $2.6\times$  accuracy gains on Modality-Collaborative and Visual tasks, respectively. These results confirm that explicit structural grounding is critical for resolving the ambiguities inherent in purely textual reasoning.

**Impact of DPO.** Comparing fine-tuned models with frozen baselines (Table 3) demonstrates that process-level DPO enhances performance across all settings, particularly for complex tasks. Notably, DPO improves Qwen2.5-VL-7B’s accuracy on collaborative tasks from 0.3589 to 0.4478. This verifies the effectiveness of aligning model outputs with expert visual reasoning trajectories.

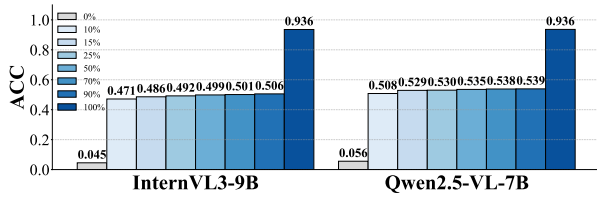


Figure 5: Tier number analysis in  $\mathcal{K}$

#### 4.4 Hyperparameter Analysis

**Subgraph Extraction.** We analyze the impact of  $k$  and  $N_{max}$  on performance for visual-modality tasks. As shown in Figure 4, Gemma-3 reaches its highest accuracy with a 2-hop neighborhood, indicating that it can leverage a broader structural context. In contrast, Qwen2.5-VL performs best with a 1-hop neighborhood, suggesting that larger subgraphs may introduce extraneous information. These observations point to the need for adapting subgraph extraction strategies to the characteristics of different VLM architectures.

**Number of Tiers in  $\mathcal{K}$ .** We examine the impact of storage granularity in  $\mathcal{K}$  across three settings: Tier 1 (Core), Tier 1+2 (Backbone), and Full Graph. As shown in Figure 5, performance positively scales with structural coverage. Tiers 1 and 2 provide efficient baselines, but adding Tier 3 significantly improves performance (e.g., 0.936 on InternVL3-9B) at the cost of higher storage. This underscores a clear trade-off: prioritizing central nodes ensures efficiency, whereas full graph coverage is essential for maximizing accuracy.

## 5 Conclusion

In this paper, we propose GraphVista, a unified framework for scalable and modality-coordinated understanding of graph structure. For scalability, it employs a hierarchical GraphRAG base to compress task-relevant information. For coordination, a planning agent routes tasks to the optimal modality, leveraging textual retrieval for properties and visual graph thoughts for topological reasoning. Future work will extend GraphVista to complex graphs with labels and semantics.

## Limitations

The proposed GraphVista framework offers a promising approach for scalable, modality-coordinated graph structure understanding, while also presenting several potential extensions that could inspire future research directions.

First, while the framework currently focuses on graph topology, its modular design is intended to facilitate extensions for handling rich semantic information. For instance, a semantic extraction module could be easily incorporated to process node attributes and edge labels. This would broaden the framework’s applicability to domains such as knowledge graph reasoning and question answering over heterogeneous networks. Additionally, for the lightweight GraphRAG base, we adopted a general and efficient implementation to ensure the framework’s versatility. We note that many advanced optimization strategies, such as sophisticated indexing and retrieval techniques, are orthogonal to our core contributions. A key advantage of GraphVista’s modular design is that these specialized modules can be readily integrated into our framework. Therefore, future work could easily incorporate state-of-the-art components to further enhance the performance of our framework, demonstrating its flexibility.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 62472400 and 62271465, the National Key R&D Program of China under Grant 2025YFC3408300, and the Suzhou Basic Research Program under Grant SYG202338.

## References

- Shuai Bai, Keqin Chen, Xuejing Liu, and 1 others. 2025. Qwen2.5-VL Technical Report. *CoRR*, abs/2502.13923.
- Albert-László Barabási and Réka Albert. 1999. Emergence of Scaling in Random Networks. *Science*.
- Yukun Cao, Zengyi Gao, Zhiyang Li, Xike Xie, S. Kevin Zhou, and Jianliang Xu. 2025a. LEGO-GraphRAG: Modularizing Graph-based Retrieval-Augmented Generation for Design Space Exploration. *PVLDB*.
- Yukun Cao, Shuo Han, Zengyi Gao, Zezhong Ding, Xike Xie, and S. Kevin Zhou. 2025b. GraphInsight: Unlocking Insights in Large Language Models

- for Graph Structure Understanding. In *ACL*, pages 12096–12134.
- Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, and Na Zou. 2025. MAIN-RAG: Multi-Agent Filtering Retrieval-Augmented Generation. In *ACL*, pages 2607–2622.
- Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024a. GraphWiz: An Instruction-Following Language Model for Graph Computational Problems. In *KDD*, pages 353–364.
- Qiguang Chen, Libo Qin, Jin Zhang, Zhi Chen, Xiao Xu, and Wanxiang Che. 2024b. M<sup>3</sup>CoT: A Novel Benchmark for Multi-Domain Multi-Step Multi-Modal Chain-of-Thought. In *ACL*, pages 8199–8221.
- Zihui Cheng, Qiguang Chen, Jin Zhang, Hao Fei, Xiaocheng Feng, Wanxiang Che, Min Li, and Libo Qin. 2025. CoMT: A Novel Benchmark for Chain of Multi-modal Thought on Large Vision-Language Models. In *AAAI*, pages 23678–23686.
- Jingcheng Deng, Zongjian Jiang, Liang Pang, Lida Chen, Keli Xu, Zihao Wei, Huawei Shen, and Xueqi Cheng. 2025. Following the Autoregressive Nature of LLM Embeddings via Compression and Alignment. In *EMNLP*.
- Jingcheng Deng, Liang Pang, Huawei Shen, and Xueqi Cheng. 2023. RegaVAE: A Retrieval-Augmented Gaussian Mixture Variational Auto-Encoder for Language Modeling. In *EMNLP (Findings)*.
- Hanxing Ding, Shuchang Tao, Liang Pang, Zihao Wei, Jinyang Gao, Bolin Ding, Huawei Shen, and Xueqi Cheng. 2025a. ToolCoder: A Systematic Code-Empowered Tool Learning Framework for Large Language Models. In *ACL*, pages 17876–17891.
- Jiayu Ding, Zhangkai Zheng, Benshuo Lin, Yun Xue, and Yiping Song. 2025b. MSG-LLM: A Multi-scale Interactive Framework for Graph-enhanced Large Language Models. In *COLING*, pages 9687–9700.
- Zezhong Ding, Yongan Xiang, Shangyou Wang, Xike Xie, and S. Kevin Zhou. 2024. Play like a Vertex: A Stackelberg Game Approach for Streaming Graph Partitioning. In *SIGMOD*.
- David Dinucu-Jianu, Jakub Macina, Nico Daheim, Ido Hakimi, Iryna Gurevych, and Mrinmaya Sachan. 2025. From Problem-Solving to Teaching Problem-Solving: Aligning LLMs with Pedagogy using Reinforcement Learning. In *EMNLP*.
- Zenghao Duan, Liang Pang, Zihao Wei, Wenbin Duan, Yuxin Tian, Shicheng Xu, Jingcheng Deng, Zhiyi Yin, and Xueqi Cheng. 2026a. Circular Reasoning: Understanding Self-Reinforcing Loops in Large Reasoning Models. In *CoRR*.
- Zenghao Duan, Zhiyi Yin, Zhichao Shi, Liang Pang, Shaoling Jing, Zihe Huang, Jiayi Wu, Yu Yan, Jingcheng Deng, Huawei Shen, and Xueqi Cheng. 2026b. Projecting Out the Malice: A Global Subspace Approach to LLM Detoxification. In *CoRR*.
- Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a Graph: Encoding Graphs for Large Language Models. In *ICLR*.
- Huawen Feng, Pu Zhao, Qingfeng Sun, Can Xu, Fangkai Yang, Lu Wang, Qianli Ma, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2025. WarriorCoder: Learning from Expert Battles to Augment Code Large Language Models. In *ACL*, pages 4955–4969.
- Zengyi Gao, Yukun Cao, Hairu Wang, Ao Ke, Yuan Feng, Xike Xie, and S Kevin Zhou. 2025. FRAG: A Flexible Modular Framework for Retrieval-Augmented Generation Based on Knowledge Graphs. In *ACL (Findings)*.
- Yuyao Ge, Shenghua Liu, Lingrui Mei, Lizhe Chen, and Xueqi Cheng. 2025. Can Graph Descriptive Order Affect Solving Graph Problems with LLMs? In *ACL*, pages 6404–6420.
- Carsten Grabow, Stefan Grosskinsky, Jürgen Kurths, and Marc Timme. 2015. Collective Relaxation Dynamics of Small-World Networks. *arXiv preprint arXiv:1507.04624*.
- Zhong Guan, Likang Wu, Hongke Zhao, Ming He, and Jianping Fan. 2025. Attention Mechanisms Perspective: Exploring LLM Processing of Graph-Structured Data. In *ICML*.
- Hanghai Guo, Jia Zhu, Shimin Di, Weijie Shi, Zhangze Chen, and Jiajie Xu. 2025. DioR: Adaptive Cognitive Detection and Contextual Retrieval Optimization for Dynamic Retrieval-Augmented Generation. In *ACL*, pages 2953–2975.
- Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data? An Empirical Evaluation and Benchmarking. *CoRR*, abs/2305.15066.
- Wenyi Hong, Wenmeng Yu, Xiaotao Gu, and 1 others. 2025. GLM-4.1V-Thinking: Towards Versatile Multimodal Reasoning with Scalable Reinforcement Learning. *CoRR*, abs/2507.01006.
- Zican Hu, Wei Liu, Xiaoye Qu, Xiangyu Yue, Chunlin Chen, Zhi Wang, and Yu Cheng. 2025. Divide and Conquer: Grounding LLMs as Efficient Decision-Making Agents via Offline Hierarchical Reinforcement Learning. In *ICML*.
- Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A General Framework for Large Language Models to Reason over Structured Data. In *EMNLP*, pages 14603–14619.

- Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and 1 others. 2024. Graph Chain-of-Thought: Augmenting Large Language Models by Reasoning on Graphs. In *ACL (Findings)*, pages 163–184.
- Jiajie Jin, Xiaoxi Li, Guanting Dong, Yuyao Zhang, Yutao Zhu, Yongkang Wu, Zhonghua Li, Qi Ye, and Zhicheng Dou. 2025. Hierarchical Document Refinement for Long-context Retrieval-augmented Generation. In *ACL*, pages 3502–3520.
- Aishwarya Kamath, Johan Ferret, Shreya Pathak, and 1 others. 2025. Gemma 3 Technical Report. *CoRR*, abs/2503.19786.
- Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. 2025. GOFA: A Generative One-for-All Model for Joint Graph Language Modeling. In *ICLR*.
- Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N. Ioannidis, Huzefa Rangwala, and Christos Faloutsos. 2025. HybGRAG: Hybrid Retrieval-Augmented Generation on Textual and Relational Knowledge Bases. In *ACL*, pages 879–893.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*.
- Jin Li, Zezhong Ding, and Xike Xie. 2025. DuetGraph: Coarse-to-Fine Knowledge Graph Reasoning with Dual-Pathway Global-Local Fusion. In *NeurIPS*.
- Lun Li, David Alderson, John C Doyle, and Walter Willinger. 2005. Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications. *Internet Math.*, 2(4):431–523.
- Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*.
- Xin Sky Li, Qizhi Chu, Yubin Chen, Yang Liu, Yaoqi Liu, Zekai Yu, Weize Chen, Chen Qian, Chuan Shi, and Cheng Yang. 2024a. GraphTeam: Facilitating Large Language Model-Based Graph Analysis via Multi-Agent Collaboration. *CoRR*, abs/2410.18032.
- Yuhan Li, Peisong Wang, Xiao Zhu, Aochuan Chen, Haiyun Jiang, Deng Cai, Victor W Chan, and Jia Li. 2024b. GLBench: A Comprehensive Benchmark for Graph with Large Language Models. In *NeurIPS*.
- Yunxin Li, Baotian Hu, Haoyuan Shi, Wei Wang, Longyue Wang, and Min Zhang. 2024c. Vision-Graph: Leveraging Large Multimodal Models for Graph Theory Problems in Visual Context. In *ICML*.
- David Liben-Nowell and Jon M. Kleinberg. 2003. The Link Prediction Problem for Social Networks. In *CIKM*.
- Zihan Luo, Xiran Song, Hong Huang, Jianxun Lian, Chenhao Zhang, Jinqi Jiang, and Xing Xie. 2024. GraphInstruct: Empowering Large Language Models with Graph Understanding and Reasoning Capability. *CoRR*, abs/2403.04483.
- Miao Peng, Nuo Chen, Zongrui Suo, and Jia Li. 2025. Rewarding Graph Reasoning Process Makes LLMs More Generalized Reasoners. In *KDD*, pages 2257–2268.
- Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. 2024. Let Your Graph Do the Talking: Encoding Structured Data for LLMs. *CoRR*, abs/2402.05862.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *NeurIPS*.
- Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A Survey of Large Language Models for Graphs. In *KDD*, pages 6616–6626.
- 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 *SIGIR*, pages 491–500.
- Jianheng Tang, Qifan Zhang, Yuhan Li, and Jia Li. 2024b. GraphArena: Benchmarking Large Language Models on Graph Computational Problems. *CoRR*, abs/2407.00379.
- Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can Language Models Solve Graph Problems in Natural Language? In *NeurIPS*.
- Meiyun Wang, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. 2025. Lost in the Distance: Large Language Models Struggle to Capture Long-Distance Relational Knowledge. In *NAACL (Findings)*.
- Yanbin Wei, Shuai Fu, Weisen Jiang, Zejian Zhang, Zhixiong Zeng, Qi Wu, James Kwok, and Yu Zhang. 2024. GITA: Graph to Visual and Textual Integration for Vision-Language Graph Reasoning. In *NeurIPS*.
- Qiming Wu, Zichen Chen, Will Corcoran, Misha Sra, and Ambuj K Singh. 2024. GraphEval2000: Benchmarking and Improving Large Language Models on Graph Datasets. *CoRR*, abs/2406.16176.
- Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. 2024. Large Language Models Can Learn Temporal Reasoning. In *ACL*.

- Siheng Xiong, Ali Payani, Yu'an Yang, and Faramarz Fekri. 2025. Deliberate Reasoning in Language Models as Structure-aware Planning with an Accurate World Model. In *ACL*.
- An Yang, Anfeng Li, Baosong Yang, and 1 others. 2025. Qwen3 Technical Report. *CoRR*, abs/2505.09388.
- Qiyong Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476.
- Zike Yuan, Ming Liu, Hui Wang, and Bing Qin. 2025a. GraCoRe: Benchmarking Graph Comprehension and Complex Reasoning in Large Language Models. In *COLING*.
- Zike Yuan, Ming Liu, Hui Wang, and Bing Qin. 2025b. MA-GTS: A multi-agent framework for solving complex graph problems in real-world applications. In *EMNLP*, pages 19286–19304.
- Qifan Zhang, Nuo Chen, Zehua Li, Miao Peng, Jing Tang, and Jia Li. 2025. Improving LLMs' Generalized Reasoning Abilities by Graph Problems. *CoRR*, abs/2507.17168.
- Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024a. LLM4DyG: Can Large Language Models Solve Spatial-Temporal Problems on Dynamic Graphs? In *KDD*, pages 4350–4361.
- Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2024b. Multi-modal Chain-of-Thought Reasoning in Language Models. *Trans. Mach. Learn. Res.*, 2024.
- Xinjian Zhao, Wei Pang, Zhongkai Xue, Xiangru Jian, Lei Zhang, Yaoyao Xu, Xiaozhuang Song, Shu Wu, and Tianshu Yu. 2025. The underappreciated power of vision models for graph structural understanding. In *NeurIPS*.
- Jinguo Zhu, Weiyun Wang, Zhe Chen, and 1 others. 2025a. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *CoRR*, abs/2504.10479.
- Yingjie Zhu, Xuefeng Bai, Kehai Chen, Yang Xiang, Jun Yu, and Min Zhang. 2025b. Benchmarking and Improving Large Vision-Language Models for Fundamental Visual Graph Understanding and Reasoning. In *ACL*, pages 30678–30701.
- Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. 2024. VGBench: Evaluating Large Language Models on Vector Graphics Understanding and Generation. In *EMNLP*.

## A Grena Benchmark

Existing graph structure understanding benchmarks suffer from three major limitations: (i) limited coverage of graph scales and topological structures; (ii) lack of task categorization grounded in fundamental graph properties and reasoning modalities; and (iii) insufficient support for modeling and evaluating multi-step visual reasoning in graph inference.

To address these gaps, we propose the **Grena Benchmark**, a large-scale, multi-task benchmark with step-level visual reasoning capability. It is designed to rigorously evaluate and advance the cognitive and reasoning abilities of VLMs in graph domains.

The core design of Grena Benchmark is reflected in the following three aspects:

### A.1 Scalability and Structural Diversity

A primary objective of Grena is to systematically evaluate the scalability graph structure understanding capabilities of VLMs. To this end, the benchmark includes graphs that span a wide range of scales, with node counts from 50 to 2,050. This diversity in size is essential for assessing how model performance varies with increasing graph size and structural complexity.

In addition to scale, Grena incorporates diverse topological structures by including both Erdős–Rényi (ER) random graphs and Barabási–Albert (BA) scale-free networks. While ER graphs provide a baseline for performance on uniform structures, BA graphs feature power-law degree distributions and hubs, closely resembling the topology of many real-world systems. This combination of scales and topologies creates a robust testbed for evaluating model generalization across heterogeneous graph structures.

### A.2 Categorization of Graph Structure Understanding Tasks

To provide a structured assessment of reasoning abilities across different modalities, Grena Benchmark organizes tasks into three categories:

- **Text-Modality Tasks.** These tasks focus on basic statistical retrieval and accessing individual node attributes without requiring reasoning over complex topology. Examples include “What is the total number of nodes?” or “What is the degree of node  $v_A$ ?” They

primarily evaluate a model’s ability to interact with the Knowledge Graph ( $\mathcal{K}$ ) for direct information retrieval.

- **Visual-Modality Tasks.** These tasks involve reasoning over local graph structures and topological relationships within a limited sub-graph. Examples include shortest-path problems (“Find the shortest path between nodes A and B”) or cycle detection (“Is node A part of any triangle?”). Unlike retrieval tasks, these require the model to utilize visual subgraph representations for efficient structural inference.
- **Modality-Collaborative Tasks.** These tasks require integrating global graph understanding with fine-grained local structural analysis, necessitating a sequence of steps that alternate between modalities. For example, calculating the graph diameter involves identifying peripheral nodes via text modality followed by estimating pairwise distances via visual modality. These tasks evaluate the model’s ability to decompose complex queries into sequential sub-tasks.

### A.3 Process-Level Supervision for Visual Reasoning

A key contribution of the Grena Benchmark is its native support for training models on step-level visual reasoning. For tasks that require a chain-of-thought process (e.g., pathfinding), the benchmark provides not only the final answers but also intermediate visual exemplars. These include the original graph and a series of “state graphs” that progressively highlight key nodes, edges, or sub-graphs. This design provides dense supervisory signals, making it suitable for both Supervised Fine-Tuning (SFT) and alignment techniques like Direct Preference Optimization (DPO), thereby enabling VLMs to learn explicit reasoning processes.

To facilitate these training paradigms, we construct a high-quality, process-centric dataset where each sample is a triplet  $(\mathbf{x}, y_w, y_l)$ , representing the input, a preferred reasoning path (Chosen), and a non-preferred one (Rejected):

- **Supervised Fine-Tuning Data (The “Chosen” Path,  $y_w$ ).** The “Chosen” paths serve as gold-standard examples for SFT. They are generated based on human-annotated, standardized reasoning templates that align with

Table 4: Comparison of Benchmarks for Evaluating graph structure understanding in Large Language Models (LLMs) and Vision-language models (VLMs)

	Max Graph Size	Graph Diversity	Multimodality	Step-Level Visual Reasoning
GraphQA (Fatemi et al., 2024)	20	✗	✗	✗
GraphInstruct (Luo et al., 2024)	35	✗	✗	✗
GraphInstruct(Graphwiz) (Chen et al., 2024a)	100	✗	✗	✗
GraphSQA (Cao et al., 2025b)	200	✓	✗	✗
VisionGraph (Li et al., 2024c)	35	✗	✓	✗
VGcure (Zhu et al., 2025b)	15	✓	✓	✗
Grena	2050	✓	✓	✓

Table 5: Categories of errors used to construct  $y_l$ .

Textual and Logical Errors	
Factual Errors	Incorrect identification of a neighbor set $\mathcal{N}(v)$ or an edge weight $w(e)$ .
Logical Errors	Faults in algorithmic reasoning (e.g., using edge count instead of weight in a shortest-path task).
Computation Errors	Mistakes in numerical operations such as counting.
Omitted Steps	Skipping essential reasoning steps.
Multimodal Errors	
Element Misrecognition	Misidentifying nodes or edges in the visual representation.
Visual Neglect	Failure to incorporate visual information, with reasoning based only on text.
Text–Visual Inconsistency	Mismatch between textual reasoning and visual grounding.
Visualization Misuse	Incorrect or missing calls to visualization functions.

the logical flow of each task. We use large-scale synthetic graphs (e.g., Erdős–Rényi and Barabási–Albert models), and programmatically validate the correctness and completeness of all reasoning steps using libraries like NetworkX.

- **Direct Preference Optimization Data (The "Rejected" Path,  $y_l$ ).** To create preference pairs  $(y_w, y_l)$  for DPO, the "Rejected" paths are systematically generated by introducing errors into the "Chosen" paths. These error-construction strategies are adapted from established textual reasoning benchmarks and extended to multimodal contexts. The resulting paths feature common reasoning pitfalls (as detailed in Table 5), teaching the model to distinguish between correct and flawed reasoning.

## B Subgraph Extraction

To mitigate the scalability challenges of large graphs, we first extract a task-relevant local subgraph  $G' = (V', E')$  from  $G$  stored in  $\mathcal{K}$ , based on

the key entities  $\mathcal{E}$ . This process is formalized as:

$$G' = f_{\text{extract}}(G, \mathcal{E}) \quad (4)$$

where the extraction strategy  $f_{\text{extract}}$  is determined by the structural characteristics of  $\mathcal{E}$  and can be categorized as follows.

**Ego-centric** For problems involving a single entity, where  $\mathcal{E} = (v_i, \emptyset)$ , we extract its  $k$ -hop neighborhood subgraph  $G'_{v_i, k} = (V', E')$ . The node set  $V'$  is defined as:

$$V' = \{v \in V \mid \text{dist}_G(v, v_i) \leq k\} \quad (5)$$

where  $\text{dist}_G(u, v)$  denotes the shortest path distance between nodes  $u$  and  $v$  in  $G$ . The edge set  $E'$  is induced from  $G$  by  $V'$ .

**Multi-centric** For problems concerning relationships between multiple entities, where  $\mathcal{E} = (v_s, v_t)$ , we extract paths connecting the source  $v_s$  and target  $v_t$ . We employ Yen’s  $K$ -shortest paths to obtain  $K$  candidate paths  $\mathcal{P} = \{P_1, \dots, P_K\}$ . We then construct a set  $V_{\mathcal{P}} = \bigcup_{j=1}^K V(P_j)$  of all nodes on these paths. Finally, a 1-hop neighborhood expansion is performed on all nodes in  $V_{\mathcal{P}}$  to

Table 6: Taxonomy of tasks in the Grena Benchmark, covering 20 distinct graph structure understanding challenges.

Task Name	Definition and Expected Output
Articulation Point Detection	Identify nodes whose removal increases the number of connected components. <b>Output:</b> List of node IDs.
Common 3rd-Order Neighbor ID	Find nodes that are exactly 3 hops away from both query nodes $u$ and $v$ . <b>Output:</b> Set of node IDs.
Connectivity Detection	Determine whether the graph is fully connected. <b>Output:</b> Boolean.
Cycle Detection	Determine whether the graph contains any cycles. <b>Output:</b> Boolean.
Edge Counting	Return the total number of edges $ E $ . <b>Output:</b> Integer.
Edge Existence Checking	Determine if a direct edge exists between a given pair of nodes $(u, v)$ . <b>Output:</b> Boolean.
Connected Edge Identification	Return all edges incident to a specific query node $v$ . <b>Output:</b> Set of edges.
Graph Diameter Calculation	Compute the longest shortest path between any two nodes in the graph. <b>Output:</b> Integer.
Highest Degree Neighbor ID	For a query node $v$ , identify the neighbor with the maximum degree. <b>Output:</b> Node ID.
Hub Node Path Finding	Identify a path between two nodes that routes through high-degree hub nodes. <b>Output:</b> Node sequence.
Maximum Degree Node ID	Identify the node(s) with the highest degree in the entire graph. <b>Output:</b> Node ID(s).
Maximum Clique Detection	Find the size or set of nodes of the largest complete subgraph. <b>Output:</b> Integer (size) or set of nodes.
Neighbor Connection Analysis	Count the number of edges within the induced subgraph of a node’s neighbors. <b>Output:</b> Integer.
Node Counting	Return the total number of nodes $ V $ . <b>Output:</b> Integer.
Node Degree Identification	Return the degree of a specific query node $v$ . <b>Output:</b> Integer.
Planarity Testing	Determine if the graph can be embedded in a plane without edge crossings. <b>Output:</b> Boolean.
Shortest Path Finding	Compute the shortest path distance and sequence between nodes $u$ and $v$ . <b>Output:</b> Path length and sequence.
Star Structure Identification	Determine if a specific subgraph forms a star topology centered at node $v$ . <b>Output:</b> Boolean.
Third-Order Neighbor ID	Return the set of nodes at exactly distance 3 from a query node $v$ . <b>Output:</b> Set of node IDs.
Triangle Counting	Count the total number of triangles (3-cycles) in the graph. <b>Output:</b> Integer.

form the final subgraph node set  $V' = \bigcup_{v \in V_p} \{u \in V \mid \text{dist}_G(u, v) \leq 1\}$ . The edge set  $E'$  is induced by  $V'$  from  $G$ .

## C Subgraph Pruning and Visualization Strategy

To maintain visual clarity and adhere to computational constraints, particularly for large graphs, the size of an extracted subgraph  $G'$  is bounded by a maximum node count,  $N_{max}$ . In instances where the initial extracted subgraph  $G' = (V', E')$  exceeds this threshold, a principled pruning procedure is invoked. Furthermore, the remaining nodes are visualized using a task-driven strategy to ensure optimal perception by the VLM.

### C.1 Ego-centric Subgraph Pruning

This approach, detailed in Algorithm 1, addresses subgraphs generated around a single central node,  $v_i$ . The core principle is to retain nodes most

relevant to this central entity. All nodes except for  $v_i$  are considered candidates for removal and are sorted into a priority list for pruning based on the hierarchical key  $(-d(v, v_i), t(v), c(v))$ . This multi-level sorting criterion prioritizes the removal of nodes based on the following order:

1. **Distance:** Nodes farthest from the central node  $v_i$  (descending distance  $d$ ) are removed first.
2. **Structural Tier:** Among nodes at the same distance, those in lower-importance tiers within the Hierarchical GraphRAG Base  $\mathcal{K}$  (ascending tier  $t(v)$ ) are prioritized for removal.
3. **Centrality:** For nodes with the same distance and tier, those with a lower centrality score (ascending centrality  $c(v)$ ) are removed.

By systematically applying these criteria, the algo-

gorithm ensures that the pruned subgraph retains the most structurally-important neighborhood around the central node.

## C.2 Multi-centric Subgraph Pruning

This strategy, presented in Algorithm 2, is designed for subgraphs constructed around a set of  $K$ -shortest paths,  $V_{\mathcal{P}}$ , connecting multiple entities. The primary objective is to preserve the integrity of these critical paths. Therefore, all nodes lying on these paths ( $V_{\mathcal{P}}$ ) are explicitly protected from pruning. Candidate nodes for removal are those not on the primary paths, and their sorting priority is determined by the key  $(t(v), c(v))$ , which prioritizes removal based on:

1. **Structural Tier:** Nodes in lower-importance tiers (ascending tier  $t(v)$ ).
2. **Centrality:** Among nodes in the same tier, those with a lower centrality score (ascending centrality  $c(v)$ ).

This approach effectively reduces the subgraph’s complexity while guaranteeing that the core relational structure between the key entities remains intact.

## C.3 Task-Driven Subgraph Visualization Strategy

The visual representation of the extracted subgraph  $G'$  plays a critical role in the VLM’s reasoning capability. We employ a **task-driven visualization strategy** where the graph layout algorithm is dynamically selected based on the task type  $T$ :

- **Hierarchical Layouts:** For sequential reasoning tasks such as *Shortest Path* or *Cycle Detection*, we utilize hierarchical layouts (e.g., Sugiyama algorithm). This layout emphasizes directionality and flow, enabling the VLM to visually trace paths layer-by-layer effectively.
- **Force-Directed Layouts:** For structural analysis tasks such as *Community Detection* or *Clustering Coefficient*, we employ force-directed algorithms (e.g., Fruchterman-Reingold). This approach naturally clusters densely connected nodes, making community structures visually distinct.

---

### Algorithm 1: Ego-centric Subgraph Pruning

---

**Input** :  $G' = (V', E')$ : Initial  $k$ -hop neighborhood subgraph.  
 $v_i$ : Central node for extraction.  
 $N_{\max}$ : Maximum allowed number of nodes.  
 $\mathcal{K}$ : Hierarchical GraphRAG Base.

**Output** :  $G'_{\text{pruned}}$ : Pruned subgraph.

```

if  $|V'| \leq N_{\max}$  then return  $G'$ 
 $V_{\text{cand}} \leftarrow V' \setminus \{v_i\}$ 
// Sort priority for removal:
Farthest, Tier 3 (Peripheral),
Lowest Centrality
Sort  $V_{\text{cand}}$  into list  $L$  by key  $(-d(v, v_i), -t(v), c(v))$ 
 $n_{\text{prune}} \leftarrow |V'| - N_{\max}$ 
 $V_{\text{prune}} \leftarrow L[1 : n_{\text{prune}}]$ 
 $V_{\text{kept}} \leftarrow V' \setminus V_{\text{prune}}$ 
 $G'_{\text{pruned}} \leftarrow$  subgraph of  $G'$  induced by  $V_{\text{kept}}$ 
return  $G'_{\text{pruned}}$ 

```

---



---

### Algorithm 2: Multi-centric Subgraph Pruning

---

**Input** :  $G' = (V', E')$ : Subgraph expanded from  $K$ -shortest paths.  
 $V_{\mathcal{P}}$ : Nodes on the  $K$ -shortest paths.  
 $N_{\max}$ : Maximum allowed number of nodes.  
 $\mathcal{K}$ : Hierarchical GraphRAG Base.

**Output** :  $G'_{\text{pruned}}$ : Pruned subgraph.

```

if  $|V'| \leq N_{\max}$  then return  $G'$ 
 $V_{\text{cand}} \leftarrow V' \setminus V_{\mathcal{P}}$ 
// Sort by priority: tier  $\uparrow$ ,
centrality  $\uparrow$ 
Sort  $V_{\text{cand}}$  into list  $L$  by key  $(t(v), c(v))$ 
 $n_{\text{prune}} \leftarrow |V'| - N_{\max}$ 
 $V_{\text{prune}} \leftarrow L[1 : n_{\text{prune}}]$ 
 $V_{\text{kept}} \leftarrow V' \setminus V_{\text{prune}}$ 
 $G'_{\text{pruned}} \leftarrow$  subgraph of  $G'$  induced by  $V_{\text{kept}}$ 
return  $G'_{\text{pruned}}$ 

```

---

**Reproducibility and Clarity.** To ensure strict reproducibility of the visual inputs, we fix the random seeds for all layout generation processes. Furthermore, to handle potential visual clutter in high-density subgraphs, we apply a **visual anti-overlap mechanism**. Following the pruning process, we dynamically adjust node repelling forces in the layout engine to ensure that no two nodes overlap and that edge crossings are minimized. This provides a clear “what-you-see-is-what-you-get” input for the Visual Graph Thoughts agent.

## D Notations

This section summarizes all notations used throughout this paper, as in Table 8.

## E Experimental Settings

Our experiments are conducted using 8 NVIDIA H20 GPUs. The framework is implemented in Python 3.11 with CUDA Version 12.6, leveraging the vLLM library for efficient inference of the

Table 7: Key Hyperparameters and Framework Configurations.

Category	Parameter	Value
<b>Model &amp; Inference Configuration</b>		
Tensor Parallel	TENSOR_PARALLEL_SIZE	1
GPU Memory Util.	GPU_MEMORY_UTILIZATION	0.4
Max Model Length	MAX_MODEL_LEN	4096
<b>Generation Sampling Parameters</b>		
Temperature	TEMPERATURE	0.01
Max Tokens	MAX_TOKENS	2048
Top P	TOP_P	0.9
<b>Subgraph Extraction</b>		
Max Subgraph Nodes	max_nodes	25
Max Hops	max_hops	2

VLMs.

## E.1 Experimental Settings and Hyperparameters

Our experiments are conducted using 8 NVIDIA H20 GPUs. The framework is implemented in Python 3.11 with CUDA Version 12.6, leveraging the VLLM library for efficient inference of the VLMs. Key hyperparameters and framework configurations are detailed in Table 7.

## E.2 Process-Level DPO Training Details

We fine-tuned  $\mathcal{M}_{VRA}$  (instantiated with Gemma-3-12B and Qwen2.5-VL-7B) using the curated preference dataset  $\mathcal{D}$ . The DPO training was performed with a regularization parameter  $\beta = 0.1$ . We utilized a learning rate of  $5e^{-7}$  with a cosine decay scheduler and an effective batch size of 16 (achieved via gradient accumulation). To ensure memory efficiency during training, we employed LoRA (Low-Rank Adaptation) with rank  $r = 64$  and  $\alpha = 128$ .

The preference dataset  $\mathcal{D}$  consists of **10,221 samples**, specifically constructed to cover diverse reasoning failure modes. The distribution of error types in the rejected trajectories ( $y_l$ ) is summarized in Table 9.

This balanced distribution ensures the model learns to robustly ground its reasoning in visual data while maintaining logical coherence.

## F Stability Analysis of DPO Training

To further verify the robustness of the Visual Reasoning Agent ( $\mathcal{M}_{VRA}$ ) within GraphVista, we conducted an in-depth analysis of the training stability of Process-level Direct Preference Optimization (DPO). This section presents detailed training metrics for Qwen2.5-VL-7B fine-tuned using DPO.

Table 8: Summary of notations used in this paper.

Notation	Description
$G = (V, E)$	Graph with node set $V$ and edge set $E$ .
$\mathcal{K}$	The Hierarchical GraphRAG Base.
$Q, T, \mathcal{E}$	NL question, its task type, and key entities.
$f_{\text{parse}}$	Semantic parsing function for questions.
$\mathcal{N}_k(v)$	$k$ -hop neighborhood of node $v$ .
$ V , N_{\text{max}}$	Total nodes in graph; max nodes in a subgraph.
$K_1\%, K_2\%$	Node percentage in Tier 1 (Core) and Tier 2 (Backbone).
$G' = (V', E')$	A subgraph extracted from the original graph $G$ .
$f_{\text{viz}}, G_{\text{image}}$	Subgraph visualization function and its visual output.
$\mathcal{M}_{VRA}$	The Visual Reasoning Agent.
$\Pi, S_t, H_t$	Reasoning plan, state at step $t$ , and history up to $t$ .
$(o_t, a_t)$	Intermediate output and action at step $t$ .
$C_{\text{retrieved}}$	Retrieved context from the GraphRAG Base $\mathcal{K}$ .
$f_{\text{retrieve}}$	Retrieval function for querying $\mathcal{K}$ .
$\mathcal{D}, (x, y_w, y_l)$	Preference dataset and sample (input, chosen, rejected).
$\pi_\theta, \pi_{\text{ref}}$	DPO policy and fixed reference models.
$r_\theta(y x)$	Ratio of policy probabilities $\pi_\theta(y x)/\pi_{\text{ref}}(y x)$ .
$\beta, \mathcal{L}_{\text{DPO}}$	DPO regularization strength and its loss function.
$f_{\text{extract}}$	The subgraph extraction function.
$\text{dist}_G(u, v)$	Shortest path distance between nodes $u$ and $v$ .
$V_{\mathcal{P}}$	Set of nodes on the $K$ -shortest paths.
$d(v, v_i), t(v), c(v)$	Node’s distance to center $v_i$ , structural tier, and centrality.

## F.1 Training Loss and Reward Curves

Figure 9 illustrates the evolution of loss and implicit rewards over 7,500 training steps.

**Analysis of Training Dynamics** The training process exhibited high stability without the divergence often observed in RL fine-tuning:

- **Loss Convergence:** The training loss (Figure 9a) displays a smooth downward trend, entering a stable convergence phase around 4,500 steps. This indicates that the model effectively learned the distribution of the preference data.
- **Reward Margin Separation:** As shown in Figure 9b, the reward values for *Chosen* paths (indicated by the blue solid line) remain relatively stable, while the reward values for *Rejected* paths (indicated by the orange dashed line) decrease significantly with training steps.

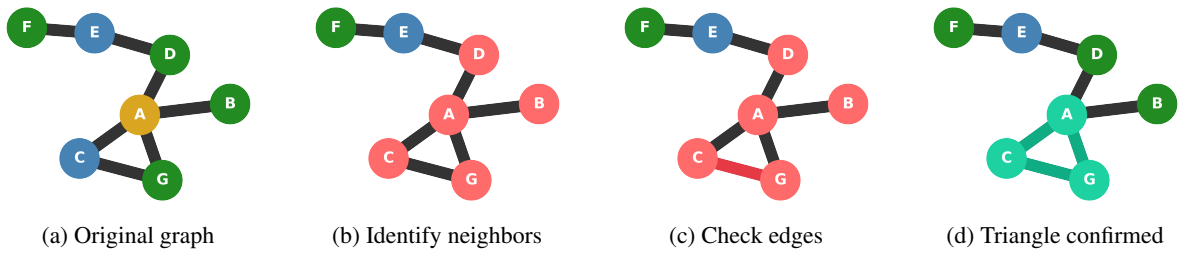


Figure 6: Visualizing the process of **Triangle Detection** for node A. The algorithm identifies node A's neighbors and then checks for edges between them to confirm a 3-cycle.

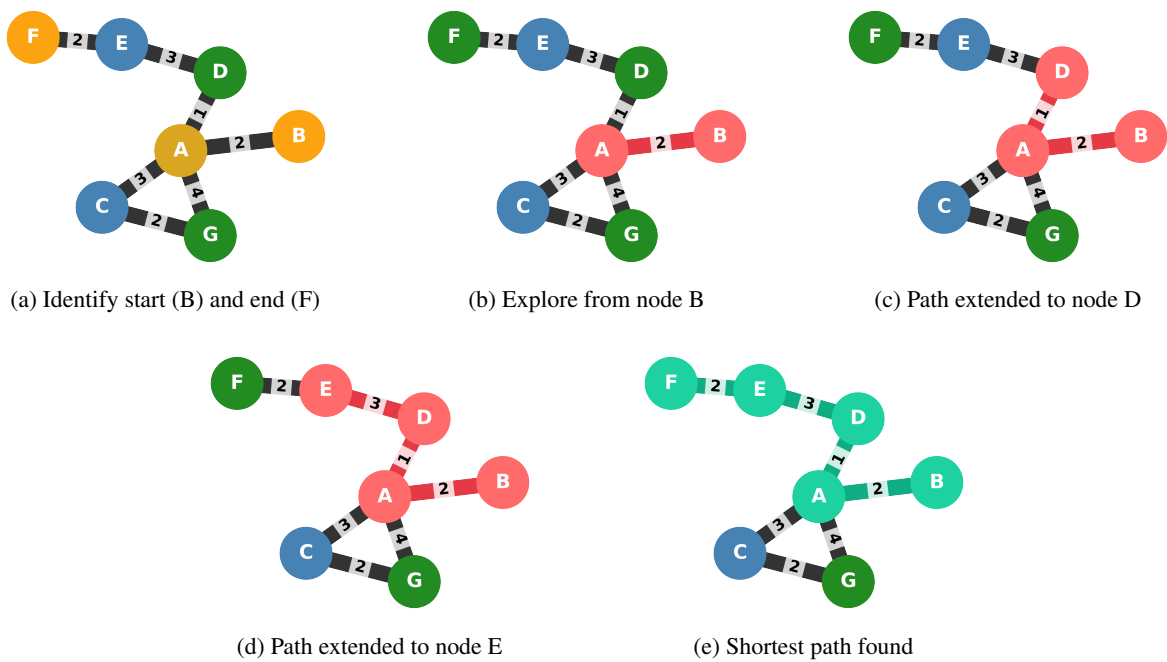


Figure 7: Visualizing a breadth-first search for the **Shortest Path** from node B to F. The algorithm explores neighbors layer-by-layer until reaching the destination.

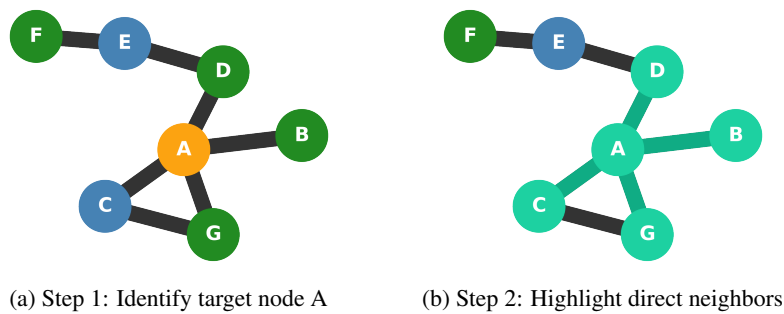


Figure 8: Visualizing **Neighbor Retrieval** for a target node A. The process identifies all nodes that are directly connected to node A.

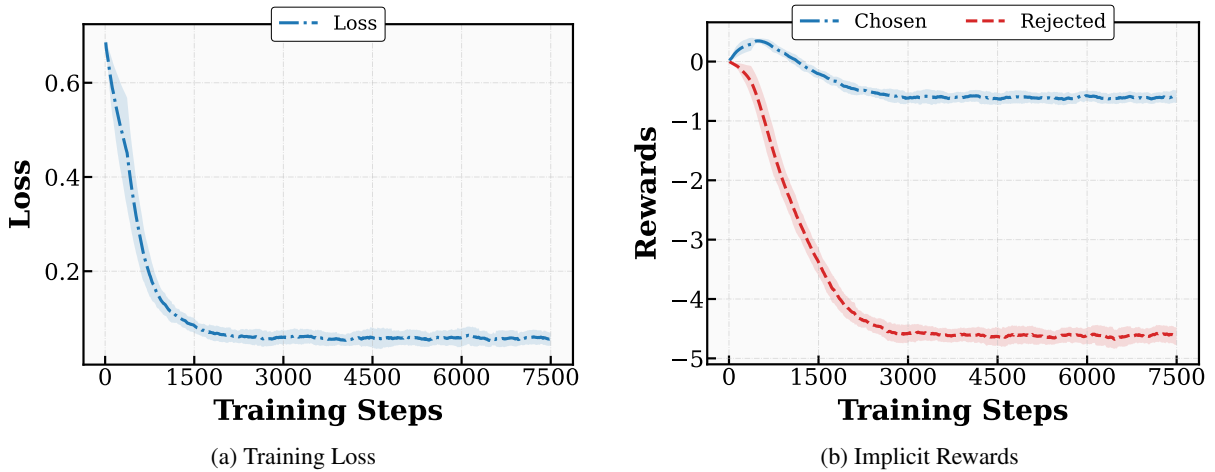


Figure 9: DPO training dynamics of Qwen2.5-VL-7B on the Grena preference dataset. The left panel shows rapid convergence of training loss; the right panel demonstrates that the model successfully widened the reward margin between Chosen and Rejected paths.

Error Type	Proportion	Count
Multimodal Errors	35%	3,577
Textual and Logical Errors	65%	6,644

Table 9: Distribution of error types in the rejected trajectories of the preference dataset.

This indicates that the optimizer maximizes the preference margin primarily by suppressing negative samples (i.e., reasoning chains containing visual hallucinations or logical errors), aligning with the theoretical expectations of DPO.

The observed stability is primarily attributed to our mild regularization strategy ( $\beta = 0.1$ ) and the high-quality process preference dataset derived from real topological structures.

### G Example of Visual Graph Thought Chains

This section provides concrete examples of the step-by-step "Visual Graph Thoughts" process, a core component of our framework for solving complex local reasoning tasks. Each figure-set deconstructs a graph-based problem into a sequence of visual states. This methodology allows the VLM to perform grounded reasoning by identifying and highlighting relevant nodes and edges at each step, culminating in a final solution.

Figure 6 illustrates the process for **Triangle Detection**. The chain begins by identifying the neighbors of a central node, then visually inspects for edges between those neighbors to confirm a

3-cycle. For **Shortest Path** finding, shown in Figure 7, the model emulates a breadth-first search, extending the path layer-by-layer from the source until the destination is reached. Finally, Figure 8 depicts the fundamental operation of **Neighbor Retrieval**, where the model identifies and highlights all nodes directly adjacent to a given target.

### H Task Parsing and Routing Mechanism

This section delineates the decision-making logic of the Planning Agent, which serves as the semantic router within the GraphVista framework. Building upon the architectural overview, we detail the criteria and operational rationale for allocating tasks across Text, Visual, and Collaborative modalities.

To balance computational efficiency with semantic flexibility, the Planning Agent employs a hierarchical **two-stage routing protocol**:

1. **Deterministic Template Routing (Primary Strategy):** The agent initially queries a library of predefined task templates (e.g., "What is the degree of node <ID>?"). Upon identifying an exact match, the task is deterministically typed and immediately routed to the designated modality, bypassing complex inference to minimize latency for standardized queries.
2. **Semantic Inference Fallback (Secondary Strategy):** In the absence of a template match, the agent resorts to semantic parsing. This mechanism analyzes the *structural depen-*

## Task Categorization Examples

1. Text-Modality Tasks (Attribute, Statistic, Retrieval)	2. Visual-Modality Tasks (Topological Reasoning)	3. Collaborative Tasks (Global-Local Composition)
<p><b>Query:</b> “What is the weight of the edge between Node 5 and 9?”</p> <p><b>Logic:</b> Targets discrete attributes. Visual parsing is redundant; direct GraphRAG retrieval is optimal.</p>	<p><b>Query:</b> “Identify a cycle of length 3 involving Node A.”</p> <p><b>Logic:</b> Requires tracing paths. Visual inspection exploits spatial recognition, superior to textual adjacency processing.</p>	<p><b>Query:</b> “Find the highest degree node and list its 2-hop neighbors.”</p> <p><b>Logic:</b> <b>Step 1 (Text):</b> Global search for target node. <b>Step 2 (Visual):</b> Local subgraph reasoning.</p>

Figure 10: Categorization guidelines used by the Planning Agent to route tasks based on structural dependency profiles.

dependency of the query to infer user intent, specifically determining whether the solution necessitates topological traversal (Visual), attribute retrieval (Text), or a synergistic combination of both (Collaborative).

**Rationale for Text-Modality Assignment.** The Text Modality is explicitly designated for tasks akin to database retrieval. This category encompasses: (1) *Existence Verification*, such as confirming the presence of nodes or edges; (2) *Attribute Retrieval*, which involves extracting specific properties (e.g., weights, labels) from the knowledge base  $\mathcal{K}$ ; and (3) *Global Aggregation*, for operations like degree counting. For these tasks, direct retrieval offers computational superiority over visual processing, effectively mitigating latency and eliminating the noise associated with rendering dense graphs for simple factual extraction.

**Rationale for Visual-Modality Assignment.** The Visual Modality is reserved for tasks necessitating "topological perception." Large Language Models (LLMs) processing linearized graph descriptions (e.g., adjacency lists) are prone to contextual attenuation ("lost-in-the-middle" phenomena) when tracing extended paths. Conversely, the visual modality projects the graph into a 2D layout where topological features—such as clusters, bridges, and cycles—become explicitly salient. This is critical for tasks like *Shortest Path* or *Motif Detection*, where the VLM can perceive connectivity patterns more effectively than through iterative textual deduction.

**Rationale for Collaborative-Modality Assignment.** Complex graph reasoning often demands

a hybrid "Search-then-Reason" paradigm. Collaborative tasks are characterized by a *global search space* (optimally handled by GraphRAG indexing) coupled with *local structural verification* (optimally handled by Visual Graph Thoughts). The Planning Agent decomposes such queries into sequential sub-goals. For instance, in diameter estimation, the Text branch first filters peripheral node candidates to reduce the search space, after which the Visual branch executes fine-grained pathfinding on the induced subgraph.

### H.1 Analysis of Task Classification Accuracy

To evaluate the reliability of the routing mechanism, we quantify the task classification error rates across different VLM backbones. Table 10 presents the category misclassification rates for the Planning Agent. The results highlight two key

VLM	Error Rate (%)
GLM-4.1V-9B	0.1650
InternVL3-9B	0.1593
Qwen2.5-VL-7B	0.2482
Gemma-3-12B	0.1697
Qwen2.5-VL-7B (DPO)	0.2495
Gemma-3-12B (DPO)	0.1713

Table 10: Task Classification Error Rates across different VLM backbones. Models fine-tuned with DPO are denoted with (DPO).

observations regarding the routing stability:

- **Impact of Model Scale:** There is a clear correlation between model capacity and routing accuracy. Larger models, such as InternVL3-

9B and GLM-4.1V-9B, consistently achieve lower error rates (approx. 0.16%) compared to the 7B-parameter baselines (approx. 0.25%). This suggests that the semantic nuance required for accurate intent classification benefits significantly from stronger foundational language understanding.

- **Robustness to DPO Fine-tuning:** The application of Direct Preference Optimization (DPO)—while crucial for enhancing the step-by-step visual reasoning detailed in the main methodology—does not degrade the semantic routing capability. The error rates for DPO-tuned models (e.g., Qwen2.5-VL-7B (DPO) at 0.2495) remain comparable to their base counterparts (0.2482). This indicates that our alignment strategy effectively isolates visual reasoning improvements without incurring an “alignment tax” on the model’s general instruction-following and planning abilities.

## I Prompt and QA Templates

This section delineates the standardized prompt templates and Question-Answering (QA) protocols utilized to rigorously evaluate the graph structure understanding capabilities of Vision-Language Models (VLMs) within the proposed **GraphVista** framework. To ensure a comprehensive assessment of both retrieval accuracy and reasoning depth, we categorize these templates into *Text-Modality Tasks* and *Visual-Modality Tasks*. This categorization strictly adheres to the task routing logic executed by the Planning Agent, as detailed in Section 3.3 of the main paper.

### I.1 Text-Modality Tasks: Retrieval via GraphRAG

The Text-Modality category encompasses tasks necessitating precise information retrieval and statistical aggregation rather than abstract topological reasoning. Upon classifying a query as text-dependent, the Planning Agent routes the task to the text-modality branch. This branch leverages the **Hierarchical GraphRAG Base** ( $\mathcal{K}$ ) to execute targeted retrieval operations. By accessing the structured textual descriptions stored within  $\mathcal{K}$  (e.g., node attributes, edge weights, and local neighborhoods), the model extracts factual information directly, thereby bypassing the potential

ambiguity associated with visual inference for non-spatial queries.

### I.2 Visual-Modality Tasks: Reasoning with Visual Graph Thoughts

This category targets tasks demanding intricate topological perception and multi-step logical inference, such as pathfinding or cycle detection. For such queries, the Planning Agent delegates execution to the visual-modality branch. This module orchestrates a pipeline consisting of: (1) *dynamic subgraph extraction* from  $\mathcal{K}$  to isolate the region of interest; (2) *task-driven visualization* to render the subgraph into a high-resolution image; and (3) the deployment of the **Visual Graph Thoughts Agent**. This agent employs a chain-of-thought mechanism grounded in the generated visual evidence to perform iterative, step-by-step reasoning, ensuring that the final answer is topologically consistent.

## System Prompt for Planning Agent

**Role Definition.** You are a Graph Analysis Planning Agent. Your goal is to parse graph-related user queries, extract key entities, classify the task type, and route the query to the most suitable modality.

**Parsing Logic.**

1. **Template Verification.** Check whether the query matches a predefined template (e.g., degree, weight, shortest path). If matched, directly output the standard JSON.
2. **Semantic Fallback.** Otherwise, analyze the structural dependency of the query and determine the required modality (*Text*, *Visual*, or *Collaborative*).

**Task Categorization Guidelines.**

- **Text Tasks.** Direct retrieval of statistics, node attributes, or global counts without complex structural reasoning. *Examples: node count, node degree, edge existence, edge weight.*
- **Visual Tasks.** Queries requiring topological reasoning, pathfinding, or local pattern recognition. *Examples: shortest path, cycle/triangle detection, common neighbors, clique detection.*
- **Collaborative Tasks.** Queries requiring global structural reasoning and decomposition into sequential text and visual sub-tasks. *Examples: graph diameter, critical node detection, connectivity analysis.*

**Output Schema.**

```
{  
  "task_type": "e.g., shortest_path, node_degree",  
  "entities": ["node IDs"] or null,  
  "modality": "Text | Visual | Collaborative",  
  "reasoning": "Brief explanation for modality choice",  
  "decomposition": ["sub-steps if Collaborative"] or null  
}
```

**Current Request:** <USER\_QUERY>

Figure 11: System prompt used by the Planning Agent for task parsing and modality routing.

## Few-Shot Demonstrations for Planning Agent

*User Input:* “What is the degree of node 15?”

```
{
  "task_type": "node_degree",
  "entities": ["15"],
  "modality": "Text",
  "reasoning": "Template match: node_degree; attribute retrieval via GraphRAG.",
  "decomposition": null
}
```

*User Input:* “Find the shortest path between node A and node B.”

```
{
  "task_type": "shortest_path",
  "entities": ["A", "B"],
  "modality": "Visual",
  "reasoning": "Template match: shortest_path; topology reasoning required.",
  "decomposition": null
}
```

*User Input:* “Analyze the network to find bottlenecks impacting flow.”

```
{
  "task_type": "bottleneck_detection",
  "entities": [],
  "modality": "Collaborative",
  "reasoning": "Fallback semantic parsing: global search followed by local verification.",
  "decomposition": [
    "Identify high-betweenness candidates (Text)",
    "Analyze local connectivity for flow restriction (Visual)"
  ]
}
```

Figure 12: Few-shot prompting templates for the Planning Agent.

## Task 1: Node Existence and Properties

**Prompt Template:** *“Determine if node <node\_id> exists in the graph. If so, what is its degree?”*

**GraphVista Behavior:** The planning agent categorizes this as a text-modality task. The model performs a targeted retrieval query against the GraphRAG Base  $\mathcal{K}$  to find the entry for <node\_id>. The retrieved context, containing the node’s 1-hop neighborhood information, is then used to generate a direct answer.

**Rationale:** This task involves retrieving explicitly stored attributes. The textual representation in the GraphRAG base is optimized for such direct lookups, making it efficient and reliable.

## Task 2: Edge Existence and Weight

**Prompt Template:** *"Is there a direct edge between node <node\_id\_1> and node <node\_id\_2>? If yes, provide its weight."*

**GraphVista Behavior:** The query is identified as a text-modality task. The system retrieves the adjacency information for <node\_id\_1> from  $\mathcal{K}$ . The VLM then processes this structured text to check for a connection to <node\_id\_2> and extracts the corresponding weight attribute.

**Rationale:** Like node property lookups, edge verification relies on retrieving specific, pre-processed information. The textual modality avoids the potential for visual ambiguity (e.g., overlapping edges or nodes) that could occur in a complex graph visualization.

## Task 3: Shortest Path Identification

**Prompt Template:** *"Find and list the sequence of nodes that form the shortest path between node <node\_id\_1> and node <node\_id\_2>."*

**GraphVista Behavior:** The planning agent categorizes this as a visual-modality task. The Visual Graph Thoughts Agent extracts a multi-centric subgraph centered around the shortest path candidates between <node\_id\_1> and <node\_id\_2>. This high-resolution subgraph is visualized. The VLM then initiates a Visual Graph Thought process, highlighting nodes and edges step-by-step in the image to trace the optimal path, verbalizing its reasoning at each step before presenting the final sequence.

**Rationale:** Textual descriptions of paths can be convoluted and difficult for VLMs to reason over effectively. A visual representation provides a "what-you-see-is-what-you-get" view of the topology, which is more intuitive for complex structural reasoning and naturally supports stepwise inference grounded in visual evidence.

## Task 4: Triangle Detection

**Prompt Template:** *"Does a 3-cycle (triangle) involving node <node\_id> exist? If so, list the nodes of one such triangle."*

**GraphVista Behavior:** This is routed to the visual modality. An ego-centric subgraph around <node\_id> is extracted from  $\mathcal{K}$  and visualized. The Visual Graph Thoughts agent first identifies the neighbors of <node\_id> in the image. It then visually inspects for edges connecting any two of these neighbors. If such an edge is found, the VLM confirms the existence of a triangle and outputs the three nodes forming the cycle.

**Rationale:** Detecting local structural patterns like cycles is inherently a topological problem. Visual inspection allows the VLM to directly perceive the connectivity pattern, whereas a text-based approach would require inefficient and error-prone traversal of adjacency lists to check for path closures.

### Task 5: Graph Diameter Calculation (Modality-Collaborative)

**Prompt Template:** *"Calculate the diameter of the graph. First identify potential peripheral nodes, and then determine the maximum shortest path distance between them."*

**GraphVista Behavior:** The Planning Agent identifies this as a collaborative task and decomposes it into two sequential sub-tasks.

- **Phase 1 (Text Modality):** The agent queries the GraphRAG Base  $\mathcal{K}$  to identify a set of "peripheral nodes" (typically defined as nodes with low centrality scores in Tier 3). This filters the search space to the most likely candidates for the diameter endpoints.
- **Phase 2 (Visual Modality):** The system extracts multi-centric subgraphs connecting these candidate pairs. The Visual Graph Thoughts agent then visually traces the shortest paths between them to calculate their distances.

Finally, the system aggregates these results and returns the maximum distance found as the graph diameter.

**Rationale:** Calculating diameter requires both global search and local pathfinding. The text modality is efficient for globally filtering candidate nodes based on statistical properties (centrality), avoiding the need to process the entire graph visually. Conversely, the visual modality excels at the subsequent topological reasoning required to accurately count path lengths between specific pairs, which is cumbersome for text-based reasoning.