GITA: Graph to Visual and Textual Integration for Vision-Language Graph Reasoning

Yanbin Wei[1,2*], **Shuai Fu**[1,3*], **Weisen Jiang**[1,2†], **Zejian Zhang**[4], **Zhixiong Zeng**[4], **Qi Wu**[3], **James T. Kwok**[2], **Yu Zhang**[1†]

[1]Department of Computer Science and Engineering, Southern University of Science and Technology
[2]Department of Computer Science and Engineering, Hong Kong University of Science and Technology
[3]Australia Institute for Machine Learning, University of Adelaide [4]Tencent
{yanbin.ust, fus.jayce, zejianzhang33, yu.zhang.ust, waysonkong}@gmail.com
barretzeng@tencent.com, qi.wu01@adelaide.edu.au, jamesk@cse.ust.hk

Abstract

Large Language Models (LLMs) are increasingly used for various tasks with graph structures. Though LLMs can process graph information in the textual format, they overlook the rich vision modality, which is an intuitive way for humans to comprehend structural information and conduct general graph reasoning. The potential benefits and capabilities of representing graph structures as visual images (i.e., *visual graph*) are still unexplored. To fill the gap, we innovatively propose an end-to-end framework, called **G**raph to v**I**sual and **T**extual Integr**A**tion (GITA), which incorporates visual graphs into general graph reasoning. Besides, we construct the **G**raph-based **V**ision-**L**anguage **Q**uestion **A**nswering (GVLQA) dataset from existing graph data, which is the first vision-language dataset for general graph reasoning. Extensive experiments on the GVLQA dataset and five real-world datasets show that GITA outperforms mainstream LLMs on general graph reasoning. Moreover, experimental results demonstrate the effectiveness of the layout augmentation on visual graphs and pretraining on the GVLQA dataset.

1 Introduction

Graph reasoning tasks are pivotal in domains such as recommendation systems [25, 60], social network analysis [7, 29], and knowledge graph reasoning [72, 46, 62]. Various architectures have been developed, from shallow embedding methods [6, 53] to advanced Graph Neural Networks (GNNs) [37, 64] and graph Transformers [71, 40, 8]. While these models excel in graph reasoning tasks, they often lack generalizability, flexibility, and user-friendliness. Achieving good performance with these models typically requires domain-specific tuning, which limits their abilities to generalize across different domains. Additionally, these models struggle to handle diverse tasks with the same architecture. Each task often requires a specialized design, including task-specific data processing and decoder, leading to limited flexibility. Lastly, unlike the Large Language Models (LLMs) that can engage in conversations with users, these models are less explainable and user-friendly.

In contrast, LLMs have shown great generalization capabilities across a wide variety of reasoning tasks [61, 67, 74, 32, 33] by encapsulating task-specific demands within a cohesive and interpretable mechanism - text prompts, under a unified architecture with minimal domain-specific adjustments. These advantages have sparked investigations into the potential of LLMs for graph reasoning. Recent developments lend credence to the notion that LLMs can indeed interpret and manipulate graph-

[*]Equal contribution.
[†]Corresponding authors.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

structured data through textual representations. For example, InstructGLM [66], GPT4Graph [21], and LLMtoGraph [44] convert graphs into textual descriptions and then use these descriptions paired with queries to enable LLMs to generate accurate responses for graph reasoning tasks. Furthermore, the introduction of benchmarks such as GraphQA [14] and NLGraph [59] is a testament to the growing interest in evaluating LLMs' effectiveness on graph reasoning tasks framed in natural languages.

Despite the development of numerous methods and benchmarks for graph reasoning on LLMs, they often overlook the valuable vision modality, which is a natural means for humans to comprehend structural information and has demonstrated its success in various visual reasoning scenarios [30, 68, 69, 34, 3, 54]. Consequently, the following questions arise: (1) *Can incorporating visual information be beneficial for general graph reasoning scenarios?* (2) *If so, how can we effectively integrate the vision modality into graph reasoning?* To the best of our knowledge, these questions remain unexplored.

To answer them, we first propose an end-to-end framework called **G**raph to v**I**sual and **T**extual Integr**A**tion (GITA)[3][4] that systematically integrates visual information into instruction-based graph reasoning, by rendering graph structures to customized visual images which are called *visual graph*. Specifically, the GITA framework has four components: a *graph visualizer* for generating visual graphs, a *graph describer* for producing textual descriptions of the graph structure, a *task-based questioner* that organizes the description and requirements of the current task into prompt instruction, and a *Vision-Language Model* (VLM) to perform vision-language graph reasoning. In the proposed GITA framework, visual information can be incorporated into many tasks with explicit or implicit graph structures, without sacrificing its versatility, flexibility, or user-friendliness. Besides, since there is no dataset for vision-supported general graph reasoning capabilities, we construct the first vision-language dataset for general graph reasoning purposes called **G**raph-based **V**ision-**L**anguage **Q**uestion **A**nswering (GVLQA)[5] based on the proposed GITA framework. The GVLQA dataset consists of 526K instances covering seven representative graph reasoning tasks, aiming to thoroughly evaluate the structure-based graph reasoning abilities of VLMs and LLMs. Extensive experiments on the GVLQA dataset and five real-world datasets demonstrate the effectiveness of the proposed GITA model. Furthermore, we delve into the effects of visual graph augmentation strategies and find that layout augmentation can dramatically boost vision-based graph reasoning performance.

Our main contributions are summarized as follows.

- We introduce an end-to-end GITA framework, innovatively integrating vision modality to boost the graph reasoning abilities of language models.
- We establish GVLQA, the first vision-language question-answering dataset for general graph reasoning purposes. It can be used to thoroughly evaluate the structure-based graph reasoning abilities of LLMs/VLMs and can also be used as pretraining data to boost the performance of downstream tasks.
- Extensive experiments on benchmark datasets across various graph reasoning tasks demonstrate the effectiveness of the proposed GITA framework and the benefits of layout augmentation on visual graphs.

2 Related Work

Graph Reasoning. Graph reasoning [5, 63] aims to answer questions based on graphs, which involves utilizing graph structures to guide the reasoning process to generate answers. Graph reasoning has a wide variety of applications in social network analysis [47, 41], bioinformatics [31, 18], chemistry [19], physics [4], knowledge graph reasoning [6], and recommendation systems [39, 26]. Many graph reasoning methods have been proposed. Early attempts [6, 53] learn node and edge representations through shallow modules, which may have only limited expressive power. Graph Neural Networks (GNNs) such as GCN [37], GAT [58], GraphSAGE [23], MPNN [19], and GIN [64] use message-passing paradigm [19] to model graph dependencies and update node features. Transformer-based

[3]Project Homepage: `v-graph.github.io`.

[4]Code Repository: `https://github.com/WEIYanbin1999/GITA/`.

[5]Dataset: `https://huggingface.co/collections/Yanbin99/`.

graph models [71, 40, 8] further propose to use self-attention to increase the expressiveness and long-range dependency. However, as discussed in Sec 1, these models may exhibit limited generalizability, flexibility, and user-friendliness.

LLMs on Graph Reasoning. There have been many attempts to use LLMs in graph reasoning. Depending on how they align the input spaces of graphs and LLMs, we categorize them into two types: Graph-to-text and Graph-to-token. Graph-to-text methods transform a graph into textual descriptions, which are concatenated with the instructions and fed to the LLM for querying. For example, InstructGLM [66] uses natural language to describe the graph and proposes instruction prompts to fine-tune the LLM. He et.al [27] applies LLMs to explain graphs for training GNNs, while Chen et.al [10] treat LLMs as enhancers to exploit text attributes or as predictors for node classification on text-attributed graphs. GPT4Graph [21] and LLMtoGraph [44] convert graphs into specific code or natural language formats by the powerful ChatGPT [48, 49]. On the other hand, Graph-to-token methods include GraphGPT [55], GraphToken [50] and LLaGA [9]. For these methods, the graph is represented as a specially designed token sequence, which is projected or merged into the LLM's token space for text-based reasoning. However, none of the aforementioned methods represent the graph structure information as images, highlighting the uniqueness of the proposed GITA framework and GVLQA dataset.

Large Vision-Language Models. Large VLMs have significantly expanded the cognitive abilities of LLMs by integrating the vision modality to address vision-language tasks. Many methods have been proposed. Some early explorations like Flamingo [2], CLIP [51], and BLIP-2 [43] use a visual encoder for processing images and align the visual and textual embeddings. Subsequent models like LLaVA [45] and MiniGPT-4 [75] combine visual and textual inputs in a single LLM for solving multimodal tasks. InstructBlip [11] proposes an instruction-aware query transformer and trains a vision-language model by instruction tuning. However, despite progress in a wide range of vision-language tasks [70, 16], using visual information in graph reasoning remains overlooked. We take the first step in this field, pushing the boundaries of VLMs in graph reasoning.

3 GITA: Graph to Visual and Textual Integration

3.1 Preliminary

Graph Reasoning. In traditional graph reasoning settings, models typically rely on two main inputs: (i) the graph structure $G = \{C, E\}$, where C and E are the set of vertices and edges, respectively; (ii) the task requirement T, encompassing specific operations or questions pertaining to the graph. Based on the information provided in G and a specific task requirement T, models are expected to output a reasonable answer A. On the other hand, in the context of instruction-based graph reasoning methods, it is necessary to convert these inputs into textual form. This transformation facilities graph reasoning within natural language, allowing for improved interpretation and harnessing the formidable reasoning capabilities of large language models.

3.2 Architecture

Overview. Different from the above graph reasoning methods, we propose a **G**raph to **I**mage-**T**xt **A**ssistant (GITA), which is the first attempt to perform graph reasoning in a vision-text-based manner. GITA comprises four pivotal components: a task-agnostic graph visualizer V, a graph describer D, a task-specific questioner Q, and a VLM reasoner R_ϕ, as illustrated in Figure 1. Firstly, V and G are designed to produce visual depictions (i.e., *visual graphs*) and textual descriptions of the graph structure inputs, respectively. Then, given the task requirement T and the textual description produced by D, Q is designed to form a task-specific query. Finally, R_ϕ receives the visual input I_G from V based on the visual graph and the textual input Q_G^T from Q, then generates answers A in natural language. In the following, we introduce the four components in detail.

Graph Visualizer. The role of the graph visualizer is to generate visual graphs from structural graphs. The image representation of a structural graph is not unique, as there can be variations in many aspects, such as backdrop colors, layouts, and node shapes. These variations may enhance the robustness of models through effective training but simultaneously increase the learning difficulty for models. Therefore, balancing *consistency* and *variety* is necessary during the graph visualization process. This trade-off is reflected in our design of graph visualizer, by maintaining consistency in

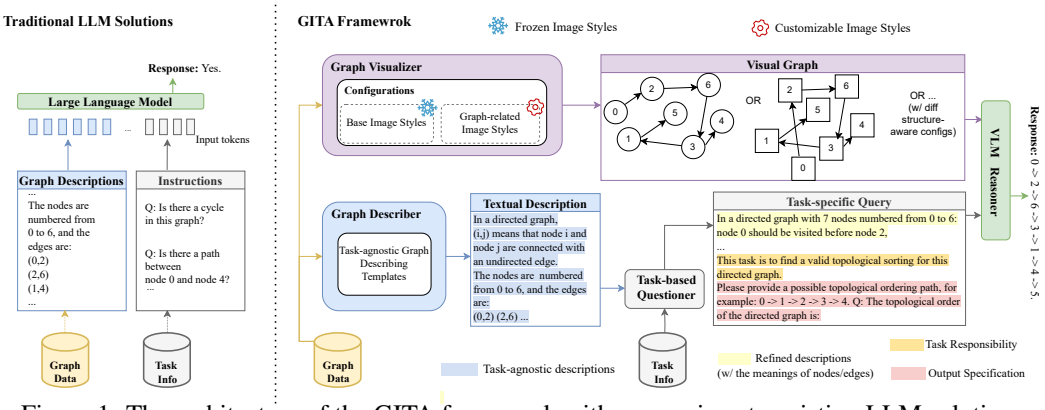


Figure 1: The architecture of the GITA framework with comparison to existing LLM solution.

basic image styles common to general images (i.e., size, resolution, backdrop) and only introducing customizable variations in four graph-related image styles unique to visual graphs (i.e., layout, node shapes, node outline styles, and edge thickness). Graph visualization in V can be formulated by the following equation:

$$I_G = V(G, \Gamma, \Delta), \tag{1}$$

where I_G denotes the visual graph derived from graph G, while Γ and Δ are the fixed basic image styles and customizable graph-related image styles, respectively.

Visualizing the entire graph can be challenging when the number of nodes or edges is very large, affecting the clarity of the images. To address this, our graph visualizer adopts the standard strategy of k-hop subgraph sampling. Specifically, k-hop subgraph sampling for a node u in the set of vertices C involves selecting a subgraph $G_u = \{C_u \subseteq N_k(u), E_u \subseteq E\}$, where $N_k(u)$ includes nodes within k steps from u and each edge (i, j) in E_u connects nodes within C_u. To generate the visual graph of the k-hop subgraph G_u centered on u, the nodes within G_u are relabeled from 0 to $|C_u| - 1$ to facilitate the generalization of visual graphs. Subsequently, this relabeled subgraph G_u is fed to the graph visualizer to generate its visual graph I_{G_u} by Eq. (1).

In practice, the graph visualizer can be implemented by a variety of graphic visualization tools, such as Graphviz [17], Matplotlib [56], and NetworkX [22]. Among them, Graphviz can automatically design the layouts of visual graphs, and is especially suitable for building large-scale datasets. Matplotlib is excellent for customizable plots with fine-grained control, and NetworkX excels in complex network analysis. We have implemented various graph visualizers using modular, plug-in architecture in GITA. Specific examples of the visual graphs generated with these tools can be found in Appendix D.

Graph Describer. The graph describer D is tasked with generating task-agnostic textual descriptions of a given graph G. To ensure clarity and fidelity of these descriptions, we meticulously craft a curated set of *graph-describing templates*. The graph description templates outlined in Appendix E are designed to cover a broad spectrum of scenarios, accommodating various graph configurations including directed or undirected graphs and those with or without node or edge weights. To generate the description for a given graph, the graph describer initially selects an appropriate template based on the graph's characteristics, such as its directionality and whether it includes node attributes or edge weights. Subsequently, this template is used by replacing placeholders with actual data, such as the number of nodes, the number of edges, and the endpoints of each edge, to craft detailed descriptions tailored to the specific graph in question. The process for D to generate textual descriptions can be formulated as follows:

$$D_G = D(G, P), \tag{2}$$

where D_G denotes the textual description generated by graph describer, and P is the graph-describing template of the graph G.

By introducing these unified and structured graph-describing templates, the graph describer is empowered to generate coherent and informative descriptions that focus on the inherent characteristics of the graph itself, independent of specific task requirements.

Questioner. The questioner Q is tailored to capture the intricate requirements of specific tasks and reflect them in its output task-specific query. In detail, Q receives the task-agnostic textual

descriptions from the graph describer and refines them to align with the task context by elucidating the concrete meanings of nodes and edges. These refined descriptions are then enriched with task responsibilities and input/output specifications to form task-specific queries. The formulation of the questioner to generate the task-specific queries can be represented as follows:

$$Q_G^T = Q(T, D_G), \tag{3}$$

where Q_G^T represents the task-specific query generated by the questioner with given the task requirement T and the textual description D_G. The construction of task-specific queries can be approached in two main ways: manual template-based construction and bootstrapping LLM agents. Manual template-based construction enriches D_G with task-specific manual templates, which is preferred for tasks with precise requirements, such as the Traveling Salesman Problem (TSP) [12], where accuracy is critical and the task definitions are well-understood. This is because it can ensure clarity and reduce the risk of errors due to its meticulous attention to details. On the other hand, bootstrapping LLM agents for automated synthesis is more economical and suitable for dynamic or bespoke tasks, such as robotic planning or complex gaming scenarios, as it can take advantage of the speed and adaptability of LLM agents to interpret context and generate appropriate queries, minimizing manual effort and enhancing responsiveness to changing conditions. Both methods are illustrated with examples in Appendix F, showcasing their applications and benefits in different scenarios.

VLM Reasoner. The VLM reasoner R_ϕ performs final graph reasoning with visual inputs I_G from V and textual inputs Q_G^T from Q, and outputs responses in natural language. This reasoning process can be represented as the following:

$$A = R(I_G, Q_G^T), \tag{4}$$

where A is the answer generated by the vision-language model R. In this work, we adopt GPT-4V and LLaVA-7B/13B as VLM reasoners. These models are regarded as representatives in the realm of closed-source and open-source VLMs, respectively.

In summary, GITA systematically incorporates the vision modality into instruction-based graph reasoning. In Appendix B, we discuss the characteristics of GITA, in aspects of generalizability, flexibility and user-friendliness.

3.3 Visual Graph Augmentation

Visual graphs generated for the same graph G can be considered as an unique data augmentation technique. Building on the four graph-related image styles introduced in the graph visualizer part of Sec 3.2, we propose the following augmentation strategies: **layout augmentation**, **node shape augmentation**, **node outline style augmentation**, and **edge thickness augmentation**. Specifically, layout augmentation involves altering the layout styles while keeping all the other settings constant. Similarly, by changing only the respective attributes, we can implement node shape augmentation, node outline style augmentation, and edge thickness augmentation. These four proposed augmentation strategies facilitate studies on the importance of each in enhancing the graph reasoning abilities of VLM reasoners.

3.4 Training

Given a visual graph I_G and a text-specific query Q_G^T, along with the target answer A_t, the VLM reasoner of GITA is trained to generate answers A. Specifically, I_G is input into the vision encoder of the VLM reasoner, resulting in a set of visual features F. If there is a dimension difference between F_v and the pretrained word embeddings, these F will be aligned with the pretrained word embedding space of the text decoder by a vision-to-text projector. Finally, the aligned visual features $F_{aligned}$ and Q_G^T are concatenated as input sequences of the text decoder.

Formally, given I_G, Q_G^T, and A_t, the VLM reasoner is trained by minimizing the following negative log-likelihood:

$$\mathcal{L}_\phi = -\sum_{i=1}^{|A|} \log\ p_\phi(A_i \mid F_{aligned}, Q_G^T, A_{<i}), \tag{5}$$

where ϕ is the trainable parameter and A_i denotes the prediction token at the i-th position. Besides, $A_{<i}$ represents the first $i-1$ predicted tokens. During the inference process, GITA is capable of accepting structure graphs as inputs and performing graph reasoning in an end-to-end manner.

4 GVLQA Dataset

In this section, we introduce the GVLQA dataset to fill the absence of a vision-language-based general graph reasoning dataset. It is designed to: 1) evaluate the graph reasoning capabilities of VLMs or LLMs; 2) help models acquire fundamental graph comprehension and reasoning abilities as a pretraining dataset.

4.1 Construction

The GVLQA dataset is created by utilizing the graph visualizerthe graph describer, and questioner in GITA to generate vision-language-based question-answer pairs for graph reasoning on an open-source graph dataset. Specifically, we first extract both the original graph structures and the ground-truth outputs from the NLGraph-full dataset [59]. Then the graph visualizer (detailed in Sec 3.2) and the graph describer (outlined in Sec 3.2) are used to generate visual graphs and textual descriptions for these original graph structures, respectively. Afterwards, the questioner (described in Sec 3.2) further improves and enriches the textual descriptions by converting them into textual queries. At the same time, it transforms the ground-truth output into text-based answers, following specific output requirements. By combining these visual graphs, textual queries, and text-based answers, we obtain the Graph-based Vision-Language Question Answering (GVLQA) dataset.

In the process of establishing GVLQA, we employed graphviz [17] to instantiate the graph visualizer. This choice is made due to its multitude of pre-defined layout algorithms, which enable convenient adjustment of visual graph layouts. Additionally, manual template-based constructed queries are utilized as the questioner because these tasks are famous with well-defined requirements.

4.2 Structure

The GVLQA dataset comprises 526K samples, each consisting of a visual graph, a textual query, and its corresponding answer. It is divided into five subsets: GVLQA-BASE, and four augmentation subsets GVLQA-AUGLY, GVLQA-AUGNS, GVLQA-AUGNO, and GVLQA-AUGET. In GVLQA-BASE, the visual graphs are uniformly styled. The remaining four augmentation subsets are derived from GVLQA-BASE through the four visual graph augmentations (Sec 3.3), varying in six different layouts, three node shapes, four node outline styles, and four degrees of edge thickness, respectively. Detailed statistics of the four subsets are shown in Table 6 of Appendix C.

Each GVLQA subset undergoes evaluation across seven graph reasoning tasks, outlined as follows.

- **Connectivity** [52] (denoted Connect): Determine whether two randomly selected nodes u and v in an undirected graph are connected.
- **Cycle** [52]: Identify whether a cycle exists in an undirected graph.
- **Topological Sort** [35] (denoted TS): Find a valid topological sort for a directed acyclic graph. Here, topological sort outputs a linear ordering of the nodes such that for every directed edge $u \leftarrow v$, node u comes before v in the ordering.
- **Shortest Path** [13] (denoted SP): Find the shortest path between two nodes in a weighted undirected graph. The shortest path between two nodes is the path connecting the two nodes with the minimum sum of edge weights along the path.
- **Maximum Flow** [15] (denoted MaxFlow): Calculate the maximum flow from a source node to a sink node in a network graph.
- **Bipartite Graph Matching** [36] (denoted BGM): Find a matching set in a bipartite graph with the largest number of edges. A matching set is a collection of edges in which no two edges share any common node.
- **Hamilton Path** [20] (denoted HP): Find a valid Hamilton path in an undirected graph. A Hamiltonian path is a path that traverses each node in a graph exactly once.

Figure 6 offers illustrations for these tasks in the GVLQA-BASE dataset. Illustrations of all the GVLQA subsets are provided in Appendix H.

5 Experiments

In this section, we extensively evaluate the performance of LLM baselines and the proposed GITA on the GVLQA-BASE and five real-world datasets. To better clarify the reasoning capabilities of solely visual graphs, we also test GITA without the textual descriptions of graphs, which can be considered as a variant of GITA and denoted as vision-only (VO). In this case, the visual graph is the only information source for graph reasoning. Additionally, we investigate the importance of visual graph augmentation (Sec 3.3) strategies, by comparison GITA-7B trained on GVLQA-BASE and on the other augmentation subsets of GVLQA (Sec 4.2). Lastly, we investigate the effectiveness of using GVLQA as the pretrained dataset on real-world datasets. The evaluation metrics for all experiments are accuracy by exact matching. For the fine-tuning setting, we fine-tune the LoRA adapters [28] for all weight matrices in the text decoder of the VLM reasoner, while keeping the vision encoder in the VLM reasoner frozen. More detailed experimental settings are in Appendix G.

5.1 Evaluation on the GVLQA-BASE Dataset

In this subsection, we perform experiments on the GVLQA-BASE dataset to compare GITA with popular LLMs including GPT-4 Turbo [49], LLaMA2-7B/13B [57], and Vicuna-7B/13B [73], under both zero-shot and fine-tuning settings. The experimental results are shown in Table 1. Based on these results, we can obtain the following observations.

Observation 1: GITA Outperforms LLM Baselines. As can be seen in Table 1, GITA consistently outperforms the LLM baselines under the same setting. This underscores its SOTA effectiveness in instruction-based graph reasoning tasks, showing robust capabilities across different parameter scales under both fine-tuning and zero-shot settings. Moreover, under the fine-tuning setting, incorporating the vision modality consistently benefits 7B models. But for the 13B models, the performance of some tasks may degrade. This could be attributed to the greater challenge of aligning representations of the visual and textual modalities in the larger 13B models compared to the 7B models, in the case of only fine-tuning LoRA adapters in the text decoder. We speculate that full training could potentially address this issue. However, we leave this as future work due to resource constraints.

Observation 2: Mainstream Open-source VLM/LLMs Lack Fundamental Graph Reasoning Abilities. The zero-shot results illustrate that prominent open-source LLMs or VLMs, including LLaMA2, Vicuna, and LLaVA, exhibit minimal graph reasoning capabilities on the GVLQA-BASE dataset. Specifically, these models produce random answers, i.e., randomly responding with either "Yes." or "No." for tasks involving Connect and Cycle, resulting in a performance close to 50%. Cur-

Table 1: Accuracy (%) comparisons on GVLQA-BASE under zero-shot and fine-tuning settings, where "VO" denotes a variant of GITA using only the vision modality.

Models	Connect	Cycle	TS	SP	MaxFlow	BGM	HP	**Avg**
Zero-shot								
LLaMA2-7B	**50.06**	49.43	0.00	0.00	0.00	0.00	0.00	14.21
Vicuna-7B	**50.06**	49.43	0.00	0.00	0.00	0.00	0.00	14.21
GITA-7B (VO)	**50.06**	**50.33**	0.00	0.00	0.00	0.00	0.00	**14.34**
GITA-7B	**50.06**	49.43	0.00	0.00	0.00	0.00	0.00	14.21
GPT-4 Turbo	76.70	49.51	19.59	35.35	**6.89**	42.11	47.04	39.60
GITA-ZS (VO)	57.76	**63.34**	5.34	4.88	1.59	46.60	10.74	27.18
GITA-ZS	**82.58**	51.46	**19.71**	**37.69**	6.00	**52.21**	**50.00**	**42.81**
Fine-tuning								
LLaMA2-7B	97.33	94.63	33.26	26.01	9.56	90.86	23.95	53.66
Vicuna-7B	97.58	95.04	34.46	25.98	9.33	91.04	25.55	54.15
GITA-7B (VO)	59.97	96.34	13.30	5.72	2.89	93.01	1.11	38.91
GITA-7B	**98.95**	**96.67**	**41.12**	**32.15**	**20.00**	**93.19**	**29.26**	**58.76**
LLaMA2-13B	98.79	93.36	33.83	27.93	12.22	91.34	33.46	55.85
Vicuna-13B	**99.35**	94.39	36.73	28.53	11.34	92.65	**34.81**	56.83
GITA-13B (VO)	58.00	**96.91**	14.45	5.72	4.89	**93.19**	1.85	39.29
GITA-13B	99.14	95.60	**38.69**	**40.47**	**20.66**	92.12	33.33	**60.00**

Table 2: Accuracy (%) comparisons across GVLQA subsets using GITA-7B (VO). ↑ denotes dramatic performance improvement.

	Connect	Cycle	TS	SP	MaxFlow	BGM	HP	**Avg**
GVLQA-BASE	59.97	96.34	13.30	5.72	2.89	93.01	1.11	38.91
GVLQA-AUGNS	59.85	96.75	14.17	6.61	3.78	91.58	1.48	39.17
GVLQA-AUGNO	54.87	96.50	14.29	5.54	**3.94**	92.83	1.11	38.44
GVLQA-AUGET	57.98	96.91	13.37	5.97	3.11	91.76	0.74	38.55
GVLQA-AUGLY	**87.18** ↑	**97.07**	**14.86**	**76.55** ↑	**3.94**	**93.19**	**70.74** ↑	**63.36** ↑

Table 3: Accuracy (%) comparisons on real-world datasets under zero-shot and fine-tuning settings, where ‡ indicates the usage of a checkpoint pretrained in the Cycle task of GVLQA-BASE.

Models	ca-GrQc	ca-HepTh	PolBlogs	Cora	CiteSeer	**Avg**
Zero-shot						
LLaMA2-7B	40.59	48.89	10.74	24.35	30.33	30.98
Vicuna-7B	41.35	50.00	8.72	26.94	29.13	31.22
GITA-7B	71.95	86.06	46.98	31.37	30.63	53.40
GITA-7B‡	**72.02**	**86.08**	**48.32**	**32.10**	**31.83**	**54.07**
Fine-tuning						
LLaMA2-7B	76.57	89.06	80.54	83.76	73.27	80.64
Vicuna-7B	78.95	89.85	80.54	84.87	74.17	81.68
GITA-7B	79.70	91.13	84.56	85.24	75.07	83.14
GITA-7B (w/ AUGLY)	79.77	91.21	**85.23**	85.24	75.68	83.43
GITA-7B‡	**80.46**	**91.68**	**85.23**	**86.35**	**76.57**	**84.06**

rent SOTA closed-source LLMs or VLMs, including GPT-4 Turbo and GPT-4V, demonstrate superior zero-shot performance compared with the aforementioned open-source models. This observation implies that current open-source LLMs and VLMs lack basic graph reasoning ability, which may be attributed to the insufficient availability of relevant training data. Such observation also enhances our motivation to propose the GVLQA dataset, with the aim of improving the graph reasoning capabilities of VLMs/LLMs.

Observation 3: Increasing Model Size Leads to Better Graph Reasoning Capabilities. The comparison of VLMs/LLMs with different parameter sizes, specifically 7B and 13B models, verify the benefits of increasing the model size for graph reasoning capabilities. In this regard, GITA-13B outperforms its counterpart with 7B parameters (GITA-7B) both on average and across four of the seven tasks. However, it is worth noting that GITA-13B does not outperform GITA-7B on the other three tasks. We hypothesize that this discrepancy may be attributed to insufficient modality alignment due to LoRA fine-tuning.

Observation 4: Vision and Text Modalities Proficient in Different Types of Graph Reasoning Tasks. We explore the individual capabilities of the visual and textual modalities within the GITA framework. The results indicate that the text and vision modalities can complement each other and contribute to better performance than individual ones, as removing either modality leads to performance drops in most cases (Vicuna & GITA (VO) and GPT-4 Turbo & GITA (VO) in Table 1). While the graph reasoning capability provided by the vision modality may not be as strong as that of the text modality in most cases, relying solely on vision still enables the model to possess basic graph reasoning abilities. Specifically, the model outperforms text-based LLMs in 2 of the 7 tasks (Cycle and BGM) when relying solely on vision. This consistent improvement across all comparison groups demonstrates the potential of the vision modality to excel in certain graph reasoning tasks, leveraging its ability to capture visual patterns like cycles and graph properties such as bipartition. In contrast, text exhibits a higher proficiency than vision modality in sequence-related graph reasoning problems, particularly on tasks such as TS, SP, and HP, which require constructing ordered node sequences.

5.2 Evaluation for the Visual Graph Augmentations

To assess the impact of the proposed visual graph augmentation strategies (including layout, node shape, node outline style, and edge thickness augmentations), we compare the performance of vision-only GITA-7B models trained on the four augmented subsets of GVLQA and on GVLQA-BASE

(without augmentation). The results are presented in Table 2. To fully utilize the visual information in visual graphs, we fine-tune the visual encoder of VLMs in addition to the vision-to-text projector and the LoRA adapters within the text decoder in this experiment.

As can be seen from the results, a significant enhancement in overall performance is observed with the introduction of layout augmentation (GVLQA-AUGLY). The average performance improves remarkably from 38.91% to 63.36%. Notably, significant improvements are observed on SP (5.72% to 76.55%), HP (1.11% to 70.74%), and Connect (59.97% to 87.18%). These findings highlight the critical role of layout augmentation in generating visual graphs. In other words, this observation suggests the potential for creating larger-scale datasets for vision-language-based graph reasoning, which could significantly contribute to advancing this field. Conversely, the other three augmentations do not yield such substantial performance improvements, further emphasizing the importance of layout augmentation in vision-language-based reasoning.

5.3 Evaluation on Real-World Datasets

In this section, we study the effectiveness of GITA on the ca-GrQC [42] and ca-HepTh [42] datasets for the link prediction task, and on the PolBlog [1], Cora [65] and CiteSeer [65] datasets for the node classification task. Table 8 in the appendix C presents the statistics of these datasets. The graph can have thousands of nodes/edges, making it infeasible to feed the entire graph into the model. Consequently, we employ k-hop subgraph sampling (with $k = 2$) discussed in Sec 3.2 to satisfy the token length restriction of LLMs and visual graph scope effectively.

The experimental results are presented in Table 3. It is evident that GITA consistently outperforms the LLM baselines, and its performance progressively improves with the addition of layout augmentation and the use of the GVLQA checkpoint. Notably, we emphasize the advantages of using GVLQA-BASE as a pretrained dataset by comparing it with GITA-7b. Performance improvements of 0.67% and 0.92% are observed in the zero-shot and fine-tuning settings, respectively. This highlights the potential application value of the proposed GVLQA dataset.

5.4 Comparison of GITA with Dedicated Graph Baselines

Table 4: Accuracy (%) comparisons among dedicated GNNs and GITAs on GVLQA-Base.

	Connect	Cycle	TS	SP	MaxFlow	BGM	HP	**Avg**
GCN	79.65	70.89	45.71	44.56	**56.44**	76.70	32.22	58.02
SAGE	82.72	73.58	44.51	**49.25**	50.67	81.00	**36.67**	59.78
GITA-7B	98.95	**96.67**	41.12	32.15	20.00	**93.19**	29.26	58.76
GITA-13B	**99.14**	95.60	38.69	40.47	20.66	92.12	33.33	**60.00**

Though GITA is designed for language-based general graph reasoning settings, which are much more user-friendly (by user-readable natural language) and general (unique model architecture for various scenarios) than the typical application of dedicated GNNs, it remains essential to conduct a comprehensive comparison with specialized GNNs to elucidate the strengths and limitations of GITA's applicability and capabilities. To this end, we assess the graph reasoning abilities of GITA against dedicated GNNs, including GCN [38] and SAGE [24], using the GVLQA-Base dataset, as detailed in Table 4. In addition, we explore and compare the effects of k-hop subgraph sampling on the proposed GITA and GNN baselines. Using the ca-Hepth dataset, we analyze the impact of increasing the number of hops k on the reasoning time and performance of both GITA and GNNs. The results are in Table 5.

Overall Graph Reasoning Ability Comparison. As shown in Table 4, compared to the dedicated GNNs, the fine-tuned GITA-7B models have comparable average graph reasoning performance, with the larger GATA-13B model performs slightly better. In particular, compared to GNNs, the GITA model shows a stronger ability in recognizing local structures in the graphs (Connect and Cycle) and to accomplish tasks with obvious layout heuristics (BGM). We believe that this advantage comes from GITA's visual perception. For SP and MaxFlow, GITA's performance is inferior to GNNs. This may be because GNNs process edge weight more effectively through the message-passing mechanism.

Scalability and Performance Variation with Different Numbers of Hops k. The inference time results are shown in Table 5. As can be seen, GITA demonstrates inferior scalability compared to the GNN baselines. Its scalability remains stable as the sampled graph size (i.e., k) increases.

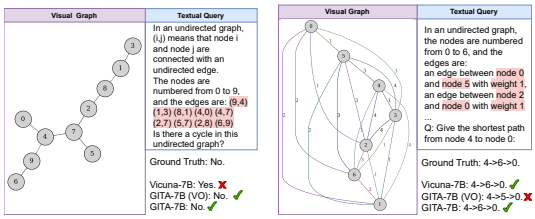
From the accuracy results in Table 5, GITA, GCN, and SAGE achieve their peak performance at $k = 2$, suggesting that a small sampled graph size suffices for optimal performance. Though the dedicated GNNs attain higher peak performance than GITA, they exhibit performance declines as k increases (e.g., 3 or 4), while GITA's performance is more stable w.r.t. k.

Table 5: Accuracy (%) and Inference Time (in parentheses) for GNNs and GITA on ca-Hepth Dataset with different subgraph sampling hop number $k \in \{1, 2, 3, 4\}$.

	GCN	SAGE	GITA-7B
k=1	93.27 (0.02s)	94.40 (0.03s)	90.33 (17.23min)
k=2	94.49 (0.04s)	94.43 (0.04s)	91.13 (17.66min)
k=3	91.10 (0.04s)	90.95 (0.18s)	90.31 (17.22min)
k=4	81.92 (0.05s)	83.60 (0.22s)	86.10 (17.01min)

5.5 Case Study

In this section, we present examples of graph information provided in both visual and textual formats, which offer some intuitive interpretations for our experimental results. Figure 2 (a) shows an example where the GITA-7B (VO) model outperforms its LLMs-based counterpart, and Figure 2 (b) shows an opposite scenario.



(a) (b)

Figure 2: A comparative case study of graph representation in vision and text modalities. All methods are trained on the GVLQA-BASE dataset.

The task depicted in Figure 2(a) is cycle detection and the correct answer is 'No'. This is predicted successfully by the vision-only GITA-7B model, while the text-based Vicuna-7B fails. In this example, recognizing cycle patterns is much easier in visual graphs, whereas text-based LLMs struggle with disordered textual descriptions of edges, which could inherently involve greater complexity and more challenges.

On the other hand, the fixed layout of visual graphs presented in GVLQA-BASE may impede the visual encoder in identifying the shortest path between two nodes, although we have verified layout augmentation can greatly improve the graph reasoning abilities of models, as shown in Sec. 5.2. This limitation might arise from the confusion caused by the visual distance within an image, without considering the weights between the nodes. For instance, in Figure 2(b), the correct answer is '4->6->0', which visually appears as a more convoluted path but numerically has a shorter path length of $3 = 1 + 2$. In contrast, the incorrect answer given by GITA-7B (vision-only) is '4->2->0', which has a higher path length cost of $4 = 1 + 3$ but visually seems like a more direct shortcut. This observation further validates the effectiveness of employing layout augmentation to enhance performance in this task. Layout variations of visual graphs play a crucial role in mitigating the visual confusion caused by the spatial arrangement within a visual graph. However, it seems more effective for text-based LLMs to handle explicitly separated nodes and weights, as illustrated by the text (in red) in Figure 2(b).

6 Conclusion

In this paper, we propose an end-to-end framework called GITA for vision-language-based graph reasoning. Extensive experiments validate the superiority of incorporating visual information into instruction-based graph reasoning. Furthermore, we conduct comparative analysis of the four proposed visual graph augmentations and identify layout augmentation as the most effective approach for enhancing visual graphs. This finding offers valuable insights for the development of larger-scale datasets aimed at facilitating vision-language-based graph reasoning. Lastly, we highlight the potential application value of the proposed GVLQA dataset as a pretrained dataset.

Acknowledgements

This work was supported by NSFC key grant 62136005 and NSFC general grant 62076118, and in part by the Research Grants Council of the Hong Kong Special Administrative Region (Grants 16200021 and 16202523).

References

[1] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[3] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

[4] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[5] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[7] Qi Cao, Huawei Shen, Jinhua Gao, Bingzheng Wei, and Xueqi Cheng. Popularity prediction on social platforms with coupled graph neural networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 70–78, 2020.

[8] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, pages 3469–3489. PMLR, 2022.

[9] Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: Large language and graph assistant. *arXiv preprint arXiv:2402.08170*, 2024.

[10] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. Exploring the potential of large language models (LLMs) in learning on graphs. *arXiv preprint arXiv:2307.03393*, 2023.

[11] W Dai, J Li, D Li, AMH Tiong, J Zhao, W Wang, B Li, P Fung, and S Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in Neural Information Processing Systems*, 2023.

[12] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[13] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1959.

[14] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *Proceedings of International Conference on Learning Representations*, 2024.

[15] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 1956.

[16] Shuai Fu, Xiequn Wang, Qiushi Huang, and Yu Zhang. Nemesis: Normalizing the soft-prompt vectors of vision-language models. In *Proceedings of International Conference on Learning Representations*, 2024.

[17] Emden R Gansner and Stephen C North. An open graph visualization system and its applications to software engineering. *Software: practice and experience*, 30(11):1203–1233, 2000.

[18] Anne-Claude Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars Juhl Jensen, Sonja Bastuck, Birgit Dümpelfeld, et al. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631–636, 2006.

[19] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[20] Ronald J Gould. Advances on the hamiltonian problem–a survey. *Graphs and Combinatorics*, 2003.

[21] Jiayan Guo, Lun Du, and Hengyu Liu. GPT4Graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*, 2023.

[22] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[23] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[25] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[26] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[27] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. In *Proceedings of International Conference on Learning Representations*, 2024.

[28] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

[29] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. Knowledge-aware coupled graph neural network for social recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4115–4122, 2021.

[30] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6700–6709, 2019.

[31] Hawoong Jeong, Sean P Mason, A-L Barabási, and Zoltan N Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[32] Weisen Jiang, Han Shi, Longhui Yu, Zhengying Liu, Yu Zhang, Zhenguo Li, and James Kwok. Forward-backward reasoning in large language models for mathematical verification. In *Findings of the Association for Computational Linguistics*, 2024.

[33] Weisen Jiang, Yu Zhang, and James Kwok. Effective structured-prompting by meta-learning and representitive verbalizer. In *International Conference on Machine Learning*, 2023.

[34] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[35] Arthur B Kahn. Topological sorting of large networks. *Communications of ACM*, 1962.

[36] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, 1990.

[37] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[38] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[39] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[40] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[41] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 462–470, 2008.

[42] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

[43] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, 2023.

[44] Chang Liu and Bo Wu. Evaluating large language models on graphs: Performance insights and comparative analysis. *arXiv preprint arXiv:2308.11224*, 2023.

[45] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 2023.

[46] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. Mask and reason: Pre-training knowledge graph transformers for complex logical queries. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1120–1130, 2022.

[47] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[48] OpenAI. GPT-3.5. Technical report, 2022.

[49] OpenAI. GPT-4 Turbo. Technical report, 2023.

[50] Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*, 2024.

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[52] Robert Sedgewick. *Algorithms in C, part 5: graph algorithms*. Pearson Education, 2001.

[53] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26, 2013.

[54] Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428, 2019.

[55] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023*, 2023.

[56] Sandro Tosi. *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[57] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[58] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[59] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? In *NeurIPS*, 2023.

[60] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the web conference 2021*, pages 878–887, 2021.

[61] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *Proceedings of Neural Information Processing Systems*, 2022.

[62] Yanbin Wei, Qiushi Huang, Yu Zhang, and James Kwok. Kicgpt: Large language model with knowledge in context for knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8667–8683, 2023.

[63] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[64] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[65] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[66] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*, 2023.

[67] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *Proceedings of International Conference on Learning Representations*, 2024.

[68] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6720–6731, 2019.

[69] Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5317–5327, 2019.

[70] Duzhen Zhang, Yahan Yu, Chenxing Li, Jiahua Dong, Dan Su, Chenhui Chu, and Dong Yu. Mm-llms: Recent advances in multimodal large language models. *arXiv preprint arXiv:2401.13601*, 2024.

[71] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

[72] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34:19172–19183, 2021.

[73] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *NeurIPS (Datasets and Benchmarks Track)*, 2023.

[74] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of International Conference on Learning Representations*, 2023.

[75] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

A Visual Modality Enhances Effectiveness by Uncovering Critical Substructures

In this section, we present a case study to highlight the complementary role of the visual modality in graph reasoning tasks. The visual modality excels at recognizing beneficial substructures or local patterns, which are crucial for effective graph reasoning. For instance, identifying the "hop number" is essential for shortest path calculations, recognizing "leaf nodes" is vital for topological sorting, and detecting "cycles" is necessary to avoid in Hamilton path construction. We extracted these substructures from the GVLQA-Base dataset and manually labeled them. By employing a frozen Vision Transformer (ViT) in the LLaVA framework with a trainable Multi-Layer Perceptron (MLP) decoder, we achieved identification accuracies of 89.92%, 95.16%, and 92.39% for hop number counting, leaf node identification, and cycle detection, respectively. In contrast, using a pre-trained BERT model with the same trainable MLP decoder resulted in significantly lower accuracies of 55.47%, 26.33%, and 60.32% for the same tasks. Therefore, the enhanced effectiveness of integrating visual and textual modalities can be attributed to the additional structural information provided by the visual modality, which facilitates the identification of these critical substructures.

B Advantages of GITA Over Traditional Graph Neural Networks

GITA offers several advantages over traditional Graph Neural Networks (GNNs) in terms of generalizability, flexibility, and user-friendliness:

Unlike GNNs, which require task-specific feature engineering and architecture adjustments, GITA employs a unified model architecture for all tasks, demonstrating its **generalizability**. By separating task specifications from graph structures, GITA can handle various graph reasoning tasks seamlessly. Additionally, it exhibits strong zero-shot capabilities, allowing it to perform well on tasks it has not been explicitly trained on, which is a feature not commonly found in traditional GNNs.

Besides, traditional GNNs often demand specialized knowledge in model architectures and coding to accommodate diverse tasks, posing a challenge for non-experts. In contrast, GITA overcomes this barrier by employing language-based templates for task adaptation, enhancing its **flexibility**. This flexibility enables GITA to effectively handle a broad spectrum of tasks, offering a framework that can be customized to specific requirements using daily language, without the necessity of profound expertise in graph neural networks.

Moreover, by leveraging existing VLMs, GITA can respond in natural language, allowing for intuitive graph reasoning with simple queries like "Is there a cycle in this graph?" This stands in contrast to the unreadable vector representations typically used in GNNs, significantly enhancing GITA's **user-friendliness**.

C Datasets Statistics

Table 6: Statistics of the GVLQA dataset.

Subset	Connect	Cycle	TS	SP	MaxFlow	BMG	HP	Total
BASE	16,410	4,100	2,910	1,560	1,500	1,860	900	29,240
AUGLY	98,460	24,600	17,460	9,360	9,000	11,160	5,400	175,440
AUGNS	49,230	12,300	8,730	4,680	4,500	5,580	2,700	87,720
AUGNO	65,640	16,400	11,640	6,240	6,000	7,440	3,600	116,960
AUGET	65,640	16,400	11,640	6,240	6,000	7,440	3,600	116,960
Total	295,380	73,800	52,380	28,080	27,000	33,480	16,200	526,320

D Illustrations for Visualization Tools in Graph Visualizer

The GITA graph visualizer incorporates a variety of implementations for existing visualization tools such as Graphviz, Matplotlib with NetworkX, and igraph, each selected for their unique capabilities

Table 7: Average numbers of nodes and edges for each task in GVLQA.

Average / Task	Connect	Cycle	TS	SP	MaxFlow	BGM	HP
#nodes	25.01	23.42	21.86	13.65	13.90	21.13	13.24
#edges	95.46	23.66	114.10	23.99	49.16	51.03	45.05

Table 8: Statistics of real-world datasets

	ca-GrQC	ca-HepTh	PolBlogs	Cora	CiteSeer
# Nodes	5,242	9,877	1,490	2,708	3,327
# Edges	14,496	25,998	19,025	5,278	4,676
domain	collaboration	collaboration	social	citation	citation
average degree	5.53	5.26	25.54	3.9	2.74

in graph rendering. These tools are implemented in our code as interchangeable modules, enhancing flexibility based on the requirements of different projects.

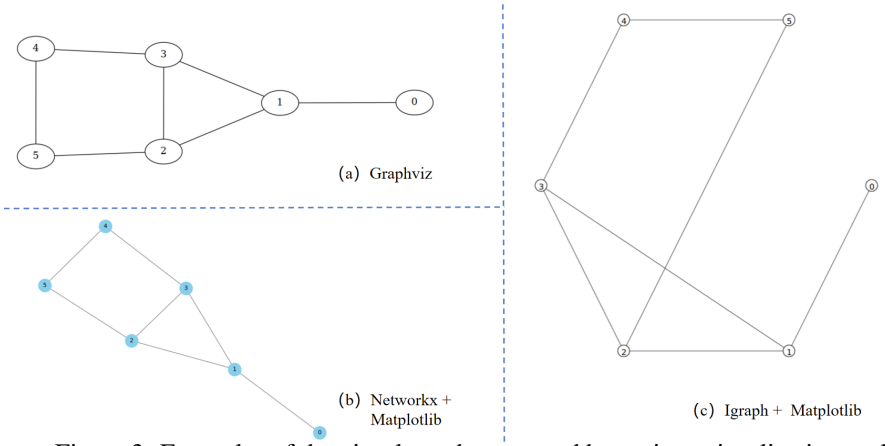


Figure 3: Examples of the visual graph generated by various visualization tools.

Figure 3 showcases some visual graphs produced by these different graph visualizer implementations.

E Graph-describing Templates

The graph describer relies on a set of unified structured templates designed to generate coherent and informative descriptions that emphasize the inherent characteristics of the graph itself, regardless of specific task requirements. These graph-describing templates cover various scenarios, including directed graphs, undirected graphs, graphs with node identities or features, and graphs with edge weights or capacities. Table 9 provides an illustration of these templates, where [P] denotes placeholders required to be filled by corresponding graph information.

F Examples of Manual-template-based and LLM-agent-bootstrapped Query Generation

Manual-template-based Query Generation. The queries Q_G^T can be generated by task-specific manual templates. These templates are manually crafted by human to supplement descriptions/instructions about 1) concrete meanings of nodes and edges, 2) task responsibilities and 3) input/output specifications into the task-agnostic graph description D_G. Therefore, the **precision** and **faith** of generated task-specific queries Q_G^T are guaranteed by human calibrations. An example of manual-template-based query generation for topological sorting is illustrated in Figure 4. In this example, placeholders [P] are used to represent information that scripts will automatically fill in.

Graph categories	Undirected	Directed
Prototype	In an undirected graph, (i,j) means that node i and node j are connected with an undirected edge. The nodes are numbered from [P] to [P], and the edges are: ([P], [P]) , ([P], [P])...	In a directed graph, (i,j) means that node i and node j are connected with a directed edge from node i to node j. The nodes are numbered from [P] to [P], and the edges are: ([P], [P]) , ([P], [P])...
W/ Node Attributes	In an undirected graph, the nodes are numbered from [P] to [P], and every node has an attribute. (i,j) means that node i and node j are connected with an undirected edge. The attributes of nodes are: node [P]: [P] node [P]: [P] ... The edges are: ([P],[P]) ([P],[P]) ...	In a directed graph, the nodes are numbered from [P] to [P], and every node has an attribute. (i,j) means that node i and node j are connected with a directed edge from node i to node j. The attributes of nodes are: node [P]: [P] node [P]: [P] ... The edges are: ([P],[P]) ([P],[P]) ...
W/ Edge Weights	In an undirected graph, the nodes are numbered from [P] to [P], and the edges are: an edge between node [P] and node [P] with weight [P], an edge between node [P] and node [P] with weight [P], ...	In a directed graph, the nodes are numbered from [P] to [P], and the edges are: an edge from node [P] to node [P] with weight [P], an edge from node [P] to node [P] with weight [P], ...
W/ Both	In an undirected graph, the nodes are numbered from [P] to [P], and every node has an attribute. The attributes of nodes are: node [P]: [P] node [P]: [P] ... And the edges are: an edge between node [P] and node [P] with weight [P], an edge between node [P] and node [P] with weight [P], ...	In a directed graph, the nodes are numbered from [P] to [P], and every node has an attribute. The attributes of nodes are: node [P]: [P] node [P]: [P] ... And the edges are: an edge from node [P] to node [P] with weight [P], an edge from node [P] to node [P] with weight [P], ...

Table 9: Graph-describing Templates for various categories.

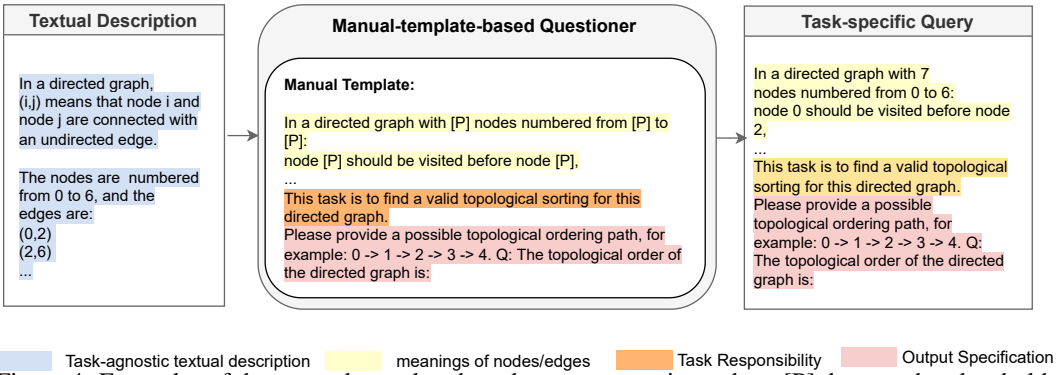


Figure 4: Examples of the manual-template-based query generation, where [P] denotes the placeholders.

LLM-agent-bootstrapped Query Generation. Figure 5 presents an example of employing a bootstrapped LLM agent, such as ChatGPT[48], for monster-hunting gaming. By incorporating task-specific information into the prompt, including node/edge meanings and task responsibilities, the LLM agent automatically generates a response that serves as the desired task-specific query. Compared to using manual templates, bootstrapping LLM agents for automated synthesis is more **flexible** and **economic** as it can take advantage of LLM agents to automatically interpret context and generate appropriate queries for various scenarios and minimize manual effort with changing conditions. Such properties make it suitable for dynamic or bespoke tasks, such as robotic planning or complex gaming scenarios.

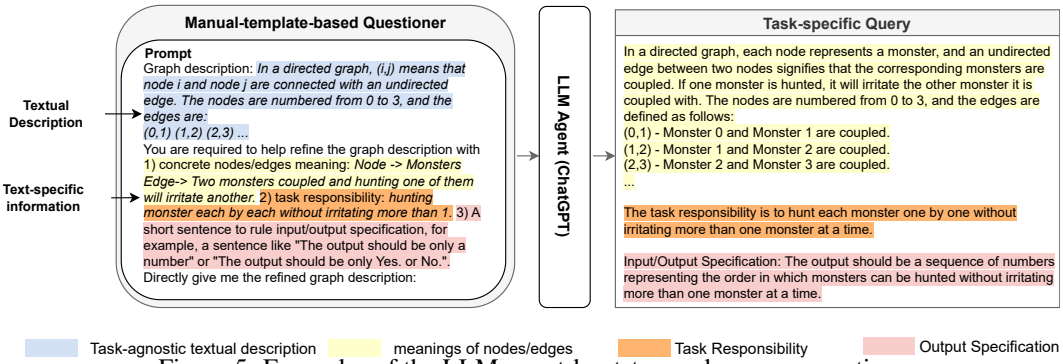


Figure 5: Examples of the LLM-agent-bootstrapped query generation.

G Experiment Settings

For all fine-tuning experiments, we use a batch size of 128 and adopt the AdamW optimizer (with a learning rate of 0.0002 and 0.00002 for the LoRA adapters within the text decoder and vision-to-text projector, respectively).

Detailed Settings for GVLQA Dataset During the evaluation, the temperature is set to 0 for all baselines. All fine-tuning experiments are conducted on an NVIDIA DGX station with 8×A100 GPUs. We split the GVLQA dataset in the ratio of 7:3 for training and testing, respectively. The accuracy (%) metrics are computed by comparing the prediction and ground truths with exact matching. We use the next-token-prediction loss to fine-tune the LoRA [28] adapters of LLMs and the vision-to-text projector. Visual graphs are encoded as visual embeddings by a visual encoder. Visual embeddings are concatenated with the embeddings of textual descriptions and instructions (i.e., questions), then fed to the text decoder to generate the answer.

Real-world Datasets Here we provide more details about the five real-world datasets used in Sec 5.3. The datasets ca-GrQC and ca-HepTh represent collaboration networks from the arXiv sections of General Relativity and Quantum Cosmology, and High Energy Physics - Theory, respectively, featuring nodes as authors and edges as co-authorships. They can be downloaded from Stanford Network Analysis Project (SNAP) website [6]. PolBlogs is a network of U.S. political blogs from February 2005, categorized by political alignment and linked by blog references. Cora and CiteSeer are both citation networks, where nodes correspond to scientific papers and edges to citations, utilized for tasks such as document classification and citation analysis, with papers categorized into various research fields. Statistics of the datasets are shown in Table 8. For each dataset, $80\%/10\%/10\%$ of the edges are randomly used for training/validation/testing, respectively.

Detailed Settings for Real-world Benchmarks In the conventional semi-supervised node classification setting, class labels are available for some nodes, which is reflected in the visual graph by coloring the nodes with a unique random color for each class. To focus on evaluating the model's ability to capture structural information, GITA filters out the influence of node features in Cora and CiteSeer datasets. For link prediction tasks on ca-GrQC and ca-HepTh datasets, GITA treats the graphs as undirected. In the test split, both the original links and their reverse links do not appear in the train and valid splits. During training and evaluation, an equal number of negative sampled links are used alongside the positive links. These negative links are sampled at each training epoch

[6] https://snap.stanford.edu/index.html

but remain fixed during evaluation. For the GVLQA pretrain checkpoint, GITA adopts the 7B cycle checkpoint finetuned on GVLQA-BASE, where the performance is nearly mature. Hyperparameter combinations for each model are determined through grid search, and the specific combinations can be found in the provided code.

H Illustrations of GVLQA subsets

In this section, we present the illustrations of the GVLQA subsets. Figure 6 provides an overview of GVLQA-BASE. Subsequently, from Figure 7 to Figure 10, we showcase the augmented visual graphs in GVLQA-AUGLY (augment layouts), GVLQA-AUGNS (augment node styles), GVLQA-AUGNO (augment node outline styles), and GVLQA-AUGET (augment edge thickness), respectively.

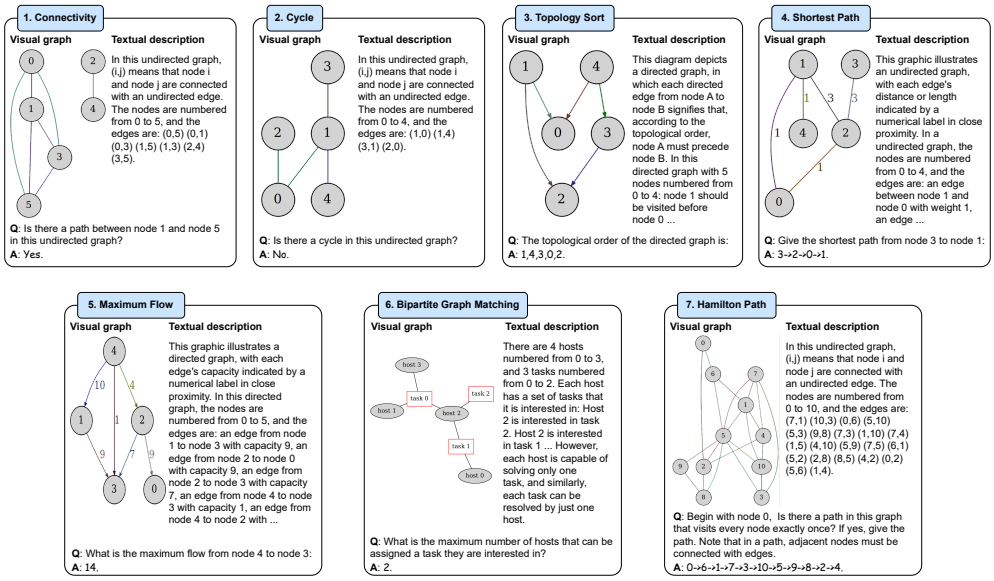


Figure 6: An overview of the GVLQA-BASE. Each figure depicts the tasks involving graph-based reasoning, showcasing a visual graph, a textual question, and the corresponding answer.

I Limitation

The GITA framework proposed in the paper, along with its experimental results, exhibits certain limitations that should be acknowledged. Firstly, when dealing with large-scale graphs, the conventional subgraph sampling strategy employed by GITA may result in imbalanced and insufficient sampling, leading to the loss of critical graph structural information. This compromise is necessary to accommodate the limited contextual length of the text-based LLM and the restricted scope of the visual graph. Secondly, due to computational constraints, the fine-tuning procedures in the paper were restricted to the LoRA framework. While this approach offers advantages, a more comprehensive fine-tuning process that considers both visual and text modalities is expected to better align the two and potentially enhance performance. Addressing these limitations should be considered as a future research direction in this field.

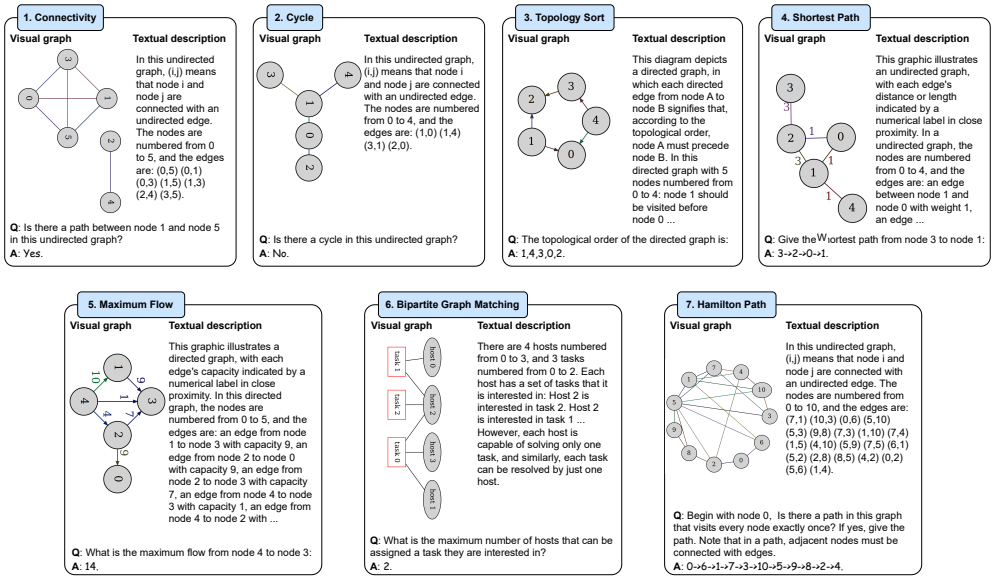


Figure 7: An overview of the GVLQA-AUGLY. Figures are akin to GVLQA-BASE but vary only in layouts.

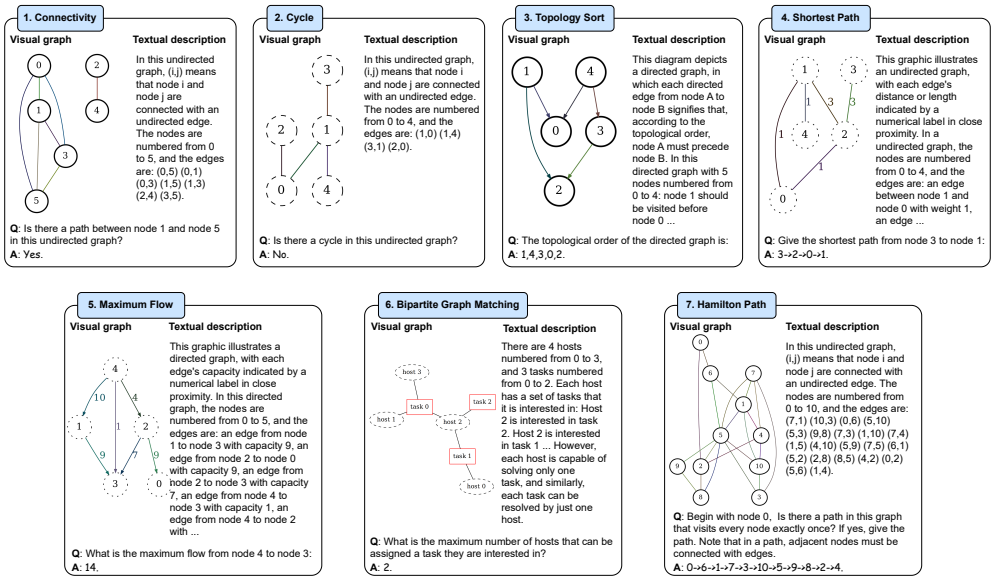


Figure 8: An overview of the GVLQA-AUGNO. Figures are akin to GVLQA-BASE but vary only in node outline styles.

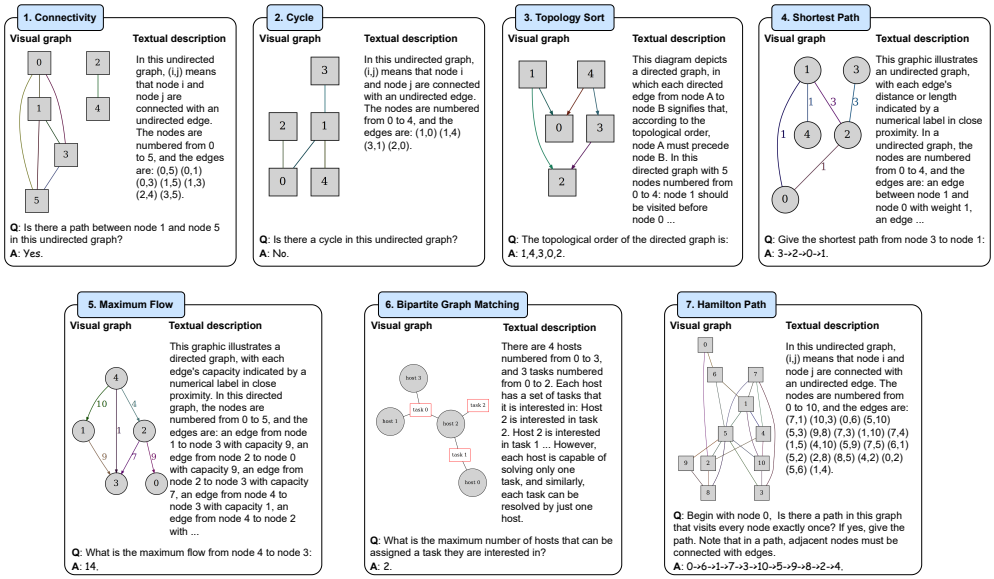


Figure 9: An overview of the GVLQA-AUGNS. Figures are akin to GVLQA-BASE but vary only in node shapes.

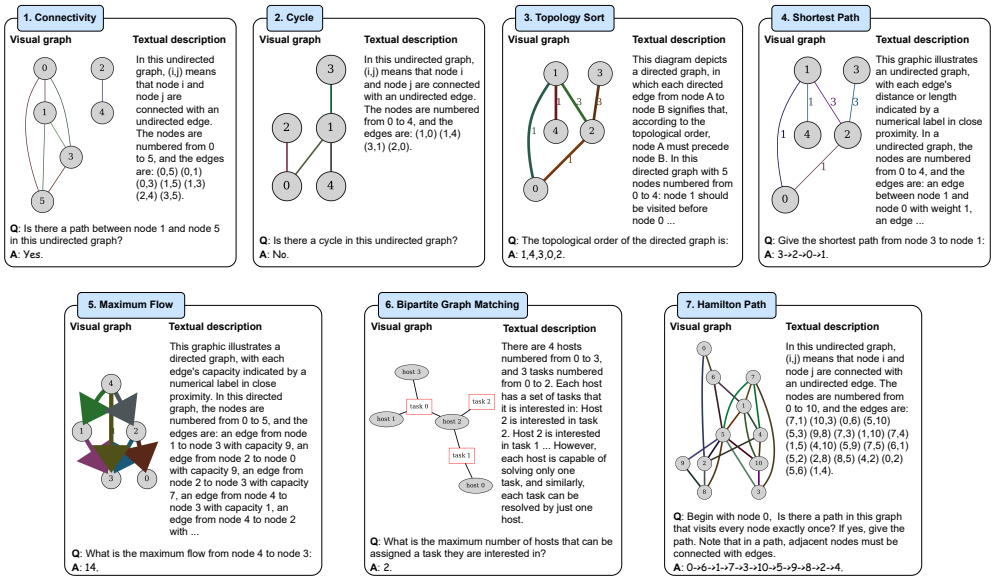


Figure 10: An overview of the GVLQA-AUGET. Figures are akin to GVLQA-BASE but vary only in edge thicknesses.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please refer to the abstract part and the contribution enumeration at the tail of the introduction part.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: Please refer to Appendix I

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not involve any theoretical result.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We show fundamental experiment settings in Section 5, and more details for experiments settings in Appendix G. Besides, we provide the complete codes as supplementary materials.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The complete codes are included, and the proposed GVLQA dataset is released with common access.

   Guidelines:

   - The answer NA means that the paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experiment details are given in both Section 5 and Appendix G.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper does not include error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the machine (type and storage) requirements in Appendix G.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We make sure the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The research does not have concerns about societal impacts because it is designed for general-purpose graph reasoning.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: The paper includes using an graph visualizer to generate abstract graph images, however, these images are focus on graph structure, without any sensitive information.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have cited necessary assets and conduct CC-BY 4.0 for our codes and datasets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The code and other supplementary materials are followed with readme and instructions.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.