GUNDAM: Aligning Large Language Models with Graph Understanding

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have achieved impressive results in processing text data, sparking interest in applying these models beyond textual data, such as graphs. There is growing interest in harnessing LLMs to understand and process graph-structured data. Existing research primarily focuses on graphs with rich textual features, such as knowledge graphs or text-attributed graphs, mainly leveraging the textual attributes while not fully exploiting the graph structure. Diverging from this line of research, this work focuses on graph reasoning tasks, aiming to evaluate and enhance the ability of LLMs to comprehend and utilize the intrinsic structural knowledge of graph data without textual attributes. We introduce the **G**raph **U**nderstanding for **N**atural Language **D**riven **A**nalytical **M**odel (GUNDAM). Specifically, we propose a simple yet effective pipeline for constructing high-quality graph reasoning dataset. By leveraging this dataset, we align LLMs to better understand the structure of graph data, enabling them to perform complex reasoning tasks by explicitly utilizing the graph's inherent structure. Our experimental evaluations on graph reasoning benchmark demonstrate that GUNDAM surpasses SOTA baselines. Furthermore, our findings identify critical factors influencing the reasoning capabilities of LLMs. We also present a theoretical analysis that illustrates how reasoning paths can improve LLMs' reasoning capabilities. [1]

1 Introduction

LLMs have achieved remarkable success in processing serialized natural language data (Touvron et al., 2023; Achiam et al., 2023; Anthropic, 2024). Recent explorations have pushed the boundaries of LLM applications beyond textual data, particularly into the realm of computer vision (Wang et al., 2024; Liu et al., 2024). Parallel to these developments, the utilization of LLMs in graph learning has emerged as a focal point of interest (Wang et al., 2023; He et al., 2024; Liu et al., 2023; Fatemi et al., 2024). Diverging from researches (Huang et al., 2024; Chen et al., 2024b) that focus on text-rich graphs (Liu et al., 2023; He et al., 2024) and primarily leverage the textual attributes of graph data, this work concentrates on pure graphs without textual attributes. We aim to explore a more fundamental and universal question: **Can LLMs truly understand graph data, particularly the structure of graphs, and rely on this understanding to perform complex reasoning tasks on graphs?**

Investigating whether LLMs can understand graph data is crucial, as graphs are fundamental data structures that represent entities and their complex interrelations, effectively modeling various real-world scenarios (Wu et al., 2020; Li et al., 2021). Enhancing LLMs' ability to understand graph data has the potential to significantly advance their general intelligence (Bubeck et al., 2023). This rationale has been explicitly or implicitly employed in various applications, such as recommendation systems (Fan et al., 2019) and knowledge graphs (Yuan et al., 2022), to facilitate the reasoning process. Recent advances (Wang et al., 2023; Chen et al., 2024a; Luo et al., 2024) have emphasized the exploration of LLMs' abilities in understanding and reasoning over graph structures. Specifically, benchmarks such as NLGraph (Wang et al., 2023) and GraphQA (Fatemi et al., 2024) have been developed to evaluate the performance of LLMs on various graph reasoning tasks. Preliminary attempts with closed-source LLMs on these benchmarks indicate that while these models are capable of addressing simple reasoning tasks, they still struggle with more complex graph reasoning challenges, as observed by Wang et al. (2023) and Fatemi et al. (2024).

Parallelly, the reasoning capabilities of open-

[1] The code and dataset for this paper are available at https://anonymous.4open.science/r/GUNDAM/.

source LLMs on graphs have garnered significant attention, with research primarily bifurcating into two categories. The first category focuses on leveraging prompt techniques to enhance the inherent reasoning abilities of LLMs for graph reasoning tasks. For instance, Wang et al. (2023) and Fatemi et al. (2024) explored prompt strategies such as Chain of Thought (CoT) and Self-consistency (SC) to augment LLMs' reasoning efficacy. Additionally, Perozzi et al. (2024a) introduced soft prompt techniques that map graph structures onto an LLM's embedding space for reasoning. These prompt-based approaches, though cost-effective, generally yield limited improvements in reasoning capabilities (Wang et al., 2023). The second research avenue involves fine-tuning LLMs with synthetically generated graph reasoning data, which tends to significantly enhance their reasoning performance. However, this method has its drawbacks. First, the quality of synthetic data is unpredictable; manually constructing graph reasoning data requires considerable expertise in graph theory, making it a costly endeavor. Observations by Wang et al. (2023) and Fatemi et al. (2024) suggest that even powerful closed-source LLMs, such as GPT-4, are inferior at graph reasoning and are therefore not capable of generating reliable graph reasoning data. Second, the specifics of what constitutes effective graph reasoning data and how to best utilize this data to maximize an LLM's capabilities in graph reasoning remain under exploration.

To this end, we introduce the **G**raph **U**nderstanding for **N**atural **L**anguage **D**riven **A**nalytical **M**odel (GUNDAM), which is designed to effectively understand graph structures and perform complex reasoning tasks over graphs. GUNDAM integrates two pivotal pipelines: high-quality graph reasoning data construction pipeline and alignment tuning pipeline. The former is designed to generate accurate answers and COT reasoning paths, providing high-quality data support for the latter. The latter initially employs a graph projection technique to encode the graph structure into the LLM, followed by fine-tuning the model using the graph reasoning datasets. The integration of these strategies enables GUNDAM to harness and expand upon the potential of LLMs in understanding and reasoning about complex graph structures. The contributions of this paper are summarized as follows:

- **Data Construction**: We have developed a pipeline for constructing high-quality graph reasoning data that significantly enhances the reasoning capabilities of training data and improves model interpretability.

- **Methodology**: We introduce GUNDAM, specifically designed to enable LLMs to comprehend graph-structured data and perform complex reasoning on graphs.

- **Understanding LLM Capabilities in Graph Reasoning**: Our experiments validate that GUNDAM achieves SOTA performance while also identifying factors that influence LLMs' graph reasoning capabilities. Additionally, we provide a theoretical analysis of how reasoning paths enhance LLMs' reasoning capabilities.

2 Related Work

2.1 LLM for Graph

In graph learning, integrating Large Language Models (LLMs) with Text-Attributed Graphs (TAGs) has led to substantial progress, particularly where text descriptions appear within nodes or the entire graph (Liu et al., 2023; Ye et al., 2023; Chen et al., 2024b; He et al., 2024; Chen et al., 2023). Recent studies focus on tasks like node classification (He et al., 2024; Liu et al., 2023; Chen et al., 2023, 2024b), link prediction (Bi et al., 2024), and graph classification (Zhao et al., 2023; Qian et al., 2023). A pivotal area is node classification in citation networks, where TAPE (He et al., 2024) uses LLMs to process paper titles and abstracts, therefore enhancing Graph Neural Network (GNN) performance. Chen et al. (2023) tackled sparse labeled data by using LLMs to generate annotated data, reducing the annotation burden. To handle graph data directly, LLaGA (Chen et al., 2024b) employs a soft prompt technique, training a Projector to map node sequences into token embeddings, adapting the LLM for graph data. For link prediction, LPNL (Bi et al., 2024) develops a scalable method using LLMs and a two-stage sampling process for accurate predictions. Qian et al. (2023) extend LLM use to molecular property prediction, broadening LLM applicability. OFA (Liu et al., 2023) offers a versatile approach that applicable to diverse graph tasks. It makes use of subgraphs centered around nodes of interest (NOIs) and creates prompt graphs, which are then integrated into the

original graph, thereby enriching the input for the GNN in an informed manner.

2.2 Reasoning on Graph

Graph reasoning, an evolving field in graph representation learning and natural language processing, focuses on using graph structures for cognitive tasks like shortest path finding and topological sorting. Notable efforts (Perozzi et al., 2024b; Chen et al., 2024a; Fatemi et al., 2024; Luo et al., 2024; Chai et al., 2023) have aimed to assess and enhance models' reasoning capabilities over graph data. Benchmarks like NLGraph (Wang et al., 2023) and GraphQA (Fatemi et al., 2024) have been crucial, which utilize closed-source models like GPT to explore the potential and limitations of LLMs in handling graph reasoning tasks. Early results from these benchmarks show that LLMs often struggle with basic graph reasoning tasks, highlighting opportunities for improving model capabilities and training methods. VisionGraph (Li et al., 2024) introduced a multimodal graph reasoning benchmark that represents graphs as images, shifting the challenge to visual Question Answering (QA) tasks and testing models' adaptability to different data modalities. Luo et al. (2024) presented GraphInstruct, a benchmark for instruction fine-tuning on graph reasoning, inspiring further studies like GraphWiz (Chen et al., 2024a), the first open-source LLM for solving graph problems through explicit reasoning. Perozzi et al. (2024a) proposed using a soft prompt technique to map graph structures into LLM embeddings via GNNs, enabling LLMs to directly process graph data. Additionally, Sun et al. (2023) and Luo et al. (2023) applied LLMs to TAGs for reasoning tasks by identifying genuine relational paths within knowledge graphs, improving the interpretability and credibility of reasoning outputs.

3 Methodology

3.1 Preliminaries

Graph A graph is denoted as $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the set of nodes and edges respectively. $(u, v, w) \in \mathcal{E}$ denotes an edge, where $u, v \in \mathcal{V}$ are the two nodes of the edge and w denotes the weight of the edge.

Graph Reasoning Graph reasoning is crucial in graph learning. It involves key operations like information propagation, feature extraction, and pattern recognition, essential for high-level reasoning and decision-making in applications such as social network analysis (Hung et al., 2020) and knowledge graph (Sun et al., 2023; Luo et al., 2023). In this paper, we focus on graph reasoning that explicitly leverages the structural aspects of graphs without external textual data. We address tasks like identifying connectivity, performing topological sorting and finding shortest paths. Details of these specific graph reasoning tasks is presented in Appendix C.2.

3.2 The Proposed Method

In this section, we introduce the GUNDAM, illustrated in Figure 1, which outlines our strategy to empower an LLM to effectively process graph data inputs and undertake complex reasoning based on graph structures. To realize this goal, we need to address three pivotal challenges: 1) How can we encode graph structures for LLM input? 2) How do we construct correct and diverse alignment data? 3) How can we enhance the graph reasoning capabilities of an LLM? We will detail our solutions to these three questions. Furthermore, we theoretically investigate the benefits brought by reasoning path.

Table 1: Performance and tokens consumption for encoding graphs of different graph2text methods.

	Connectivity		Cycle		Shortest Path	
	Acc↑	Tokens↓	Acc↑	Tokens↓	Acc↑	Tokens↓
A	91.11	223.17	93.72	68.93	37.50	129.75
E	83.56	148.23	69.63	53.09	42.19	120.03
I	99.19	345.33	92.67	184.58	31.25	153.72

A, E, and I denote Adjacency, Expert, and Incident methods, respectively.

Graph Projection LLMs are not inherently equipped to process graph data directly. To facilitate LLMs in comprehending graph data and performing reasoning tasks, it is essential to first transform the graph data into a format that can be understood by LLMs. Extensive research (Perozzi et al., 2024a; Fatemi et al., 2024) has focused on developing encoding strategies to render graph data compatible with LLMs. Fatemi et al. (2024) presents several graph2text methods, such as Adjacency, Expert and Incident. However, these methods are designed for unweighted graphs. In this paper, we extend the Adjacency method to describe weighted graphs, converting graph data into a format suitable for processing by LLMs. Specifically, we represent the graph structure using triples of the form (u, v, w), as detailed in Section 3.1. For ex-

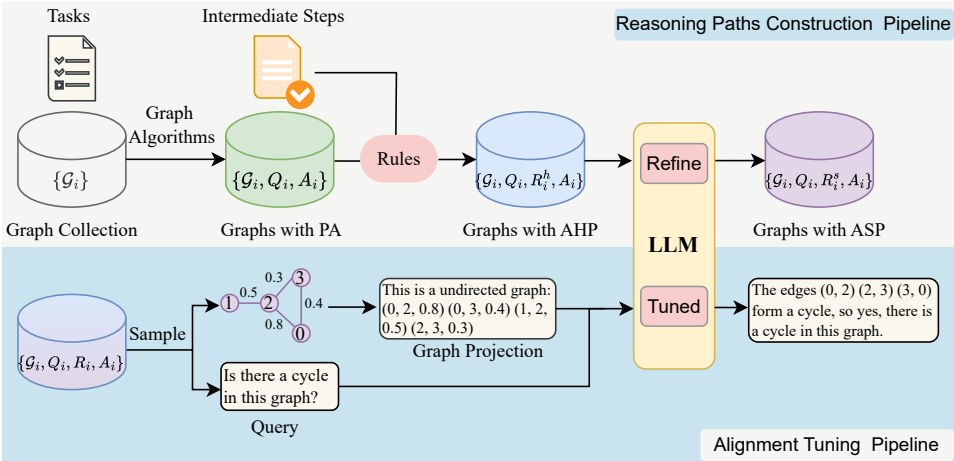


Figure 1: The overall framework of GUNDAM. This framework consists of two pipelines: 1) The upper part is the Reasoning Paths Construction Pipeline, which is designed to generate high-quality graph reasoning data. 2) The lower part is the Alignment Tuning Pipeline, which aims to align the LLM's understanding and reasoning capabilities on graph structures.

ample, the graph from Figure 1 can be represented as a sequence of such triples, as shown in Example: Graph Projection.

Example: Graph Projection

This is a undirected graph, where (u, v, w) denotes that node u and node v are connected by an undirected edge with the weight w.
(0, 2, 0.8) (0, 3, 0.4) (1, 2, 0.5) (2, 3, 0.3)

We also experimented with several other graph2text methods. Considering the tradeoff between performance and efficiency, we ultimately selected the Adjacency method, as shown in Table 1. Further details can be found in Appendix E.1.

Graph Reasoning Paths Recent work (Merrill and Sabharwal, 2023a) has shown that transformer decoders, without any intermediate steps, can only solve problems that lie within a fairly small circuit complexity class (Merrill and Sabharwal, 2023b). Therefore, constructing answers with intermediary steps is crucial. Existing studies (Hsieh et al., 2023; Chen et al., 2024a) demonstrate that integrating CoT reasoning pathways can significantly enhance the reasoning capabilities of LLMs. However, constructing accurate and diverse reasoning data presents a notable challenge in graph reasoning tasks. Conventional methods, such as human or AI-powered annotation, exhibit distinct drawbacks. Manual annotation necessitates fundamental expertise in graph theory, leading to prohibitively high costs. Conversely, directly adopting AI tools for annotation fails to ensure accuracy. We analyze this phenomenon in our experiments in Section 4, where even a powerful model like GPT-4 exhibits limited graph reasoning capability. To address these issues, we have devised a straightforward yet effective method to construct graph reasoning data, illustrated in Figure 1.

To assure the accuracy of the answers, we utilize established graph algorithms to solve the graph reasoning problems, meticulously recording both the solution processes and the answers. For example, we use the breadth-first search (BFS) algorithm for the connectivity task, the depth-first search (DFS) algorithm for the loop detection task, and the Dijkstra algorithm for the shortest path task. We refer to these directly obtained correct answers as plain answer (PA).

Example: PA

[**Question**] Is there a path between node 0 and node 1?
[**Plain Answer**] The answer is yes.

With the solution processes obtained by the graph algorithm, we can construct manually designed reasoning paths using rules, which we denote as Answer with Hard Path (AHP).

However, the AHPs tends to result in uniform answer formats, potentially causing the model to overfit to this specific format (Zhang et al., 2024).

This lack of diversity of output expressions in training can lead to a decrease in the model's generalization performance. To address this issue, we employ LLMs to rewrite the AHP-generated answers, thereby enhancing their diversity. We refer to the answers obtained through this method as Answers with Soft Path (ASP). It is noteworthy that when provided with AHP, the task facing the LLM shifts from graph reasoning to sentence rewriting, which is easier versus the reasoning task.

Example: AHP

[**Question**] Is there a path between node 0 and node 1?
[**Answer with Hard Path**] Node 0 is connected to node 2, node 2 is connected to node 1, we can follow the path: 0->2->1. Yes, there a path between node 0 and node 1.

Example: ASP

[**Question**] Is there a path between node 0 and node 1?
[**Answer with Soft Path**] To find out if there is a path between node 0 and node 1, examine the connections given in the graph. Start from node 0, which directly connects to nodes 2 and 3. From node 2, the neighbors are 0, 3, and 1. Following this path, a connection exists from node 0 to node 2, and from node 2 to node 1. Thus, the path from node 0 to node 1 is through node 2, confirming that a path exists between node 0 and node 1.

Alignment Tuning We fine-tune LLMs using datasets that include graph reasoning paths to align them better with graph understanding and enable explicitly reasoning based on graph structures. Our training data set, $\mathcal{D} = \{(\mathcal{G}_i, T_i, Q_i, R_i, A_i)\}_{i=1}^{N}$, consists of N quintuples where each element represents a graph $\mathcal{G}_i$, a task T_i, a query Q_i, a reasoning path R_i and an answer A_i respectively.

Initially, we employ the Graph Projection method to transform each graph $\mathcal{G}_i$ into a textual sequence, denoted as $S_{\mathcal{G}_i} = f_P(\mathcal{G}_i)$, where f_P is the Graph Projection function. The reasoning path R_i and answer A_i are obtained through above reasoning path construction method, expressed as $(R_i, A_i) = f_R(\mathcal{G}_i, T_i, Q_i)$, where f_R is the reasoning path construction function. The training objective is to maximize the probability of generating the correct answers based on this structured input:

$$\max_{G_\theta} p(A_i|S_{\mathcal{G}_i}, T_i, Q_i, R_i), \tag{1}$$

where G_θ is the parameter of GUNDAM. To enable GUNDAM to generate intermediate reasoning processes that aid in predicting the final answer, we formulate this as $(\hat{R}_i, \hat{A}_i) = f_G(S_{\mathcal{G}_i}, T_i, Q_i)$, where f_G is the inference function of GUNDAM. The training objective function is defined as

$$\mathcal{L} = \mathcal{L}_A + \lambda \mathcal{L}_R, \tag{2}$$

where $\mathcal{L}_\text{A} = \frac{1}{N}\sum_{i=1}^{N} \ell\left(\hat{A}_i, A_i\right)$ represents the answer prediction loss, and $\mathcal{L}_\text{R} = \frac{1}{N}\sum_{i=1}^{N} \ell\left(\hat{R}_i, R_i\right)$ is the reasoning path generation loss. ℓ is the cross-entropy loss between the predicted and target tokens and λ is a hyperparameter.

3.3 Theoretical Analysis

In this section, we provide a theoretical analysis of how CoT reasoning paths can enhance the reasoning capabilities of LLMs, thereby facilitating the generation of correct answers with greater ease.

Theorem 1. *Given the following conditions:*

1. *Non-triviality: The reasoning path R provides non-trivial information about the responses Z, such that $H(R|Z) > 0$.*
2. *Relevance: The reasoning path R contains information relevant to the correct answer a that is not fully captured by the response Z, such that $I(a; R|Z) > 0$.*

Then it follows that $H(a|Z, R) < H(a|Z)$.

In Theorem 1, $H(\cdot|\cdot)$ represent the conditional entropy and $I(\cdot;\cdot)$ denotes the mutual Information. The conditions coincides with observations from previous studies (Merrill and Sabharwal, 2023a; Chiang et al., 2023a) that point to LLMs' shortcomings in handling sequential reasoning challenges, such as simulating finite state machines, determining connectivity in graphs, or solving matrix equations. The reasoning typically requires a series of logical steps and transformations that a simple direct model output Z might not fully capture. Therefore, the reasoning path R contains additional information that diminishes the uncertainty regarding the correct answer a. We provide a proof of Theorem 1 in Appendix A.

4 Experiment

In this section, we conduct experiments to address two critical research questions ($\mathcal{RQ}$):

- $\mathcal{RQ}1$: How does GUNDAM perform in graph reasoning tasks compared to open-source and closed-source LLMs and current SOTA GraphLLMs?
- $\mathcal{RQ}2$: What factors significantly impact the graph reasoning capabilities of LLMs?

4.1 Dataset and Experimental Settings

Dataset We conducted experimental validation on the NLGraph benchmark (Wang et al., 2023), a comprehensive graph reasoning benchmark designed to evaluate performance across a spectrum of graph reasoning tasks. This benchmark encompasses eight distinct levels and complexities of tasks, namely: Connectivity, Cycle, Topological Sort, Shortest Path, Maximum Flow, Bipartite Graph Matching, Hamilton Path, and Graph Neural Networks. Detailed statistical information about the dataset and elaborate descriptions of the eight graph reasoning tasks are provided in Appendix C.

Baselines In our experimentation, we selected three categories of methods as baselines for comparison: 1) open-source and closed-source LLMs, including GPT-3.5 (version gpt-3.5-turbo-1106), GPT-4 (Achiam et al., 2023) (version gpt-4-0125-preview), Vincuna (Chiang et al., 2023b) (version vicuna-7b-v1.5), and Llama3 (Dubey et al., 2024) (version Llama 3.1 8B Instruct); 2) Graph LLMs for classification, including GraphGPT (Tang et al., 2023) and HiGPT (Tang et al., 2024); and 3) Graph LLMs for reasoning, including NLGraph (Wang et al., 2023), Ge et al. (2024), GraphWiz, and GraphWiz-DPO (Chen et al., 2024a).

Settings We follow the settings of NLGraph (Wang et al., 2023) for the dataset split, prompt, and the evaluation of the experimental results. Specifically, for all the eight tasks, we use accuracy as the evaluation metric. All experiments are conducted on an 8∗A800 machine. The learning rate is set to 2e-5 and the hyperparameter λ is set to 1. We utilized Vicuna and Llama3 as the base models for our models, respectively denoted as GUNDAM-V and GUNDAM-L in our study. More detailed baseline and experiment settings are available in Appendix D.

4.2 Main Results ($\mathcal{RQ}1$)

We conducted experimental validations on eight graph reasoning tasks in a zero-shot setting. We report the average accuracy for each task, as well as the overall average accuracy across all tasks, as shown in Table 2. We summarize the following observations: **Firstly**, GUNDAM-V and GUNDAM-L demonstrated commendable performance across all eight tasks, surpassing the advanced generative model GPT-4 with parameter sizes of 7B and 8B, respectively. They exceeded their corresponding base models (Vicuna and Llama3) by 38.36% and 49.53%, validating the effectiveness of our proposed reasoning path construction and Alignment Tuning in enhancing LLMs' capabilities in graph reasoning. **Secondly**, The two open-source models, Vicuna and Llama3, performed poorly across all tasks. The two closed-source models, GPT-3.5 and GPT-4, demonstrated basic graph reasoning capabilities, significantly outperforming the open-source models on simpler tasks such as connectivity. However, GPT-3.5 struggled with more complex graph reasoning tasks like Maximum Flow and GNN. Although GPT-4 showed improved results compared to GPT-3.5, its performance was still suboptimal. **Thirdly**, The performance of the models GraphGPT and HiGPT was similar to that of the open-source models, suggesting that these models, designed primarily for node classification tasks, may rely heavily on the textual attributes of nodes, making them less effective for graph reasoning tasks that require an understanding of graph structure. **Fourthly**, Compared to the baselines of Graph LLMs for classification, the four baselines of Graph LLMs for reasoning showed significantly better results. This improvement can be attributed to their design, which takes graph structure into account, demonstrating a certain level of graph structure-based reasoning capability.

Notably, GUNDAM-L achieved exceptionally high accuracy in the GNN task, suggesting that LLMs can effectively simulate two-layer graph convolution operations on relatively small-scale graphs, which involve complex multi-step reasoning. We provide more detailed experimental results at each difficulty level in the Appendix E.2. Additionally, we provide more detailed case studies in Appendix E.4, demonstrating that GUNDAM not only generates correct answers but also produces reasoning paths that elucidate how these answers are derived. This enhances the interpretability of

Table 2: Model performance under ZERO-SHOT.

Model	Size	Connectivity	Cycle	Shortest Path	Topo. Sort	Max. Flow	Matching	Hamilton	GNN	Overall Avg.
GPT-3.5	/	65.50	59.69	20.31	28.89	3.45	16.67	6.90	0.00	25.18
GPT-4	/	90.84	59.16	54.69	31.11	10.34	41.67	10.34	33.33	41.44
Vicuna	7B	58.22	52.88	3.12	0.00	0.00	0.00	0.00	0.00	14.28
Llama3	8B	57.14	53.93	15.62	0.00	6.90	0.00	0.00	0.00	16.70
GraphGPT	7B	54.99	46.07	3.12	0.00	6.90	2.38	0.00	0.00	14.18
HiGPT	7B	57.95	56.02	4.69	0.74	8.62	1.19	0.00	0.00	16.15
NLGraph	/	86.82	70.33	54.15	25.92	8.62	39.20	32.76	31.00	43.60
Ge et al. (2024)	/	74.39	58.12	59.38	-	-	-	-	-	-
GraphWiz	7B	63.07	52.88	35.94	0.00	**25.86**	0.00	6.90	0.00	23.08
GraphWiz-DPO	7B	68.46	59.69	59.38	1.48	20.69	0.00	0.00	0.00	26.21
GUNDAM-V	7B	94.88	94.76	42.19	45.19	13.79	48.81	36.21	43.59	52.43
GUNDAM-L	8B	**99.46**	**96.86**	**60.94**	**63.70**	17.24	**57.14**	**39.66**	**94.87**	**66.23**

[1] The best results are indicated in bold, while the second-best results are underlined.
[2] Methods using closed-source models with unspecified parameters are denoted by "/".
[3] For results where the original papers did not report specific outcomes and no public resources are available to reproduce these results, a "-" is used.

the model's outputs.

4.3 Ablation Study ($\mathcal{RQ}2$)

In this section, we investigate the impact of reasoning paths, prompt techniques, and training data difficulty on the reasoning abilities. Unless specified otherwise, all subsequent experiments use Vicuna as the base model, i.e., GUNDAM-V.

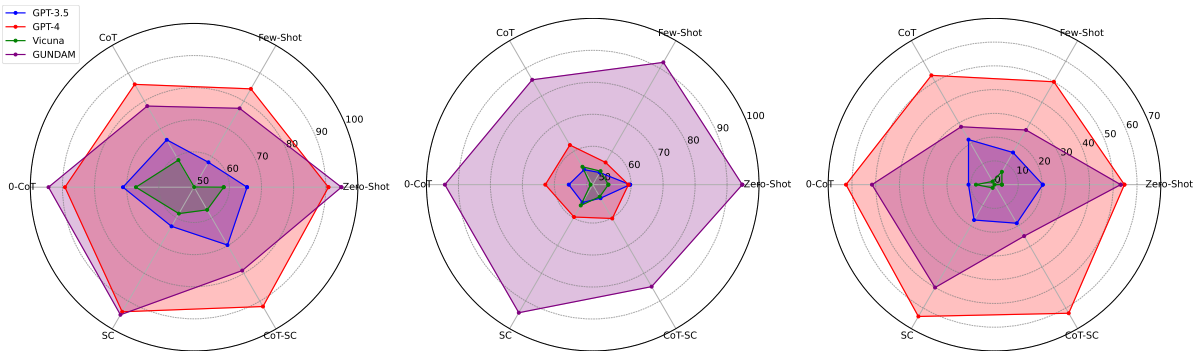
Table 3: Impact of reasoning path.

	Connectivity	Cycle	Shortest Path
Random	50.00	50.00	5.54
PA	55.53	55.50	25.00
AGP	87.33	56.02	12.50
AHP	91.11	93.72	37.50
ASP	91.11	96.61	39.06

Reasoning Path In this subsection, we further investigate how reasoning paths influence the graph reasoning performance of LLMs. We conducted experiments on the Connectivity, Cycle, and Shortest Path tasks, with the average accuracy presented in Table 3. More detailed results are provided in Appendix E.3. We use AGP to denote Answers with Generated Path, where reasoning paths are generated by GPT-4. Additionally, following NLGraph (Wang et al., 2023), we include a Random baseline, which randomly assigns "Yes" or "No" to the Connectivity and Cycle tasks, with an expected accuracy of 50%. For the Shortest Path task, we randomly pick a valid path and the sum of the weights along the path as the answer.

The results from models trained on PA, AGP, AHP, and ASP surpassed random outcomes, indicating that aligning LLMs with data containing correct answers generally enhances their reasoning capabilities to various extents. PA showed moderate performance across the three tasks, suggesting that data with only answers and no intermediate processes provide limited improvement in LLMs' reasoning abilities. AGP performed well in the Connectivity task but was less effective in the Cycle and Shortest Path tasks. This is likely because AGP's reasoning paths, generated by GPT-4, are reliable in simpler tasks such as Connectivity, where GPT-4 can generate correct paths. However, GPT-4 struggles with more complex reasoning tasks, leading to incorrect reasoning paths and, consequently, incorrect answers in the Cycle and Shortest Path tasks. Both AHP and ASP consistently demonstrated superior performance across all tasks, reaffirming that integrating reasoning paths can indeed strengthen LLMs' reasoning abilities, provided these paths are accurate. This also confirms that our methodology for constructing reasoning paths is effective. Furthermore, while AHP and ASP achieved comparable performance, models aligned with ASP data could generate more diverse responses, showcasing the added benefit of ASP in enhancing response variability.

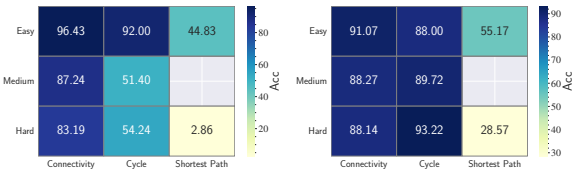
Prompt We further explore the impact of prompt techniques on the graph reasoning performance of LLM. We conducted experiments using five distinct prompt techniques, with results depicted in Figure 2. Typically, zero-shot configurations yielded relatively good or even the best results. However, advanced prompting techniques such as CoT and



(a) Connectivity. (b) Cycle. (c) Shortest Path.

Figure 2: Comparison of the performance for different prompt techniques.

SC resulted in negligible performance improvements. These findings suggest that merely employing prompt techniques struggles to significantly enhance LLMs' understanding and reasoning capabilities on graphs. In contrast, our proposed Alignment Tuning offers a viable solution to effectively bolster the graph reasoning abilities of LLMs. This insight highlights the limitations of conventional prompting strategies and underscores the importance of tailor-made adaptations such as Alignment Tuning to fully leverage LLMs in complex reasoning tasks involving graph-structured data.



(a) Easy to Hard. (b) Hard to Easy.

Figure 3: Transfer Experiment.

Data Difficulty To investigate the impact of data difficulty on model performance, we conducted transfer experiments using datasets of varying difficulties. Specifically, we implemented two scenarios: 1) easy to hard, where we trained models with easy-level data and tested them on datasets of increasing difficulty to evaluate the transferability of reasoning capabilities; and 2) hard to easy, where we trained models on hard-level data and tested them on on varying difficulties. The results are presented in Figure 3.

For simpler reasoning tasks such as Connectivity, models trained on easy data demonstrated effective transferability to other difficulty levels. However, for more complex reasoning tasks, the effectiveness of transfer from easy data was limited. Conversely, models trained on hard data showed significant transferability across all tested difficulty levels, suggesting that using hard data in training substantially enhances the LLM's graph reasoning capabilities. This effectiveness may be attributed to the intricate reasoning processes invoked by hard data, which likely induces the model to generate more intermediate steps, thereby enhancing reasoning depth and robustness. This observation aligns with findings in Merrill and Sabharwal (2023a), which analyzed how the length of a transformer's chain of thought impacts its reasoning power. Moreover, while models trained solely on hard data performed better than those trained on easy data, they underperformed compared to models trained on a mixed-difficulty dataset. This indicates that, although hard data contributes significantly to enhancing reasoning capabilities, a hybrid training approach incorporating datasets of various difficulties yields the best results.

5 Conclusion

To enhance the ability of LLMs to understand graph data and perform reasoning tasks based on graph structures, we introduce GUNDAM. It employs a Graph Projection method to convert graph structures into textual formats that LLMs can process and constructs CoT reasoning data via graph algorithms. Furthermore, we propose Alignment Tuning to effectively align LLMs with graph reasoning tasks. Experiments conducted on eight graph reasoning tasks demonstrate the efficacy of GUNDAM, validating its utility in enabling sophisticated graph-based reasoning capabilities in LLMs.

6 Limitations

In this paper, we focus on enhancing the ability of LLMs to understand graph data and perform reasoning tasks based on graph structures. However, our study has several limitations. First, we conducted experiments only on small-scale graphs with fewer than 100 nodes and on benchmarks consisting of eight graph reasoning tasks. Whether our conclusions can be extended to larger-scale graphs and a broader range of graph reasoning tasks remains an open question. This also calls for the development of larger-scale graph reasoning benchmarks. Second, we used LLMs with 7B or 8B size as the base models. It is anticipated that using larger-scale base models could yield better results due to their more powerful capabilities.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *Arxiv Preprint Arxiv:2303.08774*.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*.

Baolong Bi, Shenghua Liu, Yiwei Wang, Lingrui Mei, and Xueqi Cheng. 2024. Lpnl: Scalable link prediction with large language models. *arXiv preprint arXiv:2401.13227*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *Arxiv Preprint Arxiv:2303.12712*.

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

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024a. Graphwiz: An instruction-following language model for graph problems. *Arxiv Preprint Arxiv:2402.16029*.

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

Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2023. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668*.

David Chiang, Peter Cholak, and Anand Pillay. 2023a. Tighter bounds on the expressivity of transformer encoders. In *International Conference on Machine Learning*, pages 5544–5562. PMLR.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023b. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See Https://Vicuna. Lmsys. Org (Accessed 14 April 2023)*, 2:6.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426.

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

Yuyao Ge, Shenghua Liu, Wenjie Feng, Lingrui Mei, Lizhe Chen, and Xueqi Cheng. 2024. Graph descriptive order improves reasoning with large language model. *arXiv*.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *The Twelfth International Conference on Learning Representations*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *Arxiv Preprint Arxiv:2305.02301*.

Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024. Can gnn be good adapter for llms? In *Proceedings of the ACM on Web Conference 2024*, pages 893–904.

Man Hung, Evelyn Lauren, Eric S Hon, Wendy C Birmingham, Julie Xu, Sharon Su, Shirley D Hon, Jungweon Park, Peter Dang, and Martin S Lipsky. 2020. Social network analysis of covid-19 sentiments: Application of artificial intelligence. *Journal of medical Internet research*, 22(8):e22590.

Shuangli Li, Jingbo Zhou, Tong Xu, Liang Huang, Fan Wang, Haoyi Xiong, Weili Huang, Dejing Dou, and Hui Xiong. 2021. Structure-aware interactive graph neural networks for the prediction of protein-ligand binding affinity. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 975–985.

Yunxin Li, Baotian Hu, Haoyuan Shi, Wei Wang, Longyue Wang, and Min Zhang. 2024. Visiongraph: Leveraging large multimodal models for graph theory problems in visual context. *arXiv preprint arXiv:2405.04950*.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Zihan Luo, Xiran Song, Hong Huang, Jianxun Lian, Chenhao Zhang, Jinqi Jiang, Xing Xie, and Hai Jin. 2024. Graphinstruct: Empowering large language models with graph understanding and reasoning capability. *arXiv preprint arXiv:2403.04483*.

William Merrill and Ashish Sabharwal. 2023a. The expresssive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*.

William Merrill and Ashish Sabharwal. 2023b. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545.

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

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

Chen Qian, Huayi Tang, Zhirui Yang, Hong Liang, and Yong Liu. 2023. Can large language models empower molecular property prediction? *arXiv preprint arXiv:2307.07443*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2023. Graphgpt: Graph instruction tuning for large language models. corr abs/2310.13023 (2023).

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024. Higpt: Heterogeneous graph language model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2842–2853.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *Arxiv Preprint Arxiv:2302.13971*.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Thirty-seventh Conference on Neural Information Processing Systems*.

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. 2024. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *Advances in Neural Information Processing Systems*, 36.

Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. 2020. Graph convolutional networks with markov random field reasoning for social spammer detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1054–1061.

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

Xu Yuan, Chengchuan Xu, Peng Li, and Zhikui Chen. 2022. Relational learning with hierarchical attention encoder and recoding validator for few-shot knowledge graph completion. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 786–794.

Liang Zhang, Katherine Jijo, Spurthi Setty, Eden Chung, Fatima Javid, Natan Vidra, and Tommy Clifford. 2024. Enhancing large language model performance to answer questions and extract information more accurately. *arXiv preprint arXiv:2402.01722*.

Haiteng Zhao, Shengchao Liu, Ma Chang, Hannan Xu, Jie Fu, Zhihong Deng, Lingpeng Kong, and Qi Liu. 2023. Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning. *Advances in Neural Information Processing Systems*, 36:5850–5887.

A Proof of theorem

Theorem 2. *Given the following conditions:*

1. *Non-triviality: The reasoning path R provides non-trivial information about the responses Z, such that $H(R|Z) > 0$.*

2. *Relevance: The reasoning path R contains information relevant to the correct answer a that is not fully captured by the response Z, such that $I(a; R|Z) > 0$.*

Then it follows that $H(a|Z, R) < H(a|Z)$.

Definitions and Notation

- $H(\cdot|\cdot)$ denotes the conditional entropy.
- $I(\cdot;\cdot)$ denotes the mutual Information.
- Z denotes the responses generated by the LLM.
- R denotes the reasoning path.
- a denotes the correct answer.

Intuition The main idea is that the reasoning path R provides additional information about the correct answer a beyond what is provided by the response Z. Therefore, having access to R should reduce the uncertainty about a, hence reducing the conditional entropy.

Proof. Consider the chain rule of conditional entropy:

$$H(a, R|Z) = H(a|Z) + H(R|a, Z)$$

and

$$H(a, R|Z) = H(R|Z) + H(a|R, Z).$$

Setting these two expressions equal, we obtain:

$$H(a|Z) + H(R|a, Z) = H(R|Z) + H(a|R, Z).$$

$$H(a|Z) - H(a|R, Z) = H(R|Z) - H(R|a, Z) = I(a; R|Z).$$

Given $I(a; R|Z) \geq 0$:

$$H(a|Z) - H(a|R, Z) = I(a; R|Z) \geq 0.$$

We have proven that:

$$H(a|Z) > H(a|R, Z),$$

Thus, this completes the proof that having access to the reasoning path R in addition to the responses Z reduces the uncertainty about the correct answer a, proving $H(a|Z, R) < H(a|Z)$. □

Table 4: Training parameter setting.

Training Parameter	Value
epochs	1
batch size	8
gradient accumulation steps	2
learning rate	2e-05
warmup ratio	0.03
model max length	4096

B Details of training

The training parameters of GUNDAM are set as shown in Table 4.

Constructing Answers with Soft Paths (ASP) essentially involves a sentence rewriting task, an area in which LLMs demonstrate substantial proficiency. For this purpose, researchers can utilize any commonly available open-source or closed-source LLM. To facilitate ASP construction using LLMs, we employ two optional prompts. These prompts strategically guide LLMs to reformulate the initial AHP into a more varied and generalized form, enhancing the diversity and robustness of the model outputs.

Prompt Templates for ASP

Template1.
Given an question [question] and its answer [answer], derive the shortest yet complete Chain of Thought reasoning path that leads to the answer.
[begin of question]
{question}
[end of question]
[begin of answer]
{answer}
[end of answer]

Template2.
Rewrite the following sentence to improve its logical flow:
{answer}

C Dataset

C.1 Dataset statistic.

Table 5: Dataset statistic.

Subset	Connectivity	Cycle	Topo. Sort	Shortest Path	Max. Flow	Matching	Hamilton	GNN
#Easy	296/56	125/25	151/29	151/29	124/26	249/51	125/25	77/23
Size	n: 5-10	n: 5-10	n: 5-10	n: 5-10	n: 5-10	n: 6-20	n: 5-10	n: 5-8
#Medium	1004/196	493/107	372/78	/	/	/	/	/
Size	n: 11-25	n: 11-25	n: 11-25	/	/	/	/	/
#Hard	561/119	341/59	152/28	165/35	168/32	177/33	167/33	124/16
Size	n: 26-35	n: 26-35	n: 26-35	n: 11-20	n: 11-20	n: 17-33	n: 11-20	n: 9-15
Total	1861/371	959/191	675/135	316/64	292/58	426/84	292/58	201/39

Size denotes the number of nodes in graphs. A/B denotes the number of samples in the training/test set respectively.

C.2 Reasoning Task

- **Connectivity**: Given an undirected graph $\mathcal{G}$ and two nodes u and v, determine whether there is a path to connect them, following $u \to v_i \to \cdots \to v_j \to v$.

- **Cycle**: Given an undirected graph $\mathcal{G}$, determine whether there is a cycle in the graph, following $u \to v_i \to \cdots \to v_j \to u$.

- **Shortest Path**: Given a weighted undirected graph $\mathcal{G}$ and two nodes u and v, the shortest path is the one that connects the two nodes and minimizes the sum of the weights of the edges, such that

$$P = \arg\min_{P \in \mathcal{P}uv} \sum_{(v_i, v_j) \in P} w_{ij},$$

  where $\mathcal{P}_{uv}$ denotes the set of all possible paths from u to v in the graph $\mathcal{G}$, and $(v_i, v_j) \in P$ represents each edge in the path P with weight w_{ij}.

- **Topological Sort**: A topological sort of a directed graph arranges its nodes in a sequence such that for every directed edge (u, v) from node u to node v, node u precedes node v in the sequence. The goal is to determine a valid topological order for a given directed graph, acknowledging that there may be multiple correct solutions. We task LLMs with producing a valid topological sort for the given graph and then use an external program to verify its accuracy.

- **Maximum Flow**: Consider a network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with two specific nodes s and t representing the source and sink, respectively. Each edge in the network has an associated capacity c, which indicates the maximum flow that can pass through it. The task is to instruct LLMs to devise a strategy to maximize the flow from the source to the sink. where s represents the flow value in the optimal plan, and t is the flow value in the solution generated by the LLMs.

- **Bipartite Graph Matching**: In an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, a matching refers to a collection of edges such that no two edges share a common node. A bipartite graph is a special type of graph where the set of nodes can be partitioned into two disjoint sets $\mathbf{U}$ and $\mathbf{V}$, with no edges between nodes within the same set. The goal is to find a matching in the bipartite graph that includes the maximum possible number of edges. We then use an external program to assess whether the solution generated by the LLMs is both valid and optimal.

- **Hamilton Path**: A Hamilton path in an undirected graph is a route that passes through each node exactly one time. For a given undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, the objective is to identify a valid Hamilton path.

- **Graph Neural Networks**: Given an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and a two-dimensional embedding $\mathbf{x}_i$ for each node, the goal is to execute ℓ layers of message passing. This involves updating each node's embedding by summing the embeddings of all its neighboring nodes. Formally, this update is represented as:

$$\mathbf{x}_i^{(\ell+1)} = \sum_{j \in \mathcal{N}_i} \mathbf{x}_j^{(\ell)},$$

  where $\mathcal{N}_i$ represents the set of neighbors for node i, and (ℓ) indicates the ℓ-th layer.

D Experiment Setting

For the FEW-SHOT based prompt techniques (including FEW-SHOT, CoT and CoT-SC), the input prompt contains K examples of pertinent questions and answers. For Connectivity and Cycle tasks, K is 4 and K is set to 5 for Shortest Path task. For the self-consistency (SC) prompt method, we sample 5 responses. All experiments are conducted on an 8*A800 machine. We employ Vicuna and Llama3 as the base model for GUNDAM, the learning rate is set to 2e-5, and the batchsize is set to 8. For evaluation, the temperature factor τ is set to 0.2 except for the self-consistency Prompt method, whose τ is 0.8. Hyperparameter λ is set to 1.

D.1 Baseline Setting

For baselines that provided experimental results on the same dataset and under the same settings, we directly extracted the reported results from their original papers for comparison. When multiple variants of the same method were available, we selected the best-reported results for comparison. For other baselines, we reproduced their results using the publicly available resources under the same settings. Specifically, for GraphGPT[2] (Tang et al., 2023), HiGPT[3] (Tang et al., 2024), GraphWiz[4], and GraphWiz-DPO[5] (Chen et al., 2024a), we conducted experiments using their publicly released model weights to ensure a fair comparison.

E Supplementary Experiment

E.1 Graph Encoding

Example: Expert

You are a graph analyst and you have been given a graph G among A, B, C and D. G has the following undirected edges:
A -> C with weight 0.8, A -> D with weight 0.4, B -> C with weight 0.5, C -> D with weight 0.3.

Example: Incident

G describes a graph among nodes 0, 1, 2 and 3. In this graph:
Node 0 is connected to nodes 2, 3 with weight 0.8, 0.4,
Node 1 is connected to nodes 2 with weight 0.5,
Node 2 is connected to nodes 3 with weight 0.3.

Table 6: Performance of different graph2text methods.

	Connectivity				Cycle				Shortest Path		
	Easy	Medium	Hard	Avg.	Easy	Medium	Hard	Avg.	Easy	Hard	Avg.
Adjacency	100.00	91.84	85.71	91.11	96.00	92.52	94.92	93.72	62.07	17.14	37.50
Expert	92.86	84.69	77.31	83.56	88.00	67.29	66.10	69.63	62.07	25.71	42.19
Incident	100.00	99.49	98.32	99.19	92.00	94.39	89.83	92.67	48.28	17.14	31.25

Table 7: Tokens consumption for encoding graphs of different graph2text methods.

	Connectivity				Cycle				Shortest Path		
	Easy	Medium	Hard	Avg.	Easy	Medium	Hard	Avg.	Easy	Hard	Avg.
Adjacency	51.42	399.54	740.33	223.17	45.00	107.58	193.84	68.93	162.23	248.50	129.75
Expert	66.34	292.51	491.73	148.23	61.20	99.16	149.31	53.09	169.31	229.87	120.03
Incident	122.04	638.45	1145.56	345.33	131.13	295.95	519.10	184.58	175.14	294.40	153.72

E.2 Main Results

We provide experimental results for all tasks at each difficulty level as a supplement, as shown in Table 8.

[2] https://huggingface.co/Jiabin99/GraphGPT-7B-mix-all
[3] https://huggingface.co/Jiabin99/HiGPT
[4] https://huggingface.co/GraphWiz/LLaMA2-7B
[5] https://huggingface.co/GraphWiz/LLaMA2-7B-DPO

Table 8: Model performance under ZERO-SHOT.

	Connectivity				Cycle				Shortest Path			Topology			
Model	Easy	Medium	Hard	Avg.	Easy	Medium	Hard	Avg.	Easy	Hard	Avg.	Easy	Medium	Hard	Avg.
GPT-3.5	73.21	69.39	55.46	65.50	80.00	55.14	59.32	59.69	34.48	8.57	20.31	79.31	20.51	0.00	28.89
GPT-4	98.21	93.88	82.35	90.84	92.00	55.14	52.54	59.16	75.86	37.14	54.69	82.76	23.08	0.00	31.11
Vicuna	67.86	59.69	51.26	58.22	52.00	49.53	59.32	52.88	6.90	0.00	3.12	0.00	0.00	0.00	0.00
Llama3	73.21	58.67	47.06	57.14	64.00	53.27	50.85	53.93	27.59	5.71	15.62	0.00	0.00	0.00	0.00
GraphGPT	60.71	56.63	49.58	54.99	40.00	44.86	50.85	46.07	0.00	5.71	3.12	0.00	0.00	0.00	0.00
HiGPT	69.64	58.67	51.26	57.95	48.00	55.14	61.02	56.02	10.34	0.00	4.69	3.45	0.00	0.00	0.74
NLGraph	93.18	84.50	82.79	86.82	80.00	70.00	61.00	70.33	68.89	29.00	54.15	70.60	18.40	0.00	25.92
Ge et al. (2024)	89.20	74.50	67.64	74.39	57.69	60.33	55.00	58.12	70.00	51.50	59.38	-	-	-	-
GraphWiz	78.57	64.29	53.78	63.07	64.00	49.53	54.24	52.88	58.62	17.14	35.94	0.00	0.00	0.00	0.00
GraphWiz-DPO	87.50	70.41	56.30	68.46	72.00	61.68	50.85	59.69	72.41	48.57	59.38	6.90	0.00	0.00	1.48
GUNDAM-V	100.00	94.90	92.44	94.88	92.00	94.39	96.61	94.76	68.97	20.00	42.19	96.55	42.31	0.00	45.19
GUNDAM-L	100.00	100.00	98.32	99.46	96.00	96.26	98.31	96.86	82.76	42.86	60.94	96.55	70.51	10.71	63.70

	Flow			Match			Hamilton			GNN			
Model	Easy	Hard	Avg.	Easy	Hard	Avg.	Easy	Hard	Avg.	Easy	Hard	Avg.	Overall Avg.
GPT-3.5	7.69	0.00	3.45	21.57	9.09	16.67	8.00	6.06	6.90	0.00	0.00	0.00	25.18
GPT-4	23.08	0.00	10.34	58.82	15.15	41.67	20.00	3.03	10.34	56.52	0.00	33.33	41.44
Vicuna	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	14.28
Llama3	3.85	9.38	6.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	16.70
GraphGPT	15.38	0.00	6.90	3.92	0.00	2.38	0.00	0.00	0.00	0.00	0.00	0.00	14.18
HiGPT	7.69	9.38	8.62	1.96	0.00	1.19	0.00	0.00	0.00	0.00	0.00	0.00	16.15
NLGraph	15.30	2.50	8.62	46.00	27.60	39.20	49.30	20.00	32.76	-	-	31.00	43.60
Ge et al. (2024)	-	-	-	-	-	-	-	-	-	-	-	-	-
GraphWiz	26.92	25.00	25.86	0.00	0.00	0.00	12.00	3.03	6.90	0.00	0.00	0.00	23.08
GraphWiz-DPO	11.54	28.12	20.69	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	26.21
GUNDAM-V	23.08	6.25	13.79	66.67	21.21	48.81	36.00	36.36	36.21	73.91	0.00	43.59	52.43
GUNDAM-L	19.23	15.62	17.24	62.75	48.48	57.14	56.00	27.27	39.66	100.00	87.50	94.87	66.23

E.3 Reasoning Path

We provide the results of the experiments at each difficulty as a supplement to Table 3, as shown in Table 9.

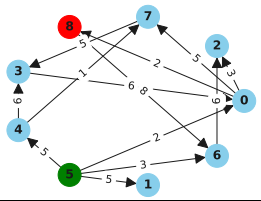
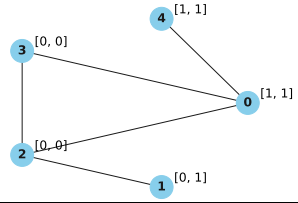
Table 9: Impact of reasoning path.

	Connectivity				Cycle				Shortest Path		
	Easy	Medium	Hard	Avg.	Easy	Medium	Hard	Avg.	Easy	Hard	Avg.
random	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	6.07	5.08	5.54
PA	53.57	51.53	63.03	55.53	56.00	54.21	57.63	55.50	41.38	11.43	25.00
AGP	92.86	87.76	84.03	87.33	56.00	59.81	49.15	56.02	20.69	5.71	12.50
AHP	100.00	91.84	85.71	91.11	96.00	92.52	94.92	93.72	62.07	17.14	37.50
ASP	96.43	90.82	89.08	91.11	96.00	93.46	96.61	94.76	51.72	28.57	39.06

E.4 Case Study

In Figure 4, we present case studies for two complex tasks: Maximum Flow and GNN. Results from the two base models are omitted for these tasks due to their inability to handle such complexity. For the Maximum Flow task, GPT-3.5 identified an algorithm capable of solving the problem but failed to detail the necessary intermediate processes, resulting in an incorrect answer. GPT-4, on the other hand, offered more detailed intermediate results but erroneously reversed the direction of the edge from node 8 to node 6, leading to incorrect intermediate steps and consequently a wrong conclusion. Conversely, GUNDAM-L not only provided correct intermediate steps but also accurately reasoned the correct answer.

Regarding the GNN task, which requires modeling two layers of graph convolution operations involving

Task	Maximum Flow	GNN
Graph		
Query	What is the maximum flow from node 5 to node 8 ?	What's the embedding of each node after two layers of simple graph convolution layer?
GPT3.5	1. Start with an initial flow of 0. 2. While there exists a path from node 5 to node 8 in the resi dual graph (the original graph in the beginning), find the ma ximum flow through this path and update the residual graph. 3. Repeat step 2 until no more paths from node 5 to node 8 e xist in the residual graph. Using the Ford-Fulkerson algorithm, we find that the maxim um flow from node 5 to node 8 is 5.	Layer 1: node 0: [1,1] + [0,0] + [0,0] = [1,1]. node 1: [0,0] = [0,0]. node 2: [1,1] + [0,1] = [1,2]. node 3: [1,1] + [1,2] = [2,3]. node 4: [1,1] = [1,1]. Layer 2: node 0: [1,1] + [2,3] + [1,2] = [4,6]. node 1: [1,2] = [1,2]. node 2: [4,6] + [0,0] = [4,6]. node 3: [1,1] + [4,6] = [5,7]. node 4: [1,1] = [2,2].
GPT4	One possible path from node 5 to node 8 would pass throug h node 6. Let's calculate the flow along this path: - Node 5 to node 6 has an edge with capacity 3. - Node 6 to node 8 has an edge with capacity 8. The bottleneck in this path is the edge from node 5 to node 6, with a capacity of 3. So, this path can contribute a maxim um flow of 3. There are no other paths available from node 5 to node 8. Thus, the maximum flow from node 5 to node 8 in this grap h is 3.	first layer: Node 0's new embedding [1,1] + [0,0] + [0,0] = [1,1] Node 1's new embedding [0,0] Node 2's new embedding [1,1] + [0,1] + [0,0] = [1,2] Node 3's new embedding [1,1] + [0,0] = [1,1] Node 4's new embedding [1,1] second layer: Node 0's new embedding [1,1] + [1,1] + [1,2] = [3,4] Node 1's new embedding [1, 2] Node 2's new embedding [3,4] + [0,0] + [1,1] = [4,5] Node 3's new embedding [3,4] + [1,2] = [4,6] Node 4's new embedding [3,4]
GUNDAM-L	From the source (node 5), we can send a maximum of 2 unit s of flow to node 8. Path 5 -> 0 -> 8: 2 units of flow 2 units of flow from node 5 to node 0 2 units of flow from node 0 to node 8 The total unit of flow arrives at node 8 is 2. So the maximu m flow from node 5 to node 8 is 2.	Layer 1: node 0: [1, 1] + [0, 0] + [0, 0] = [1, 1]. node 1: [0, 0] = [0, 0]. node 2: [1, 1] + [0, 1] + [0, 0] = [1, 2]. node 3: [1, 1] + [0, 0] = [1, 1]. node 4: [1, 1] = [1, 1]. Layer 2: node 0: [1, 1] + [1, 1] + [1, 2] = [3, 4]. node 1: [1, 2] = [1, 2]. node 2: [1, 1] + [0, 0] + [1, 1] = [2, 2]. node 3: [1, 1] + [1, 2] = [2, 3]. node 4: [1, 1] = [1, 1].

For the case of GNN task, due to its complexity requiring the generation of extensive intermediate steps, we have excerpted only the key segments of the model's output.

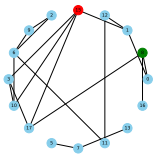
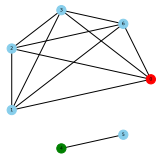
Figure 4: Case study.

complex multi-step reasoning, GPT-3.5 made errors in simulating even the first layer of convolution. It incorrectly updated the embedding of node 3 using the post-update embedding of node 2 as if they were the original embedding. GPT-4 accurately modeled the first layer but replicated the same error as GPT-3.5 in the second layer, wrongly using the updated embedding of node 0 for updating nodes 2, 3, and 4 in that layer, leading to incorrect final embedding for these nodes. GUNDAM-L, however, executed correct feature updates at each layer, resulting in the correct outcomes. From these case studies, it is apparent that GPT-4 exhibits stronger reasoning capabilities than GPT-3.5 but still encounters challenges with tasks requiring complex and multi-step reasoning, occasionally leading to erroneous reasoning outcomes. In contrast, GUNDAM-L demonstrated robust graph reasoning proficiency, effectively understanding and manipulating complex graph structures through multi-step reasoning. Furthermore, GUNDAM-L not only produces correct answers but also generates a reasoning path that elucidates how those answers were derived, enhancing the interpretability of the model's outputs.

We provide further insights with two case studies illustrated in Table 10. For the first case, considering the graph on the left. All four models correctly indicated the presence of a path, assuring the connectivity between the two nodes. However, the Vicuna-7B made an error in its reasoning process by suggesting a non-existent connection from node 17 to node 16. GPT-3.5, GPT-4, and GUNDAM provided correct answers with different valid paths. Although these paths were not the shortest, they were indeed existent and verified the connectivity as required. In the second case, both GPT-4 and GUNDAM accurately concluded the absence of any connecting path, whereas GPT-3.5 and Vicuna-7B generated incorrect responses. GPT-3.5 mistakenly introduced a non-existent edge (6,4), and Vicuna-7B incorrectly proposed

an edge (6,5). In analyzing other error cases of GPT3.5, it was also found that GPT3.5 tends to construct non-existent edges to make two nodes connected when no path exists in the graph.

Table 10: Cases from Connectivity Task.

Graph		
Query	Is there a path between node 15 and node 8?	Is there a path between node 0 and node 4?
GPT3.5	Yes, there is a path between node 15 and node 8 in the provided graph. The path is as follows: 15-1-0-8	Yes, there is a path between node 0 and node 4 in the given graph. We can see that the following path connects these two nodes: 0-3-6-1-2-3-6-4.
GPT4	Yes, there is a path between node 15 and node 8. The path can be as follows: 15-3-17-8.	No, there is no path between node 0 and node 4 in the given graph. The nodes 0 and 4 are in separate disconnected subgraphs.
Vicuna-7B	***\[1] Starting from node 15, we can follow the connections to node 17, then to node 16, and finally to node 8. Therefore, there is a path between node 15 and node 8.	***\[1] Starting from node 0, we can follow the edges as follows: (0,6) (6,3) (3,2) (2,1) (1,6) (6,5) (5,4) Therefore, the answer is Yes, there is a path between node 0 and node 4.
GUNDAM-V	Node 15 is connected to node 1, node 1 is connected to node 0, node 0 is connected to node 8, we can follow the path: 15-1-0-8. Yes, there a path between node 15 and node 8.	No, there is no path between node 0 and node 4.

[1] ***\ indicates that unimportant generated content has been omitted.