



# GraphWiz: An Instruction-Following Language Model for Graph Computational Problems

Nuo Chen\*

Data Science and Analytics Thrust, The Hong Kong  
University of Science and Technology (Guangzhou)  
Guangzhou, China  
chennuo26@gmail.com

Yuhan Li\*

Data Science and Analytics Thrust, The Hong Kong  
University of Science and Technology (Guangzhou)  
Guangzhou, China  
yuhanli98@gmail.com

Jianheng Tang\*

The Hong Kong University of Science and Technology  
Data Science and Analytics Thrust, The Hong Kong  
University of Science and Technology (Guangzhou)  
Hong Kong SAR, China  
jtangbf@connect.ust.hk

Jia Li†

Data Science and Analytics Thrust, The Hong Kong  
University of Science and Technology (Guangzhou)  
Guangzhou, China  
jiale@ust.hk

## ABSTRACT

Large language models (LLMs) have achieved impressive success across various domains, but their capability in understanding and resolving complex graph problems is less explored. To bridge this gap, we introduce GraphInstruct, a novel instruction-tuning dataset aimed at enabling language models to tackle a broad spectrum of graph problems through explicit reasoning paths. Utilizing GraphInstruct, we build GraphWiz, an open-source language model capable of solving various graph computational problems while generating clear reasoning processes. To further enhance the model's performance and reliability, we integrate the Direct Preference Optimization (DPO) framework within the graph problem-solving context. The improved model, GraphWiz-DPO, achieves an average accuracy of 65% across nine tasks with different complexity levels, surpassing GPT-4 which has an average accuracy of 43.8%. Our study also investigates the relationship between training data volume and model performance, emphasizing the risk of overfitting as data volume increases. Additionally, we explore the transferability of the proposed model across different tasks and datasets, demonstrating its robust zero-shot generalization capability. GraphWiz offers a new blueprint and valuable insights for developing LLMs specialized in graph reasoning and problem-solving.<sup>1</sup>

\* Authors contributed equally to this research.

† Corresponding author.

<sup>1</sup> Codes and data are available at <https://github.com/nuochenpku/Graph-Reasoning-LLM>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

KDD '24, August 25–29, 2024, Barcelona, Spain

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0490-1/24/08

<https://doi.org/10.1145/3637528.3672010>

## CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; • **Mathematics of computing** → **Graph algorithms**.

## KEYWORDS

graph algorithms, large language models, instruction tuning

### ACM Reference Format:

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024. GraphWiz: An Instruction-Following Language Model for Graph Computational Problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*, August 25–29, 2024, Barcelona, Spain. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3637528.3672010>

## 1 INTRODUCTION

After witnessing the remarkable capabilities of Large Language Models (LLMs) in text processing, the research community is now exploring their applicability across diverse modalities such as vision, audio, tabular, and spatio-temporal data [8, 19, 27, 40, 42, 51]. Graphs, with their non-Euclidean nature and relationship-centric characteristics, present a particularly challenging yet promising frontier for LLM exploration. The synergy between graphs and LLMs has sparked considerable interest due to their potential for bi-directional benefits: integrating graph-based approaches can advance the reasoning abilities of LLMs, enabling them to tackle complex logical tasks such as mathematical problem-solving [5, 48] and commonsense reasoning [44]. Conversely, LLMs offer powerful capabilities to augment graph analysis, particularly in the realm of semantically-rich, text-attributed networks [10, 31, 37].

While the intersection of graphs and LLMs is an emerging field, it remains uncertain *to what extent LLMs can comprehend a graph merely from the context*. Current graph machine learning tasks, such as node classification and link prediction, primarily require LLMs to focus on the semantic aspects of graphs [6, 7, 13, 14, 29, 30, 38, 45, 49]. This often entails a basic understanding of local subgraph structures. Although benchmarks like mathematical problem-solving [5, 35, 48] and knowledge-oriented question-answering [4, 45–47] necessitate multi-hop reasoning within a graph, they do not always demand an in-depth understanding of the entire graph's structure.

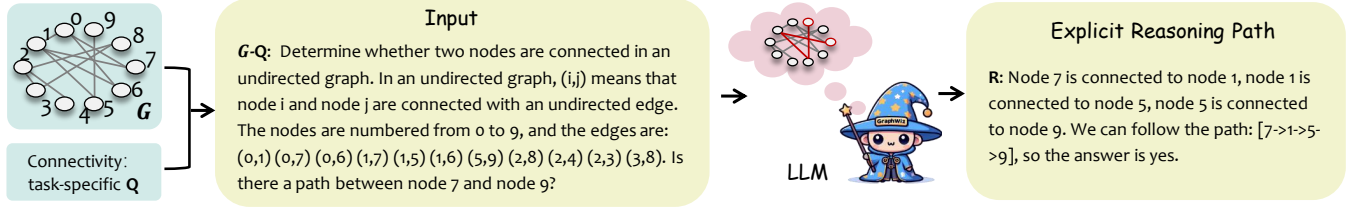


Figure 1: An example of solving the Connectivity task explicitly within natural language via LLMs.

To address this, newer benchmarks like GraphQA [10, 24] and NLGraph [37] introduce more diverse and complex *graph computational problems* to LLMs, demanding a more profound grasp of graph structures—challenges typically addressed by specific computational algorithms. Figure 1 showcases giving an undirected graph and two nodes in this graph, the LLM needs to answer whether these two nodes are connected through a path. Other similar problems include finding the shortest path and subgraph isomorphism, etc.

Current methodologies for tackling graph problems with language models largely rely on the art of prompt design, seeking to enhance the performance of LLMs through well-crafted prompts [10, 37]. Notably, this strategy has shown more effectiveness in closed-source LLMs, such as GPT-4 [22] and PaLM 2 [1]. In contrast, open-source LLMs like LLaMA [34] demonstrate considerably weaker results using the same methods. Additionally, complex prompts that transform edges into sentences can lengthen the context, restricting model’s scalability for larger graphs. Indeed, previous studies [3, 10, 24, 37] have limited graph sizes to very small scales (e.g., fewer than 20 nodes and 100 edges). Existing methods also lack the capability to control or understand the output reasoning process. This raises concerns about whether the model is accurately deducing answers through logical reasoning or simply making correct guesses. *Significantly, there is a noticeable gap in the availability of an open-source language model proficient in explicitly and accurately solving these graph problems.*

In this paper, we aspire to enhance the graph problem-solving abilities of current open-source LLMs and finetune a single model that can solve various types of graph problems and meanwhile output explicit reasoning paths, as the example of connectivity shown in Figure 1. However, developing instruction-following models for graph problems is challenging due to the absence of a robust training corpus equipped to aid models in producing explicit reasoning paths. Such paths are critical for understanding and evaluating the model’s reasoning process. To tackle this challenge, we introduce a novel instruction-tuning dataset named GraphInstruct, which is designed to serve as a foundational corpus, enabling language models to not only comprehend graph problems but also articulate clear and logical reasoning paths. GraphInstruct is constructed with the “self-augment” idea: we first sample multiple explicit predictions for each graph problem sample using few-shot Chain-of-Thought (CoT) [42] prompting method based on GPT-4, and then finetune a smaller LLM on these predictions. Given a graph problem, the finetuned model is used to augment the original corpus by generating diverse reasoning paths. This self-augmentation process is crucial in enhancing the diversity and complexity of the reasoning paths,

thereby enriching the dataset and making it more representative of various graph problems. GraphInstruct offers 72,785 training samples across nine graph problem tasks, ranging from linear and polynomial complexity to NP-complete, extending the scope and scale of previous benchmarks.

Upon constructing a corpus specialized for explicit reasoning in graph problems, we fine-tuned current open-source LLMs, including the LLaMA 2 families [33, 34] and Mistral [15, 16]. The resulting model, GraphWiz, achieves superior performances in solving various graph computational problems. Our training strategy involves mixed-task instruction tuning and directly preference optimization (DPO) alignment. This dual-focused approach ensures the model not only imitates the best examples but also actively avoids common errors. As a result, the most advanced version of GraphWiz-DPO achieves an average accuracy of 65% across all tasks, significantly outperforming GPT-4, which has an average accuracy of 43.8%. Finally, we delve into the nuanced relationship between training volume, model performance, and overfitting risks. We find that while increasing training data volume generally leads to improved model performance, there is a threshold beyond which the benefits diminish and the risk of overfitting becomes pronounced. Additionally, we explore the potential of transferring the model’s reasoning ability across different graph computational problems.

The contributions of this paper are summarized as follows:

- We collect the first large-scale instruction-tuning dataset named GraphInstruct specifically designed for training LLMs on a variety of graph computational tasks. This dataset enables the trained models to output explicit reasoning paths and arrive at final answers, significantly enhancing the models’ interpretability.
- We introduce GraphWiz, the first open-source LLM specialized for solving graph problems of various types and scales through explicit reasoning. This model markedly outperforms the current best closed-source model, GPT-4, demonstrating superior capability in understanding complex graph structures and addressing related tasks.
- We analyze potential factors that impact model performance in detail, such as the volume of training data and the sampling strategy for dispreferred samples within the DPO framework. This analysis provides valuable insights into model optimization and performance enhancement across diverse graph tasks.

## 2 PRELIMINARY AND RELATED WORKS

**Instruction Tuning** Instruction Tuning [5, 21, 23, 32, 41] is a process where LLMs are explicitly trained to understand and execute instructions provided in natural language. This approach leverages

the models’ inherent ability to process and generate language, refining it to follow specified tasks more accurately. Concretely, it involves providing a model with a set of instructions or prompts that define specific tasks it should perform. These instructions are formulated in natural language and are designed to guide the model in understanding and executing various tasks. The process then involves adjusting the model’s parameters to optimize its performance on these tasks. To date, it’s recognized as an essential approach to enhance the LLMs’ ability to understand and execute a broad range of tasks based on natural language instructions. In this work:

We aim at leveraging instruction-tuning to build a powerful instruction-following LLM that can understand textual descriptions of graph structures and solve various graph problems explicitly in natural language.

In a general scenario, consider an undirected or directed graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ , where  $\mathcal{V}$  and  $\mathcal{E}$  represent the set of nodes and edges, respectively. Each graph is paired with a task-specific instruction or prompt question  $Q$ . Let  $\mathcal{M}$  denote the current generative LLM, which takes each  $\mathcal{G}$ - $Q$  pair as inputs and returns the step-by-step textual reasoning path  $\mathcal{R}$ , as illustrated in Figure 1. Formally,  $\mathcal{R} = \mathcal{M}(\mathcal{G}, Q)$ , where  $\mathcal{R}$  consists of multiple tokens or sentences, commonly referred to as the Chain-of-Thought (CoT) in natural language processing. For certain graph problem tasks, additional information is included in  $\mathcal{G}$ , such as the edge weight  $w$  in identifying shortest paths.

**Directly Preferred Optimization** Although current LLMs demonstrate remarkable capabilities across a diverse set of tasks due to extensive training datasets and instruction tuning strategies, they are not immune to issues such as generating misleading information and producing offensive content. Reinforcement Learning from Human Feedback (RLHF) [23, 28, 50], aims to mitigate these issues by leveraging human feedback to guide the model’s learning process. Despite the efficacy of RLHF with PPO [28] in aligning LLMs with diverse human preferences, this approach necessitates the use of four distinct sub-models, which complicates and increases the cost of training. As an alternative, DPO [25] suggests distilling a referential *SFT* policy,  $r_{\text{ref}}$ , by accentuating preference differences, which is a simple approach for policy optimization. The DPO method optimizes the policy  $\pi_0$  using a contrasting pair of outputs  $(y_u, y_l)$ , as described in the equation below:

$$\mathcal{L}_{\text{DPO}}(\pi_0; r_{\text{ref}}) = -\mathbb{E}_{(x, y_u, y_l) \sim D} \left[ \log \sigma \left( \beta \log \frac{\pi_0(y_u|x)}{\pi_0(y_l|x)} - \beta \log \frac{r_{\text{ref}}(y_u|x)}{r_{\text{ref}}(y_l|x)} \right) \right], \quad (1)$$

where  $y_u$  is favored over  $y_l$ , and  $\beta$  is a hyperparameter. Of note, DPO requires a separate training corpus that belongs to a similar domain with  $r_{\text{ref}}$ .

## 3 METHODOLOGY

### 3.1 GraphInstruct Collection

In the pursuit of enhancing LLMs’ capabilities to tackle graph computational problems with explicit reasoning paths, we develop a new dataset named GraphInstruct. This section outlines the detailed

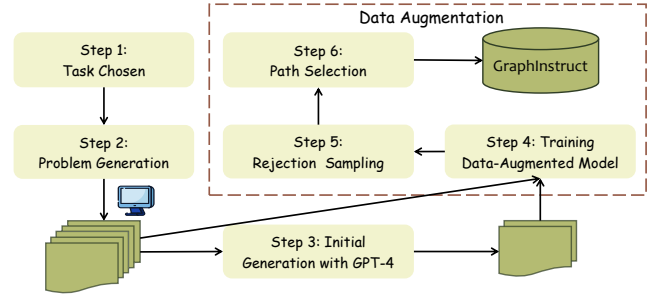


Figure 2: The overview of GraphInstruct collection process.

procedures followed to compile and enhance this dataset, including graph task selection, graph problem generation, and subsequent data augmentation. An overview of our dataset collection process is illustrated in Figure 2.

**3.1.1 Graph Task Selection.** In this work, we meticulously choose a diverse set of nine graph computational tasks that encompass different levels of computational complexity. We include four linear complexity tasks: *Connectivity*, *Cycle Detection*, *Bipartite Graph Checking*, and *Topological Sort*; three polynomial complexity tasks: *Shortest Path*, *Maximum Triangle Sum*, and *Maximum Flow*; and two NP-Complete tasks: *Hamilton Path* and *Subgraph Matching*. These tasks are defined in Table 1, with detailed explanations provided in Appendix A. These nine graph tasks provide a comprehensive exploration of algorithmic graph theory, allowing us to enhance the theoretical understanding of graph algorithms and address a broad range of practical applications.

**3.1.2 Graph Problem Generation.** To create a diverse and challenging suite of graph problems for model training and testing, we adopt a programming-aid approach inspired by [11] that generates random graph problems for each predefined task. Each task is paired with a unique template designed to capture the distinct characteristics of the graphs, such as being directed or undirected and whether the edges contain weights. We employ the Erdős-Rényi (ER) model [9] to generate random graphs. Specifically, the ER model accepts two parameters: the number of nodes  $n$  and the probability  $p$  of an edge existing between any two nodes. For each pair of nodes, the generator randomly decides whether to create an edge between them with probability  $p$ , resulting in a graph with an average edge density of  $p$ . We utilize the NetworkX library [12] to generate the random graphs and determine the solutions to the graph tasks.

To enhance the quality and diversity of generated problems, we implement specific constraints during the problem generation phase. (1) **Diverse Distributions.** Each task is crafted to include graph problems with varied distributions. We specify five combinations of node counts and edge densities for each task to introduce varying difficulty levels. The node ranges for each task are detailed in Table 2. It is observed that edge density impacts tasks differently. For example, increasing the edge density in Shortest Path problems generally raises the difficulty by adding more potential routes. Conversely, in Cycle Detection, a higher density may lead to more prevalent cycles, thus simplifying detection. (2) **Length Constraints.** Considering the token limitations of most current

**Table 1: Overview of nine tasks in our GraphInstruct benchmark with problem definition, time complexity of representative algorithms, graph type (weighted/directed), node size range, and task difficulty.  $|\mathcal{V}|$  and  $|\mathcal{E}|$  indicate the number of nodes and edges in the graph.**

Problem	Definition	Time Complexity	Weighted?	Directed?	Node Range	Difficulty
Cycle Detection	Detect if a given graph $\mathcal{G}$ contains any cycles.	$O( \mathcal{V}  +  \mathcal{E} )$	✗	✗	[2, 100]	Easy
Connectivity	Assess if two nodes $u$ and $v$ in a given graph $\mathcal{G}$ are connected via a path.	$O( \mathcal{V}  +  \mathcal{E} )$	✗	✗	[2, 100]	Easy
Bipartite Graph Check	Judge if a given graph $\mathcal{G}$ is bipartite.	$O( \mathcal{V}  +  \mathcal{E} )$	✗	✓	[2, 100]	Easy
Topological Sort	Find a topological ordering of vertices in a directed acyclic graph $\mathcal{G}$ .	$O( \mathcal{V}  +  \mathcal{E} )$	✗	✓	[2, 50]	Easy
Shortest Path	Compute the shortest path between two specific nodes $u$ and $v$ in a given graph $\mathcal{G}$ .	$O( \mathcal{E}  +  \mathcal{V} \log \mathcal{V} )$	✓	✗	[2, 100]	Medium
Maximum Triangle Sum	Find the maximum sum of weights for any connected triplet of vertices in a given graph $\mathcal{G}$ .	$O( \mathcal{V} ^3)$	✓	✗	[2, 25]	Medium
Maximum Flow	Calculate the maximum flow from a source node $s$ to a sink node $t$ in a directed graph $\mathcal{G}$ .	$O( \mathcal{V} ^2\sqrt{ \mathcal{E} })$	✓	✓	[2, 50]	Medium
Hamilton Path	Determine if a given graph $\mathcal{G}$ has a Hamiltonian path that visits each vertex exactly once.	NP-Complete	✗	✗	[2, 50]	Hard
Subgraph Matching	Verify if there exists a subgraph in $\mathcal{G}$ that is isomorphic to a given graph $\mathcal{G}'$ .	NP-Complete	✗	✓	[2, 30]	Hard

**Table 2: Statistics of our GraphInstruct corpus, including total samples ( $\mathcal{G}-Q$ ), nodes ( $\mathcal{V}$ ) and reasoning paths  $\mathcal{R}$ .**

Tasks		Easy				Medium			Hard		Sum.
		cycle	connect	bipartite	topology	shortest	triangle	flow	hamilton	Subgraph	
Train	Total $\mathcal{G}\text{-}Q$	3,717	2,687	2,013	902	1,392	2,756	405	2,097	1,435	17,158
	Total $\mathcal{V}$	84,980	79,853	58,860	10,146	23,204	14,714	4,333	33,284	7,847	315,051
	Total $\mathcal{R}$	13,122	10,001	9,324	4,481	5,859	13,483	747	8,454	6,274	72,785
Test	Total $\mathcal{G}\text{-}Q$	400	400	400	400	400	400	400	400	400	3,600
	Total $\mathcal{V}$	19,570	19,500	19,515	9,449	19,449	4,990	10,024	9,732	6,594	118,823

**Table 3: Comparison between ours and other typical datasets.**

Datasets	Include Training Set?	Include CoTs?	Tasks	Node Scale	Edge Scale
NLGraph	No	No	8	9-35	10-30
GraphQA	No	No	12	5-20	1-100
<b>GraphInstruct</b>	<b>Yes</b>	<b>Yes</b>	<b>9</b>	<b>2-100</b>	<b>5-500</b>

open-source LLMs (e.g., 4096 tokens), we eliminate excessively long graph problems to maintain compatibility, ensuring no graph exceeds 100 nodes. (3) **Unique Instances.** To build a robust dataset, it is crucial that each problem within the training and testing sets is unique. (4) **Efficient Graph Description.** In the textual descriptions of each problem, we only specify the total number of nodes and depict edges using tuples  $(u, v)$ , where  $u$  and  $v$  represent the connected nodes. As illustrated in Figures 1, 8 and Table ??, this graph description language maximize graph sizes within the limits of input length constraints.

We curate an initial set of 27,000 graph problems ( $\mathcal{G}-Q$ ) for training, distributed across nine tasks, with 3,000 pairs allocated to each. Furthermore, we prepare 3,600 graph problems ( $\mathcal{G}-Q$ ) for testing, also divided among nine tasks, with each task receiving 400 pairs. Notably, we only annotate  $\mathcal{R}$  for the training problems.

**3.1.3 Generation of Explicit Reasoning Paths.** The most distinctive aspect of GraphInstruct is that each  $\mathcal{G}-Q$  pair is coupled with an explicit reasoning path,  $\mathcal{R}$ . Given the intensive effort required to manually annotate these paths for graph tasks, we utilize GPT-4 to generate preliminary reasoning paths. Each  $\mathcal{G}-Q$  is fed into GPT-4 with a CoT prompt designed to draw out an explicit reasoning path, concluding with “The answer is,” to direct the model towards a definitive solution. To promote diversity in responses, we employ a sampling strategy with a temperature setting of 0.9, repeating the procedure three times for each problem. We then select only those  $\mathcal{G}-Q$  pairs that demonstrate accurate reasoning paths—those leading to the correct answer—to form our initial dataset,  $D_1$ , which includes fewer than 8,000 samples.

**Data Augmentation with Rejection Sampling.** Observations indicate that GPT-4 struggles with numerous graph tasks, such as yielding fewer than 100 correct samples for the Maximum Flow task in  $D_1$ . To enrich the dataset with a broader range of reasoning paths, we implement a rejection sampling strategy [5]. Using  $D_1$ , we develop a specialized data augmentation model, designated as  $M_1$ , serving two primary functions: (1) **Increasing  $\mathcal{G}$ - $Q$  quantity:**  $M_1$  is aimed at correctly answering questions that GPT-4 previously mishandled, thus increasing the number of viable  $\mathcal{G}$ - $Q$  pairs. (2) **Generating diverse paths:**  $M_1$  is also designed to generate multiple diverse reasoning paths for the same  $\mathcal{G}$ - $Q$ , thereby enhancing the diversity of our dataset. Specifically,  $M_1$  processes the initial 27,000 graph problems, with the goal of producing multiple reasoning paths per problem through repeated sampling, aiming to significantly broaden the dataset’s diversity.

**Diverse Reasoning Path Selection.** A pivotal aspect of rejection sampling involves selecting correct and diverse reasoning paths to ensure the dataset’s quality and utility. Balancing accuracy with diversity presents significant challenges in this phase. To tackle these, we employ refined strategies divided into string-based and semantic-based approaches for selecting distinct generated reasoning paths:

- **String-based strategies:** These include using *edit distance*, *TF-IDF* [2], and *Jaccard similarity* to evaluate the uniqueness and relevance of the reasoning paths, ensuring a broad representation of problem-solving techniques.
- **Semantic-based strategies:** We utilize Sentence-BERT [26] to encode the reasoning paths and apply *cosine similarity* and *K-means clustering* [20] to assess their semantic coherence and diversity, encouraging distinct and logically coherent paths.

In our string-based and semantic-based selection strategies, except for K-means clustering, the methodology necessitates a reference or anchor sample to effectively measure the diversity of reasoning paths. This anchor acts as the pivot for assessing the diversity of other paths. We select the *longest correct reasoning path generated for each problem* as the anchor, based on the assumption that it is likely to encompass a comprehensive and detailed reasoning process [17], thus serving as an ideal reference for evaluating the diversity of other paths.

After applying each selection strategy, we compile the final GraphInstruct,  $D$ , by incorporating the reasoning paths that exhibit the greatest diversity compared to the anchor path, as determined by each strategy. In contrast, K-means clustering groups the generated reasoning paths based on their semantic similarity, from which we identify and select a representative path from each cluster, filtering out repetitive reasoning paths and ensuring the left paths remain diverse and distinct. This approach guarantees that GraphInstruct contains not only correct solutions to graph problems but also a broad spectrum of reasoning styles and methodologies. GraphInstruct fosters the development of LLMs capable of generating explicit and interpretable reasoning paths, significantly enhancing their problem-solving proficiency across a diverse array of graph computational problems. Through this endeavor, we aim to bridge the gap between the current capabilities of LLMs and the complex demands of graph problem-solving.

**3.1.4 GraphInstruct Statistics.** GraphInstruct encompasses 17,158 graph-question ( $\mathcal{G}$ - $Q$ ) pairs, 313,051 nodes ( $\mathcal{V}$ ), and 72,785 reasoning paths ( $\mathcal{R}$ ), with each graph problem potentially linked to multiple paths. Easy tasks are more prevalent, while tasks like Maximum Flow and Topology Sort are less common. This imbalance is primarily due to the difficulty of generating correct reasoning paths, even with advanced models such as GPT-4. Reasoning paths are pivotal in our dataset, as a single  $\mathcal{G}$ - $Q$  pair may correspond to multiple unique paths, thereby enriching the diversity of GraphInstruct. Additionally, the test set introduces more complex challenges, where an increased number of nodes generally indicates greater difficulty in graph-related problems. This comprehensive dataset forms a robust foundation for training and testing models across a range of graph computational challenges, from basic to advanced levels, as the comparison in Table 3.

## 3.2 GraphWiz

Based on the GraphInstruct dataset, we develop GraphWiz, which is designed to enhance the capabilities of current LLMs in solving graph computational problems with explicit reasoning paths. The training methodology for GraphWiz involves a novel two-phase process. Initially, we employ Mixed-Task Instruction Tuning to refine the model’s ability to interpret and solve a diverse array of graph problems. The subsequent phase, DPO Alignment, further sharpens the model’s reasoning by training it to differentiate between more and less effective problem-solving paths. This approach, especially the application of DPO in the realm of graph problems, marks a significant advancement in teaching LLMs not only to generate explicit answers but also to develop and adhere to a logical reasoning process that mimics expert problem-solving behavior.

**3.2.1 Phase 1: Mixed-Task Instruction Tuning.** The initial phase of our training, called Mixed-Task Instruction Tuning, involves training the model simultaneously on all nine graph tasks from the GraphInstruct dataset. This strategy is designed to provide the model with comprehensive capabilities to solve a diverse array of graph problems by adhering to task-specific instructions. Additionally, it leverages the synergies among different tasks to enhance overall model performance.

During this phase, we employ Supervised Fine-Tuning (SFT) with a conventional language modeling loss. The model is trained to process both the graph  $\mathcal{G}$  and its corresponding prompt question  $Q$  as inputs and to generate textual reasoning paths  $\mathcal{R}$  as outputs. The SFT loss is computed based on the discrepancy between the model’s predicted reasoning paths and the actual reasoning paths in the dataset, formally defined as follows:

$$\mathcal{L}_{LM} = - \sum_{i=1}^N \sum_{j=1}^M \log P(\mathcal{R}_{i,j} | \mathcal{G}_i, Q_i; \theta) \quad (2)$$

where  $N$  represents the number of examples,  $M$  is the total number of reasoning paths for  $\mathcal{G}_i$ - $Q_i$ , and  $\theta$  are the parameters of the model.

This optimization process trains the model to map graph structures and textual prompts to their corresponding reasoning paths, thereby fostering a deep understanding of how to navigate and solve graph problems using explicit reasoning. The model developed in Phase 1 is named  $M(\theta)_{\text{sft}}$ .



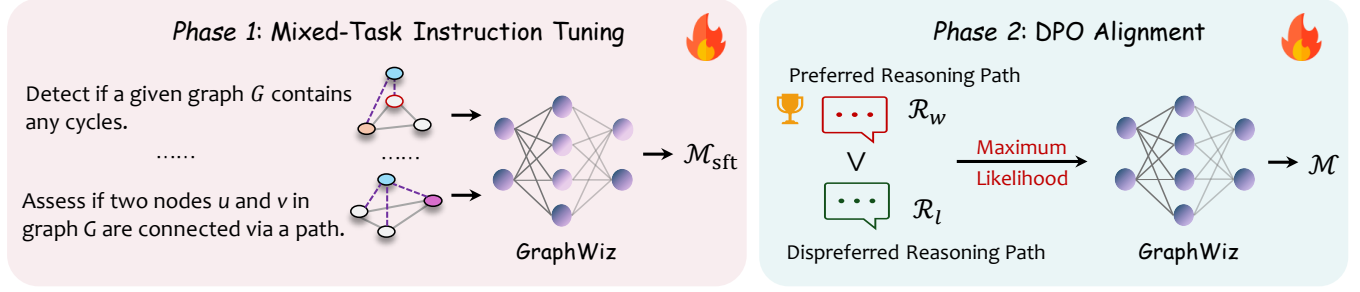


Figure 3: Training pipelines of GraphWiz.

**3.2.2 Phase 2: DPO Alignment of Reasoning Abilities.** The second phase of training incorporates Direct Preference Optimization (DPO) to enhance the model’s reasoning capabilities. DPO refines the learned SFT model,  $\mathcal{M}(\theta)_{\text{sft}}$ , by aligning it more closely with preferred outcomes using a training corpus similar to the SFT data domain. Specifically, DPO employs input pairs labeled  $(\mathcal{R}_w, \mathcal{R}_l)$ , where  $\mathcal{R}_w$  and  $\mathcal{R}_l$  denote the preferred and less preferred reasoning paths, respectively. In the context of graph problem-solving, the challenge is: *how do we obtain  $\mathcal{R}_w$  and  $\mathcal{R}_l$  for each  $\mathcal{G}$ - $Q$  pair?*

To generate these pairs for each  $\mathcal{G}$ - $Q$ , we start by generating a new set of 9,000 graph problems. Using the SFT model  $\mathcal{M}(\theta)_{\text{sft}}$ , we infer outcomes 20 times for each problem. Consistent with the approach in Section 3.1.3, we select the longest correct path as the preferred reasoning path ( $\mathcal{R}_w$ ). For less preferred paths ( $\mathcal{R}_l$ ), the selection process includes: (1) Using the preferred sample as an anchor, we employ the string-based and semantic-based selection methods (excluding K-means) described in Section 3.1.3 to identify the *closest incorrect path* to the preferred one. (2) We then use Majority Voting [39] to determine the most frequently occurring incorrect path across these strategies, which serves as our dispreferred sample.

This methodology not only ensures the identification of reasoning paths that align with the model’s understanding but also targets hard-negative examples similar to the preferred paths but are fundamentally incorrect. This approach significantly enhances the effectiveness of the DPO stage by rigorously challenging the model to distinguish and preferentially align its reasoning capabilities toward more accurate and logical problem-solving strategies. Finally, we collect 6,166  $\mathcal{G}$ - $Q$ - $\mathcal{R}_w$ - $\mathcal{R}_l$  samples for our DPO training corpus, with detailed statistics available in the Appendix, Table 7. The formal training objective for DPO is defined as follows:

$$\mathcal{L}_{\text{DPO}}(\mathcal{M}(\theta); \mathcal{M}(\theta)_{\text{sft}}) = -\mathbb{E}_{(x, \mathcal{R}_w, \mathcal{R}_l) \sim D} \left[ \log \sigma \left( \beta \log \frac{\mathcal{M}(\theta)(\mathcal{R}_w|x)}{\mathcal{M}(\theta)(\mathcal{R}_l|x)} - \beta \log \frac{\mathcal{M}(\theta)_{\text{sft}}(\mathcal{R}_w|x)}{\mathcal{M}(\theta)_{\text{sft}}(\mathcal{R}_l|x)} \right) \right], \quad (3)$$

where  $x$  represents the concatenation of  $\mathcal{G}$  and  $Q$ .

## 4 EXPERIMENTS

In this section, we evaluate the proposed GraphWiz by addressing the following research questions: **RQ1**: How does GraphWiz perform on these graph tasks, particularly in comparison to the most powerful closed-source model, such as GPT-4? **RQ2**: What is the

impact of training data volume variations on GraphWiz’s performance? **RQ3**: How transferable is GraphWiz across different graph tasks? **RQ4**: How do changes in the number of nodes in a graph affect GraphWiz’s performance, and what is the maximum complexity of graphs it can effectively handle? **RQ5**: How does hyperparameter  $\beta$  influence model performance.

### 4.1 Experimental Settings

Experimentally, we define baselines across two principal settings: (1) **In-Context Learning**, where the model parameters remain fixed. We utilize state-of-the-art closed-source LLMs as robust baselines, including GPT-3.5 (turbo-0613) and GPT-4 (2023-03-14-preview). Their effectiveness is evaluated using Chain-of-Thought (CoT) prompting, which provides the model with a sequence of examples that sequentially address task solutions. These models are then tasked to replicate this reasoning in their outputs for new problems. We assess their performance under zero-shot and 2-shot scenarios. For zero-shot prompts, the phrase “Let’s think step by step” is included to stimulate the generation of autonomous CoTs. (2) **Training with Naive Labels**: Our focus here is on training smaller-scale LLMs with simplified labels—direct answers like ‘yes’ or ‘no’, without detailed reasoning paths. This approach is termed **Naive-SFT**. Additionally, we test three Graph Neural Network (GNN) models—Graph Convolutional Network (GCN) [18], Graph Isomorphism Network (GIN) [43], and Graph Attention Network (GAT) [36]—on seven of the nine tasks using identical training and test sets as GraphWiz. Topology and Hamilton are excluded as they require path-based reasoning, which GNNs are not originally support.

**4.1.1 Implementation.** In this work, we use open-source LLaMA 2-7B/13B and Mistral-7B as backbone models, allowing us to build GraphWiz in multiple scales and architectures. Our codebase is built on DeepSpeed and Huggingface Library. For all models, we set the learning rate, epochs, and max length as  $2e-5$ , 3, and 2048, running on NVIDIA 8\*A800 GPUs. The batch size is set to 16 or 32. We use the alpaca-instruction format during training. For the GraphInstruct generation in Section 3.1.3, we use 2-shot CoT prompting for GPT-4. For rejection sampling to augment the data, we train LLaMA 2-13B model for this purpose. Specifically, we sample 30 times with a temperature of 0.9 for each sample, pursuing a high diversity of reasoning paths. For testing, we set temperature as 0 and maximum output tokens as 1024, ensuring the stable and reasonable generation. All GNNs use a hidden dimension size of 16,

**Table 4: Performances of GraphWiz and other baselines on GraphInstruct test set.**

Categories	Algorithms	Easy				Medium			Hard		Average
		cycle	connect	bipartite	topology	shortest	triangle	flow	hamilton	subgraph	
Closed-Source	GPT-4 (zero-shot)	38.75	17.00	65.25	5.00	9.25	5.75	3.25	59.25	45.50	27.67
	GPT-3.5 (2-shot)	51.25	43.75	70.75	4.50	3.50	17.25	8.50	54.25	43.00	32.97
	GPT-4 (2-shot)	52.50	62.75	74.25	25.25	18.25	31.00	7.75	75.75	46.75	43.81
Graph Neural Networks	GCN	84.00	74.00	82.00	-	5.75	6.75	9.25	-	68.00	-
	GIN	87.50	73.00	85.25	-	7.25	7.30	12.00	-	66.50	-
	GAT	87.50	79.25	85.25	-	7.25	7.50	12.50	-	66.25	-
Mistral-7B	Naive SFT	73.75	83.50	78.50	1.00	23.00	47.00	28.75	31.75	41.25	46.56
	<b>GraphWiz</b>	92.00	89.50	72.00	19.00	31.25	38.75	29.25	26.50	85.50	53.75
	<b>GraphWiz-DPO</b>	85.50	79.50	85.50	85.25	12.50	29.00	35.50	62.75	48.50	58.22
LLaMA 2-7B	Naive SFT	73.75	83.50	41.25	4.00	9.50	30.00	16.50	69.00	75.45	44.81
	<b>GraphWiz</b>	91.50	87.00	74.00	18.00	28.00	38.25	24.50	52.25	82.25	55.08
	<b>GraphWiz-DPO</b>	89.00	82.50	84.75	46.75	24.00	52.75	43.50	81.50	77.25	65.00
LLaMA 2-13B	Naive SFT	73.75	83.75	59.00	0.50	11.75	34.75	24.25	59.75	54.75	44.69
	<b>GraphWiz</b>	94.75	87.00	78.00	28.00	27.75	36.00	24.50	59.00	81.50	57.39
	<b>GraphWiz-DPO</b>	87.50	88.50	88.25	72.75	22.00	48.75	43.75	46.50	77.00	63.89

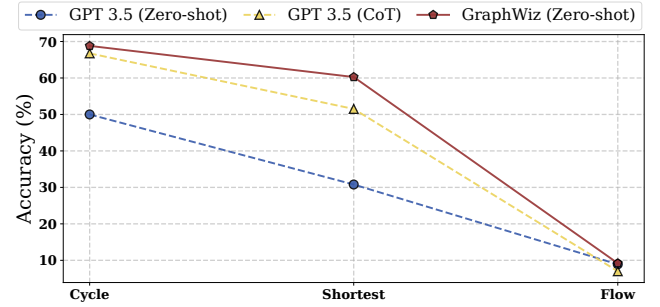
**Table 5: GraphWiz performances with increasing ratios of the graph to reasoning path ( $\mathcal{G}:\mathcal{R}$ ), which is indicative of the data volume and diversity of reasoning paths available for training.  $\mathcal{G}:\mathcal{R}=1:5$  means the whole corpus of GraphInstruct.**

Backbone	$\mathcal{G}:\mathcal{R}$	Easy	Medium	Hard	Average
Mistral-7B	1:1	61.81	32.92	36.50	46.56
	1:2	68.00	33.17	38.50	49.83
	1:3	65.88	34.25	57.50	53.47
	1:4	68.75	31.92	55.75	53.58
	1:5	68.13	31.50	56.00	53.75
LLaMA 2-7B	1:1	58.63	28.92	38.13	44.17
	1:2	68.00	26.25	61.75	53.00
	1:3	69.56	31.92	60.50	54.00
	1:4	68.00	29.25	64.00	54.36
	1:5	67.63	30.00	67.25	55.08

2 layers, and 1-dimensional random input features from a Normal distribution.

## 4.2 Performance of GraphWiz (RQ1)

We repeat our evaluation three times and report the average results in Table 4 and Figure 4. Notably, our models exhibit exceptional results across various backbones, significantly surpassing GPT-4’s performance. This consistency is maintained across a range of tasks, from easy to difficult. DPO further enhances the average performance across tasks. However, DPO might adversely affect specific tasks, indicating that while DPO generally improves model reasoning, it may require further tuning to avoid negative impacts on certain problem types. Although Naive SFT shows commendable


**Figure 4: Performance of GraphWiz (Mistral) on NLGraph test set. The results for GPT-3.5 were obtained from its original paper [37]. In the CoT setting, the cycle task uses 4 examples, while the shortest and flow tasks use 5 examples.**

performance, even slightly better than GPT-4, it is unclear if the outcomes are from random guessing or actual task understanding.

**Dataset Transfer.** In addition to our constructed test set, we also utilize NLGraph [37] as part of our testing suite. Given the imbalanced distribution of graph task samples within NLGraph, we select three tasks for evaluation: Cycle Detection, Maximum Flow, and Shortest Path. We directly test our models without any example prompts or further training. Our approach also demonstrates robust cross-dataset transfer capabilities. Figure 4 illustrates that both GraphWiz maintains a higher level of accuracy compared with GPT-3.5, underscoring the effectiveness of our methodology across datasets with varying distributions.

**Comparison with GNNs.** GNNs have competitive results on simple graph binary classification tasks like Cycle, Bipartite, and Connectivity. However, as task complexity increases, requiring

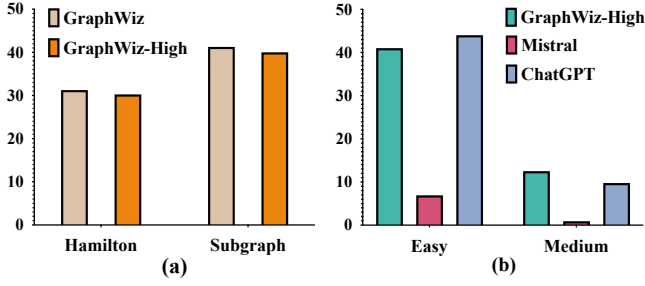


Figure 5: (a) Model performance in two high-complexity tasks; (b) Average model performance of in easy and medium graph tasks. The backbone of GraphWiz is Mistral.

multi-step numerical reasoning, GNNs’ performance markedly declines compared to GraphWiz. Beyond the results, we highlight three fundamental differences between GNNs and GraphWiz: (1) *Unification*: GNNs require separate training for each task, often necessitating distinct model structures to accommodate different task inputs. In contrast, a single GraphWiz model is capable of handling all tasks effectively. (2) *Generalization*. GraphWiz has the potential to perform zero-shot task transfer: train on some tasks and test on other tasks, a capability absent in GNNs. (3) *Explainability*. Although both models perform similarly on simple tasks such as cycle detection, GraphWiz offers added value by providing step-by-step solutions and identifying specific locations of the cycle, thereby enhancing its explainability. While combining GNNs and LLMs may offer mutual benefits, it also introduces new challenges in model design and joint training. Addressing these challenges in solving graph computational problems remains an area for future work.

### 4.3 Results of GraphWiz with Reasoning Path Increasing (RQ2)

Table 5 illustrates the average performance of two variants of GraphWiz, Mistral 7B and LLaMA 2-7B, across various task categories. Detailed task results are provided in the Appendix, Table 8. The performance metrics consider varying ratios of graph to reasoning path ( $\mathcal{G}:\mathcal{R}$ ), which represent the data volume and diversity of reasoning paths employed during training. Across all tasks, both models exhibit consistent performance enhancements in correlation with an expanded training corpus. This trend suggests that a greater variety of reasoning paths typically boosts model efficacy across graph-based problems.

Nevertheless, specific tasks such as Triangle Sum and Hamilton Path display negligible or even reduced accuracy improvements with larger data volumes. For instance, the accuracy for GraphWiz (Mistral-7B) in the Triangle Sum task declines from 47.00% at a 1:1 ratio to 38.75% at a 1:5 ratio, potentially indicating issues such as overfitting where the model excessively learns specific training data patterns at the expense of generalization to new problems. This observation underscores the importance of careful monitoring and validation during model training to ensure broad applicability and effectiveness.

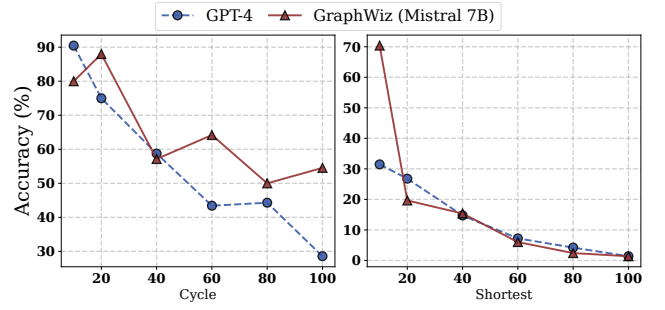


Figure 6: Performance of GPT-4 (2 shot) and GraphWiz based on Mistral 7B with different graph sizes (nodes range) on two tasks. In this work, we use the number of nodes as an indicator of graph complexity.

Furthermore, the  $\mathcal{G}:\mathcal{R}$  ratios in tasks such as Maximum Flow do not strictly conform to anticipated increments like 1:2 or 1:3. Specifically, in the Maximum Flow task, the GraphInstruct dataset records a  $\mathcal{G}:\mathcal{R}$  ratio below 1:2 (see Table 2). Despite these deviations, both models can exhibit improved performance in this task, indicating that data volume increases in certain tasks might impact results in others. This suggests the potential for cross-task learning, where models apply reasoning paths learned from one task to enhance performance on another, even without a proportional increase in the data volume for the latter.

### 4.4 Transferability of GraphWiz (RQ3)

To explore the transferability of GraphWiz, we establish an additional model variant to explore this aspect: GraphWiz-High. This model is trained exclusively on two high-complexity (NP-Complete) graph tasks: Hamilton Path and Subgraph Matching. To investigate its transferability, we conduct two comparative experiments:

**Comparison on High-Complexity Tasks.** We first compare GraphWiz-High with the regular GraphWiz on high-complexity tasks. Figure 5 (a) indicates that GraphWiz performs better, validating the effectiveness of mixed-task training. This outcome also suggests that the model is capable of transferring skills learned from other tasks to specific high-complexity tasks.

**Zero-Shot Transfer Capability.** We further test the zero-shot transfer capability of GraphWiz-High on low and medium-complexity tasks it has never been trained on. As can be seen from Figure 5 (b), GraphWiz-High shows a significant performance improvement compared to the Mistral-7B backbone. Even when compared to GPT-3.5, our model managed to maintain comparable performance. Considering the vast difference in the number of parameters between GPT-3.5 and GraphWiz-High, this indicates that ours possesses a commendable ability to generalize across tasks.

### 4.5 Results of GraphWiz with Increasing Problem Complexity (RQ4)

To address the questions regarding how the model’s performance varies with different graph sizes and to figure out the largest graph size the model can effectively solve, we present Figure 6 showcasing the performance of the GraphWiz model on the best-performing



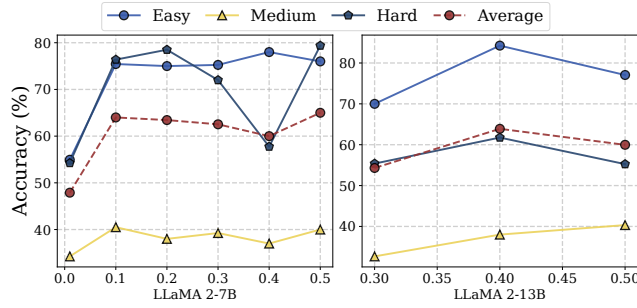


Figure 7: Hyperparameter study of  $\beta$ .

task (a) Cycle Detection, and the most challenging tasks (b) Shortest Path. From the figure, we draw the following conclusions: (1) Both GraphWiz and GPT-4 exhibit a decrease in performance as the size of the graph increases. However, our model consistently outperforms GPT-4 across most graph sizes, indicating a more robust understanding and processing of graph structures. (2) We observe that the performance on the latter task, Shortest Path, needs improvement, with a notable decline as the number of nodes increases. This decline can likely be attributed to two main factors: The latter tasks demand high reasoning and memory capabilities as higher time complexity, as well as strong computational skills, which may pose an additional challenge to the models' capacities; Experimentally, we find that both models primarily rely on enumeration to arrive at solutions. Consequently, as the graph size enlarges, the required enumerative reasoning grows exponentially, leading to a significant drop in accuracy when the number of nodes exceeds 60, after which they exhibit almost no accuracy.

These observations suggest that while GraphWiz shows a clear advantage over GPT-4 in handling graph-related tasks, there is an upper bound of complexity—particularly evident in tasks that require numerical computation—at which the performance of even the most advanced models starts to diminish significantly.

#### 4.6 Sensitivity Analysis (RQ5)

We perform sensitivity analysis to understand the influence of hyperparameter  $\beta$  on GraphWiz. Figure 7 presents the average performance of our models based on LLaMA 2-7B and 13B versions in *easy*, *medium* and *hard* tasks. The results indicate that the optimal value of  $\beta$  varies depending on the task difficulty and the model size. A higher  $\beta$  seems to favor performance on Hard tasks to some extent, but this is not a strictly linear relationship and does not hold consistently across different model sizes. This study highlights the importance of hyperparameter tuning in achieving the best possible model performance across a range of tasks. The analysis suggests that careful tuning of  $\beta$  is necessary to achieve the best trade-off performances.

#### 4.7 Case Study

In this subsection, we present a case study of a complex cycle detection problem that involves an undirected graph with **89 nodes** in Figure 8, a scale that poses substantial challenges even for human solvers. For a human to detect a cycle in such an extensive graph would require external aids or substantial time, as it is impractical

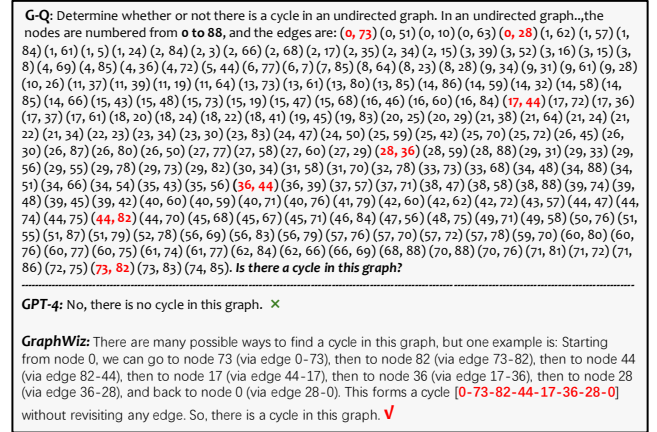


Figure 8: A typical case of GraphWiz and GPT-4.

to resolve purely through mental computation. We could observe that GPT-4 outputs a very short incorrect answer, which could arise from the model's constraints in processing long inputs or from an erroneous interpretation of the graph's intricate structure. This limitation reflects the challenges that conventional LLMs encounter when adapting to graph-theoretic problems. In contrast, GraphWiz correctly detects a cycle, providing a clear and detailed reasoning path. This demonstration of GraphWiz's spatial reasoning and memory retention capabilities is significant. It indicates that the model has effectively internalized the principles of graph theory to the extent that it can autonomously navigate through and reason about large-scale and complex graph structures. This case study is a testament to the feasibility of employing GraphWiz for sophisticated graph problems. The model's ability to accurately resolve such complex tasks suggests it has substantial practical applications, offering a robust tool for researchers and practitioners.

## 5 CONCLUSION

This paper tackles the challenge of equipping LLMs not only to process but to reason explicitly on a variety of graph computational problems. We introduce GraphInstruct, a novel instruction-tuning dataset encompassing a broad spectrum of graph problems, designed to enhance the interpretability and reasoning capabilities of LLMs. We subsequently fine-tune open-source LLMs on this dataset, notably from the LLaMA 2 and Mistral series, leading to the creation of GraphWiz. This model not only surpasses GPT-4 in performance but also showcases the potential for cross-task transferability of reasoning skills. Our experimental analysis highlights the critical balance needed in training volume to maximize model effectiveness and proposes a promising direction on cross-task transfer learning. Future endeavors will focus on devising more efficient model architectures and training methods to further improve GraphWiz's performance and generalization capabilities.

## ACKNOWLEDGMENTS

This work was supported by NSFC Grant No. 62206067, HKUST-HKUST(GZ) 20 for 20 Cross-campus Collaborative Research Scheme C019 and Guangzhou-HKUST (GZ) Joint Funding Scheme 2023A03J0673.

## REFERENCES

- [1] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv:2305.10403* (2023).
- [2] Prafulla Bafna, Dhanya Pramod, and Anagha Vaidya. 2016. Document clustering: TF-IDF approach. In *ICEEOT*. 61–66.
- [3] 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:2310.05845* (2023).
- [4] Nuo Chen, Hongguang Li, Junqing He, Yinan Bao, Xinshi Lin, Qi Yang, Jianfeng Liu, Ruyi Gan, Jiaxing Zhang, Baoyuan Wang, and Jia Li. 2023. Orca: A Few-shot Benchmark for Chinese Conversational Machine Reading Comprehension. In *Findings of EMNLP*. 15685–15699.
- [5] Nuo Chen, Zinan Zheng, Ning Wu, Linjun Shou, Ming Gong, Yangqiu Song, Dongmei Zhang, and Jia Li. 2023. Breaking language barriers in multilingual mathematical reasoning: Insights and observations. *arXiv:2310.20246* (2023).
- [6] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv:2307.03393* (2023).
- [7] 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:2310.04668* (2023).
- [8] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *CoRR* abs/2110.14168 (2021).
- [9] Paul Erdős, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci* 5, 1 (1960), 17–60.
- [10] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a Graph: Encoding Graphs for Large Language Models. In *NeurIPS 2023 Workshop*.
- [11] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *ICML*. 10764–10799.
- [12] Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).
- [13] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2023. Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning. *arXiv:2305.19523* (2023).
- [14] Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2023. Can llms effectively leverage graph structural information: When and why. *arXiv:2309.16595* (2023).
- [15] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv:2310.06825* (2023).
- [16] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv:2401.04088* (2024).
- [17] Mingyu Jin, Qinkai Yu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, Mengnan Du, et al. 2024. The Impact of Reasoning Step Length on Large Language Models. *arXiv:2401.04925* (2024).
- [18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
- [19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. *arXiv:2304.02643* (2023).
- [20] Tupti M Kodinariya, Prashant R Makwana, et al. 2013. Review on determining number of Cluster in K-Means Clustering. *International Journal* 1, 6 (2013), 90–95.
- [21] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *arXiv:2304.08485* (2023).
- [22] OpenAI. 2023. GPT-4 Technical Report. *Arxiv* (2023). <https://doi.org/10.48550/arXiv.2303.08774>
- [23] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *NeurIPS* 35 (2022), 27730–27744.
- [24] Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. 2024. Let Your Graph Do the Talking: Encoding Structured Data for LLMs. *arXiv:2402.05862* (2024).
- [25] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv:2305.18290* (2023).
- [26] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-JCNLP*. 3982–3992.
- [27] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv:2211.05100* (2022).
- [28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv:1707.06347* (2017).
- [29] Yanchao Tan, Hang Lv, Xinyi Huang, Jiawei Zhang, Shiping Wang, and Carl Yang. 2024. MuseGraph: Graph-oriented Instruction Tuning of Large Language Models for Generic Graph Mining. *arXiv:2403.04780* (2024).
- [30] 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. *arXiv:2310.13023* (2023).
- [31] Jianheng Tang, Kangfei Zhao, and Jia Li. 2023. A Fused Gromov-Wasserstein Framework for Unsupervised Knowledge Graph Entity Alignment. In *Findings of ACL*. 3320–3334.
- [32] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model.
- [33] 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:2302.13971* (2023).
- [34] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shriti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288* (2023).
- [35] Trieu H Trinh, Yuhui Wu, Quoc V Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature* 625, 7995 (2024), 476–482.
- [36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *ICLR*.
- [37] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can Language Models Solve Graph Problems in Natural Language?. In *NeurIPS*.
- [38] Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024. InstructGraph: Boosting Large Language Models via Graph-centric Instruction Tuning and Preference Alignment. *arXiv:2402.08785* (2024).
- [39] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv:2203.11171* (2022).
- [40] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khoshabi, and Hannaneh Hajishirzi. 2022. Self-Instruct: Aligning Language Model with Self Generated Instructions.
- [41] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *ICLR*.
- [42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS* 35 (2022), 24824–24837.
- [43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? *ICLR* (2019).
- [44] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv:2210.03629* (2022).
- [45] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. *arXiv:2308.07134* (2023).
- [46] Chenyu You, Nuo Chen, Fenglin Liu, Shen Ge, Xian Wu, and Yuxian Zou. 2022. End-to-end Spoken Conversational Question Answering: Task, Dataset and Model. In *Findings of NAACL*. 1219–1232.
- [47] Chenyu You, Nuo Chen, and Yuxian Zou. 2021. Self-supervised Contrastive Cross-Modality Representation Learning for Spoken Question Answering. In *Findings of EMNLP*. 28–39.
- [48] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhao Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv:2309.05653* (2023).
- [49] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. GraphText: Graph reasoning in text space. *arXiv:2310.01089* (2023).
- [50] Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. 2023. Secrets of rlhf in large language models part i: Ppo. *arXiv:2307.04964* (2023).
- [51] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv:2205.10625* (2022).

## A TASK DEFINITION

In this section, we provide a detailed definition of each task in our GraphInstruct benchmark. For binary classification tasks, we filter the data to create a balanced set of graphs with both positive and negative labels.

- **Task 1: Cycle Detection.** In an undirected graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ , the task is to detect the existence of a cycle. A cycle can be defined as a sequence of vertices  $v_1, v_2, \dots, v_k$  with  $k \geq 3$ , that forms a closed loop, meaning  $v_1 = v_k$ . Additionally, for all  $1 \leq i < k$ , each vertex  $v_i$  must be distinct from the others, and there must be an edge connecting  $v_i$  to  $v_{i+1}$ .
- **Task 2: Connectivity.** Given an undirected graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ , the task is to assess if two randomly chosen nodes  $u$  and  $v$  are connected through a sequence of edges.
- **Task 3: Bipartite Graph Check.** This task is to determine if a directed graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$  is bipartite. A graph is considered bipartite if its nodes can be split into two distinct sets  $U$  and  $V$  such that no two nodes within the same set are adjacent.
- **Task 4: Topological Sort.** The task entails finding a valid topological sort of a directed graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ . In a topological sort, nodes are linearly ordered such that for each directed edge  $(u, v)$  from node  $u$  to node  $v$ , the node  $u$  is positioned before  $v$  in the sequence.
- **Task 5: Shortest Path.** The task requires identifying the shortest path between two nodes in an undirected, weighted graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, w\}$ , where  $w : \mathcal{E} \rightarrow \mathbb{R}^+$  assigns a positive weight to each edge. The goal is to find a path connecting the two nodes such that the total sum of the edge weights along this path is minimized.
- **Task 6: Maximum Triangle Sum.** Given an undirected, weighted graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, l\}$ , where  $l : \mathcal{V} \rightarrow \mathbb{R}^+$  is a function assigning a positive weight to each node, the task involves finding a triangle, a cycle of three connected vertices  $(v_1, v_2, v_3)$ , that maximizes the weight sum  $l(v_1) + l(v_2) + l(v_3)$ .
- **Task 7: Maximum Flow.** Consider a directed, weighted graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, c\}$ , where  $c : \mathcal{E} \rightarrow \mathbb{R}^+$  is a function assigning a positive capacity to each edge, representing the maximum flow

that the edge can support. Given a source node  $v_s$  and a sink node  $v_t$  in  $\mathcal{G}$ , the task is to devise a plan to maximize the flow from the source  $s$  to the sink  $t$ .

- **Task 8: Hamilton Path.** This task is to determine whether there is a Hamilton path in an undirected graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ . A Hamilton path is defined as a path that traverses each node in the graph exactly once.
- **Task 9: Substructure Matching.** Given two graphs,  $\mathcal{G}$  and  $\mathcal{G}'$ , where  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$  and  $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}'\}$ , the task is to determine whether there exists a subgraph within  $\mathcal{G}$  that is isomorphic to  $\mathcal{G}'$ .

## B ADDITIONAL DATASET INFORMATION

Table 7 shows the detailed statistics of our DPO training corpus. Table 6 shows the training and testing instruction of our model. Due to the space limit, we only keep all task prompts used for testing GPT-4 and ChatGPT in our GitHub Repository.

## C ADDITIONAL EXPERIMENTS

**Results on other graph types.** GraphWiz is primarily trained on Erdős-Rényi (ER) graphs due to their representation as general random graphs. To evaluate the generalization ability of GraphWiz, we tested it on the Barabási-Albert (BA) model, Path graph, Star graph, and Complete graph. Each graph type included 200 examples for each of the six tasks (Cycle, Connectivity, Bipartite, Hamilton, Shortest, and Flow). The results demonstrate that GraphWiz-Mistral-7B achieves an average accuracy of 68.12% on BA graphs, 54.48% on Path graphs, 63.17% on Star graphs, and 83.33% on Complete graphs, all of which outperform the 47.93% accuracy on ER graphs. These findings indicate that GraphWiz trained on ER graphs can effectively generalize to other graph types. Notably, GraphWiz achieves 100% accuracy on the Cycle, Connectivity, and Bipartite tasks for Complete graphs.

**Effect on the number of reasoning paths.** As shown in Table 5, Table 8 provides detailed results of our model’s performance on each task as the ratio of the graph to reasoning paths increases.

**Table 6: Training and testing prompts of GraphWiz in our experiments.**

<b>Input Prompts</b>	Below is an instruction that describes a task. \n Write a response that appropriately completes the request. \n \n ### Instruction: \n {Graph-Prompt Question} \n \n ### Response:
----------------------	---

**Table 7: Statistics of our DPO corpus, including total samples ( $\mathcal{G}\text{-}\mathcal{Q}$ ), nodes ( $\mathcal{V}$ ) and reasoning paths  $\mathcal{R}$ .**

Tasks	Easy				Medium			Hard		Sum.
	cycle	connect	bipartite	topology	shortest	triangle	flow	hamilton	Subgraph	
Total $\mathcal{G}\text{-}\mathcal{Q}\text{-}\mathcal{R}_w\text{-}\mathcal{R}_l$	982	972	689	490	551	545	405	962	570	6,166
Total $\mathcal{V}$	22,132	20,705	11,546	12,802	21,670	5,842	12,863	24,846	10,050	142,456

**Table 8: GraphWiz performances with increasing ratios of the graph to reasoning path ( $\mathcal{G}\text{:}\mathcal{R}$ ), which is indicative of the data volume and diversity of reasoning paths available for training.  $\mathcal{G}\text{:}\mathcal{R}=1:5$  means the whole corpus of GraphInstruct.**

$\mathcal{G}:\mathcal{R}$	Easy				Medium			Hard		Avg.
	cycle	connect	bipartite	topology	shortest	triangle	flow	hamilton	substructure	
GraphWiz (Mistral-7B)										
1:1	80.25	87.50	78.50	1.00	23.00	47.00	28.75	31.75	41.25	46.56
1:2	78.25	85.00	81.75	27.00	28.00	47.00	24.50	34.25	42.75	49.83
1:3	90.75	88.25	62.75	21.75	30.75	39.75	32.25	30.75	84.25	53.47
1:4	90.25	87.00	78.00	19.75	26.00	43.25	26.50	26.50	85.00	53.58
1:5	92.00	89.50	72.00	19.00	31.25	38.75	29.25	26.50	85.50	53.75
GraphWiz (LLaMA 2-7B)										
1:1	77.75	84.00	71.75	1.00	18.50	46.25	22.00	31.25	45.00	44.17
1:2	94.00	85.25	76.25	17.25	23.50	35.75	19.50	40.50	83.00	53.00
1:3	93.00	89.00	72.50	18.25	24.75	40.50	22.50	45.00	83.25	54.00
1:4	92.75	86.00	77.75	21.75	23.25	41.50	25.25	40.25	80.75	54.36
1:5	91.50	87.00	74.00	18.00	28.00	38.25	24.50	52.25	82.25	55.08

**Table 9: Performance of GraphWiz-Mistral-7B on different graph types.**

	Cycle	Connectivity	Bipartite	Hamilton	Shortest	Flow
ER	92	89.5	19	26.5	31.3	29.3
BA	99	92	68.5	51.2	59	39
Path	55	90	34.3	77.6	37	33
Star	73	100	21.2	44.3	73.5	67
Complete	100	100	100	73.9	94	32.1