# G1: Teaching LLMs to Reason on Graphs with Reinforcement Learning

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Although Large Language Models (LLMs) have demonstrated remarkable progress, their proficiency in graph-related tasks remains notably limited, hindering the development of truly general-purpose models. Previous attempts, including pretraining graph foundation models or employing supervised fine-tuning, often face challenges such as the scarcity of large-scale, universally represented graph data. We introduce G1, a simple yet effective approach demonstrating that Reinforcement Learning (RL) on synthetic graph-theoretic tasks can significantly scale LLMs' graph reasoning abilities. To enable RL training, we curate Erdős, the largest graph reasoning dataset to date comprising 50 diverse graph-theoretic tasks of varying difficulty levels, 100k training data and 5k test data, all drived from real-world graphs. With RL on Erdős, G1 obtains substantial improvements in graph reasoning, where our finetuned 3B model even outperforms Qwen2.5-72B-Instruct (24x size). RL-trained models also show strong zero-shot generalization to unseen tasks, domains, and graph encoding schemes, including other graph-theoretic benchmarks as well as real-world node classification and link prediction tasks, without compromising general reasoning abilities. Our findings offer an efficient, scalable path for building strong graph reasoners by finetuning LLMs with RL on graph-theoretic tasks, which combines the strengths of pretrained LLM capabilities with abundant, automatically generated synthetic data, suggesting that LLMs possess graph understanding abilities that RL can elicit successfully.

## 1 Introduction

Large Language Models (LLMs) have achieved widespread success [3, 13] but exhibit notable limitations in reasoning about graph-structured data, a critical capability for achieving general-purpose intelligence. Proficient graph reasoning is essential for numerous applications, yet even state-of-the-art LLMs like OpenAI's o1 [30] demonstrate significant deficiencies, with reported accuracies as low as 58.49% on graph connectivity tests [55].

Initial efforts to enhance LLMs' graph understanding explored various natural language encoding schemes [12, 6, 10], but these yielded only modest improvements. Alternative strategies have involved instruction tuning [27, 53] or preference tuning [4, 42] on curated graph datasets. Others attempted to build specialized graph foundation models through pretraining [28, 21, 26]; however, these are often limited by the lack of large-scale, universal graph representations suitable for diverse graph types. Different from prior work, we believe LLMs pretrained on Internet-scale data already possess graph reasoning ability, and we can elicit it through their own trial and error without human data.

In this work, we are the first to explore the use of Reinforcement Learning (RL) to solve graph reasoning tasks. We chose graph-theoretic problems as a testbed as they allow direct verification of generated answers to produce rule-based rewards for RL training, which is shown to be key for the success of DeepSeek R1 in math and coding problems [13]. We collect the largest-to-date

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Table 1: An overview of 50 graph-theoretic tasks in our dataset Erdős (100k train, 5k test), alongside with the difficulty distribution, and the accuracy of the base model Qwen2.5-7B-Instruct and our RL-trained G1-7B model. A complete description of tasks are in Appendix O.2.

| Difficulty | Tasks | Ratio | Base Model Acc | G1 Acc |
|---|---|---|---|---|
| **Easy** | Node Number, Dominating Set, Common Neighbor, Edge Number, Neighbor, BFS, Has Cycle, DFS, Minimum Spanning Tree, Edge Existence, Is Regular, Degree, Is Tournament, Density | 29.16% | 57.16% | **95.07%** |
| **Medium** | Adamic Adar Index, Clustering Coefficient, Connected Component Number, Bipartite Maximum Matching, Local Connectivity, Jaccard Coefficient, Min Edge Covering, Is Eularian, Degree Centrality, Is Bipartite, Resource Allocation Index | 22.91% | 42.55% | **88.91%** |
| **Hard** | Max Weight Matching, Closeness Centrality, Traveling Salesman Problem, Strongly Connected Number, Shortest Path, Center, Diameter, Barycenter, Radius, Topological Sort, Periphery, Betweenness Centrality, Triangles, Average Neighbor Degree, Harmonic Centrality, Bridges | 33.33% | 18.87% | **50.44%** |
| **Challenging** | Isomophic Mapping, Global Efficiency, Maximal Independent Set, Maximum Flow, Wiener Index, Hamiltonian Path, Min Vertex Cover | 14.58% | 3.29% | **23.57%** |

graph-theoretic problem set, Erdős, spanning a wide spectrum of difficulty levels, as shown in Table O.2. Through RL training, our model G1 achieves substantial performance improvements on the Erdős benchmark, with gains of up to 46% over baseline models. Notably, our G1-7B model attains competitive performance with state-of-the-art reasoning models like OpenAI's o3-mini, and G1-3B easily rivals Qwen2.5-72B-Instruct by noticeable margins. Notably, texttt G1 models exhibit strong zero-shot generalization on unseen graph tasks and domains, improving base models' performance on other graph-theoretic benchmarks and real-world graphs without deteriorating general reasoning ability, indicating a synergetic improvement of LLMs' graph reasoning abilities through RL. We also study various aspects of the training process, such as the influence of data mixture, supervised initialization, and the use of chain-of-thought [45].

G1 charts a data-efficient and scalable course for developing LLMs with strong graph reasoning. By demonstrating that RL can unlock latent graph understanding within general-purpose LLMs using synthetic data, our work suggests a possible paradigm shift away from reliance on heterogeneous real-world graphs to build graph foundation models. This paves the way for more versatile AI systems capable of sophisticated reasoning across diverse data modalities.

## 2 Erdős: A Comprehensive Collection of Graph-theoretic Reasoning Tasks on Real-world Graphs

To facilitate rule-based Reinforcement Learning of LLMs (aka. Reinforcement Learning from Verifiable Rewards (RLVR)) on graphs, we construct a diverse, large-scale collection of graph-theoretic reasoning tasks. We name it Erdős to remember Paul Erdős, a seminal figure with diverse contributions to graph theory. Compared to real-world graph tasks, these graph-theoretic tasks allow clear rule-based determination of rewards for the answers sampled from LLMs. We categorize these tasks into **Easy, Medium, Hard, and Challenging**, based on their inherent problem complexity as well as current LLMs' ability to solve them (see a full list in Table 1). For the training split, there are a total of 100,000 question-answer pairs, evenly distributed across tasks with 2,000 examples each. We also reserve 5,000 test pairs with different questions for evaluation. We include a detailed comparison of Erdős with other graph reasoning benchmarks in Appendix O.1. Erdős can serve as a dataset for training LLMs as well as a benchmark for evaluating LLMs on graph-theoretic tasks.

Specifically, we curate 50 graph-theoretic reasoning tasks available on NetworkX [14], one of the most widely used libraries for graph processing, and construct, as we know, the most comprehensive collection so far. The tasks vary in difficulty level and cover a wide range of answer types, including boolean, integer, float, node list, edge list, and node mapping. For *answer generation.*, we utilize the default solvers of NetworkX to automatically solve the problem. If there are multiple solutions to each question, we use NetworkX-based programs to verify the correctness, avoiding both costly human labeling and potential bias of LLM judges. For *graph sources.*, we utilize the real-world graphs from the Network Repository [33], the largest network repository with thousands of donations

in 30+ domains. We downsample the graphs by random walk with a restart strategy, generating subgraphs with sizes from 5 to 35 nodes, following common settings in previous work [41, 55, 37]. For *language encoding*, we choose to describe the graph structure in a unified edge list format, *e.g.*, $(1, 2), (2, 3), \ldots$. In later experiments of Section 4.2, we show that our model trained on a single graph description method can even positively transfer to other formats. Details about benchmark construction are provided in Appendix B.

# 3 Training LLMs to Reason on Graphs

In this section, we introduce the training pipeline that we explored for training `G1`. We design proper rule-based rewards for different graph tasks, while intentionally keeping the RL algorithm general and consistent with previous work. Similar to DeepSeek R1 [13], the training of `G1` is very simple: it consists of a Reinforcement Learning phase for rewarding correct rollouts with the GRPO algorithm [36], and an *optional* SFT phase for warming up the model in the beginning (without which we call `G1`-Zero). We find that the SFT phase is generally beneficial for learning more challenging tasks, whose initial accuracy with the base model is close to zero.

**RL Algorithm.** Following common practice [13], we use the Group Relative Policy Optimization (GRPO) [36] algorithm for RL training. We design the following rule-based outcome reward model (ORM) for our training on graph-theoretic tasks, with a combination of value match, set matching, and algorithmic verification for different problems:

- *Strict value matching*. For tasks that have a unique ground truth value, e.g., node counting, the policy receives a reward of +1 only when the generated answer is identical to the ground truth in terms of numerical value, e.g., 0.5 and 1/2, otherwise it receives a reward of 0.
- *Jaccard Index for set matching.* For problems whose answer is not a single value $\hat{s}$ but an unordered set, e.g., common neighbors of two nodes, the reward is defined as the Jaccard Index between the generated set $\hat{s}$ and the ground truth $s$, i.e., $|s \cap \hat{s}|/|s \cup \hat{s}|$. In this way, the model can receive intermediate rewards for imperfect solutions.
- *Algorithmic verification.* Lastly, for problems that have multiple correct solutions (e.g., shortest paths) and it is not feasible to enumerate all of them, we implement algorithmic verifiers to check the correctness of the proposed solutions. For instance, we determine the validness of a Hamiltonian path proposed by the policy by checking whether all the edges in the path exist and each node is visited exactly once.

**Optional Warm-up with Supervised Fine-tuning.** During RL training, we have noticed that for some challenging tasks like isomorphic mapping (see Table 1), the initial accuracy of the base model is often so low that we frequently end up with only incorrect rollouts, producing no useful signal for RL training. We find that introducing a short warm-up phase with supervised fine-tuning effectively improves overall learning efficiency. Specifically, we consider two types of supervised fine-tuning: (1). *Direct-SFT*: the first is direct supervised fine-tuning on question-answer pairs $(q, a)$, where $q$ is the textual description of the problem and $a$ is the final answer without any intermediate reasoning steps. (2). *CoT-SFT*: we collect reasoning trajectories via sampling $(q, c, a)$ triplets from another model [54], where $c$ represents the Chain-of-Thought (CoT) reasoning steps in natural language that lead to the final answer $a$, and use them to fine-tune the base model.

# 4 Experiments

## 4.1 Benchmarking `G1` on Graph-theoretic Reasoning Tasks

**Setup.** As shown in Table 2, in the interest of academic compute budgets, we focus on comparing relatively small models. We include strong proprietary models (of unknown sizes) like GPT-4o-mini (non-reasoning) and OpenAI o3-mini (state-of-the-art reasoning), open-source instruction models like Qwen2.5-Instruct series (3B, 7B, 72B) [38], Qwen2.5-Math-Instruct [52], LLaMA-3 series (3B, 8B, 70B) [1], and a strong baseline DeepSeek-R1-Distill-Qwen-7B [13] that is distilled from DeepSeek R1 with 671B parameters. Additionally, for reference, we incorporate previous training strategies for graph reasoning tasks such as GraphWiz-RFT and GraphWiz-DPO [4]. We finetune our model from Qwen2.5-Instruct models (3B and 7B) for 300 steps with batch size 512 on a cluster of 8$\times$A800 GPUs, using our dataset `Erdős`. More experimental details can be found in Appendix E.

Table 2: Test accuracy (%) comparison of different LLMs of varying sizes on our Erdős benchmark tasks. In all experiments we use Qwen2.5-Instruct models as our base model (marked below). We report the average accuracy across all tasks in the *Average* column, and full results for each task are provided in Appendix M.4.

| **Model** | Easy | Medium | Hard | Challenging | **Average** |
|---|---|---|---|---|---|
| Proprietary (Unknown Parameters) | | | | | |
| GPT-4o-mini | 76.20 | 72.07 | 28.81 | 3.34 | 47.60 |
| OpenAI o3-mini (w/ tool use) | 74.83 | 83.49 | 59.28 | 43.22 | 64.90 |
| 3B Parameters | | | | | |
| Llama-3.2-3B-Instruct | 36.50 | 21.45 | 6.81 | 1.14 | 17.32 |
| Qwen2.5-3B-Instruct (base model) | 45.71 | 30.18 | 9.44 | 1.29 | 22.72 |
| Direct-SFT-3B (Ours) | 74.43 | 75.27 | **43.69** | **14.43** | 53.78 |
| CoT-SFT-3B (Ours) | 65.57 | 67.64 | 29.44 | 4.57 | 43.56 |
| **G1-3B (Ours)** | **94.86** | **84.64** | 41.25 | 7.57 | **59.76** (+37.04) |
| 7B Parameters | | | | | |
| Llama-3.1-8B-Instruct | 49.21 | 30.45 | 13.69 | 1.43 | 25.10 |
| Qwen2.5-7B-Instruct (base model) | 57.36 | 42.55 | 18.87 | 3.29 | 32.06 |
| Qwen2.5-Math-7B-Instruct | 52.79 | 39.64 | 14.82 | 2.46 | 28.94 |
| DeepSeek-R1-Distill-Qwen-7B | 71.79 | 73.73 | 39.12 | 16.57 | 51.64 |
| GraphWiz-7B-RFT | 14.57 | 13.73 | 1.38 | 0.47 | 7.70 |
| GraphWiz-7B-DPO | 20.36 | 19.09 | 1.44 | 0.78 | 10.59 |
| Direct-SFT-7B (Ours) | 73.57 | 75.91 | 39.12 | 10.71 | 51.76 |
| CoT-SFT-7B (Ours) | 72.57 | 75.73 | 38.50 | 11.00 | 51.34 |
| **G1-7B (Ours)** | **95.07** | **88.91** | **50.44** | **23.57** | **66.16** (+34.10) |
| 70B Parameters | | | | | |
| Llama-3.1-70B-Instruct | 68.07 | 55.45 | 31.87 | 4.44 | 42.28 |
| Qwen2.5-72B-Instruct | 71.71 | 67.81 | 33.37 | 8.22 | 47.16 |

**Performance.** As shown in Table 2, our proposed model G1-7B consistently outperforms most proprietary, open-source, and graph training counterparts by significant margins across all difficulty levels. With a notable average accuracy of 66.16%, G1-7B outperforms GPT-4o-mini (47.60%) by 18.56%, reaching competitive performance to a cutting-edge reasoning model like o3-mini (64.90%) that underwent much heavier training. Notably, our small variant G1-3B, delivers a strong average performance of 59.76%, surpassing open-source models including Qwen2.5-72B-Instruct (47.16%) and Llama-3.1-70B-Instruct (42.28%) with 20× parameters. We also evaluate the GPT-4o model on a randomly sampled subset of Erdős due to the cost budget. As shown in Appendix Table 13, G1-7B surpasses GPT-4o across all difficulty levels, exceeding it by over 10% on average (65.29% vs. 55.13%), further validating the strong graph reasoning capabilities of the G1 models. In Appendix K, we provide a robustness test where G1 variants demonstrate consistently small standard deviations among multiple runs and phrasing changes. In Appendix C, we further analyze the influence of *data mixture* and *SFT warmup* on G1's reasoning performance.

**Scaling G1 to 32B.** To demonstrate the scalability of our approach, we extended the training methodology to develop G1-Zero-32B from Qwen2.5-32B-Instruct. As shown in Table 14, G1-Zero-32B achieves a **27.96%** improvement in average accuracy (from 47.10% to 75.06%), with particularly notable gains in harder categories: +31.87% on Hard problems and +26.43% on Challenging problems. Furthermore, Appendix Table 15 demonstrates that G1-Zero-32B not only preserves but slightly enhances mathematical performance on standard benchmarks, which shows modest improvements on both GSM8K (+0.08%) and MATH (+4.00%).

**Scaling G1 to larger graphs.** To verify the transferability of G1 to larger graphs, we construct a new test set of 5,000 graphs with 36-100 nodes, with other settings kept the same, which ensures there is no overlap between training and test data. Appendix Table 11 shows that both G1-3B and G1-7B achieve strong zero-shot generalization to these larger graphs without additional training, significantly outperforming the baselines across difficulty levels. These results demonstrate our method's effective scalability beyond the original training distribution. For larger graphs with thousands of nodes, G1 is bottlenecked by the context window limit of underlying LLMs, detailed in Appendix H.

Table 3: Test accuracy (%) of G1-Zero-32B and Qwen2.5-32B-Instruct on Erdős.

| | Easy | Medium | Hard | Challenging | Average |
|---|---|---|---|---|---|
| Qwen2.5-32B-Instruct | 70.57 | 68.73 | 33.38 | 9.00 | 47.10 |
| **G1-Zero-32B** | **97.79** | **93.00** | **65.25** | **35.43** | **75.06** |

Table 4: Test accuracy (%) by computational complexity on the GraphWiz benchmark.

| Model | Linear | Poly | NP-Complete | Avg. |
|---|---|---|---|---|
| Llama-3.2-3B-Instruct | 29.80 | 3.00 | 2.50 | 19.80 |
| Qwen2.5-3B-Instruct (base) | 40.25 | 9.58 | **69.12** | 36.44 |
| **G1-3B** | **58.06** | **26.75** | **69.12** | **50.08** |
| Llama-3.1-8B-Instruct | 54.00 | 5.67 | 32.12 | 33.03 |
| DeepSeek-R1-Distill-Qwen-7B | 57.69 | 31.42 | 70.88 | 51.86 |
| GraphWiz-7B-RFT | 67.56 | 29.83 | 43.38 | 49.61 |
| GraphWiz-7B-DPO | 63.88 | **36.25** | 39.50 | 49.25 |
| Qwen2.5-7B-Instruct (base) | 49.06 | 17.92 | **76.12** | 44.69 |
| **G1-7B** | **68.00** | 32.25 | 72.62 | **57.11** |

Table 5: Test accuracy (%) by computational complexity on the GraphArena benchmark.

| Model | Poly-Time | | NP-Complete | | Avg. |
|---|---|---|---|---|---|
| | Easy | Hard | Easy | Hard | |
| Llama-3.2-3B-Instruct | 22.25 | 6.75 | 8.00 | 0.66 | 8.40 |
| Qwen2.5-3B-Instruct (base) | 31.50 | 14.50 | 17.33 | 1.50 | 14.85 |
| **G1-3B** | **57.50** | **26.75** | **24.66** | **1.83** | **24.80** |
| Llama-3.1-8B-Instruct | 47.00 | 21.25 | 22.00 | 2.16 | 20.90 |
| DeepSeek-R1-Distill-Qwen-7B | 66.0 | 22.75 | 34.83 | 1.50 | 28.65 |
| GraphWiz-7B-RFT | 2.25 | 0.75 | 0.83 | 0.00 | 0.85 |
| GraphWiz-7B-DPO | 0.25 | 1.00 | 0.66 | 0.16 | 0.49 |
| Qwen2.5-7B-Instruct (base) | 62.00 | 35.75 | 28.83 | 2.16 | 28.84 |
| **G1-7B** | **77.50** | **44.25** | **47.33** | **8.50** | **41.10** |

Table 6: Test accuracy (%) on Node Classification and Link Prediction benchmarks.

| Model | Node | | Link | | Avg. |
|---|---|---|---|---|---|
| | Cora | PubMed | Cora | PubMed | |
| Llama-3.2-3B-Instruct | 68.77 | 75.20 | 60.40 | 57.60 | 64.79 |
| Qwen2.5-3B-Instruct (base) | 70.83 | 75.08 | 62.15 | 58.38 | 65.66 |
| CoT-SFT-3B | 75.97 | 81.47 | 75.70 | **71.52** | 75.12 |
| **G1-3B** | **77.25** | **83.88** | **78.97** | 69.75 | **75.16** |
| Llama-3.1-8B-Instruct | 70.90 | 75.00 | 50.60 | 46.10 | 59.53 |
| DeepSeek-R1-Distill-Qwen-7B | 76.50 | 81.25 | 68.03 | 78.72 | 78.80 |
| Qwen2.5-7B-Instruct (base) | **79.30** | 85.35 | **88.22** | 88.67 | 85.50 |
| CoT-SFT-7B | 73.20 | 83.25 | 64.70 | 68.12 | 73.17 |
| **G1-7B** | 79.20 | **86.20** | 87.98 | **91.88** | **87.29** |

Table 7: Test accuracy (%) on reasoning benchmarks beyond graph-related tasks.

| Model | GSM8K | MATH | MMLU-pro |
|---|---|---|---|
| Llama-3.2-3B-Instruct | 71.03 | 42.40 | 13.50 |
| Qwen2.5-3B-Instruct (base) | **81.95** | **62.20** | **38.53** |
| CoT-SFT-3B | 75.36 | 56.00 | 34.85 |
| **G1-3B** | 79.30 | 61.80 | 37.11 |
| Llama-3.1-8B-Instruct | 74.45 | 44.80 | 32.02 |
| DeepSeek-R1-Distill-Qwen-7B | 86.03 | **87.20** | 37.21 |
| Qwen2.5-7B-Instruct (base) | 86.27 | 69.80 | 45.75 |
| CoT-SFT-7B | 83.85 | 65.80 | 44.79 |
| **G1-7B** | **87.49** | 71.80 | **48.56** |

### 4.2 Transferability of G1 to Unseen Tasks and Domains

In this section, we evaluate *zero-shot* generalization of G1 to unseen domains, tasks, and data formats. Detailed benchmark description and complete evaluation setups are provided in Appendix F.

#### 4.2.1 G1's Transferability to Other Graph Reasoning Benchmarks

**Setup**. We consider two additional graph reasoning benchmarks, *GraphWiz* [4] and *GraphArena* [37], which bring three major shifts that challenge our model: 1) different distributions of the underlying graphs 2) tasks unseen during training 3) unfamiliar graph encoding formats, e.g., the GraphArena benchmark represents nodes with human names instead of integers.

**Results.** The performance across models is reported in Table 4 and Table 5. On the GraphWiz benchmark, G1-7B achieves the highest overall accuracy (57.11%) among all models, outperforming DeepSeek-R1-Distill-Qwen-7B (51.86%) and even models specifically trained on GraphWiz data such as GraphWiz-7B-RFT (49.61%). The smaller variant G1-3B also achieves comparable performance with DeepSeek-R1-Distill-Qwen-7B. Similar results can be found on the GraphArena benchmark (Table 5) with a different graph encoding scheme. These results demonstrate that G1 has strong zero-shot generalization ability to unseen graph encoding methods, graph distributions, and graph tasks. Full results for GraphWiz and GraphArena are shown in Appendix M.1 and Appendix M.3.

#### 4.2.2 G1 on Real-world, Non-graph-theoretic Graph-reasoning Tasks

**Baseline.** For real-world graph tasks, we consider two standard problems: node classification and link prediction. We adopt the benchmarks introduced by Wang et al. [44], which are constructed by subsampling from the widely used Cora and PubMed citation graphs. Each instance includes a description of the target node (or node pair) containing the paper ID and title, along with the textual and structural information of neighboring nodes. These benchmarks emphasize the model's ability to leverage both local graph structure and textual attributes for effective prediction.

**Results.** As shown in Table 6, our model G1 significantly outperforms both open-source and distilled baselines across tasks and model sizes. In the 3B model category, G1-3B surpasses the base model (Qwen2.5-3B-Instruct) by a large margin—especially in link prediction on Cora (+16.82%) and node

classification on PubMed (+8.8%). In the 7B model category, G1-7B achieves the highest average score of 87.29%, ranking first on PubMed dataset in both node classification and link prediction tasks. Overall, G1 consistently demonstrates strong generalization across real-world graph tasks where graph-text reasoning is required.

#### 4.2.3 G1's Reasoning Ability beyond Graphs

**Setup.** We next extend our investigations of G1's abilities beyond graph-based tasks. We consider two mathematics benchmarks, GSM8K [8] and MATH [16]. Additionally, we include MMLU-Pro [43], which is a massive multi-task benchmark covering disciplines such as chemistry, economics, and computer science. We believe the three benchmarks collectively provide a comprehensive assessment of G1's reasoning capabilities.

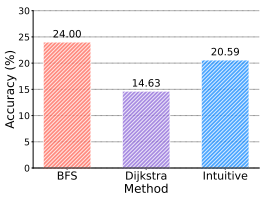
**Results.** In table 7, we first notice that the CoT-SFT training on graph reasoning trajectories leads to a non-negligible degradation in general abilities, which could be attributed to the fact that SFT *memorizes* pattern instead of incentivizing truly generalizable skills [5]. Remarkably, the subsequent reinforcement learning stage—despite being trained exclusively on graph tasks—restores the reasoning abilities of both the 3B and the 7B model. G1-7B even surpasses the performance of the initial Qwen-7B checkpoint in all of the three benchmarks (87.49% v.s. 86.27% for GSM8K, 72.8% v.s. 69.8% for MATH, and 48.56% v.s. 45.75% for MMLU-pro). Interestingly, G1-7B also outperforms Qwen-7B-Instruct on several non-STEM tasks like Economy (68.76 v.s. 46.87), which are intuitively less related to graph reasoning (see Appendix M.2 for full MMLU-Pro results). We further provide a detailed analysis on the transferability of G1 to mathematics tasks in Appendix L, showing G1 mainly improves the numerical calculation and utilization of known information.

### 4.3 Understanding the Benefits of RL Training for Graph Reasoning

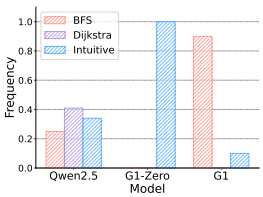
To understand how RL training helps graph reasoning, we take *shortest path* (a Hard task) as a case study. Specifically, we study the behaviors of three models: Qwen2.5-3B-Instruct (base), G1-Zero-3B (RL only), and G1-3B (SFT & RL).

We identify three primary approaches adopted by the models to solve the problem: 1) Breadth-First Search (BFS), 2) Dijkstra's algorithm, and 3) Intuitive deductions. Figure 1a shows the distribution of these approaches alongside their corresponding accuracies for Qwen2.5-3B-Instruct. On *unweighted* graphs, BFS is the most efficient method and yields the highest performance. In contrast, Dijkstra's algorithm is best suited for *weighted* graphs, where it correctly accounts for edge costs. However, its reliance on a min-priority queue and a distance list introduces computational complexity, which appears to challenge Qwen2.5-3B-Instruct and results in its lowest observed accuracy. For example, as shown in Appendix Figure 3 (left), the model falsely states that node 4 has no edges (node 4 is connected to node 7) while updating the distance list. Interestingly, intuitive approaches—where the model attempts to visually estimate or heuristically trace paths—can also produce correct answers by a noticeable accuracy, particularly on small graphs.

We proceed by observing that RL training significantly reshapes the models' graph reasoning strategies: RL-trained models largely abandon Dijkstra and prefer a combination of BFS and intuitive search. As shown in Figure 1b and Appendix Figure 3 (middle), G1-Zero-3B navigates the graph in a manner akin to human heuristics—sequentially checking neighbors and adjusting paths dynamically. G1-3B primarily adopts a neat BFS-style algorithm as in Figure 1b and Appendix Figure 3 (right), executing it with high precision, occasionally resorting to intuitive strategies for simple graphs. To conclude, our case study highlights how RL training enhances graph reasoning by guiding LLMs toward more model-aware strategies that are adaptive to their inherent capabilities [47].




(a) The accuracy of different graph reasoning patterns for shortest path on Qwen2.5-3B-Instruct.




(b) Frequency of different graph reasoning patterns for Qwen2.5-3B-Instruct, G1-Zero-3B and G1-3B.

Figure 1: Reasoning patterns for the shortest path task.

# References


[1] AI, M. Llama3 foundation models. https://www.llama.com/models/llama-3/, 2024.

[2] Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *science*, 286(5439): 509–512, 1999.

[3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020.

[4] Chen, N., Li, Y., Tang, J., and Li, J. Graphwiz: An instruction-following language model for graph computational problems. In *SIGKDD*, 2024.

[5] Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., and Ma, Y. Sft memorizes, rl generalizes: A comparative study of foundation model post-training, 2025. URL https://arxiv.org/abs/2501.17161.

[6] Chu, X., Xue, H., Tan, Z., Wang, B., Mo, T., and Li, W. Graphsos: Graph sampling and order selection to help llms understand graphs better. *arXiv e-prints*, pp. arXiv–2501, 2025.

[7] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[8] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[9] Dai, X., Qu, H., Shen, Y., Zhang, B., Wen, Q., Fan, W., Li, D., Tang, J., and Shan, C. How do large language models understand graph patterns? a benchmark for graph pattern comprehension. In *ICLR*, 2025.

[10] Das, D., Gupta, I., Srivastava, J., and Kang, D. Which modality should i use–text, motif, or image?: Understanding graphs with large language models. In *NAACL*, 2024.

[11] Erdös, P. Erdös-rényi model. *Publ. Math. Debrecen*, pp. 290–297, 1959.

[12] Fatemi, B., Halcrow, J., and Perozzi, B. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*, 2023.

[13] Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[14] Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, 2008.

[15] Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[16] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. In *NeurIPS*, 2021.

[17] Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[18] Huang, X., Zhang, J., Li, D., and Li, P. Knowledge graph embedding based question answering. In *WSDM*, 2019.

[19] Karalias, N. and Loukas, A. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In *NeurIPS*, 2020.

[20] Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[21] Kong, L., Feng, J., Liu, H., Huang, C., Huang, J., Chen, Y., and Zhang, M. Gofa: A generative one-for-all model for joint graph language modeling. In *ICLR*, 2025.

[22] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *SIGOPS*, 2023.

[23] Li, X., Chen, W., Chu, Q., Li, H., Sun, Z., Li, R., Qian, C., Wei, Y., Shi, C., Liu, Z., et al. Can large language models analyze graphs like professionals? a benchmark, datasets and models. In *NeurIPS*, 2024.

[24] Li, Y., Pan, Z., Lin, H., Sun, M., He, C., and Wu, L. Can one domain help others? a data-centric study on multi-domain reasoning via reinforcement learning. *arXiv preprint arXiv:2507.17512*, 2025.

[25] Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[26] Liu, H., Feng, J., Kong, L., Liang, N., Tao, D., Chen, Y., and Zhang, M. One for all: Towards training one graph model for all classification tasks. In *ICLR*, 2024.

[27] Luo, Z., Song, X., Huang, H., Lian, J., Zhang, C., Jiang, J., and Xie, X. Graphinstruct: Empowering large language models with graph understanding and reasoning capability. *arXiv preprint arXiv:2403.04483*, 2024.

[28] Mao, H., Chen, Z., Tang, W., Zhao, J., Ma, Y., Zhao, T., Shah, N., Galkin, M., and Tang, J. Position: Graph foundation models are already here. In *ICML*, 2024.

[29] Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nova, A., et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[30] OpenAI, :, Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., Iftimie, A., Karpenko, A., Passos, A. T., Neitz, A., Prokofiev, A., Wei, A., Tam, A., Bennett, A., Kumar, A., Saraiva, A., Vallone, A., Duberstein, A., Kondrich, A., Mishchenko, A., Applebaum, A., Jiang, A., Nair, A., Zoph, B., Ghorbani, B., Rossen, B., Sokolowsky, B., Barak, B., McGrew, B., Minaiev, B., Hao, B., Baker, B., Houghton, B., McKinzie, B., Eastman, B., Lugaresi, C., Bassin, C., Hudson, C., Li, C. M., de Bourcy, C., Voss, C., Shen, C., Zhang, C., Koch, C., Orsinger, C., Hesse, C., Fischer, C., Chan, C., Roberts, D., Kappler, D., Levy, D., Selsam, D., Dohan, D., Farhi, D., Mely, D., Robinson, D., Tsipras, D., Li, D., Oprica, D., Freeman, E., Zhang, E., Wong, E., Proehl, E., Cheung, E., Mitchell, E., Wallace, E., Ritter, E., Mays, E., Wang, F., Such, F. P., Raso, F., Leoni, F., Tsimpourlas, F., Song, F., von Lohmann, F., Sulit, F., Salmon, G., Parascandolo, G., Chabot, G., Zhao, G., Brockman, G., Leclerc, G., Salman, H., Bao, H., Sheng, H., Andrin, H., Bagherinezhad, H., Ren, H., Lightman, H., Chung, H. W., Kivlichan, I., O'Connell, I., Osband, I., Gilaberte, I. C., Akkaya, I., Kostrikov, I., Sutskever, I., Kofman, I., Pachocki, J., Lennon, J., Wei, J., Harb, J., Twore, J., Feng, J., Yu, J., Weng, J., Tang, J., Yu, J., Candela, J. Q., Palermo, J., Parish, J., Heidecke, J., Hallman, J., Rizzo, J., Gordon, J., Uesato, J., Ward, J., Huizinga, J., Wang, J., Chen, K., Xiao, K., Singhal, K., Nguyen, K., Cobbe, K., Shi, K., Wood, K., Rimbach, K., Gu-Lemberg, K., Liu, K., Lu, K., Stone, K., Yu, K., Ahmad, L., Yang, L., Liu, L., Maksin, L., Ho, L., Fedus, L., Weng, L., Li, L., McCallum, L., Held, L., Kuhn, L., Kondraciuk, L., Kaiser, L., Metz, L., Boyd, M., Trebacz, M., Joglekar, M., Chen, M., Tintor, M., Meyer, M., Jones, M., Kaufer, M., Schwarzer, M., Shah, M., Yatbaz, M., Guan, M. Y., Xu, M., Yan, M., Glaese, M., Chen, M., Lampe, M., Malek, M., Wang, M., Fradin, M., McClay, M., Pavlov, M., Wang, M., Wang, M., Murati, M., Bavarian, M., Rohaninejad, M., McAleese, N., Chowdhury, N., Chowdhury, N., Ryder, N., Tezak, N., Brown, N., Nachum, O., Boiko, O., Murk, O., Watkins, O., Chao, P., Ashbourne, P., Izmailov, P., Zhokhov, P., Dias, R., Arora, R., Lin, R., Lopes, R. G., Gaon, R., Miyara, R., Leike, R., Hwang, R., Garg, R., Brown, R., James, R., Shu, R., Cheu, R., Greene, R., Jain, S., Altman, S., Toizer, S., Toyer, S., Miserendino, S., Agarwal, S., Hernandez, S., Baker, S., McKinney, S., Yan, S., Zhao, S., Hu, S., Santurkar, S., Chaudhuri, S. R., Zhang,

S., Fu, S., Papay, S., Lin, S., Balaji, S., Sanjeev, S., Sidor, S., Broda, T., Clark, A., Wang, T., Gordon, T., Sanders, T., Patwardhan, T., Sottiaux, T., Degry, T., Dimson, T., Zheng, T., Garipov, T., Stasi, T., Bansal, T., Creech, T., Peterson, T., Eloundou, T., Qi, V., Kosaraju, V., Monaco, V., Pong, V., Fomenko, V., Zheng, W., Zhou, W., McCabe, W., Zaremba, W., Dubois, Y., Lu, Y., Chen, Y., Cha, Y., Bai, Y., He, Y., Zhang, Y., Wang, Y., Shao, Z., and Li, Z. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

[31] Perozzi, B., Fatemi, B., Zelle, D., Tsitsulin, A., Kazemi, M., Al-Rfou, R., and Halcrow, J. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*, 2024.

[32] Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025.

[33] Rossi, R. A. and Ahmed, N. K. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL http://networkrepository.com.

[34] Sanford, C., Fatemi, B., Hall, E., Tsitsulin, A., Kazemi, M., Halcrow, J., Perozzi, B., and Mirrokni, V. Understanding transformer reasoning capabilities via graph algorithms. In *NeurIPS*, 2024.

[35] Sato, R., Yamada, M., and Kashima, H. Approximation ratios of graph neural networks for combinatorial problems. In *NeurIPS*, 2019.

[36] Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[37] Tang, J., Zhang, Q., Li, Y., and Li, J. Grapharena: Benchmarking large language models on graph computational problems. In *ICLR*, 2025.

[38] Team, Q. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

[39] Veličković, P., Ying, R., Padovano, M., Hadsell, R., and Blundell, C. Neural execution of graph algorithms. In *ICLR*, 2020.

[40] Wang, H., Wang, K., Yang, J., Shen, L., Sun, N., Lee, H.-S., and Han, S. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2020.

[41] Wang, H., Feng, S., He, T., Tan, Z., Han, X., and Tsvetkov, Y. Can language models solve graph problems in natural language? In *NeurIPS*, 2023.

[42] Wang, J., Wu, J., Hou, Y., Liu, Y., Gao, M., and McAuley, J. Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment. In *ACL*, 2024.

[43] Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., and Chen, W. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

[44] Wang, Y., Dai, X., Fan, W., and Ma, Y. Exploring graph tasks with pure llms: A comprehensive benchmark and investigation. *arXiv preprint arXiv:2502.18771*, 2025.

[45] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[46] Wu, Q., Chen, Z., Corcoran, W., Sra, M., and Singh, A. K. Grapheval2000: Benchmarking and improving large language models on graph datasets. *arXiv preprint arXiv:2406.16176*, 2024.

[47] Wu, Y., Wang, Y., Du, T., Jegelka, S., and Wang, Y. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*, 2025.

[48] Xu, H., Jian, X., Zhao, X., Pang, W., Zhang, C., Wang, S., Zhang, Q., Monteiro, J., Sun, Q., and Yu, T. Graphomni: A comprehensive and extendable benchmark framework for large language models on graph-theoretic tasks. *arXiv preprint arXiv:2504.12764*, 2025.

[49] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[50] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.

[51] Xu, K., Li, J., Zhang, M., Du, S. S., Kawarabayashi, K.-i., and Jegelka, S. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.

[52] Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T., Ren, X., and Zhang, Z. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[53] Ye, R., Zhang, C., Wang, R., Xu, S., and Zhang, Y. Language is all a graph needs. In *ECAL*, 2024.

[54] Yuan, Z., Yuan, H., Li, C., Dong, G., Lu, K., Tan, C., Zhou, C., and Zhou, J. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

[55] Yuan, Z., Liu, M., Wang, H., and Qin, B. Gracore: Benchmarking graph comprehension and complex reasoning in large language models. In *COLING*, 2025.

[56] Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *NeurIPS*, 2018.

# A Related Work

**Graph Reasoning.** Graph reasoning problems fall into two categories: domain-specific, which require understanding both graph structures and node/link attributes, *e.g.*, node classification, link prediction, and knowledge-based QA [15, 56, 18]; and domain-agnostic, also called *graph-theoretic problems*, which focus solely on structural reasoning but find a lot of practical uses in various domains, *e.g.*, shortest paths, Hamiltonian paths, graph isomorphism [50, 35]. For the latter problems that we study in this paper, people have studied the use of RL [29, 40] or unsupervised learning [19], often in conjunction with Graph Neural Networks (GNNs) [20, 49] that align with the solution structure [51]. Yet these models are often built to solve each problem alone. Recently, Sanford et al. [34] prove and validate the priority of the transformer models compared to GNNs on complex graph reasoning tasks requiring long-range dependencies. In this work, we focus on building general-purpose graph reasoners that could solve a range of graph-theoretic problems by exploiting the strength of LLM pretraining, and find that the ability also generalizes to the former domain-specific graph tasks.

**Benchmarking LLMs on Graph Reasoning.** There is a growing interest in evaluating LLMs' graph reasoning abilities. NLGraph [41] evaluate LLMs on graph-theoretic tasks and discover preliminary yet brittle reasoning abilities in the face of spurious correlations and large graphs. Later, GraphArena [37] and GraCoRe [55] include a broader task coverage and recently released LLMs, finding that even OpenAI o1-mini struggles a lot with complex tasks. Moreover, GraphEval2000 [46] and ProGraph [23] emphasize code-oriented problem solving using library-based prompts, and GraphOmni [48] unify varying graph types, encodings, and prompt styles for a comprehensive evaluation. Overall, these benchmarks suggest that LLMs overall demonstrate moderate success on simple tasks but struggle with abstraction, generalization, and larger or more complex graph instances. Nevertheless, these datasets are either too small (e.g., thousands of examples) or not diverse enough (e.g., 8 tasks in NLGraph) for training general-purpose graph reasoners, which motivates the design of `Erdős`.

**Improving LLMs on Graph Reasoning.** A major concern when using LLMs for graph tasks is the mismatch of data structure: LLMs take text sequences as input, while graphs have no natural order. Fatemi et al. [12] analyzed different graph encoding schemes for LLMs, such as adjacency lists and real-name networks, revealing that no single strategy proved universally optimal across all tasks and models. Subsequent explorations with different linearization orders [6], graph embeddings [31], or input modalities [10] have generally resulted in only modest improvements. Another thread of research proposes post-training LLMs using instruction tuning [27, 53] or preference tuning [4, 42, 39] on curated datasets of graph problems. However, the creation of diverse, high-quality instruction datasets at scale is challenging and expensive and requires extra supervision. Furthermore, models trained via distillation may only learn to memorize patterns and overfit to graph tasks [5]; in Section 4.2, we show that previous instruction-tuned models exhibit dramatic failures when generalizing to other data formats and reasoning tasks, while our RL training yields consistently better performance.

**Reinforcement Learning for LLMs Reasoning.** Recent advances have demonstrated that LLMs can attain strong reasoning abilities in math and coding domains through RL, with representative work like OpenAI o1 [30] and DeepSeek R1 [13]. However, as discussed above, even o1 struggles a lot with graph reasoning tasks [55] and it is thus yet unclear whether RL can reliably and scalably improve LLMs' graph reasoning abilities. Our findings on `G1` first confirm the effectiveness of RL on graph reasoning as well and suggest that applying RL to diverse graph-theoretic tasks with verifiable rewards is a scalable path for eliciting generalizable graph reasoning abilities of LLMs.

# B Benchmark Details

To facilitate rule-based Reinforcement Learning of LLMs (aka. Reinforcement Learning from Verifiable Rewards (RLVR)) on graphs, we construct a diverse, large-scale collection of graph-theoretic reasoning tasks. We name it `Erdős` to remember Paul Erdős, a seminal figure with diverse contributions to graph theory. Compared to real-world graph tasks, these graph-theoretic tasks allow clear rule-based determination of rewards for the answers sampled from LLMs. We categorize these tasks into **Easy, Medium, Hard, and Challenging**, based on their inherent problem complexity as well as current LLMs' ability to solve them (see a full list in Table 1). For the training split, there are a total of 100,000 question-answer pairs, evenly distributed across tasks with 2,000 examples each. We also reserve 5,000 test pairs with different questions for evaluation. We include a detailed comparison of `Erdős` with other graph reasoning benchmarks in Appendix O.1. `Erdős` can serve as a dataset for

training LLMs as well as a benchmark for evaluating LLMs on graph-theoretic tasks. Below is a more detailed description of the data collection process.

**Graph-theoretic Tasks.** We curate 50 graph-theoretic reasoning tasks available on NetworkX [14], one of the most widely used libraries for graph processing, and construct, as we know, the most comprehensive collection so far. In the difficulty level, the tasks vary from easy determination of graph attributes like node number counting, to well-known NP-hard problems like the traveling salesman problem. This collection includes both tasks for general graphs and tasks specific to directed graphs or weighted graphs, and covers a wide range of answer types, including boolean, integer, float, node list, edge list, and node mapping.

**Answer Generation.** To generate the golden answer for each problem, we utilize the default solvers of NetworkX to automatically solve the problem. If there are multiple solutions to each question, we use NetworkX-based programs to verify the correctness of each generated solution. The procedure ensures rigorous rewarding attribution, avoiding both costly human labeling and potential bias and hacking brought by LLM judges.

**Graph Sources.** Most previous graph-theoretic datasets or benchmarks [41, 27, 4] consider random graphs, following Erdős-Rényi model [11] or Barabási–Albert model [2]. However, these random graph models are often far from graphs encountered in real-world practice. To mitigate this gap, we utilize the real-world graphs from the Network Repository [33], the largest network repository with thousands of donations in 30+ domains. As these graphs can be very large and infeasible for LLMs, we downsample the graphs by random walk with a restart strategy, generating subgraphs with sizes from 5 to 35 nodes, following common settings in previous work [41, 55, 37].

**Language Encoding**. There are multiple ways to translate the graph structure into languages that LLMs can understand. Previous works explore serialized formats such as adjacency matrix, edge list, or graph embeddings [12, 9, 53], but fail to find a consistently good method. Here, we choose to describe the graph structure in a unified edge list format, *e.g.*, $(1, 2), (2, 3), \ldots$. In later experiments of Section 4.2, we show that our model trained on a single graph description method can even positively transfer to other formats.

# C Training Analysis

In this section, we further analyze the influence of two training factors on `G1`'s reasoning performance.

**Data Mixture.** In Table 2, we observe that although `G1`-3B achieves strong overall performance, it is outperformed by Direct-SFT-3B on the *Hard* and *Challenging* subsets. We hypothesize that this gap arises from imbalanced reward signals across different difficulty levels during RL training. Since correct rollouts are much easier to obtain on simpler tasks, the policy tends to allocate more of its constrained probability ratios as well as KL budget to optimize for *Easy* and *Medium* tasks, thereby maximizing the overall reward. To test this hypothesis, we introduce `G1`-Hard-3B, which is trained exclusively on *Hard* and *Challenging* tasks during RL. As shown in Table 8, this model achieves the highest accuracy on *Hard* (48.50%) and *Challenging* (17.43%) tasks, surpassing both `G1` and Direct-SFT. These results support our claim, suggesting that the suboptimal performance of `G1`-3B on challenging tasks is a natural consequence of the uniformly weighted reward function, rather than a shortcoming of `G1` training pipeline. Notably, despite being trained only on hard tasks, `G1`-Hard-3B also generalizes to *Easy* and *Medium* tasks (69.36% and 70.64%), far exceeding the baseline Qwen2.5-3B-Instruct. This indicates that learning to solve difficult tasks confers transferable reasoning skills that benefit performance on simpler problems. To better balance the optimization process across difficulty levels, we further explore reward-weighting strategies in Appendix N.

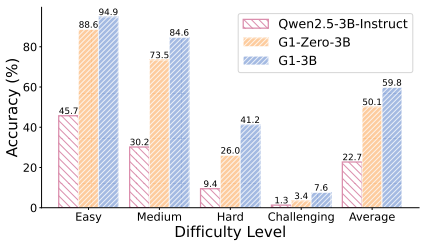



Figure 2: Test accuracy comparison of `G1`-3B and `G1`-Zero-3B on our benchmark. We include results for -7B in Appendix G.

**SFT Warmup.** We study the role of SFT as a cold-start mechanism for RL, evaluating its impact on both performance and response behavior. To isolate the effect of SFT, we compare two variants: `G1`-Zero-3B that is directly trained from the base model Qwen2.5-3B-Instruct with RL, and `G1`-3B that initializes RL from the CoT-SFT checkpoint. As shown in Figure 2, training RL directly from

Table 8: Test accuracy (%) on our benchmark. ⋆ denotes the tasks are excluded in model training. G1-Hard-3B is only RL-trained on Hard and Challenging tasks.

| Category | Model | Easy | Medium | Hard | Challenging | Average |
|---|---|---|---|---|---|---|
| **Base Model** | Qwen2.5-3B-Instruct | 45.71 | 30.18 | 9.44 | 1.29 | 22.72 |
| **Ours** | Direct-SFT-3B | 74.43 | 75.27 | 43.69 | 14.43 | 53.78 |
| | G1-3B | **94.86** | **84.64** | 41.25 | 7.57 | **59.76** |
| | **G1-Hard-3B** | 69.36⋆ | 70.64⋆ | **48.50** | **17.43** | 53.30 |

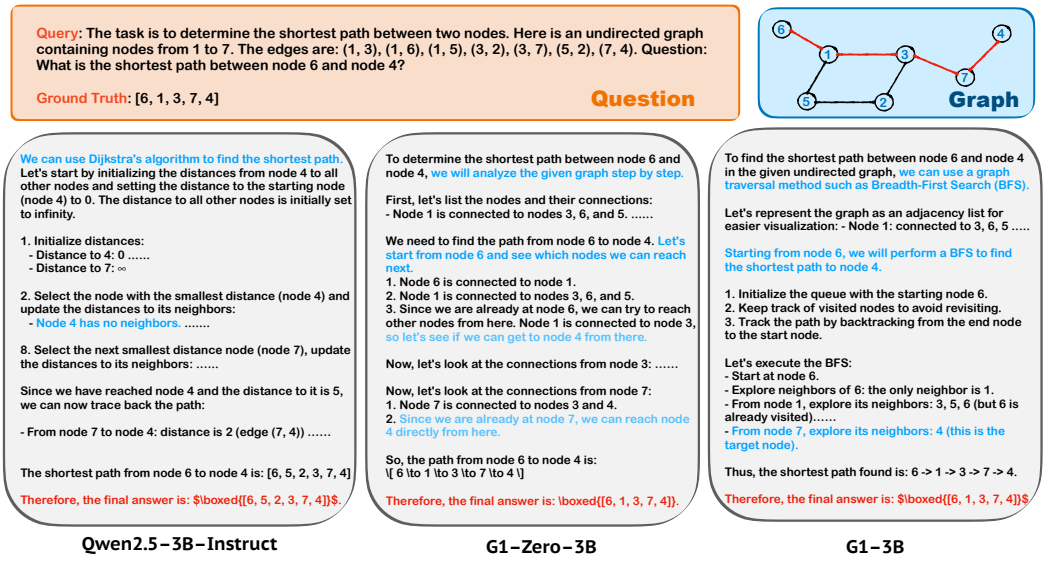



Figure 3: An intuitive illustration of the differences in solution strategies employed by Qwen2.5-3B-Instruct, G1-Zero-3B, and G1-3B for a shortest path problem.

the base model achieves surprisingly strong performance, aligning with recent findings in Deepseek-R1-Zero [13]. Meanwhile, initializing RL with CoT-SFT provides clear and consistent improvements across all difficulty levels, with an average accuracy of 59.8% compared to 50.1% of G1-Zero-3B. Besides, we notice that relative improvements become larger as the difficulty increases. In addition to performance gains, we also observe that models initialized by CoT-SFT present more precise reasoning patterns, illustrated by the case study in the following section.

# D Understanding the Benefits of RL Training for Graph Reasoning

We identify three primary approaches adopted by the models to solve the problem: 1) Breadth-First Search (BFS), 2) Dijkstra's algorithm, and 3) Intuitive deductions. Figure 1a shows the distribution of these approaches alongside their corresponding accuracies for Qwen2.5-3B-Instruct. On *unweighted* graphs, BFS is the most efficient method and yields the highest performance. In contrast, Dijkstra's algorithm is best suited for *weighted* graphs, where it correctly accounts for edge costs. However, its reliance on a min-priority queue and a distance list introduces computational complexity, which appears to challenge Qwen2.5-3B-Instruct and results in its lowest observed accuracy. For example, as shown in Figure 3 (left), the model falsely states that node 4 has no edges (node 4 is connected to node 7) while updating the distance list. Interestingly, intuitive approaches—where the model attempts to visually estimate or heuristically trace paths—can also produce correct answers by a noticeable accuracy, particularly on small graphs.

# E Training Details

## E.1 Rejection Sampling

We randomly extract a subset with 100 examples per task from the training dataset, and use Qwen2.5-32B-Instruct to sample on the subset for k = 8 times with a temperature of 1.0. We filter the responses by keeping the reasoning steps that lead to the right answer. If the task is difficult and the filtered responses are insufficient, we resample the subset with a different random seed and repeat the process above. In the end, we obtain around 4,500 training examples ($\sim$90 per task) for the SFT phase.

## E.2 Supervised Fine-tuning

The detailed training configurations of Naive SFT and RFT are presented in Table 9.

Table 9: Training configurations of Naive-SFT and RFT. In this table, batch size is abbreviated to BSZ, Max-Length refers to the maximum response length during training and Data Num. reports the number of training examples.

| Setting | LR | Weight Decay | BSZ | Max-Length | Data Num. | Epoch |
|---|---|---|---|---|---|---|
| Naive-SFT | 1e-5 w/ 1% warm-up | 1e-2 | 64 | 512 | 98.7k | 1 |
| RFT | 1e-5 w/ 1% warm-up | 1e-2 | 64 | 3072 | 4.4k | 2 |

## E.3 Reinforcement Learning

**Configurations for training and evaluation**. Our experiments primarily adopt Qwen-2.5-3B/7B-Instruct [32] for their moderate sizes and strong reasoning performance. For GRPO training, we set $\epsilon$ to be 0.02, $\beta$ to be 0.001, group size $G$ to be 5, and context length to be 4096 unless otherwise specified. We additionally incorporate an entropy loss of weight 0.001 to encourage the policy to explore. Lastly, we train the models on 8xA800 GPUs with batch size of 512. During evaluation, we use the vLLM [22] engine for efficient inference. For DeepSeek-R1-Distill-Qwen-7B, we set the maximum token generation length to 4096 tokens except for DeepSeek-R1-Distill-Qwen-7B, which is extended to 30768 for its prolonged thinking process. Sampling is configured with a temperature of 0.6, top-p of 0.95, and top-k of 30.

The detailed RL training configurations are presented in Table 10.

Table 10: Training configurations for Naive-SFT and RFT. For abbreviation, we refer the coefficient for entropy loss as Ent. in this table. We report (batch size)/(number of gradient accumulation steps) in the BSZ column, and the temperature for on-policy sampling as $T$.

| Model | LR | $\epsilon$ | $\lvert G\rvert$ | $\beta$ | $\gamma$ | $T$ | Ent. | BSZ | Max-Length | Data Num. | Steps |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RL-3B | 1e-6 | 0.2 | 5 | 1e-3 | 1.0 | 1.0 | 1e-3 | 512/4 | 4096 | 98.7k | 300 |
| SFT-RL-3B | 1e-6 | 0.2 | 5 | 1e-3 | 1.0 | 1.0 | 1e-3 | 512/4 | 4096 | 98.7k | 300 |
| SFT-RL-Hard-3B | 1e-6 | 0.2 | 16 | 5e-4 | 1.0 | 1.0 | 5e-4 | 512/8 | 8192 | 49.3k | 150 |
| SFT-RL-7B | 1e-6 | 0.2 | 5 | 1e-3 | 1.0 | 1.0 | 1e-3 | 512/8 | 4096 | 98.7k | 300 |

# F Evaluation Details

## F.1 Benchmark Introduction

**GraphWiz** [4]. GraphWiz employs the Erdős-Rényi (ER) model to generate random graphs and describe graphs in the edge-list formation like $(u, v)$. The tasks 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*. A prompt example is shown in the following:

**Maximum Triangle Sum Example in GraphWiz**

Find the maximum sum of the weights of three interconnected nodes. In an undirected graph, [i, k] means that node i has the weight k. (i,j) means that node i and node j are connected with an undirected edge. Given a graph, you need to output the maximum sum of the weights of three interconnected nodes. Q: The nodes are numbered from 0 to 4, weights of nodes are: [0, 8] [1, 5] [2, 3] [3, 6] [4, 3], and the edges are: (0, 4) (0, 3) (0, 1) (1, 3) (1, 2) (3, 4). What is the maximum sum of the weights of three nodes?

**Node Classification and Link Prediction** [44]. We adopt the benchmarks introduced by Wang et al. [44], which are constructed by subsampling from the widely used Cora and PubMed citation graphs. Each instance includes a description of the target node (or node pair) containing the paper ID and title, along with the textual and structural information of neighboring nodes. For node classification, we consider two cases that the description includes the attributes of the target node and those of its 2-hop neighbors, with or without labels. For link prediction, we consider two cases where target nodes are described using their own node attributes along with those of their 2-hop neighbors (excluding the other targeting node), with or without titles. For each task, we randomly sample 2,000 examples per case from the benchmark and report the average performance. A representative example for node classification is shown below:

**Node Classification Example**

You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from pubmed dataset. You need to understand the graph and the task definition and answer the question.

## Target node: Paper id: 10695 Title: Haplotype structures and large-scale association testing of the 5' AMP-activated protein kinase genes PRKAA2, PRKAB1, and PRKAB2 [corrected] with type 2 diabetes.

Known neighbor papers at hop 1 (partial, may be incomplete):
Paper id: 1155 Title: Computational disease gene identification: a concert of methods prioritizes type 2 diabetes and obesity candidate genes. Label: Type 2 diabetes

Known neighbor papers at hop 2 (partial, may be incomplete):
Paper id: 9816 Title: Mitochondrial dysfunction and type 2 diabetes. Label: Type 2 diabetes
Paper id: 1683 Title: A genome-wide search for type II diabetes susceptibility genes in Chinese Hans. Label: Type 2 diabetes
Paper id: 9916 Title: Genomewide search for type 2 diabetes-susceptibility genes in French whites: evidence for a novel susceptibility locus for early-onset diabetes on chromosome 3q27-qter and independent replication of a type 2-diabetes locus on chromosome 1q21-q24.
Paper id: 3793 Title: Association of amino acid variants in the activating transcription factor 6 gene (ATF6) on 1q21-q23 with type 2 diabetes in Pima Indians. Label: Type 2 diabetes
Paper id: 4788 Title: Altered glycolytic and oxidative capacities of skeletal muscle contribute to insulin resistance in NIDDM. Label: Type 2 diabetes

Please predict the most likely type of the Target node. Your answer should be chosen from: Type 1 diabetes Type 2 diabetes Experimentally induced diabetes

**GraphArena** [37]. GraphArena samples subgraphs from real-world graphs, including knowledge graphs, social networks, and molecular structures. The tasks include four polynomial-time tasks, *Common Neighbor*, *Shortest Distance*, *Connected Component*, *Graph Diameter*, and six NP-complete tasks, *Maximum Clique Problem (MCP)*, *Maximum Independent Set (MIS)*, *Minimum Vertex Cover (MVC)*, *Maximum Common Subgraph (MCS)*, *Graph Edit Distance (GED)*, and *Traveling Salesman Problem (TSP)*. Each problem is contextualized within the real-world setting of the graph with an example presented as below:

**Connected Component Example in GraphArena**

You are required to identify all connected components in the given social network and output one representative node from each component. Within a connected component, any node can be reached from any other node through the edges in the graph. Different connected components are isolated from each other.

**Problem to Solve**
- Names in the network: Veronica Garcia, Katherine Brennan, Angel Chavez, Steven Martin, Brett Johnson, Megan Banks, Julia Dominguez, Rachel Mitchell - Fiendship connections: Veronica Garcia to Brett Johnson, Veronica Garcia to Megan Banks, Katherine Brennan to Brett Johnson, Katherine Brennan to Megan Banks, Angel Chavez to Megan Banks, Angel Chavez to Rachel Mitchell, Steven Martin to Megan Banks, Brett Johnson to Megan Banks, Megan Banks to Julia Dominguez, Megan Banks to Rachel Mitchell.

Identify all connected components in this network. Note that for each connected component, you should only output one of its nodes. Present your answer in the following format: [UserA, UserB, UserC, UserD, ...]

**GSM8K** [7]. GSM8K is a dataset of 8.5K high quality linguistically diverse grade school math word problems created by human problem writers. We report the accuracies on the 1K test problems and the dataset is downloaded via `https://huggingface.co/datasets/openai/gsm8k`.

**Example in GSM8K**

Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

**MATH500.** The dataset contains a subset of 500 problems from the MATH benchmark that OpenAI created in their Let's Verify Step by Step paper [25]. We download the dataset via `https://huggingface.co/datasets/HuggingFaceH4/MATH-500`.

**Example in MATH500**

Let $z = 2 + \sqrt{2} - (3 + 3\sqrt{2})i$, and let $c = 2 - 3i$. Let $w$ be the result when $z$ is rotated around $c$ by $\frac{\pi}{4}$ counter-clockwise.
[asy]
unitsize(0.6 cm);
pair C, W, Z;
Z = (2 + sqrt(2), -3 - 3*sqrt(2));
C = (2,-3);
W = rotate(45,C)*(Z);
draw(Z–C–W);
dot("$c$", C, N);
dot("$w$", W, SE);
dot("$z$", Z, S);
label("$\frac{\pi}{4}$", C + (0.6,-1));
[/asy]
Find $w$.

**MMLU-Pro.** MMLU-Pro is enhanced version of the Massive Multitask Language Understanding benchmark. It covers a wide range of disciplines, including Math, Law, Engineering, Health, Phycology, etc. We download the dataset via `https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro/viewer/default/test?q=Health&row=5903`.

**Health Example in MMLU-pro**

Question: Food supplements, including trace minerals and vitamins are frequently advertised with promising health benefits. Which of the following substance could be consumed in excess, i.e. well above the recommended daily requirement?

Options: [ "Vitamin C", "Vitamin D", "Zinc", "Vitamin A" ]

### F.2 Inference Configuration

For inference, we adopt the vLLM framework [22]. We set the temperature to be 0.06 and the context window to be 4096 for our evaluations unless otherwise specified.

### F.3 Prompt and Answer Extraction

To facilitate answer extraction, we adopt the prompt shown in F.3 to guide the models to reason step by step and place their answers within \boxed{}. We extract the last \boxed{} shown in the model responses and do necessary format normalizations to retrieve the answer, which includes operations like converting LaTeX-style fraction numbers to float numbers.

**Problem Instructions**

{Question Description}
Approach the problem methodically. Ensure all conclusions are based on precise calculations and logical deductions. Feel free to explore various solution methods and cross-check results for consistency. Maintain dynamic thinking and always verify each step of your reasoning.
Present the final answer in \boxed{} format, like this: $\boxed{ANSWER}$, where ANSWER is the final result or expression.
Think carefully and break down the problem step by step.

## G Comparison between G1-Zero-7B and G1-7B

In Section **??**, we study the role of SFT as a cold-start mechanism for RL by comparing two variants: G1-Zero-3B that is directly trained from the base model Qwen2.5-3B-Instruct with RL, and G1-3B that initializes RL from the CoT-SFT checkpoint. We observe that G1-Zero-3B already achieves surprisingly strong performance, while G1-3B presents clear and consistent improvements across all difficulty levels. Here, we provide additional results for comparing G1-Zero-7B and G1-7B. As shown in Figure 4, for *Easy* and *Medium* tasks, the benefit brought by CoT-SFT initialization is marginal, with G1-Zero-7B (96.9%) even surpassing G1-7B (95.1%) on *Easy* tasks. However, on *Hard* and *Challenging* tasks, CoT-SFT as a preliminary step has definite benefits by improving G1-Zero-7B from 13.7% to 23.6% on *Challenging* tasks. This observation agrees with the case in -3B. Moreover, the average gap between G1-Zero-7B and G1-7B is less than -3B case, indicating G1-7B can possibly be further improved with CoT-SFT generated by a stronger teacher model rather than Qwen2.5-32B-Instruct. We leave this exploration for further work.

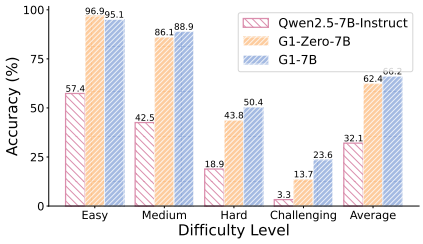



Figure 4: Test accuracy comparison of G1-7B and G1-Zero-7B on our benchmark.

## H Transferability of G1 to Larger Graphs

To verify the transferability of G1 to larger graphs, we construct a new test set of 5,000 graphs with 36-100 nodes, with other settings kept the same, which ensures there is no overlap between training and test data. Table 11 shows that both G1-3B and G1-7B achieve strong zero-shot generalization to these larger graphs without additional training, significantly outperforming the baselines across

difficulty levels. These results demonstrate our method's effective scalability beyond the original training distribution.

Currently, we limit our analysis to smaller graphs because of the context window limit of underlying LLMs (*e.g.*, Qwen2.5-7B-Instruct). The token count scales quadratically with the number of nodes. As shown in Table 12, a 200-node graph often exceeds 32k tokens, surpassing the maximum effective context window of many open-source LLMs [17]. Long-context understanding is actively studied in the LLM literature, and there are some cutting-edge proprietary variants (*e.g.*, OpenAI's GPT-4.1) supporting inputs of over 1M tokens (though graphs of 2000 nodes need 4M). Due to computational constraints (demanding a huge GPU memory), it is hard for us to evaluate on a very long context. We believe our approach can be scaled to larger graphs with the rapid progress of long-context studies.

Table 11: Zero-shot generalization (accuracy in percentage) of G1 to larger graphs with 36-100 nodes.

| | Easy | Medium | Hard | Challenging | Average |
|---|---|---|---|---|---|
| Qwen2.5-3B-Instruct | 27.98 | 28.53 | 5.26 | 0.29 | 16.74 |
| **G1-3B** | **79.39** | **65.66** | **18.46** | **3.74** | **44.29** |
| Qwen2.5-7B-Instruct | 37.86 | 41.56 | 9.17 | 1.17 | 23.94 |
| **G1-7B** | **76.65** | **70.67** | **23.16** | **5.22** | **46.46** |

Table 12: Token numbers of graph with different node sizes. We generate random graphs by Erdős–Rényi model with an edge probability of 0.2. For each node number, we generate 10 graphs and report the mean of the token numbers. For tokenization, we utilize the tokenizer of Qwen2.5-Instruct.

| **Node Number** | 30 | 50 | 100 | 200 | 500 | 1000 | 2000 |
|---|---|---|---|---|---|---|---|
| **Token Number** | 641.4 | 1941.5 | 7842.8 | ∼35k | ∼230k | ∼1M | ∼4M |

# I GPT-4o Results on Erdős

Due to the cost budget, we randomly sample 20 examples per task from Erdős's evaluation set to construct a subset of 1,000 samples. As shown in Table 13, G1-3B performs comparably to GPT-4o on average (57.37% vs. 55.13%), while G1-7B surpasses GPT-4o across all difficulty levels, exceeding it by over 10% on average. This comparison further validates the strong graph reasoning capabilities of the G1 models.

Table 13: Test accuracy%) of GPT-4o and G1 variants on a subset of Erdős.

| | Easy | Medium | Hard | Challenging | Average |
|---|---|---|---|---|---|
| GPT-4o-2024-11-20 | 82.50 | 81.82 | 44.06 | 12.14 | 55.13 |
| **G1-3B** | **96.43** | 85.45 | 41.88 | 5.71 | 57.37 |
| **G1-7B** | **96.43** | **88.64** | **52.50** | **23.57** | **65.29** |

# J Results of G1-Zero-32B

To demonstrate the scalability of our approach, we extended our training methodology to develop G1-Zero-32B from Qwen2.5-32B-Instruct. The results showcase substantial improvements across all difficulty levels while maintaining computational efficiency. As shown in Table 14, G1-Zero-32B demonstrates remarkable improvements on the Erdősbenchmark across all difficulty categories. The model achieves a **27.96%** improvement in average accuracy (from 47.10% to 75.06%), with particularly notable gains in harder categories: +31.87 points on Hard problems and +26.43 points on Challenging problems. These results indicate that our method scales effectively to larger models while maintaining consistent performance improvements across varying problem complexities.

Furthermore, Table 15 demonstrates that G1-Zero-32B not only preserves but slightly enhances mathematical performance on standard benchmarks. The model shows modest improvements on both

GSM8K (+0.08 points) and MATH (+4.00 points), confirming that our reasoning-focused training does not compromise existing mathematical capabilities and may even provide synergistic benefits. The training process is completed in 40 hours on 32×A100 GPUs, demonstrating the practical feasibility of scaling our approach to larger model architectures without prohibitive computational costs.

Table 14: Test accuracy (%) of G1-Zero-32B and Qwen2.5-32B-Instruct on `Erdős`.

| | Easy | Medium | Hard | Challenging | Average |
|---|---|---|---|---|---|
| Qwen2.5-32B-Instruct | 70.57 | 68.73 | 33.38 | 9.00 | 47.10 |
| G1-Zero-32B | **97.79** | **93.00** | **65.25** | **35.43** | **75.06** |

Table 15: Test accuracy (%) of G1-Zero-32B and Qwen2.5-32B-Instruct on math tasks.

| | GSM8K | MATH |
|---|---|---|
| Qwen2.5-32B-Instruct | 90.67 | 76.80 |
| G1-Zero-32B | **90.75** | **80.80** |

# K Experiments for Robustness Verification

## K.1 Multi-runs Robustness

To rigorously evaluate robustness, we conducted 32 repeated runs with different random seeds. The results in Table 16 demonstrate consistently small standard deviations (<1% across all models and difficulty levels), confirming the stability of our method against potential randomness in LLM outputs.

Table 16: Test accuracy (%) for 32 runs with different random seeds.

| | Easy | Medium | Hard | Challenging |
|---|---|---|---|---|
| Qwen2.5-3B-Instruct | $45.65 \pm 0.51$ | $30.88 \pm 0.36$ | $10.36 \pm 0.21$ | $1.54 \pm 0.29$ |
| G1-3B | $94.96 \pm 0.32$ | $83.22 \pm 0.24$ | $41.48 \pm 0.40$ | $7.96 \pm 0.64$ |
| Qwen2.5-7B-Instruct | $57.50 \pm 0.13$ | $44.92 \pm 0.03$ | $19.90 \pm 0.31$ | $3.45 \pm 0.22$ |
| G1-7B | $95.66 \pm 0.12$ | $88.89 \pm 0.16$ | $50.76 \pm 0.53$ | $24.46 \pm 0.84$ |

## K.2 Prompt Robustness

For prompt robustness, we rigorously test prompt sensitivity by having GPT-4o generate three semantically equivalent prompt variants. Table 17 shows minimal performance variance (<1.5% standard deviation) across all models and difficulty levels, confirming our benchmark's stability to phrasing changes.

# L Analysis of `G1`'s Transferbility to Mathematics Tasks

We compare the generation results of `G1`-7B and Qwen2.5-7B-Instruct on the math benchmarks case by case. As a summary, `G1`-7B improves over Qwen2.5-7B-Instruct on more accurate *numerical calculation* (Instance 1) and more comprehensive *utilization of the given information* (Instance 2). It's reasonable since the training of `G1` includes complex numerical calculations (e.g., shortest path) and careful processing of questions (e.g., traversing the whole graph descriptions to find key edges).

In fact, the transferability between different reasoning domains has also been observed in other works and is actively under research. For example, Chu et al. [6] demonstrates that reinforcement learning exhibits strong generalization even under distribution shifts, as it encourages models to learn robust, task-agnostic reasoning strategies rather than overfitting to surface-level patterns. Additionally, RL training has been shown to foster mutual reinforcement across seemingly distinct tasks, *e.g.*, optimizing on puzzles can help reasoning on math [24]. Similarly, due to the similarity between math and graph problems, they would also require similar reasoning skills (such as guessing, backtracking,

Table 17: Test accuracy (%) on different prompts, mean and standard deviation reported.

| | Easy | Medium | Hard | Challenging |
|---|---|---|---|---|
| Qwen2.5-3B-Instruct | $44.26 \pm 0.63$ | $30.52 \pm 1.08$ | $10.71 \pm 0.41$ | $1.24 \pm 0.07$ |
| G1-3B | $94.86 \pm 0.69$ | $83.03 \pm 0.42$ | $41.08 \pm 1.35$ | $9.67 \pm 1.05$ |
| Qwen2.5-7B-Instruct | $56.48 \pm 1.17$ | $56.48 \pm 1.17$ | $18.81 \pm 0.79$ | $3.10 \pm 0.18$ |
| G1-7B | $95.48 \pm 0.18$ | $88.39 \pm 0.67$ | $51.27 \pm 0.13$ | $25.14 \pm 1.04$ |

and verification). Therefore, solving graph problems could have positive transferability to math domains.

**Instance 1 (Calculation Correctness)**

**Question**: Toula went to the bakery and bought various types of pastries. She bought 3 dozen donuts, which cost 68 dollars per dozen, 2 dozen mini cupcakes, which cost 80 dollars per dozen, and 6 dozen mini cheesecakes for 55 dollars per dozen. How much was the total cost?

**Ground Truth**: 694

**Answer of Qwen2.5-7B-Instruct**: - Calculate the cost of the donuts: Toula bought 3 dozen donuts. The cost per dozen donuts is 68 dollars. Therefore, the total cost for the donuts is: 3 dozen $\times$ 68 dollars/dozen $=$ 180 dollars.
- Calculate the total cost: Add the costs of the donuts, mini cupcakes, and mini cheesecakes: 180 dollars $+$ 160 dollars $+$ 330 dollars $=$ 670 dollars.

**Answer of G1-7B**: - Cost of Donuts: Toula bought 3 dozen donuts. The cost per dozen donuts is 68 dollars. - Therefore, the total cost for the donuts is: 3 dozen $\times$ 68 per dozen $= 3 \times 68 = 204$.
- Now, we add up the costs of all the pastries to find the total cost: \$204 (donuts) $+$ \$160 (mini cupcakes) $+$ \$330 (mini cheesecakes) $=$ \$694.

**Instance 2 (Utilization of Known Conditions)**

**Question**: On Mondays, Wednesdays, and Fridays, college student Kimo has three 1-hour classes each day. On Tuesdays and Thursdays, he has two 2-hour classes each day. In one semester, there are 16 weeks of school. In one semester, how many hours does Kimo spend attending classes?

**Ground Truth**: 272

**Answer of Qwen2.5-7B-Instruct**: - Calculate the number of hours Kimo spends in classes each week: On Mondays, Wednesdays, and Fridays, Kimo has 3 classes, each lasting 1 hour. 3 classes $\times$ 1 hour/class $=$ 3 hours. On Tuesdays and Thursdays, Kimo has 2 classes, each lasting 2 hours. 2 classes $\times$ 2 hours/class $=$ 4 hours. Therefore, the total number of hours Kimo spends in classes each week is: 3 hours $+$ 4 hours $=$ 7 hours. - Calculate the number of hours Kimo spends in classes in one semester: There are 16 weeks in the semester. The total number of hours Kimo spends in classes in one semester is: 7 hours/week$\times$16 weeks $=$ 112 hours.

**Answer of G1-7B**: - Step 1: Calculate Weekly Class Hours. - Mondays, Wednesdays, and Fridays: Each day, Kimo has 3 classes, each lasting 1 hour. Therefore, for each of these days, the total class time is: 3 classes/day $\times$ 1 hour/class $=$ 3 hours/day. Since there are 3 days (Monday, Wednesday, Friday): 3 hours/day $\times$ 3 days $=$ 9 hours. - Tuesdays and Thursdays: Each day, Kimo has 2 classes, each lasting 2 hours. Therefore, for each of these days, the total class time is: 2 classes/day $\times$ 2 hours/class $=$ 4 hours/day. Since there are 2 days (Tuesday, Thursday) 4 hours/day $\times$ 2 days $=$ 8 hours. - Total Weekly Class Hours: Adding the hours from all the days: 9 hours $+$ 8 hours $=$ 17 hours/week. - Step 2: Calculate Total Class Hours in One Semester. There are 16 weeks in the semester. Therefore, the total class hours in one semester is: 17 hours/week $\times$ 16 weeks $=$ 272 hours.

# M Detailed Experimental Results

## M.1 Detailed Results for GraphWiz

We present the test accuracy for each task in the GraphWiz benchmark in Table 18. G1-7B achieves the highest overall accuracy (57.11%) among all models and reaches the top in 5/7 tasks. It outperforms DeepSeek-R1-Distill-Qwen-7B (51.86%) and even models specifically trained on GraphWiz data such as GraphWiz-7B-RFT (49.61%). Moreover, the smaller variant G1-3B ranks first on all tasks among models of similar parameters, surpassing the base model (Qwen2.5-3B-Instruct) by 13.64% on average and achieves comparable performance with DeepSeek-R1-Distill-Qwen-7B. The results in the GraphWiz benchmark verify the strong zero-shot generalization ability of our G1 models.

Table 18: Test accuracy (%) on the GraphWiz benchmark.

| Model | cycle | connect | bipartite | topology | shortest | triangle | flow | hamilton | subgraph | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| Llama-3.2-3B-Instruct | 32.00 | 53.75 | 25.75 | 7.50 | 2.75 | 3.75 | 2.50 | 38.25 | 12.00 | 19.80 |
| Qwen2.5-3B-Instruct (base) | 58.00 | 60.50 | 38.50 | 4.00 | 5.75 | 15.50 | 7.50 | 75.00 | **63.25** | 36.44 |
| **G1-3B** (Ours) | **91.00** | **64.00** | **64.25** | **13.00** | **14.00** | **23.25** | **43.00** | **96.00** | 42.25 | **50.08** |
| GraphWiz-RFT-7B | 88.00 | **90.25** | 72.25 | 19.75 | **28.00** | 36.75 | 24.75 | 2.50 | **84.25** | 49.61 |
| GraphWiz-DPO-7B | 86.50 | 82.25 | 71.75 | 15.00 | 26.75 | **37.00** | 45.00 | 0.00 | 79.00 | 49.25 |
| Llama-3.1-8B-Instruct | 64.75 | 81.00 | 58.75 | 11.50 | 3.50 | 4.25 | 9.25 | 19.25 | 45.00 | 33.03 |
| DeepSeek-R1-Distill-Qwen-7B | 87.00 | 90.00 | 42.75 | 11.00 | 18.25 | 36.00 | 40.00 | 84.75 | 57.00 | 51.86 |
| Qwen2.5-7B-Instruct (base) | 79.00 | 72.25 | 40.75 | 4.25 | 13.50 | 28.75 | 11.50 | 91.25 | 61.00 | 44.69 |
| **G1-7B** (Ours) | **92.00** | 80.00 | **75.75** | **24.25** | 21.00 | 29.50 | **46.25** | **95.25** | 50.00 | **57.11** |

## M.2 Detailed Results for MMLU-Pro

We present the detailed results for our evaluations on MMLU-Pro in Table 19. We first notice that although G1 models share close accuracies with their base model on average, they excel at notably different disciplines: G1-3B does the best in Physics (56.18%) while G1-7B is good at CS (53.32%). Interestingly, RL training on graph problems in some cases improves G1over Qwen on non-STEM subjects such as Health (53.0% v.s. 37.65%) for 3B models and Business (62.76% v.s. 53.91%) for 7B models.

Table 19: Test accuracy (%) on the MMLU-Pro benchmark.

| Model | Physics | Chem. | Econ. | Other | Math | Philo. | History | Busi. | Psycho. | Law | Engin. | Health | CS | Bio. | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Llama-3.2-3B-Instruct | 7.18 | 14.79 | 15.91 | 13.39 | 6.50 | 13.69 | 18.54 | 11.28 | 23.91 | 15.40 | 9.89 | 14.03 | 13.25 | 9.71 | 13.51 |
| Qwen2.5-3B-Instruct (base) | 38.49 | 31.18 | **46.21** | 37.34 | **58.92** | 31.06 | 31.23 | **45.25** | **46.24** | 18.07 | 19.40 | 37.65 | **41.22** | **54.25** | **38.54** |
| CoT-SFT-3B | 35.70 | 13.99 | 32.25 | 38.72 | 53.29 | 34.41 | 25.65 | 30.04 | 18.16 | **42.71** | 28.08 | 39.22 | 36.34 | 46.03 | 34.23 |
| **G1-3B** (Ours) | **56.18** | **42.46** | 16.26 | **43.73** | 37.78 | **44.55** | **36.10** | 31.80 | 41.46 | 20.95 | **34.42** | **53.00** | 28.86 | 30.18 | 37.12 |
| Llama-3.1-8B-Instruct | 28.79 | 17.13 | 33.96 | 34.03 | 32.28 | 41.83 | 24.91 | 18.80 | 43.89 | 46.45 | 35.28 | 36.10 | 31.75 | 28.26 | 32.02 |
| DeepSeek-R1-Distill-Qwen-7B | 39.75 | 11.72 | 19.20 | 49.81 | 40.80 | 19.95 | 23.35 | 25.65 | **47.39** | 30.30 | **72.76** | 36.84 | 34.59 | 49.51 | 37.21 |
| Qwen2.5-7B-Instruct (base) | 44.17 | 48.53 | 46.87 | **55.89** | **65.80** | 21.44 | **54.53** | 53.91 | 27.04 | 49.50 | 42.64 | **53.66** | 33.27 | 35.96 | 45.75 |
| CoT-SFT-7B | 44.36 | **55.51** | 44.61 | 29.82 | 51.08 | **64.84** | 45.97 | 41.45 | 46.42 | 37.01 | 33.87 | 45.61 | 21.44 | **52.01** | 44.54 |
| **G1-7B** (Ours) | **46.43** | 51.19 | **68.76** | 40.94 | 47.70 | 53.90 | 32.40 | **62.76** | 25.61 | **49.88** | 51.50 | 51.71 | **53.32** | 36.07 | **48.56** |

## M.3 Detailed Results for GraphArena

We report the detailed results for evaluations on the easy/hard problems from GraphArena in Table 20 and Table 21 respectively. We observe that G1 models perform equally or better compared to the other models on all tasks but *Distance*, in which G1 performs slightly worse than the Qwen models.

## M.4 Detailed Results for Erdős

In Table 22, we show the performance for each task in Erdősfor our models and baselines in detail.

Table 20: Test accuracy (%) on the **easy** problems from the GraphArena benchmark.

| Model | Connected | Diameter | Distance | Neighbor | GED | TSP | MCP | MCS | MIS | MVC |
|---|---|---|---|---|---|---|---|---|---|---|
| Llama-3.2-3B-Instruct | 8.00 | 16.00 | 15.00 | 50.00 | 9.00 | 2.00 | 15.00 | 10.00 | 7.00 | 5.00 |
| Qwen2.5-3B-Instruct (base) | 20.00 | 11.00 | **47.00** | 48.00 | **37.00** | **17.00** | 3.00 | **41.00** | 4.00 | 2.00 |
| **G1-3B** (Ours) | **52.00** | **42.00** | **47.00** | **89.00** | 30.00 | **17.00** | **27.00** | 20.00 | **32.00** | **22.00** |
| LLaMA2-7B-RFT | 0.00 | 7.00 | 1.00 | 1.00 | 4.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| LLaMA2-7B-DPO | 0.00 | 1.00 | 0.00 | 0.00 | 3.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| Llama-3.1-8B-Instruct | 33.00 | 29.00 | 45.00 | 81.00 | 24.00 | 14.00 | 32.00 | 18.00 | 24.00 | 20.00 |
| DeepSeek-R1-Distill-Qwen-7B | 77.00 | 41.00 | 64.00 | 82.00 | 22.00 | 30.00 | 44.00 | 40.00 | **56.00** | 17.00 |
| Qwen2.5-7B-Instruct (Ours) | 79.00 | 15.00 | **70.00** | 84.00 | 22.00 | 22.00 | 39.00 | 41.00 | 28.00 | 21.00 |
| **G1-7B** (Ours) | **86.00** | **63.00** | 62.00 | **99.00** | **30.00** | **38.00** | **52.00** | **51.00** | 50.00 | **63.00** |

Table 21: Test accuracy (%) on the **hard** problems from the GraphArena benchmark.

| Model | Connected | Diameter | Distance | Neighbor | GED | TSP | MCP | MCS | MIS | MVC |
|---|---|---|---|---|---|---|---|---|---|---|
| Llama-3.2-3B-Instruct | 0.00 | 1.00 | 7.00 | 19.00 | 3.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 |
| Qwen2.5-3B-Instruct (base) | 4.00 | 4.00 | **28.00** | 22.00 | **7.00** | 0.00 | **1.00** | 0.00 | 0.00 | 1.00 |
| **G1-3B** (Ours) | **19.00** | **12.00** | 25.00 | **51.00** | 3.00 | 0.00 | 0.00 | 0.00 | **1.00** | **7.00** |
| LLaMA2-7B-RFT | 0.00 | 2.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| LLaMA2-7B-DPO | 0.00 | 3.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Llama-3.1-8B-Instruct | 8.00 | 4.00 | 19.00 | 54.00 | **3.00** | 1.00 | 2.00 | 0.00 | 0.00 | 7.00 |
| DeepSeek-R1-Distill-Qwen-7B | 18.00 | 4.00 | 33.00 | 36.00 | 1.00 | 0.00 | 3.00 | 0.00 | 1.00 | 4.00 |
| Qwen2.5-7B-Instruct (base) | 27.00 | 4.00 | **44.00** | 68.00 | 2.00 | 0.00 | **5.00** | 0.00 | 1.00 | 5.00 |
| **G1-7B** (Ours) | **31.00** | **27.00** | 35.00 | **84.00** | **3.00** | 0.00 | 3.00 | 0.00 | **6.00** | **39.00** |

Table 22: Accuracy comparison of models across tasks

| Task | GPT-4o | o3-mini | Llama-3B | Qwen-3B | DSFT-3B | CSFT-3B | G1-3B | Llama-8B | Qwen-7B | Math-7B | R1-7B | GWiz-R | GWiz-D | DSFT-7B | CSFT-7B | G1-7B | Llama-70B | Qwen-72B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| node_number | 100.00 | 100.00 | 83.00 | 94.00 | 100.00 | 97.00 | 100.00 | 99.00 | 99.00 | 94.00 | 100.00 | 0.00 | 0.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| dominating_set | 29.41 | 64.71 | 57.00 | 23.00 | 72.00 | 31.00 | 99.00 | 37.00 | 27.00 | 21.00 | 27.00 | 34.00 | 28.00 | 74.00 | 68.00 | 99.00 | 24.00 | 44.00 |
| common_neighbor | 73.68 | 73.68 | 23.00 | 44.00 | 71.00 | 71.00 | 91.00 | 56.00 | 52.00 | 48.00 | 79.00 | 0.00 | 0.00 | 76.00 | 80.00 | 93.00 | 91.52 | 89.99 |
| edge_number | 72.22 | 77.78 | 9.00 | 31.00 | 31.00 | 59.00 | 96.00 | 16.00 | 58.00 | 38.00 | 74.00 | 0.00 | 0.00 | 39.00 | 34.00 | 97.00 | 72.00 | 66.00 |
| neighbor | 84.21 | 63.16 | 26.00 | 36.00 | 87.00 | 82.00 | 91.00 | 42.00 | 65.00 | 64.00 | 94.00 | 4.00 | 2.00 | 89.00 | 89.00 | 93.00 | 93.05 | 98.53 |
| bfs | 52.17 | 17.39 | 0.00 | 3.00 | 52.00 | 30.00 | 95.00 | 5.00 | 12.00 | 9.00 | 12.00 | 0.00 | 0.00 | 43.00 | 44.00 | 98.00 | 25.00 | 53.00 |
| has_cycle | 80.00 | 100.00 | 51.00 | 51.00 | 98.00 | 63.00 | 89.00 | 46.00 | 55.00 | 54.00 | 83.00 | 65.00 | 83.00 | 95.00 | 93.00 | 98.00 | 64.00 | 54.00 |
| dfs | 73.33 | 33.33 | 0.00 | 9.00 | 61.00 | 43.00 | 100.00 | 12.00 | 27.00 | 10.00 | 23.00 | 0.00 | 0.00 | 52.00 | 50.00 | 99.00 | 29.00 | 49.00 |
| minimum_spanning_tree | 38.10 | 42.86 | 5.00 | 8.00 | 62.00 | 17.00 | 81.00 | 17.00 | 15.00 | 14.00 | 46.00 | 0.00 | 0.00 | 65.00 | 65.00 | 66.00 | 28.00 | 39.00 |
| edge_existence | 100.00 | 100.00 | 60.00 | 80.00 | 100.00 | 97.00 | 100.00 | 73.00 | 96.00 | 82.00 | 100.00 | 52.00 | 56.00 | 98.00 | 97.00 | 100.00 | 98.00 | 100.00 |
| is_regular | 100.00 | 95.00 | 88.00 | 95.00 | 98.00 | 98.00 | 100.00 | 92.00 | 96.00 | 99.00 | 98.00 | 27.00 | 58.00 | 99.00 | 99.00 | 100.00 | 99.00 | 100.00 |
| degree | 95.45 | 100.00 | 26.00 | 58.00 | 94.00 | 93.00 | 95.00 | 72.00 | 77.00 | 79.00 | 96.00 | 0.00 | 0.00 | 88.00 | 82.00 | 99.00 | 94.00 | 82.00 |
| is_tournament | 100.00 | 88.89 | 47.00 | 75.00 | 99.00 | 99.00 | 99.00 | 80.00 | 86.00 | 87.00 | 100.00 | 22.00 | 58.00 | 100.00 | 100.00 | 99.00 | 99.00 | 94.00 |
| density | 68.18 | 90.91 | 36.00 | 33.00 | 17.00 | 38.00 | 92.00 | 42.00 | 38.00 | 40.00 | 81.00 | 0.00 | 0.00 | 12.00 | 15.00 | 97.00 | 51.00 | 51.00 |
| adamic_adar_index | 92.31 | 88.46 | 1.00 | 6.00 | 74.00 | 89.00 | 94.00 | 12.00 | 39.00 | 22.00 | 82.00 | 3.00 | 1.00 | 76.00 | 75.00 | 98.00 | 52.00 | 64.00 |
| clustering_coefficient | 72.22 | 94.44 | 13.00 | 31.00 | 71.00 | 56.00 | 82.00 | 25.00 | 44.00 | 36.00 | 65.00 | 6.00 | 10.00 | 67.00 | 66.00 | 88.00 | 49.00 | 69.00 |
| connected_component_number | 60.87 | 82.61 | 9.00 | 27.00 | 85.00 | 63.00 | 79.00 | 34.00 | 35.00 | 30.00 | 79.00 | 0.00 | 0.00 | 80.00 | 81.00 | 92.00 | 64.00 | 66.00 |
| bipartite_maximum_matching | 40.74 | 48.15 | 3.00 | 19.00 | 53.00 | 47.00 | 82.00 | 13.00 | 12.00 | 3.00 | 42.00 | 0.00 | 0.00 | 76.00 | 73.00 | 87.00 | 29.00 | 37.00 |
| local_connectivity | 96.15 | 100.00 | 57.00 | 62.00 | 93.00 | 86.00 | 90.00 | 53.00 | 74.00 | 79.00 | 82.00 | 53.00 | 66.00 | 97.00 | 98.00 | 96.00 | 77.00 | 69.00 |
| jaccard_coefficient | 100.00 | 100.00 | 23.00 | 48.00 | 81.00 | 84.00 | 95.00 | 44.00 | 77.00 | 70.00 | 95.00 | 3.00 | 5.00 | 78.00 | 76.00 | 100.00 | 87.00 | 93.00 |
| min_edge_covering | 10.53 | 31.58 | 1.00 | 2.00 | 23.00 | 17.00 | 51.00 | 0.00 | 1.00 | 1.00 | 37.00 | 0.00 | 0.00 | 18.00 | 17.00 | 50.00 | 16.00 | 15.00 |
| is_eularian | 86.36 | 95.45 | 78.00 | 81.00 | 95.00 | 89.00 | 98.00 | 82.00 | 81.00 | 89.00 | 92.00 | 33.00 | 59.00 | 93.00 | 93.00 | 97.00 | 90.00 | 80.00 |
| degree_centrality | 71.43 | 85.71 | 0.00 | 7.00 | 81.00 | 79.00 | 89.00 | 4.00 | 8.00 | 23.00 | 87.00 | 0.00 | 0.00 | 80.00 | 84.00 | 97.00 | 49.00 | 88.00 |
| is_bipartite | 68.00 | 92.00 | 49.00 | 39.00 | 92.00 | 55.00 | 79.00 | 53.00 | 52.00 | 43.00 | 76.00 | 51.00 | 67.00 | 93.00 | 90.00 | 80.00 | 62.00 | 67.00 |
| resource_allocation_index | 94.12 | 100.00 | 2.00 | 10.00 | 80.00 | 79.00 | 92.00 | 15.00 | 45.00 | 40.00 | 86.00 | 2.00 | 2.00 | 77.00 | 80.00 | 92.00 | 36.00 | 78.00 |
| max_weight_matching | 11.11 | 27.78 | 2.00 | 3.00 | 25.00 | 22.00 | 24.00 | 7.00 | 12.00 | 2.00 | 40.00 | 0.00 | 0.00 | 25.00 | 25.00 | 43.00 | 24.00 | 26.00 |
| closeness_centrality | 0.00 | 31.58 | 0.00 | 1.00 | 8.00 | 6.00 | 9.00 | 4.00 | 3.00 | 3.00 | 14.00 | 0.00 | 0.00 | 4.00 | 5.00 | 11.00 | 13.00 | 11.00 |
| traveling_salesman_problem | 36.84 | 89.47 | 8.00 | 24.00 | 29.00 | 40.00 | 43.00 | 17.00 | 41.00 | 41.00 | 62.00 | 3.00 | 1.00 | 25.00 | 20.00 | 51.00 | 47.00 | 43.00 |
| strongly_connected_number | 13.33 | 73.33 | 4.00 | 5.00 | 63.00 | 24.00 | 58.00 | 3.00 | 11.00 | 7.00 | 35.00 | 0.00 | 0.00 | 55.00 | 56.00 | 59.00 | 9.00 | 10.00 |
| shortest_path | 69.23 | 38.46 | 11.00 | 19.00 | 74.00 | 51.00 | 62.00 | 31.00 | 35.00 | 11.00 | 62.00 | 3.00 | 0.00 | 77.00 | 78.00 | 70.00 | 62.00 | 60.00 |
| center | 19.05 | 66.67 | 4.00 | 8.00 | 25.00 | 13.00 | 25.00 | 6.00 | 8.00 | 9.00 | 26.00 | 0.00 | 0.00 | 24.00 | 25.00 | 35.00 | 29.72 | 41.48 |
| diameter | 17.65 | 94.12 | 12.00 | 8.00 | 55.00 | 31.00 | 46.00 | 14.00 | 31.00 | 27.00 | 39.00 | 3.00 | 4.00 | 41.00 | 39.00 | 49.00 | 5.00 | 0.00 |
| barycenter | 7.69 | 69.23 | 9.00 | 15.00 | 56.00 | 26.00 | 39.00 | 20.00 | 22.00 | 11.11 | 29.00 | 1.01 | 1.01 | 49.00 | 50.00 | 47.00 | 53.71 | 47.61 |
| radius | 68.75 | 87.50 | 12.00 | 23.00 | 66.00 | 47.00 | 56.00 | 26.00 | 34.00 | 35.00 | 52.00 | 1.00 | 2.00 | 63.00 | 58.00 | 68.00 | 5.00 | 1.00 |
| topological_sort | 60.00 | 48.00 | 10.00 | 14.00 | 76.00 | 38.00 | 67.00 | 25.00 | 25.00 | 21.00 | 64.00 | 6.00 | 5.00 | 74.00 | 71.00 | 78.00 | 73.00 | 74.00 |
| periphery | 29.41 | 58.82 | 1.00 | 3.00 | 33.00 | 16.00 | 22.00 | 1.00 | 11.00 | 6.00 | 25.00 | 0.00 | 0.00 | 27.00 | 29.00 | 31.00 | 50.06 | 47.78 |
| betweenness_centrality | 18.18 | 50.00 | 4.00 | 4.00 | 38.00 | 30.00 | 39.00 | 24.00 | 1.00 | 5.00 | 6.00 | 1.00 | 2.00 | 38.00 | 37.00 | 39.00 | 7.00 | 4.00 |
| triangles | 35.29 | 58.82 | 13.00 | 4.00 | 54.00 | 48.00 | 67.00 | 12.00 | 30.00 | 21.00 | 54.00 | 0.00 | 0.00 | 42.00 | 40.00 | 79.00 | 43.00 | 55.00 |
| avg_neighbor_degree | 66.67 | 61.11 | 16.00 | 17.00 | 36.00 | 55.00 | 68.00 | 26.00 | 30.00 | 29.00 | 58.00 | 3.00 | 6.00 | 31.00 | 36.00 | 82.00 | 62.00 | 64.00 |
| harmonic_centrality | 7.69 | 84.62 | 2.00 | 3.00 | 17.00 | 15.00 | 19.00 | 3.00 | 5.00 | 8.00 | 37.00 | 1.00 | 2.00 | 9.00 | 7.00 | 30.00 | 7.00 | 22.00 |
| bridges | 0.00 | 9.09 | 1.00 | 0.00 | 44.00 | 9.00 | 16.00 | 0.00 | 3.00 | 1.00 | 5.00 | 0.00 | 0.00 | 42.00 | 40.00 | 23.00 | 28.57 | 29.92 |
| isomophic_mapping | 0.00 | 4.00 | 0.00 | 0.00 | 10.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 11.00 | 11.00 | 12.00 | 1.00 | 1.00 |
| global_efficiency | 4.76 | 71.43 | 0.00 | 1.00 | 10.00 | 3.00 | 1.00 | 0.00 | 0.00 | 2.00 | 11.00 | 0.00 | 0.00 | 4.00 | 4.00 | 2.00 | 1.00 | 3.00 |
| maximal_independent_set | 5.56 | 55.56 | 2.00 | 1.00 | 26.00 | 2.00 | 13.00 | 2.00 | 2.00 | 3.00 | 31.00 | 0.00 | 0.00 | 19.00 | 24.00 | 79.00 | 7.00 | 13.00 |
| maximum_flow | 0.00 | 80.95 | 5.00 | 2.00 | 8.00 | 10.00 | 7.00 | 3.00 | 6.00 | 1.10 | 12.00 | 3.30 | 5.49 | 9.00 | 7.00 | 9.00 | 4.00 | 10.00 |
| wiener_index | 0.00 | 73.68 | 0.00 | 1.00 | 14.00 | 6.00 | 8.00 | 0.00 | 4.00 | 4.00 | 22.00 | 0.00 | 0.00 | 6.00 | 3.00 | 13.00 | 7.00 | 7.00 |
| hamiltonian_path | 0.00 | 4.76 | 0.00 | 1.00 | 11.00 | 3.00 | 2.00 | 1.00 | 2.00 | 1.09 | 3.00 | 0.00 | 0.00 | 10.00 | 10.00 | 5.00 | 5.00 | 12.00 |
| min_vertex_cover | 13.04 | 60.87 | 1.00 | 3.00 | 22.00 | 8.00 | 21.00 | 3.00 | 9.00 | 6.00 | 21.00 | 0.00 | 0.00 | 16.00 | 18.00 | 42.00 | 10.00 | 21.00 |

# N Discussion on Reward Weighting

In Section C, we analyze the factor of data mixture by introducing a model G1-Hard-3B trained exclusively on *Hard* and *Challenging* tasks. We observe that G1-Hard-3B effectively improves performance on hard tasks, while on easier tasks still lags behind G1-3B (Table 23).

In this section, we further explore a *soft* data mixture strategy that scales the reward for each task according to its difficulty. In detail, we fix the scaling factor $s$ as 0.2, 0.4, 0.6, and 0.8 for *Easy*, *Medium*, *Hard* and *Challenging* tasks, respectively, and name the resulting model as G1-Soft-3B. As shown in Table 23, G1-Soft-3B achieves a balance between G1-3B and G1-Hard-3B. On easy tasks, G1-Soft-3B largely surpasses G1-Hard-3B and is on par with G1-3B which applies uniform scaling across all tasks. For hard tasks, G1-Soft-3B outperforms G1-3B (*e.g.*, 11.71% v.s 7.57% for *Challenging* tasks), but there is still a gap to G1-Hard-3B. The results show the soft scaling method take effects, but the RL optimization remains dominated by easy tasks. This suggests that further reducing the reward scaling factor for easy tasks or a dynamic weighting strategy could be beneficial—a direction we leave for future work.

Table 23: Test accuracy (%) on our benchmark. $\star$ denotes the tasks are excluded in model training. G1-Hard-3B is only RL-trained on Hard and Challenging tasks. G1-Soft-3B is trained on all tasks but with different reward scaling factors based on the task difficulty.

| **Category** | **Model** | Easy | Medium | Hard | Challenging | Average |
|---|---|---|---|---|---|---|
| **Base Model** | Qwen2.5-3B-Instruct | 45.71 | 30.18 | 9.44 | 1.29 | 22.72 |
| **Ours** | Direct-SFT-3B | 74.43 | 75.27 | <u>43.69</u> | <u>14.43</u> | 53.78 |
| | G1-3B | <u>94.86</u> | **84.64** | 41.25 | 7.57 | <u>59.76</u> |
| | G1-Hard-3B | 69.36$^\star$ | 70.64$^\star$ | **48.50** | **17.43** | 53.30 |
| | G1-Soft-3B | **96.07** | <u>83.55</u> | 40.88 | 11.71 | **60.38** |

# O Detailed Description of Erdős

## O.1 Comparing Erdős with Other Graph Reasoning Benchmarks for LLMs

There is a growing interest in evaluating LLMs' graph reasoning abilities. NLGraph [41] evaluate LLMs on graph-theoretic tasks and discover preliminary yet brittle reasoning abilities in the face of spurious correlations and large graphs. Later, GraphArena [37] and GraCoRe [55] include a broader task coverage and recently released LLMs, finding that even OpenAI o1-mini struggles a lot with complex tasks. Moreover, GraphEval2000 [46] and ProGraph [23] emphasize code-oriented problem solving using library-based prompts, and GraphOmni [48] unify varying graph types, encodings, and prompt styles for a comprehensive evaluation. Overall, these benchmarks suggest that LLMs overall demonstrate moderate success on simple tasks but struggle with abstraction, generalization, and larger or more complex graph instances. Nevertheless, these datasets are either too small (e.g., thousands of examples) or not diverse enough (e.g., 8 tasks in NLGraph) for training general-purpose graph reasoners, which motivates the design of Erdős. We show the detailed comparison of existing graph reasoning benchmarks for LLM with our Erdősin Table 24.

Table 24: Comparison of existing graph-theoretic reasoning benchmarks for LLM with our Erdős.

| Benchmark | #Tasks | # Q-A Samples | Graph Types | Node Size |
|---|---|---|---|---|
| NLGraph [41] | 8 | 5,902 | Synthetic | 5 to 35 |
| GraphWiz [4] | 9 | 3,600 | Synthetic | 2 to 100 |
| GraphArena [37] | 10 | 10,000 | Real-world | 4 to 50 |
| GraCoRe [55] | 19 | 5,140 | Synthetic & Real-world | 8 to 30 |
| GraphOmni [48] | 6 | 241,726 | Synthetic | 5 to 30 |
| Erdős**(ours)** | 50 | 100,000 | Real-world | 5 to 35 |

### O.2 Full list of tasks in Erdős

Table 25: Benchmark exmaples

| Task | Prompt | Answer |
|---|---|---|
| **adamic adar index** | The task is to determine the Adamic-Adar index of two nodes in a graph.<br>The Adamic-Adar index is the sum of the inverse logarithm of the degrees of the common neighbors of the two nodes.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 9. The edges are: (1, 5), (1, 4), (1, 8), (1, 2), (1, 3), (1, 7), (5, 2), (5, 3), (5, 4), (5, 9), (5, 6), (4, 8), (4, 9), (4, 7), (8, 2), (8, 3), (8, 6), (8, 7), (8, 9), (2, 3), (2, 7), (2, 6), (3, 9), (3, 7), (7, 6), (7, 9).<br>Question: What is the Adamic-Adar index between node 4 and node 6?<br>You need to format your answer as a float number. | 1.5859 |
| **avg neighbor degree** | The task is to determine the average degree of the neighbors of a node in the graph.<br>Here is an undirected graph containing nodes from 1 to 8. The edges are: (1, 7), (1, 8), (1, 4), (7, 8), (8, 5), (2, 3), (2, 6), (3, 5).<br>Question: What is the average neighbor degree of node 2 in the graph?<br>You need to format your answer as a float number. | 1.5 |
| **barycenter** | The task is to determine the barycenter of a graph.<br>The barycenter of a graph is also called the median. It includes the node that minimizes the sum of shortest path lengths to all other nodes.<br>The input graph is guaranteed to be connected.<br>Here is an undirected graph containing nodes from 1 to 7. The edges are: (1, 2), (1, 6), (1, 5), (1, 7), (1, 4), (2, 6), (2, 5), (2, 7), (2, 4), (6, 4), (6, 5), (6, 7), (7, 3), (7, 4).<br>Question: What is the barycenter of the graph?<br>You need to format your answer as a list of nodes in ascending order, e.g., [node-1, node-2, ..., node-n]. | [1, 2, 6, 7] |
| **betweenness centrality** | The task is to determine the betweenness centrality of a node in the graph.<br>Betweenness centrality of a node *u* is the sum of the fraction of all-pairs shortest paths that pass through *u*.<br>Here is an undirected graph containing nodes from 1 to 9. The edges are: (1, 6), (1, 4), (1, 8), (1, 9), (6, 2), (6, 7), (4, 7), (4, 5), (8, 3), (8, 5), (8, 7), (9, 3), (9, 5), (2, 7).<br>Question: What is the betweenness centrality of node 5 in the graph?<br>You need to format your answer as a float number. | 0.0679 |
| **bfs** | The task is to determine the breadth-first search (BFS) traversal order given a starting node.<br>Stop when the BFS cannot be continued.<br>Here is an undirected graph containing nodes from 1 to 7. The edges are: (1, 2), (1, 5), (2, 3), (2, 4), (5, 3), (5, 4), (3, 4), (4, 7), (7, 6).<br>Question: What is the breadth-first search (BFS) traversal order for the starting node 1?<br>You need to format your answer as a list of edges, e.g., [(u1, v1), (u2, v2), ..., (un, vn)]. | [(1, 2), (1, 5), (2, 3), (2, 4), (4, 7), (7, 6)] |

Continuing table 25

| Task | Prompt | Answer |
|---|---|---|
| **bipartite maximum matching** | The task is to determine the maximal matching in a bipartite graph.<br>The input graph is guaranteed to be a bipartite graph.<br>Here is an undirected graph containing nodes from 1 to 4. The edges are: (1, 3), (1, 4), (2, 3), (2, 4).<br>Question: What is the bipartite maximal matching of the bipartite graph?<br>You need to format your answer as a list of edges in ascending dictionary order, e.g., [(u1, v1), (u2, v2), ..., (un, vn)]. | [(1, 3), (2, 4)] |
| **bridges** | The task is to find all bridges of a graph.<br>A bridge is an edge in a graph whose removal increases the number of connected components.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 5. The edges are: (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5).<br>Question: What are the bridges of the graph?<br>You need to format your answer as a list of edges in ascending dictionary order, e.g., [(u1, v1), (u2, v2), ..., (un, vn)]. | [] |
| **center** | The task is to determine the center of a graph.<br>The center of a graph includes the node that minimizes the maximum distance to any other nodes in the graph.<br>The input graph is guaranteed to be connected.<br>Here is an undirected graph containing nodes from 1 to 6. The edges are: (1, 5), (5, 2), (2, 6), (6, 4), (3, 4).<br>Question: What is the center of the graph?<br>You need to format your answer as a list of nodes in ascending order, e.g., [node-1, node-2, ..., node-n]. | [2, 6] |
| **closeness centrality** | The task is to determine the closeness centrality of a node in the graph.<br>For a node *u*, closeness centrality is the reciprocal of the average shortest path distance to *u* over all *n-1* reachable nodes. For directed graphs, it computes the incoming distance to *u*.<br>Here is an undirected graph containing nodes from 1 to 8. The edges are: (1, 3), (3, 6), (2, 8), (2, 6), (8, 6), (8, 7), (4, 7), (7, 5).<br>Question: What is the closeness centrality of node 2 in the graph?<br>You need to format your answer as a float number. | 0.4667 |
| **clustering coefficient** | The task is to compute the clustering coefficient for a given node.<br>For unweighted graphs, the clustering of a node is the fraction of possible triangles through that node that exist.<br>Here is an undirected graph containing nodes from 1 to 7. The edges are: (1, 4), (1, 5), (1, 3), (4, 2), (4, 3), (4, 5), (4, 6), (4, 7), (5, 2), (5, 3), (5, 6), (5, 7), (2, 6), (2, 7), (6, 7).<br>Question: What is the clustering coefficient of node 6?<br>You need to format your answer as a float number. | 1.0 |
| **common neighbor** | The task is to determine common neighbors between two nodes in the graph.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 7. The edges are: (1, 7), (1, 6), (1, 4), (1, 5), (7, 2), (7, 3), (6, 2), (4, 3), (5, 3).<br>Question: What are the common neighbors between node 2 and node 3?<br>You need to format your answer as a list of nodes in ascending order, e.g., [node-1, node-2, ..., node-n]. | [7] |

Continuing table 25

| Task | Prompt | Answer |
|---|---|---|
| **connected component number** | The task is to determine the number of connected components in an undirected graph.<br>A connected component is a subgraph where any two nodes are connected to each other by paths.<br>Here is an undirected graph containing nodes from 1 to 10. The edges are: (1, 4), (1, 7), (1, 5), (1, 9), (1, 10), (1, 6), (1, 2), (4, 2), (4, 3), (4, 8), (4, 5), (4, 9), (4, 10), (7, 2), (7, 3), (7, 5), (7, 6), (7, 8), (7, 9), (5, 2), (5, 3), (5, 8), (5, 9), (5, 10), (9, 2), (9, 3), (9, 6), (9, 8), (9, 10), (10, 2), (10, 3), (10, 8), (6, 2), (6, 3), (6, 8), (2, 8), (2, 3).<br>Question: How many connected components are there in the graph?<br>Your answer should be an integer. | 1 |
| **degree** | The task is to determine the degree of a node in the graph.<br>For the undirected graph, you should count the edge between two nodes only once.<br>Here is an undirected graph containing nodes from 1 to 6. The edges are: (1, 6), (6, 5), (2, 3), (2, 4), (3, 5).<br>Question: What is the degree of node 6 in the graph?<br>Your answer should be an integer. | 2 |
| **degree centrality** | The task is to determine the degree centrality of a node in the graph.<br>Degree centrality for a node is the fraction of nodes it is connected to.<br>Here is an undirected graph containing nodes from 1 to 7. The edges are: (1, 2), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (2, 6), (4, 3), (4, 5), (4, 7), (5, 3).<br>Question: What is the degree centrality of node 3 in the graph?<br>You need to format your answer as a float number. | 0.5 |
| **density** | The task is to determine the density of the graph.<br>Density is defined as the ratio of the number of edges in the graph to the number of possible edges.<br>Here is an undirected graph containing nodes from 1 to 5. The edges are: (1, 2), (1, 3), (2, 3), (2, 4), (2, 5), (3, 4), (4, 5).<br>Question: What is the density of the graph?<br>You need to format your answer as a float number. | 0.7 |
| **dfs** | The task is to determine the depth-first search (DFS) traversal order given a starting node.<br>Stop when the DFS cannot be continued.<br>Here is an undirected graph containing nodes from 1 to 9. The edges are: (1, 2), (1, 3), (1, 6), (3, 9), (4, 8), (4, 5), (8, 7).<br>Question: What is the depth-first search (DFS) traversal order for the starting node 1?<br>You need to format your answer as a list of edges, e.g., [(u1, v1), (u2, v2), ..., (un, vn)]. | [(1, 2), (1, 3), (3, 9), (1, 6)] |
| **diameter** | The task is to determine the diameter of a graph.<br>The diameter of a graph is the longest shortest path between any two nodes in the graph.<br>The input graph is guaranteed to be connected.<br>Here is an undirected graph containing nodes from 1 to 7. The edges are: (1, 5), (1, 7), (1, 4), (5, 6), (2, 6), (2, 3).<br>Question: What is the diameter of the graph?<br>You need to format your answer as a float number. | 5 |

Continuing table 25

| Task | Prompt | Answer |
|---|---|---|
| **dominating set** | The task is to determine the dominating set of a graph.<br>A dominating set is a subset of nodes such that every node in the graph is either in the set or adjacent to a node in the set.<br>For directed graphs, any node not in the dominating set must be a successor of a node within the set.<br>Here is an undirected graph containing nodes from 1 to 7. The edges are: (1, 2), (1, 5), (1, 6), (1, 7), (2, 3), (2, 4), (5, 6), (7, 3), (7, 4).<br>Question: What is the dominating set of the graph?<br>You need to format your answer as a list of nodes in ascending order, e.g., [node-1, node-2, ..., node-n]. | [1, 3, 4] |
| **edge existence** | The task is to determine if there is an edge connecting two nodes.<br>For an undirected graph, determine if there is an edge between nodes *u* and *v*. For a directed graph, determine if there is an edge from *u* to *v*.<br>Here is an undirected graph containing nodes from 1 to 8. The edges are: (1, 2), (1, 6), (3, 8), (3, 4), (8, 4), (8, 5), (8, 7), (4, 7), (4, 5), (7, 5).<br>Question: Is there an edge between node 5 and node 3?<br>Your answer should be Yes or No. | No |
| **edge number** | The task is to determine the number of edges in the graph.<br>For the undirected graph, you should count the edge between two nodes only once.<br>Here is an undirected graph containing nodes from 1 to 10. The edges are: (1, 10), (1, 8), (10, 7), (8, 6), (2, 5), (2, 4), (2, 6), (5, 4), (5, 9), (4, 3), (4, 9), (3, 7).<br>Question: How many edges are there in the graph?<br>Your answer should be an integer. | 12 |
| **global efficiency** | The task is to determine the global efficiency of a graph.<br>Global efficiency is the average efficiency of all pairs of nodes.<br>The efficiency of a pair of nodes is the multiplicative inverse of the shortest path distance between the nodes.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 7. The edges are: (1, 5), (1, 4), (5, 2), (2, 7), (7, 3), (3, 6).<br>Question: What is the global efficiency of the graph?<br>You need to format your answer as a float number. | 0.5310 |
| **hamiltonian path** | The task is to return a Hamiltonian path in a directed graph.<br>A Hamiltonian path is a path in a directed graph that visits each vertex exactly once.<br>The input graph is guaranteed to be directed and tournable.<br>Here is a directed graph containing nodes from 1 to 8. The edges are: (2, 1), (2, 4), (2, 5), (2, 6), (2, 7), (1, 3), (1, 4), (1, 7), (3, 2), (3, 7), (3, 8), (4, 3), (4, 5), (4, 7), (5, 1), (5, 3), (5, 8), (6, 1), (6, 3), (6, 4), (6, 5), (7, 5), (7, 6), (8, 1), (8, 2), (8, 4), (8, 6), (8, 7).<br>Question: Return a Hamiltonian path in the graph.<br>You need to format your answer as a list of nodes, e.g., [node-1, node-2, ..., node-n]. | [2, 1, 4, 5, 3, 8, 7, 6] |

Continuing table 25

| Task | Prompt | Answer |
|---|---|---|
| **harmonic centrality** | The task is to determine the harmonic centrality of a node in the graph.<br>Harmonic centrality of a node *u* is the sum of the reciprocal of the shortest path distances from all other nodes to u.<br>Here is a directed graph containing nodes from 1 to 8. The edges are: (6, 2), (6, 1), (6, 4), (6, 5), (6, 3), (7, 8).<br>Question: What is the harmonic centrality of node 3 in the graph?<br>You need to format your answer as a float number. | 1.0 |
| **has cycle** | The task is to determine if the graph has a cycle.<br>Here is an undirected graph containing nodes from 1 to 9. The edges are: (1, 2), (1, 4), (1, 5), (2, 4), (2, 5), (4, 9), (5, 3), (3, 6), (3, 8), (6, 8), (9, 7).<br>Question: Does the graph have a cycle?<br>Your answer should be Yes or No. | Yes |
| **is bipartite** | The task is to determine if the graph is bipartite.<br>A bipartite graph is a graph whose nodes can be divided into two disjoint sets such that no two graph vertices within the same set are adjacent.<br>Here is an undirected graph containing nodes from 1 to 6. The edges are: (1, 4), (4, 3), (2, 5), (2, 3), (5, 6), (3, 6).<br>Question: Is the graph bipartite?<br>Your answer should be Yes or No. | Yes |
| **is eularian** | The task is to determine if the graph is Eulerian.<br>An Eulerian graph is a graph that contains an Eulerian circuit, which is a cycle that visits every edge exactly once.<br>Here is an undirected graph containing nodes from 1 to 6. The edges are: (1, 5), (1, 3), (1, 2), (1, 4), (5, 2), (3, 2), (3, 4), (3, 6), (2, 4), (4, 6).<br>Question: Is the graph Eulerian?<br>Your answer should be Yes or No. | Yes |
| **is regular** | The task is to determine if the graph is regular.<br>A regular graph is a graph where every node has the same degree.<br>Here is an undirected graph containing nodes from 1 to 10. The edges are: (1, 5), (1, 7), (1, 10), (5, 2), (5, 10), (7, 8), (7, 10), (3, 9), (3, 8), (3, 4), (9, 4), (4, 6).<br>Question: Is the graph regular?<br>Your answer should be Yes or No. | No |
| **is tournament** | The task is to determine if the graph is a tournament.<br>A tournament is a directed graph where every pair of nodes is connected by a single directed edge.<br>The input graph is guaranteed to be directed.<br>Here is a directed graph containing nodes from 1 to 10. The edges are: (1, 2), (2, 1), (2, 4), (4, 2), (4, 3), (3, 1), (5, 2), (5, 4), (6, 2), (6, 5), (7, 8), (8, 6), (9, 7), (10, 7).<br>Question: Is the graph a tournament?<br>Your answer should be Yes or No. | No |

Continuing table 25

| Task | Prompt | Answer |
|---|---|---|
| **isomophic mapping** | Given a pair of isomorphic graphs, determine the node correspondence between the two graphs.<br>The first graph is: G describes an undirected graph among 0, 1, 2, 3, 4, 5, and 6. In this graph: Node 0 is connected to nodes 6, 3, 4. Node 1 is connected to nodes 4, 5, 6. Node 2 is connected to nodes 3, 4. Node 3 is connected to nodes 0, 2, 5. Node 4 is connected to nodes 0, 1, 2. Node 5 is connected to nodes 1, 3. Node 6 is connected to nodes 0, 1.<br>The second graph is: G describes an undirected graph among 102, 106, 105, 101, 103, 100, and 104. In this graph: Node 100 is connected to nodes 106, 101. Node 101 is connected to nodes 102, 105, 100. Node 102 is connected to nodes 104, 101, 103. Node 103 is connected to nodes 102, 106, 105. Node 104 is connected to nodes 102, 106. Node 105 is connected to nodes 101, 103. Node 106 is connected to nodes 103, 100, 104.<br>Provide a node matching dictionary such as {Graph1 #Node1: Graph2 #Node1, Graph1 #Node2: Graph2 #Node2, ...} | {0: 102, 3: 101, 2: 105, 4: 103, 1: 106, 5: 100, 6: 104} |
| **jaccard coefficient** | The task is to determine the Jaccard coefficient of two nodes in a graph.<br>The Jaccard coefficient is the size of the intersection divided by the size of the union of the neighbors of the two nodes.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 5. The edges are: (1, 2), (1, 3), (2, 5), (2, 3), (3, 5), (5, 4).<br>Question: What is the Jaccard coefficient between node 2 and node 4?<br>You need to format your answer as a float number. | 0.3333 |
| **local connectivity** | The task is to determine the local connectivity of two nodes in the graph.<br>Local connectivity is whether there exists at least one path between the two nodes.<br>Here is a directed graph containing nodes from 1 to 7. The edges are: (1, 7), (7, 6), (3, 1), (4, 3), (5, 4), (6, 2).<br>Question: What is the local connectivity between node 7 and node 4 in the graph?<br>Your answer should be Yes or No. | No |
| **max weight matching** | The task is to determine the maximum weight matching of a graph.<br>A matching is a set of edges without common vertices. A maximal matching cannot add more edges and still be a matching. The weight of a matching is the sum of the weights of its edges. If not specified, all edges have equal edge weights.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 7. The edges are: (1, 7), (7, 5), (2, 4), (2, 5), (4, 3), (3, 6).<br>Question: What is the maximum weight matching of the graph?<br>You need to format your answer as a list of edges in ascending dictionary order, e.g., [(u1, v1), (u2, v2), ..., (un, vn)]. | [(2, 4), (5, 7), (6, 3)] |

Continuing table 25

| Task | Prompt | Answer |
|---|---|---|
| **maximal independent set** | The task is to determine the maximal independent set guaranteed to contain a given node in the graph.<br>An independent set is a set of nodes such that the subgraph induced by these nodes contains no edges. A maximal independent set is an independent set such that it is not possible to add a new node and still get an independent set.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 6. The edges are: (1, 2), (1, 6), (1, 3), (2, 3), (2, 4), (2, 5), (3, 5), (4, 5).<br>Question: What is the maximal independent set that includes node 4 of the graph?<br>You need to format your answer as a list of nodes in ascending order, e.g., [node-1, node-2, ..., node-n]. | [3, 4, 6] |
| **maximum flow** | The task is to determine the value of the maximum flow for the given source node and sink node.<br>The maximum flow is the greatest amount of flow that can be sent from the source to the sink without violating capacity constraints.<br>Here is a directed graph containing nodes from 1 to 5. The edges are: (2, 5, 8), (3, 1, 9), (3, 5, 3), (4, 2, 4). (u ,v, w) denotes the edge from node *u* to node *v* has a capacity of *w*.<br>Question: What is the value of the maximum flow from node 3 to node 2?<br>You need to format your answer as a float number. | 0.0 |
| **min edge covering** | The task is to determine the minimum edge covering of a graph.<br>An edge cover is a set of edges such that every vertex in the graph is incident to at least one edge in the set. The minimum edge cover is the edge cover with the smallest number of edges.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 9. The edges are: (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (2, 5), (3, 4), (3, 6), (3, 7), (3, 8), (3, 5), (4, 7), (4, 8), (5, 6), (5, 7), (6, 7), (6, 9), (7, 9).<br>Question: What is the minimum edge covering of the graph?<br>You need to format your answer as a list of edges in ascending dictionary order, e.g., [(u1, v1), (u2, v2), ..., (un, vn)]. | [(2, 1), (5, 2), (7, 4), (8, 3), (9, 6)] |
| **min vertex cover** | The task is to determine the minimum vertex cover of a graph.<br>A vertex cover is a set of nodes such that every edge in the graph is incident to at least one node in the set.<br>Here is an undirected graph containing nodes from 1 to 5. The edges are: (1, 2), (2, 3), (3, 5), (5, 4).<br>Question: What is the minimum vertex cover of the graph?<br>You need to format your answer as a list of nodes in ascending order, e.g., [node-1, node-2, ..., node-n]. | [2, 5] |
| **minimum spanning tree** | The task is to determine the minimum spanning tree of a graph.<br>A minimum spanning tree is a subset of the edges that connects all vertices in the graph with the minimum possible total edge weight. If not specified, all edges have equal edge weights.<br>The input graph is guaranteed to be undirected and connected.<br>Here is an undirected graph containing nodes from 1 to 9. The edges are: (1, 2), (1, 8), (1, 5), (1, 6), (1, 4), (1, 7), (1, 9), (2, 5), (2, 6), (2, 4), (2, 7), (2, 3), (8, 3), (8, 4), (8, 6), (8, 7), (5, 3), (5, 4), (5, 6), (5, 7), (5, 9), (6, 3), (6, 4), (6, 7), (6, 9), (4, 3), (4, 7), (4, 9), (7, 9), (9, 3).<br>Question: What is the minimum spanning tree of the graph?<br>You need to format your answer as a list of edges in ascending dictionary order, e.g., [(u1, v1), (u2, v2), ..., (un, vn)]. | [(1, 2), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (2, 3)] |

Continuing table 25

| Task | Prompt | Answer |
|---|---|---|
| **neighbor** | The task is to determine the neighbors of a node in the graph.<br>For directed graph, you should return the successors of the node.<br>Here is an undirected graph containing nodes from 1 to 10. The edges are: (1, 3), (1, 9), (1, 6), (1, 7), (3, 2), (3, 8), (3, 9), (6, 7), (2, 10), (10, 8), (4, 5).<br>Question: What are the neighbors of node 2 in the graph?<br>You need to format your answer as a list of nodes in ascending order, e.g., [node-1, node-2, ..., node-n]. | [3, 10] |
| **node number** | The task is to determine the number of nodes in the graph.<br>Here is an undirected graph containing nodes from 1 to 10. The edges are: (1, 10), (1, 3), (10, 6), (10, 8), (3, 7), (3, 4), (2, 7), (2, 5), (2, 9), (5, 9), (5, 8), (9, 4), (8, 6).<br>Question: How many nodes are there in the graph?<br>Your answer should be an integer. | 10 |
| **periphery** | The task is to determine the periphery of a graph.<br>The periphery of a graph is the set of nodes with the maximum eccentricity. The eccentricity of a node is the maximum distance from this node to all other nodes in the graph.<br>The input graph is guaranteed to be connected.<br>Here is an undirected graph containing nodes from 1 to 6. The edges are: (1, 3), (3, 2), (3, 4), (3, 5), (3, 6).<br>Question: What is the periphery of the graph?<br>You need to format your answer as a list of nodes in ascending order, e.g., [node-1, node-2, ..., node-n]. | [1, 2, 4, 5, 6] |
| **radius** | The task is to determine the radius of a graph.<br>The radius of a graph is the minimum eccentricity of any node in the graph. The eccentricity of a node is the maximum distance from this node to all other nodes in the graph.<br>The input graph is guaranteed to be connected.<br>Here is an undirected graph containing nodes from 1 to 5. The edges are: (1, 2), (2, 3), (3, 4), (3, 5), (4, 5).<br>Question: What is the radius of the graph?<br>You need to format your answer as a float number. | 2 |
| **resource allocation index** | The task is to determine the resource allocation index of two nodes in a graph.<br>The resource allocation index of two nodes is the sum of the inverse of the degrees of the common neighbors of the two nodes.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 5. The edges are: (1, 2), (1, 3), (2, 3), (3, 4), (3, 5), (4, 5).<br>Question: What is the resource allocation index between node 1 and node 4?<br>You need to format your answer as a float number. | 0.25 |
| **shortest path** | The task is to determine the shortest path between two nodes.<br>The input nodes are guaranteed to be connected.<br>Here is an undirected graph containing nodes from 1 to 6. The edges are: (1, 2), (1, 3), (2, 4), (2, 3), (2, 5), (3, 4), (3, 5), (4, 6).<br>Question: What is the shortest path between node 1 and node 6?<br>You need to format your answer as a list of nodes, e.g., [node-1, node-2, ..., node-n]. | [1, 2, 4, 6] |

Continuing table 25

| Task | Prompt | Answer |
|---|---|---|
| **strongly connected number** | The task is to determine the number of strongly connected components in a directed graph.<br>A strongly connected component is a maximal subgraph where every node is reachable from every other node.<br>Here is a directed graph containing nodes from 1 to 6. The edges are: (2, 5), (5, 1), (3, 4), (6, 2).<br>Question: How many strongly connected components are there in the graph?<br>Your answer should be an integer. | 6 |
| **topological sort** | The task is to determine the topological sort of a directed acyclic graph (DAG).<br>Here is a directed graph containing nodes from 1 to 6. The edges are: (1, 6), (1, 5), (1, 4), (1, 3), (1, 2).<br>Question: What is the topological sort of the directed acyclic graph (DAG)?<br>You need to format your answer as a list of nodes, e.g., [node-1, node-2, ..., node-n]. | [1, 6, 5, 4, 3, 2] |
| **traveling salesman problem** | The task is to determine the minimal cost of the Traveling Salesman Problem (TSP).<br>The Traveling Salesman Problem asks for the shortest possible route that visits each vertex exactly once and returns to the starting vertex.<br>The input graph is guaranteed to be a complete graph.<br>Here is an undirected graph containing nodes from 1 to 8. The edges are: (1, 2, 9), (1, 3, 3), (1, 4, 6), (1, 5, 8), (1, 6, 7), (1, 7, 4), (1, 8, 9), (2, 3, 10), (2, 4, 11), (2, 5, 5), (2, 6, 11), (2, 7, 1), (2, 8, 9), (3, 4, 11), (3, 5, 1), (3, 6, 9), (3, 7, 2), (3, 8, 9), (4, 5, 8), (4, 6, 3), (4, 7, 4), (4, 8, 8), (5, 6, 3), (5, 7, 3), (5, 8, 10), (6, 7, 8), (6, 8, 1), (7, 8, 10). (u ,v, w) denotes the edge from node *u* to node *v* has a weight of *w*.<br>Question: What is the minimal cost of the Traveling Salesman Problem on the graph?<br>You need to format your answer as a float number. | 27.0 |
| **triangles** | The task is to find the number of triangles that include a specific node as one vertex.<br>A triangle is a set of three nodes that are all connected to each other.<br>The input graph is guaranteed to be undirected.<br>Here is an undirected graph containing nodes from 1 to 8. The edges are: (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (4, 5), (4, 6), (4, 7), (4, 8), (5, 6), (5, 7), (5, 8), (6, 7), (6, 8), (7, 8).<br>Question: How many triangles include node 1 in the graph?<br>Your answer should be an integer. | 21 |

Continuing table 25

| Task | Prompt | Answer |
|---|---|---|
| **weighted minimum spanning tree** | The task is to determine the minimum spanning tree of a weighted graph.<br>A minimum spanning tree is a subset of the edges that connects all vertices in the graph with the minimum possible total edge weights. If not specified, all edges have equal edge weights.<br>The input graph is guaranteed to be undirected and connected.<br>Here is an undirected graph containing nodes from 1 to 5. The edges are: (1, 4, 5), (2, 4, 11), (2, 3, 10), (3, 4, 2), (3, 5, 2). (u ,v, w) denotes the edge from node *u* to node *v* has a weight of *w*.<br>Question: What is the minimum spanning tree of the weighted graph?<br>You need to format your answer as a list of edges in ascending dictionary order, e.g., [(u1, v1), (u2, v2), ..., (un, vn)]. | [(1, 4), (2, 3), (3, 4), (3, 5)] |
| **weighted shortest path** | The task is to determine the shortest path between two nodes of a weighted graph.<br>The input nodes are guaranteed to be connected.<br>Here is a directed graph containing nodes from 1 to 8. The edges are: (1, 2, 5), (1, 4, 3), (1, 7, 9), (2, 3, 10), (2, 4, 10), (3, 1, 11), (3, 4, 2), (3, 5, 6), (4, 1, 1), (4, 2, 4), (4, 6, 8), (4, 8, 2), (5, 1, 7), (5, 2, 11), (5, 6, 2), (5, 7, 5), (5, 8, 11), (6, 1, 7), (6, 2, 11), (6, 3, 4), (6, 5, 1), (6, 8, 11), (7, 1, 3), (7, 2, 8), (7, 4, 7), (7, 6, 6), (7, 8, 3), (8, 1, 11), (8, 2, 7), (8, 4, 5), (8, 7, 5). (u ,v, w) denotes the edge from node *u* to node *v* has a weight of *w*.<br>Question: What is the shortest path between node 1 and node 5?<br>You need to format your answer as a list of nodes, e.g., [node-1, node-2, ..., node-n]. | [1, 4, 6, 5] |
| **wiener index** | The task is to determine the Wiener index of a connected graph.<br>The Wiener index of a graph is the sum of the shortest-path distances between each pair of reachable nodes. For pairs of nodes in undirected graphs, only one orientation of the pair is counted.<br>In the input graph, all node pairs are guaranteed to be reachable.<br>Here is an undirected graph containing nodes from 1 to 5. The edges are: (1, 2), (1, 4), (2, 3), (4, 5), (3, 5).<br>Question: What is the Wiener index of the graph?<br>You need to format your answer as a float number. | 15.0 |

# P Discussion

In this paper, we explored the use of Reinforcement Learning to improve LLMs' reasoning abilities on graph reasoning and demonstrate significant improvements across a spectrum of tasks with various difficulty levels, showing that graph reasoning of LLMs can be elicited via RL training (even with only 300 steps). We also comprehensively evaluate the transferability of RL-trained models to unseen graph reasoning tasks, real-world graph tasks, and general reasoning tasks, observing strong zero-shot generalization. These results support our hypothesis that training LLMs on diverse synthetic graph-theoretic tasks via RL offers a scalable, generalizable path toward robust graph reasoning. As a first step, this approach may guide the development of efficient, general-purpose graph reasoners.

In future work, we aim to explore dynamic difficulty scheduling during RL training to address the sample inefficiency issue. On a broader scale, we also plan to extend our approach to the following scenarios: (1) Handling larger graphs with thousands of nodes, aligning with the long-context reasoning challenges. (2) Incorporating visual inputs (*e.g.*, images depicting graphs) to enhance real-world applicability. (3) Adapting `G1` to more practical domains such as logistics, knowledge graph reasoning, and tabular problem-solving, where structured reasoning is critical.