Can LLM Graph Reasoning Generalize beyond Pattern Memorization?

Yizhuo Zhang[*1] **Heng Wang**[*2] **Shangbin Feng**[*1]
Zhaoxuan Tan[3] **Xiaochuang Han**[1] **Tianxing He**[4] **Yulia Tsvetkov**[1]
[1]University of Washington [2]Xi'an Jiaotong University
[3]University of Notre Dame [4]Tsinghua University
mattyz@uw.edu wh2213210554@stu.xjtu.edu.cn shangbin@cs.washington.edu

Abstract

Large language models (LLMs) demonstrate great potential for problems with implicit graphical structures, while recent works seek to enhance the graph reasoning capabilities of LLMs through specialized instruction tuning. The resulting "graph LLMs" are evaluated with in-distribution settings only, thus it remains underexplored whether LLMs are learning generalizable graph reasoning skills or merely memorizing patterns in the synthetic training data. To this end, we propose the NLGIFT benchmark, an evaluation suite of LLM graph reasoning generalization: whether LLMs could go beyond *semantic*, *numeric*, *structural*, *reasoning* patterns in the synthetic training data and improve utility on *real-world* graph-based tasks. Extensive experiments with two LLMs across four graph reasoning tasks demonstrate that while generalization on simple patterns (semantic, numeric) is somewhat satisfactory, LLMs struggle to generalize across reasoning and real-world patterns, casting doubt on the benefit of synthetic graph tuning for real-world tasks with underlying network structures. We explore three strategies to improve LLM graph reasoning generalization, and we find that while post-training alignment is most promising for real-world tasks, empowering LLM graph reasoning to go beyond pattern memorization remains an open research question.[1]

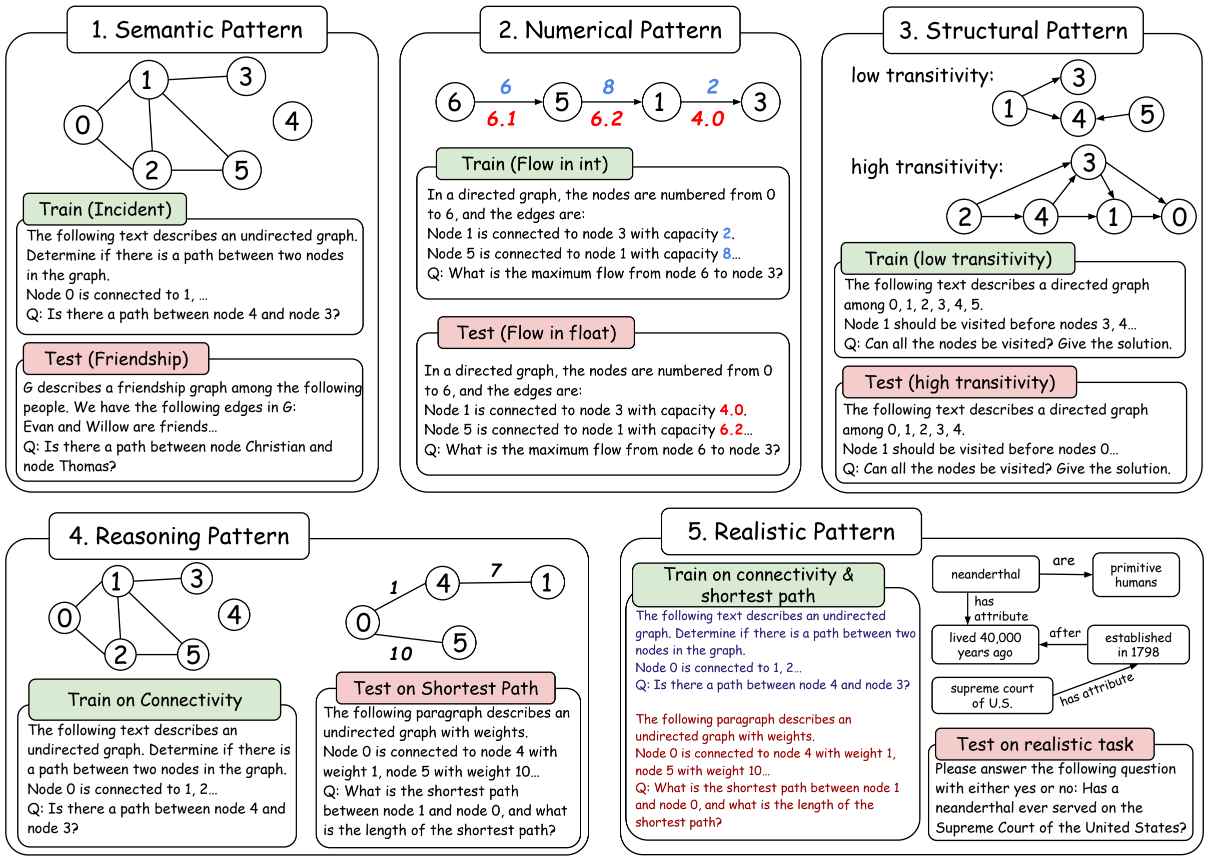
1 Introduction

Large Language Models (LLMs) are increasingly employed for tasks at the intersection of language and structure such as multi-hop QA (Geva et al., 2021; Ding et al., 2023) and structured commonsense reasoning (Sakaguchi et al., 2021; Madaan et al., 2022). These problems are often described in natural language and have implicit graphical structures (Geva et al., 2021; Ding et al., 2023; Saha et al., 2021; Sakaguchi et al., 2021), where LLMs' graph reasoning capabilities are tested. While LLMs *do* possess preliminary abilities to represent and reason with graphs (Wang et al., 2023), they also face challenges such as hallucinations (Huang et al., 2023b) and prompt sensitivity (Fatemi et al., 2024) when dealing with structured data.

Existing works seek to improve LLM graph reasoning mainly through better prompting (Fatemi et al., 2024) or instruction tuning (Wang et al., 2024; Chen et al., 2024a), while the latter line of training-based approaches is generally more effective in producing specialized models for graph-based applications (Tang et al., 2023; Wang et al., 2024; Chen et al., 2024a; Luo et al., 2024). However, these approaches are often evaluated in in-distribution settings, while robust graph reasoners should go beyond training sets to encode general and transferable graph reasoning capabilities in model parameters. Consequently, we ask: *Are LLMs graph reasoners or merely pattern regurgitators?* More concretely, *Can LLM graph reasoning go beyond memorizing patterns in the training data and perform well in out-of-distribution contexts?* The answer to this question has profound implications for LLM reliability in structured contexts since real-world graph-based problems are diverse, heterogeneous, and constantly evolving.

To this end, we propose **NLGIFT**, a comprehensive testbed of **N**atural **L**anguage **G**raph reasoning with sh**ift**ing patterns. NLGIFT contains 37,000 problems in total, where LLMs are instruction tuned on a subset of problems with distribution $\mathcal{D}_{train}$ and evaluated on both in-distribution $\mathcal{D}_{train}$ and out-of-distribution $\mathcal{D}_{test}$ test sets. NLGIFT features five types of patterns where LLMs should generalize beyond: 1) *semantic*, 2) *numerical*, 3) *structural*, 4) *reasoning*, and 5) *real-world patterns*. The first four settings focus on transferring across patterns in the synthetic graph data, while the real-world pattern focuses on the transfer from synthetic

[*]equal contribution
[1]Code and data are publicly available at https://github.com/MatthewYZhang/NLGift.



The following text describes an undirected graph. Determine if there is a path between two nodes in the graph.
Node 0 is connected to 1, ...
Q: Is there a path between node 4 and node 3?

G describes a friendship graph among the following people. We have the following edges in G:
Evan and Willow are friends...
Q: Is there a path between node Christian and node Thomas?

In a directed graph, the nodes are numbered from 0 to 6, and the edges are:
Node 1 is connected to node 3 with capacity 2.
Node 5 is connected to node 1 with capacity 8...
Q: What is the maximum flow from node 6 to node 3?

In a directed graph, the nodes are numbered from 0 to 6, and the edges are:
Node 1 is connected to node 3 with capacity 4.0.
Node 5 is connected to node 1 with capacity 6.2...
Q: What is the maximum flow from node 6 to node 3?

The following text describes a directed graph among 0, 1, 2, 3, 4, 5.
Node 1 should be visited before nodes 3, 4...
Q: Can all the nodes be visited? Give the solution.

The following text describes a directed graph among 0, 1, 2, 3, 4.
Node 1 should be visited before nodes 0...
Q: Can all the nodes be visited? Give the solution.

The following text describes an undirected graph. Determine if there is a path between two nodes in the graph.
Node 0 is connected to 1, 2...
Q: Is there a path between node 4 and node 3?

The following paragraph describes an undirected graph with weights.
Node 0 is connected to node 4 with weight 1, node 5 with weight 10...
Q: What is the shortest path between node 1 and node 0, and what is the length of the shortest path?

The following text describes an undirected graph. Determine if there is a path between two nodes in the graph.
Node 0 is connected to 1, 2...
Q: Is there a path between node 4 and node 3?

The following paragraph describes an undirected graph with weights.
Node 0 is connected to node 4 with weight 1, node 5 with weight 10...
Q: What is the shortest path between node 1 and node 0, and what is the length of the shortest path?

Please answer the following question with either yes or no: Has a neanderthal ever served on the Supreme Court of the United States?

Figure 1: Overview of the NLGIFT Benchmark, featuring five types of graph reasoning patterns that are increasingly challenging in order. We present an example for each pattern to show the transfer from training to test sets.

graph problems to real-world tasks with graph implications such as structured commonsense reasoning (Saha et al., 2021; Sakaguchi et al., 2021) and multi-hop QA (Geva et al., 2021; Ding et al., 2023). These five patterns are increasingly challenging in order and offer a progressive testbed of LLM graph reasoning generalization.

To quantify the success of generalization, NLGIFT establishes two standards: 1) the *basic* standard: *Significant Transfer*, i.e., the tuned model's improvement on out-of-distribution test sets over the zero-shot untuned model is statistically significant; 2) the *strong* standard: *Strong Recovery*, i.e., when evaluated on out-of-distribution data, the tuned LLMs could substantially recover the gains of in-distribution training. Robust graph reasoners should ideally meet both *basic* and *strong* generalization standards across the five patterns.

Extensive experiments on NLGIFT with two LLMs demonstrate that on easier patterns (semantic, numerical, and structural), LLMs achieve the *basic* standard 75% of the time but only reach the *strong* standard in 35% of settings. On reasoning patterns, LLMs only achieve *basic* generalization 33% of the time while *never* qualify for the *strong* standard of generalization. What's worse, on the most challenging real-world patterns, LLMs achieve *basic* generalization in 6% of the settings while graph instruction tuning is counterproductive in 69% of cases, casting doubt on the benefit of synthetic graph data. Further analysis reveals that task composition and keyword frequency in training corpus greatly impact graph reasoning generalization.

We explore three preliminary strategies to augment LLM graph reasoning generalization: *code mixing*, *machine-generated CoTs*, and *post-training alignment*. While post-training alignment is most promising on real-world tasks, empowering LLMs with general and transferable graph reasoning abilities remains an open research question. To sum up, our key contributions include presenting the NLGIFT benchmark, evaluating LLM graph reasoning generalization with diverse tasks and patterns, as well as exploring preliminary solutions to mitigate the profound generalization challenges.

2 NLGIFT Benchmark

To examine whether LLMs are capable of robust reasoning with graph problems rather than memorizing training patterns, we present the NLGIFT Benchmark (Figure 1). Specifically, we select four representative graph reasoning tasks: connectivity, shortest path, topological sort, and maximum flow. We then design five patterns that LLMs should generalize beyond. We present the five patterns in ascending order of difficulty so that they can serve as a progressive testbed.

2.1 Graph reasoning patterns

Semantic Patterns The semantic pattern involves representing the graph problem with different natural language descriptions such as "edge (2,3)" or "Bob and Amy are friends". When trained on one type of semantic representation, robust graph reasoners should achieve similar levels of performance when testing with different semantic representations. Specifically, we employ four typical representation methods: adjacency, incident, graph expert, and friendship (Fatemi et al., 2024).

- *Adjacency*: We list all the edges in the graph with natural language. (i.e., node 1 is connected to node 3. node 1 is connected to 2)
- *Incident*: We describe the connectivity of each node in a single sentence. (i.e., node 1 is connected to 2, 3, 4)
- *Graph Expert*: We employ the prompt "You are a graph analyst" as a prefix, and then use letters to represent the nodes. (i.e., A->B, A->C)
- *Friendship*: We describe the nodes as people and the edges as the friendship between people. (i.e., Alice and Carol are friends)

Numerical Patterns For the numerical pattern, we aim to investigate whether different number distributions in the edge attributes (i.e., edge weight, edge capacity) might affect LLM graph reasoning. Specifically, we employ three different number distributions: small integers (from 1 to 10), large integers (from 11 to 100), and floats with one floating point (from 1.0 to 10.0). Trained on graph problems with one numerical distribution of node/edge attributes, robust graph reasoners should achieve similar performance on another set of problems with different numerical distributions.

Structural Patterns Graphs are often diverse and heterogeneous, featuring varying levels of size, centrality, and other structural variations. We therefore investigate the impact of structural patterns, if changes in graph structural properties might affect the performance of LLM's reasoning abilities. We specifically design three different criteria to quantify a network's structural features.

- *Graph Size*: Small-size graphs have 3 to 10 nodes while large-size graphs have 11 to 25 nodes.
- *Graph Generator*: we use two different graph generator algorithms to generate graphs, specifically, Erdos-Renyi (Erdős et al., 1960) and Barabasi-Albert (Barabási and Albert, 1999).
- *Graph Transitivity*: we calculate the graph transitivity (Luce and Perry, 1949; Wasserman and Faust, 1994) and partition them into low-transitivity and high-transitivity subsets. Graph transitivity (T) is calculated as $\mathrm{T} = \frac{3\times\text{number of triangles}}{\text{number of all triplets}}$.

Reasoning Patterns For the previous three patterns, we train and test LLMs on the same graph tasks, while a robust graph reasoner should learn universal principles of graph reasoning to generalize across different graph reasoning problems. To evaluate this, we use the four graph reasoning tasks (§2), instruction-tune the LLM on one synthetic graph task and evaluate it on both the same task and three other graph tasks.

Real-world Patterns While the previous four patterns all train and test on synthetic graph data, we argue that *the ultimate goal of graph synthetic tuning is to benefit real-world problems with underlying graph structures*: after all, these synthetic problems could be solved with 100% accuracy by conventional algorithms. Thus for the real-world pattern, we fine-tune LLMs with synthetic graph data, and then evaluate with real-world problems that have implicit graph structures. We specifically employ two types of datasets for evaluation:

- *Multi-Hop QA*: Multi-hop QA involves answering questions that require multi-hop reasoning, which is inherently related to synthetic graph problems such as connectivity or shortest path, as the solving process can be viewed as navigating through a network of concepts and relations or trying to find the shortest path between two concepts using existing relations. We adopt StrategyQA (Geva et al., 2021) and Knowledge Crosswords (Ding et al., 2023) for the multi-hop QA task.

train/test	Connectivity				Shortest Path			
	Adjacency	Friendship	Expert	Incident	Adjacency	Friendship	Expert	Incident
LLaMA2-7B								
ADJACENCY	.672 (+29%)	.518 (0%)	.512 (-2%)	.660 (+27%)	.212 (+1225%)	.122 (+578%)	.152 (+850%)	.132 (+725%)
FRIENDSHIP	.500 (-4%)	.694 (+34%)	.686 (+31%)	.540 (+4%)	.028 (+75%)	.242 (+1244%)	.086 (+438%)	.042 (+163%)
EXPERT	.482 (-8%)	.548 (+6%)	.624 (+20%)	.484 (-7%)	.044 (+175%)	.028 (+56%)	.184 (+1050%)	.046 (+188%)
INCIDENT	.662 (+27%)	.512 (-1%)	.502 (-4%)	.728 (+40%)	.132 (+725%)	.084 (+367%)	.124 (+675%)	.244 (+1425%)
ZERO-SHOT	.522	.516	.522	.520	.016	.018	.016	.016
ChatGPT								
ADJACENCY	.958 (+25%)	.826 (+12%)	.880 (+16%)	.912 (+24%)	.542 (+132%)	.516 (+153%)	.512 (+129%)	.546 (+101%)
FRIENDSHIP	.892 (+17%)	.950 (+29%)	.892 (+18%)	.896 (+21%)	.426 (+82%)	.476 (+133%)	.486 (+117%)	.518 (+90%)
EXPERT	.930 (+22%)	.874 (+19%)	.922 (+22%)	.916 (+24%)	.472 (+102%)	.400 (+96%)	.526 (+135%)	.568 (+109%)
INCIDENT	.808 (+6%)	.780 (+6%)	.772 (+2%)	.964 (+31%)	.324 (+38%)	.390 (+91%)	.354 (+58%)	.616 (+126%)
ZERO-SHOT	.764	.736	.756	.738	.234	.204	.224	.272

Table 1: Results for the semantic pattern where colors indicate *Significant Transfer*, *Strong Recovery*, and in-distribution results. LLMs can generalize across different semantic representations of the same graph problems to some extent and achieve *Significant Transfer* 68.8% of the time. The larger ChatGPT is stronger in generalization, achieving *Significant Transfer* and *Strong Recovery* in 21 and 7 cases compared to LLaMA2's 12 and 3.

- *Structured Commonsense Reasoning*: This task tests the intersection of commonsense knowledge and structured reasoning, where LLMs need to figure out the dependency structure of events (*e.g.* open the fridge before taking milk from the fridge) and generate a reasonable plan. This is strongly correlated with synthetic graph problems such as topological sort, where different steps have a constraint of order. We specifically employ ExplaGraphs (Saha et al., 2021) and Proscript (Sakaguchi et al., 2021) for this task.

Finally, we obtain the NLGift Benchmark with 33,000 synthetic problems and 4,000 realistic problems. Details of NLGift and the four real-world datasets can be found in Appendix B.2.

2.2 Generalization Metrics

We instruction-tune a pre-trained language model f_θ with initial parameters θ on one distribution of training data $\mathcal{D}_{train}$. Formally, this is expressed as:

$$\theta_{\mathcal{D}_{train}} = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}_{train}}[L(f_\theta(x), y)],$$

where L represents the loss function, and (x, y) are input-output pairs in instruction tuning. We evaluate the tuned model on a different data distribution $\mathcal{D}_{test}$ to test its generalization ability, which is defined as a *generalization pair*.

$$\mathrm{Acc}(\mathcal{D}_{test} \mid \theta_{\mathcal{D}_{train}}) = \mathbb{E}_{(x,y)\sim\mathcal{D}_{test}}\mathcal{I}(f_{\theta_{\mathcal{D}_{train}}}(x), y),$$

where $\mathcal{I}(f_{\theta_{\mathcal{D}_{train}}}(x), y)$ is the indicator function of whether the $f_{\theta_{\mathcal{D}_{train}}}(x)$ and y have the same reasoning path and reach the same answer.

We use two standards to quantify the success of LLM graph reasoning generalization: 1) *Significant Transfer* and 2) *Strong Recovery*. Concretely, *Significant Transfer* is when the performance of the tuned model on $\mathcal{D}_{test}$ is significantly better than the original untuned model, which is tested by proportional z-test (Seabold and Perktold, 2010):

$$\mathrm{Acc}(\mathcal{D}_{test} \mid \theta_{\mathcal{D}_{train}}) \gg \mathrm{Acc}(\mathcal{D}_{test} \mid \theta) \quad (\mathrm{p} < 0.01)$$

where $\mathrm{Acc}(\mathcal{D}_{test} \mid \theta)$ refers to untuned model's performance on $\mathcal{D}_{test}$ data.

The higher criteria, *Strong Recovery*, means the model's performance on out-of-distribution data should largely match the performance with in-distribution training. Specifically, we define Performance Gap Recovered (PGR) following Burns et al. (2023), and achieving *Strong Recovery* requires that PGR is no less than a threshold parameter λ:

$$\mathrm{PGR} = \frac{\mathrm{Acc}(\mathcal{D}_{test} \mid \theta_{\mathcal{D}_{train}}) - \mathrm{Acc}(\mathcal{D}_{test} \mid \theta)}{\mathrm{Acc}(\mathcal{D}_{test} \mid \theta_{\mathcal{D}_{test}}) - \mathrm{Acc}(\mathcal{D}_{test} \mid \theta)} \geq \lambda$$

In the following experiments, we empirically set the PGR threshold $\lambda = 0.8$ while we also experiment with other values in Appendix A.

3 Experiment Settings

We evaluate the graph reasoning capabilities of two LLMs: ChatGPT (GPT-3.5-TURBO) (Ouyang et al., 2022) and LLaMA2-7B (META-LLAMA/LLAMA-2-7B-CHAT-HF) (Touvron et al., 2023). We empirically employ temperature $\tau = 1$ and $\tau = 0.9$ for the two models respectively to

train/test	Shortest Path			Maximum Flow		
	Small Int	Large Int	Float	Small Int	Large Int	Float
			LLAMA2-7B			
SMALL INT	.330 (+617%)	.330 (+432%)	.294 (+407%)	.168 (+140%)	.080 (+208%)	.060 (+114%) *
LARGE INT	.252 (+448%)	.244 (+294%)	.242 (+317%)	.116 (+66%)	.064 (+146%)	.064 (+129%)
FLOAT	.388 (+743%)	.368 (+494%)	.398 (+586%)	.060 (-14%)	.062 (+138%)	.052 (+86%)
ZERO-SHOT	.046	.062	.058	.070	.026	.028
			CHATGPT			
SMALL INT	.620 (+48%)	.638 (+54%)	.588 (+76%)	.262 (+122%)	.094 (+31%) *	.108 (+35%) *
LARGE INT	.604 (+44%)	.628 (+52%)	.580 (+74%)	.122 (+3%)	.092 (+28%)	.092 (+15%)
FLOAT	.60 (+43%)	.634 (+53%)	.584 (+75%)	.148 (+25%)	.096 (+33%) *	.100 (+25%)
ZERO-SHOT	.420	.414	.334	.118	.072	.080

Table 2: Results for the numerical pattern. In this table, four cells are marked with *, indicating corner cases where they achieve *Strong Recovery* but fail to achieve *Significant Transfer* due to the low-performance in zero-shot settings. For different numerical distributions, easier tasks (the shortest path task) show better transfer results compared to more complex tasks (the maximum flow task).

sample solutions. For CHATGPT instruction tuning, we fine-tune the model for 3 epochs and employ default hyperparameters with the OpenAI fine-tuning API. For LLAMA2-7B instruction tuning, we quantize our model in 4-bit quantization and use QLoRA (Dettmers et al., 2024) for efficient fine-tuning with lora_alpha of 16. We train for 10 epochs with a batch size of 4, and a learning rate of 1e-4 with a warmup ratio of 0.03. For the first four patterns (testing on synthetic graph problems), we use 500 training samples and 500 testing samples. For the real-world pattern, we use 1,000 samples as training data and 1,000 real-world tasks as testing data. We use zero-shot prompting by default and additionally append format instructions for the reasoning and real-world pattern. More details about experiment settings can be found in Appendix B.

4 Results

We evaluate LLM graph reasoning generalization with NLGIFT across diverse data patterns: while generalization on simple patterns (*semantic*, *numerical* and *structural*) is somewhat satisfactory, LLMs struggle to generalize across the more challenging reasoning and real-world patterns, casting doubt on the benefit of synthetic graph tuning for real-world tasks with underlying network structures.

Semantic Patterns We present the results for semantic pattern generalization in Table 1. Out of the 48 generalization pairs, 33 achieved *Significant Transfer* and 10 achieved *Strong Recovery* of in-distribution performance. The larger and more capable CHATGPT can transfer better when testing on out-of-distribution data, with 21 out of 24 achieving *Significant Transfer*, compared to 12 out of 24 for Llama-2-7B. However, even CHATGPT can only achieve 7 out of 24 *Strong Recovery*, indicating that LLMs rely at least partially on natural language patterns to reason on graphs.

Moreover, the semantic representation 'incident' can represent the graph most robustly, with 3 out of 4 best performance when testing on in-distribution data, achieving as much as 17.6% of improvement over other in-distribution test performance. The 'incident' representation shows fewer fluctuations as it achieves 9 out of 12 *Significant Transfer* when testing on out-of-distribution data. This indicates that LLM graph reasoning might be impacted by semantic representations, where some ways of describing graph problems work better than others. We thus employ the 'incident' representation by default in the following experiments.

Numerical Patterns We present the results with the three different number distributions in Table 2. For the shortest path task, both models demonstrate strong performance on in-distribution and out-of-distribution accuracy. However, the maximum flow task is much more challenging since the zero-shot performance is much lower and only 3 out of 12 transfer settings achieve both *Significant Transfer* and *Strong Recovery*, indicating that both models show limited transfer capabilities for harder graph reasoning problems. In addition, the absolute performance gain for in-distribution evaluation is limited for large integer and float distributions (average of 2.6% accuracy increase) compared to

train/test	LLaMA2-7B				ChatGPT			
	Shortest Path		Topological Sort		Shortest Path		Topological Sort	
	Graph Size							
	Small	Large	Small	Large	Small	Large	Small	Large
Small	.656 (+507%)	.208 (+767%)	.884 (+391%)	.022 (+∞) *	.834 (+21%)	.492 (+51%)	.942 (+163%)	.466 (+959%)
Large	.756 (+600%)	.336 (+1300%)	.536 (+198%)	.344 (+∞)	.854 (+23%)	.650 (+99%)	.720 (+101%)	.752 (+1609%)
Zero-shot	.108	.024	.180	.000	.692	.326	.358	.044
	Graph Generator Algorithm							
	ER	BA	ER	BA	ER	BA	ER	BA
ER	.380 (+604%)	.340 (+750%)	.656 (+343%)	.496 (+700%)	.622 (+45%)	.632 (+65%)	.844 (+213%)	.776 (+203%)
BA	.372 (+589%)	.390 (+875%)	.564 (+281%)	.718 (+1058%)	.594 (+39%)	.648 (+69%)	.530 (+96%)	.928 (+263%)
Zero-shot	.054	.040	.148	.062	.428	.384	.270	.256
	Graph Transitivity							
	Low	High	Low	High	Low	High	Low	High
Low	.426 (+407%)	.232 (+955%)	.756 (+278%)	.364 (+1200%)	.692 (+50%)	.420 (+44%)	.886 (+175%)	.658 (+391%)
High	.284 (+238%)	.196 (+791%)	.720 (+260%)	.374 (+1236%)	.624 (+36%)	.482 (+65%)	.924 (+187%)	.814 (+507%)
Zero-shot	.084	.022	.200	.028	.460	.292	.322	.134

Table 3: Results for the structural pattern. The colors have the same meaning as that in Table 1. One cell is marked with *, indicating a corner case where PGR cannot be calculated. The graph size setup has the worst transfer performance, suggesting that graph size has the biggest impact on LLMs reasoning capabilities.

12.1% accuracy increase for small integer evaluation. This indicates that the impact of numerical distributions varies based on task complexity.

Structural Patterns As shown in Table 3, it is evident that among the three different graph structure aspects, *graph size* has the most significant impact on the reasoning abilities of language models, with only 2 out of 8 achieved *Strong Recovery*. Also, the results of training on small graphs and testing on large graphs show that the average PGR is less than 60%. This suggests that it may not be effective to train on small graphs and expect it will generalize to larger graphs. On the other hand, different types of graph generators and different transitivity levels have weak influence on graph reasoning capabilities, with 10 out of 16 showing *Strong Recovery*. This indicates that LLM graph reasoning could transfer across graph types and transitivity, but not size: when creating a synthetic graph training set, it is crucial to include a wide range of problems with varying network sizes.

Reasoning Patterns From Table 4, there is no *Strong Recovery* and only 8 out of 24 *Significant Transfer* achieved. More importantly, the average improvements of out-of-distribution performance are -12% for LLaMA2-7B and 19% for ChatGPT, significantly lower than average in-distribution (>280% for LLaMA2-7B and >100% for ChatGPT). The results show that LLMs might only memorize the reasoning pattern about specific tasks from training data, but cannot successfully transfer general graph reasoning capabilities to other graph reasoning tasks. The two models show entirely different transferring capabilities, making it hard to discuss relationships between tasks.

train/test	Connectivity	Topological Sort	Shortest Path	Maximum Flow
	LLaMA2-7B			
Connectivity	.728 (+40%)	.038 (-74%)	.058 (+7%)	.124 (+77%)
Topological Sort	.470 (-10%)	.656 (+343%)	.008 (-85%)	.028 (-60%)
Shortest Path	.656 (+26%)	.094 (-36%)	.380 (+604%)	.140 (+100%)
Maximum Flow	.584 (+12%)	.052 (-65%)	.032 (-41%)	.168 (+140%)
Zero-shot	.520	.148	.054	.070
	ChatGPT			
Connectivity	.964 (+31%)	.368 (+36%)	.512 (+22%)	.142 (+20%)
Topological Sort	.890 (+21%)	.844 (+213%)	.380 (-10%)	.136 (+15%)
Shortest Path	.804 (+9%)	.328 (+21%)	.620 (+48%)	.130 (+10%)
Maximum Flow	.830 (+12%)	.484 (+79%)	.396 (-6%)	.262 (+122%)
Zero-shot	.738	.270	.420	.118

Table 4: Results for the reasoning pattern. We find very weak or even negative transfer, where only 8 out of 24 cases achieve *Significant Transfer* and 9 cases where out-of-distribution training is counterproductive.

Real-World Patterns The first four patterns are based on synthetic graph data, while the ultimate goal of graph instruction tuning is to boost performance on real-world tasks with implicit graph structures. However, as illustrated in Figure 2, there are barely any improvements after instruction tuned on related synthetic tasks or all synthetic tasks. Also, for some real-world tasks like Proscript, we see a significant drop of an average of 12.5% for both models after instruction tuning. The results suggest that current LLMs struggle to transfer their learned

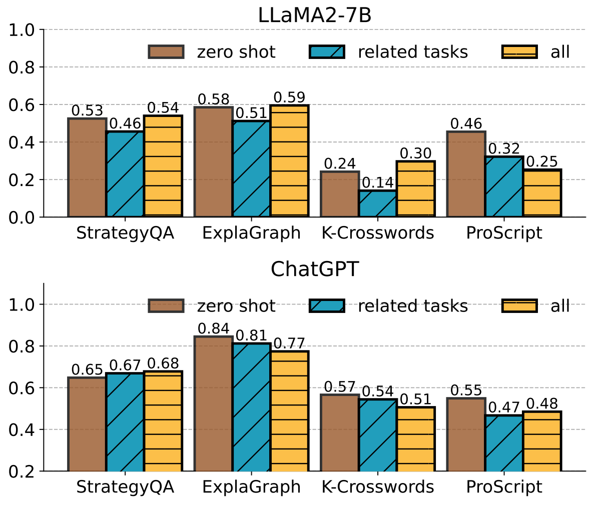


Figure 2: Results for real-world patterns, where the LLM is either untuned (zero shot), tuned with graph tasks related to the real-world problem (related tasks), or on the mixture of all synthetic tasks (all). We find no obvious benefits or even negative transfers of synthetic graph tuning for real-world graphical problems.

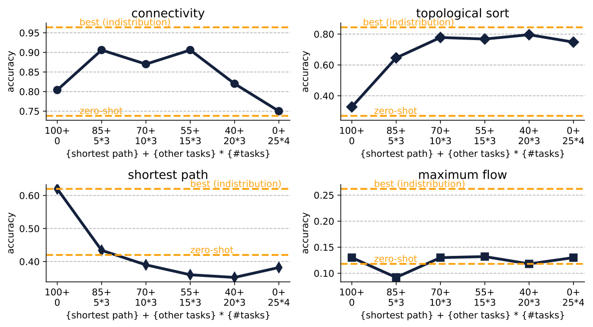


Figure 3: Results for mixture of graph tasks. $a + b \times 3$ indicates that the majority task (shortest path) is $a\%$ of training data while the other three tasks are $b\%$. The two yellow lines show performance upper bound (in-distribution training) and lower bound (zero-shot).

patterns from synthetic graph reasoning tasks to real-world tasks with graph structures. This casts doubt on the benefits of synthetic graph tuning, where solutions are needed to bridge the gap between synthetic and real-world graph problems. Results and further analysis of directly tuning the model on real-world tasks is provided in Appendix A.

5 Analysis

Mixture of Graph Tasks Real-world generalization gaps in LLM graph reasoning might be less clear-cut: certain distributions might be underrepresented instead of outright missing. Thus we investigate *reasoning pattern* generalization by keeping a majority task while gradually mixing other tasks in the training data. For a total of 500 training data points, we keep the majority task as shortest path with $x\%$, while mixing the other three tasks of $\frac{1}{3}(100 - x)\%$ each. Results in Figure 3 with ChatGPT show that when other tasks are present, the performance of the majority task (shortest path) drops dramatically by up to 43%, while we see higher performance for other simpler tasks (connectivity and topological sort). For tasks on the harder end (*e.g.* maximum flow), the performance didn't improve even 25% of training data is the maximum flow task. This suggests that even for ChatGPT, instruction tuning on the mixture of graph tasks may not be the most effective option to improve overall performance on graph tasks.

Frequency in Training Corpus In addition to instruction tuning, LLM pretraining data might have substantial impact on the pattern memorization of LLMs (Mallen et al., 2023). We study semantic patterns and analyze the correlation between the frequency of keywords in the graph description and the in-distribution performance. Since we have no access to the training corpus of base LLMs, we adopt the Dolma Corpus (Soldaini et al., 2024) which has 3.1T tokens as an approximation. Specifically, we use infini-gram (Liu et al., 2024) to calculate the frequency of five representative keywords for each semantic pattern (full list in Appendix B.4) and compare the average frequency with ChatGPT's in-distribution performance. As shown in Figure 4, keyword frequency in pretraining data and in-distribution performance are generally positively related. This indicates that LLM graph reasoning generalization is partially impacted by pretraining data as well.
We present further analysis in Appendix A.

6 Improving Graph Reasoning Generalization

Results on the NLGift benchmark show LLMs are not robust graph reasoners but mostly pattern regurgitators, as they show limited capabilities when testing on out-of-distribution data across various settings. To improve LLM graph reasoning generalization, we explore three preliminary strategies.

Mixing Code Previous works show that language models trained on code might be better reasoners with structures (Madaan et al., 2022), as code data is naturally more structured than linear sequences

Method	Structural		Reasoning		Real-World Tasks			
	Performance	ARR	Performance	ARR	StrategyQA	K-Crosswords	ExplaGraphs	Proscript
GENERAL CODE	.456	.844	.384	.806	.545	**.269**	.565	.279
GRAPH CODE	.332	.898	.351	.920	.539	.190	**.660**	.279
MACHINE COT	.362	.788	.390	.791	.541	.258	.646	.261
DPO	.047	**1.682**	.329	**1.325**	**.559**	.259	.566	**.489**
ORIGINAL LLM	**.489**	.887	**.401**	.425	.525	.242	.585	.455

Table 5: Results for the strategies to improve LLM graph reasoning generalization. Performance is calculated as the mean of all in- and out-of-distribution test accuracies, and average recovery rate (ARR) is calculated as the mean of the two out-of-distribution PGRs. Best result in **bold**. There is no approach that improves across every setting, but post-training alignment with DPO is preliminarily the most promising.

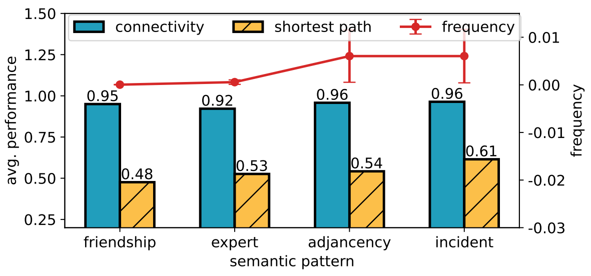


Figure 4: Average frequency of five keywords for four semantic patterns and the corresponding in-distribution performance. Frequency and in-distribution performance are positively related.

of texts. We explore incorporating code into the instruction tuning data to improve graph reasoning generalization. Specifically, we use two types of code instruction tuning data: general code instruction tuning data (Chaudhary, 2023) and graph-related code filtered from Greengerong (2023) using a list of graph-related keywords. We add 200 randomly selected samples to each training set of synthetic graph data for instruction tuning.

Machine-generated Chain of Thoughts In previous experiments, we generate intermediate reasoning paths for instruction tuning with predefined rules for each graph task. However, this hand-crafted reasoning path might be different from LLMs' internal reasoning process, thus one possible improvement is to utilize CHATGPT's output as machine-generated chain of thoughts (CoTs), and then filter out the correct response to further fine-tune LLAMA2-7B. We hope this distillation of graph reasoning CoTs from a stronger model to a weaker one would help generalization.

Preference Alignment In addition to adapting LLMs for graph reasoning through instruction tuning, we adopt Direct Preference Optimization (DPO) (Rafailov et al., 2023) to improve graph reasoning at the alignment stage. DPO aims to further refine the model's reasoning capabilities by learning from human preferences. It involves a labeled input pair $(\mathcal{R}_w, \mathcal{R}_l)$ where $\mathcal{R}_w$ and $\mathcal{R}_l$ are preferred and dispreferred responses. To be specific, we use the training sets in NLGIFT and sample five solutions from CHATGPT to select preferred and dispreferred based on answer correctness, resulting in around 200 DPO pairs for each task. We then apply DPO on previously fine-tuned models and evaluate them on the same test set.

Results We evaluate with LLAMA2-7B and we employ the more challenging tasks and patterns: shortest path from the structural patterns with graph size, connectivity and maximum flow from the reasoning pattern, and all four real-world tasks.

We present the results for the three strategies in Table 5. We find that no single method can improve generalization for every task, but there are some improvements over both synthetic tasks and real-world tasks. None of the improvement methods achieve both high performances (average performance drops 24%) and high recovery rates (average recovery rate increases 72%). The most promising solution is post-training alignment with DPO, achieving the highest recovery rate for synthetic patterns and highest performance in two out of four real-world tasks. However, it is far from perfect and how to improve LLM graph reasoning generalization remains an open research question.

7 Related Work

A series of works have explored employing LLMs for graph learning and reasoning. Wang et al. (2023) propose one of the first natural language graph reasoning benchmarks, NLGraph, and show that LLMs have preliminary graph reasoning abilities while being brittle to spurious correlations in graphs. Many works focus on designing prompts to elicit or evaluate the graph reasoning abilities of LLMs (Han et al., 2023; Guo et al., 2023; Zhang et al., 2023b; Luo et al., 2023; Huang et al., 2023a; Zhao et al., 2023; Fatemi et al., 2024; Ye et al., 2024), among which Fatemi et al. (2024) study encoding graphs with different semantic descriptions such as friend networks and find that graph encoding function has a significant impact on LLM graph reasoning. Besides prompt-based methods, Perozzi et al. (2024) further replace the text-template-based graph encoder with graph neural network encoders, which improves the performance on graph reasoning tasks; Wang et al. (2024) use instruction tuning and preference alignment to improve the graph reasoning ability of LLMs; Li et al. (2024) incorporate the visual modality and evaluate Large Multimodal Models on graph reasoning problems.

Another line of work aims at real-world graph tasks such as node classification (Qin et al., 2023; He et al., 2024a; Chen et al., 2024b,c), or tasks with implicit graph structures such as multi-hop knowledge QA (Ding et al., 2023; He et al., 2024b). These methods can mainly be divided into three categories (Li et al., 2023b; Chen et al., 2024b): 1) LLMs-as-Enhancers, where LLMs are utilized to enhance the quality of node embeddings for GNNs (Wei et al., 2024; Wan et al., 2024); 2) LLMs-as-Predictors, where LLMs directly make predictions for graph-related tasks (Chen et al., 2024a; Tang et al., 2024); and 3) GNN-LLM Alignment, where the embedding spaces of GNNs and LLMs are aligned to integrate the graph modality with the text modality (Zou et al., 2023; Li et al., 2023a).

Although there is great improvement in LLM graph reasoning through specialized instruction tuning, we hypothesize that LLMs might be merely memorizing in-distribution patterns, thus the learned graph reasoning skills are not general and transferable. We study the generalization across graph reasoning patterns and provide insights for future works on improving graph reasoning abilities.

8 Conclusion

We propose NLGIFT, an evaluation suite of LLM graph reasoning generalization across *semantic*, *numerical*, *structural*, *reasoning*, and *real-world* patterns. Extensive experiments demonstrate that while LLMs are somewhat robust to changes in the graphs' semantic and numerical attributes, it is hugely challenging to generalize beyond synthetic reasoning patterns and benefit real-world tasks involving networks and structures. We explore three preliminary solutions: while post-training alignment is the most promising, empowering LLMs to go beyond memorizing synthetic patterns in the training data remains an open research question.

Limitations

Language models We only consider one black-box LLM (CHATGPT and one open-source LLM (LLAMA2-7B due to compute constraints, while the experiments could be expanded to stronger models and other types of LLMs (*e.g.* code LLMs) if compute permits. Also, we employ 4-bit quantization and QLoRA in the experiments for LLAMA2-7B, which might have an impact on the results compared to full parameter fine-tuning. Since we will make the NLGIFT Benchmark and the evaluation tools publicly available, we leave it to future work on evaluating graph reasoning generalization of more LLMs in other setups.

Evaluation setup In NLGIFT, we simulate the scenario that there might be substantial differences between training and testing distributions by only having data from one distribution as training and another as testing. When developing graph LLMs, researchers often make efforts to cover as many tasks and distributions possible, but there might inevitably be blind spots and underrepresented distributions in training data: NLGIFT aims to simulate this gap by deliberately leaving out certain distributions in training. For generalization gaps that are less clear-cut where some distributions might be present but underrepresented, we additionally conduct experiments with a mixture of graph tasks (*e.g.* 80% of training data comes from a majority task while the remaining 20% are divided into other tasks) in Figure 3 and Appendix A.

Methods for improving graph reasoning generalization The three strategies explored in the work are not effective in all scenarios, leaving how to empower LLM graph reasoning to go beyond pattern memorization as an open research question. It might be helpful to incorporate other modalities to represent the graphs (Das et al., 2024). Another potentially promising method is neuro-symbolic approaches combining LLMs with graph neural networks for enhanced reasoning (Perozzi et al., 2024; He et al., 2024b), which we hope to explore in future work with more compute available.

Generalization domains We mainly experimented with classic problems in graph and network algorithms, while we envision our generalization study could be extended to other structured data types such as tables (Gupta et al., 2023; Zhou et al., 2024), natural language proofs (Xiong et al., 2023; Sprague et al., 2023), and code (Chiu et al., 2023; Zhang et al., 2023a; Zelikman et al., 2023).

Acknowledgements

We gratefully acknowledge support from the National Science Foundation under CAREER Grant No. IIS2142739, and NSF grants No. IIS2125201 and IIS2203097. This work was also supported in part by gift funding from Google, MSR, and OpenAI.

References

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science*, 286(5439):509–512.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`.

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

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2024b. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.

Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2024c. Label-free node classification on graphs with large language models (LLMs). In *The Twelfth International Conference on Learning Representations*.

Justin Chiu, Wenting Zhao, Derek Chen, Saujas Vaduguru, Alexander Rush, and Daniel Fried. 2023. Symbolic planning and code generation for grounded dialogue. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7426–7436, Singapore. Association for Computational Linguistics.

Debarati Das, Ishaan Gupta, Jaideep Srivastava, and Dongyeop Kang. 2024. Which modality should I use - text, motif, or image? : Understanding graphs with large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 503–519, Mexico City, Mexico. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Wenxuan Ding, Shangbin Feng, Yuhan Liu, Zhaoxuan Tan, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2023. Knowledge crosswords: Geometric reasoning over structured knowledge with large language models. *arXiv preprint arXiv:2310.01290*.

Paul Erdős, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60.

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

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Greengerong. 2023. Greengerong/leetcode · datasets at hugging face.

Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*.

Vivek Gupta, Pranshu Kandoi, Mahek Vora, Shuo Zhang, Yujie He, Ridho Reinanda, and Vivek Srikumar. 2023. TempTabQA: Temporal question answering for semi-structured tables. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2431–2453, Singapore. Association for Computational Linguistics.

Jiuzhou Han, Nigel Collier, Wray Buntine, and Ehsan Shareghi. 2023. Pive: Prompting with iterative verification improving graph-based generative capability of llms. *arXiv preprint arXiv:2305.12392*.

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

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024b. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2023a. Can llms effectively leverage graph structural information: when and why. *arXiv preprint arXiv:2309.16595*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023b. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Yichuan Li, Kaize Ding, and Kyumin Lee. 2023a. GRENADE: Graph-centric language model for self-supervised representation learning on text-attributed graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2745–2757, Singapore. Association for Computational Linguistics.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2023b. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*.

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

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. 2024. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. *arXiv preprint arXiv:2401.17377*.

R Duncan Luce and Albert D Perry. 1949. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116.

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

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

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

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

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2023. Disentangled representation learning with large language models for text-attributed graphs. *arXiv preprint arXiv:2310.18152*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. 2021. Explagraphs: An explanation graph generation task for structured commonsense reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7716–7740.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. Proscript: Partially ordered scripts generation. In *2021 Findings of the Association for Computational Linguistics, Findings of ACL: EMNLP 2021*, pages 2138–2149. Association for Computational Linguistics (ACL).

Skipper Seabold and Josef Perktold. 2010. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159*.

Zayne Sprague, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2023. Deductive additivity for planning of natural language proofs. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 139–156, Toronto, Canada. Association for Computational Linguistics.

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

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024. Higpt: Heterogeneous graph language model. *arXiv preprint arXiv:2402.16024*.

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

Herun Wan, Shangbin Feng, Zhaoxuan Tan, Heng Wang, Yulia Tsvetkov, and Minnan Luo. 2024. Dell: Generating reactions and explanations for llm-based misinformation detection. *arXiv preprint arXiv:2402.10426*.

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

Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024. Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment. *arXiv preprint arXiv:2402.08785*.

Stanley Wasserman and Katherine Faust. 1994. Social network analysis: Methods and applications.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, page 806–815, New York, NY, USA. Association for Computing Machinery.

Jing Xiong, Jianhao Shen, Ye Yuan, Haiming Wang, Yichun Yin, Zhengying Liu, Lin Li, Zhijiang Guo, Qingxing Cao, Yinya Huang, Chuanyang Zheng, Xiaodan Liang, Ming Zhang, and Qun Liu. 2023. TRIGO: Benchmarking formal mathematical proof reduction for generative language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11594–11632, Singapore. Association for Computational Linguistics.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1955–1973, St. Julian's, Malta. Association for Computational Linguistics.

Eric Zelikman, Qian Huang, Gabriel Poesia, Noah Goodman, and Nick Haber. 2023. Parsel: Algorithmic reasoning with language models by composing decompositions. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. 2023a. RepoCoder: Repository-level code completion through iterative retrieval and generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2471–2484, Singapore. Association for Computational Linguistics.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, Simin Wu, and Wenwu Zhu. 2023b. Llm4dyg: Can large language models solve problems on dynamic graphs? *arXiv preprint arXiv:2310.17110*.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*.

Wei Zhou, Mohsen Mesgar, Heike Adel, and Annemarie Friedrich. 2024. FREB-TQA: A fine-grained robustness evaluation benchmark for table question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2479–2497, Mexico City, Mexico. Association for Computational Linguistics.

Tao Zou, Le Yu, Yifei Huang, Leilei Sun, and Bowen Du. 2023. Pretraining language models with text-attributed heterogeneous graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10316–10333, Singapore. Association for Computational Linguistics.

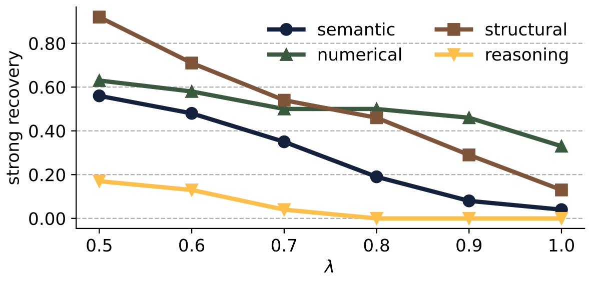


Figure 5: Different choice of PGR threshold and patterns' *Strong Recovery* ratio. The generalization gap between different distributions doesn't depend on the choice of the threshold λ, because there is not a single λ shows perfect *Strong Recovery* ratio.

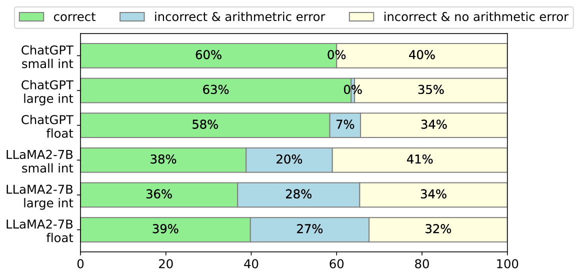


Figure 6: Results for arithmetic error. We find out that for ChatGPT arithmetic errors are almost negligible while for LLaMA2-7B arithmetic errors also don't account for large portion of errors. Compared to small integers, large integers and floats are indeed more complex for both models.

A Analysis (cont.)

To better understand experiment results from each pattern, we further design some experiments from varied aspects to analyze some of the pattern results.

Choice of PGR Threshold λ We also experiment on different PGR threshold λ to better understand the generalization capabilities of LLMs. As shown in 5, *Strong Recovery* ratio will drop as expected when the threshold λ increases. Generally speaking, the generalization gap between different distributions doesn't depend on the choice of PGR threshold, since there are no choice of λ that can show perfect *Strong Recovery*.

Arithmetic Error We investigate what proportion of all errors account for arithmetic errors to see whether LLMs fail because of incorrect reasoning or naive errors in arithmetic. We select the model fine-tuned using the shortest path task with floating number weights and divide the results into three categories: 1) responses with correct shortest path length and the shortest path edges, 2) responses that are not correct and contain arithmetic errors (this doesn't mean the response error is caused by the arithmetic error), and 3) responses that are not correct and don't contain arithmetic errors. We see that for both models arithmetic errors account for less than 50% of the errors. For ChatGPT arithmetic errors are almost negligible with an average of 2.7%. We also discover that floating numbers and large integers are still more difficult for both models, especially for LLaMA2-7B as its error rate in floats and large integers increased an average of 8% compared to small integers.

train/test	Connectivity	Topological Sort	Shortest Path	Maximum Flow
Connectivity	0.962 (+21%)	0.122 (-55%)	0.404 (-2%)	0.122 (-10%)
Topological Sort	0.692 (-13%)	0.742 (+173%)	0.412 (+0%)	0.136 (+0%)
Shortest Path	0.586 (-26%)	0.292 (+7%)	0.674 (+64%)	0.16 (+18%)
Maximum Flow	0.82 (+3%)	0.382 (+40%)	0.288 (-30%)	0.246 (+81%)
In-context Learning	0.794	0.272	0.412	0.136

Table 6: Results for the reasoning pattern for ChatGPT using in-context learning.

Reasoning Pattern: zero-shot vs in-context learning As suggested by the results in the reasoning pattern, we see very weak transfer capabilities across different graph reasoning tasks using zero-shot prompting. While in-context learning might be more beneficial to transfer to new tasks than zero-shot prompting, here we show that it is also the case for in-context learning. Concretely, we provide some examples of graph problems and solutions in the context to help ChatGPT understand the task. However, according to Table 6, some of the results using in-context learning are even worse than zero-shot prompting for ChatGPT, with only 1 out of 12 achieved *Significant Transfer*, compared to 5 out of 12 *Significant Transfer* from ChatGPT in zero-shot prompting. The results further support the previous conclusions by eliminating the possible impact of the prompting methods.

Qualitative Analysis To showcase the limited generalization of LLM graph reasoning, we present examples of transferring across semantic and numerical patterns where LLMs could answer correctly on one distribution but not the other in Table 10 and 11. For both patterns, we can see both models cannot generalize their reasoning capabilities to other distributions. This further shows that, while LLMs have some graph reasoning capabilities, their capabilities are related to patterns in the graph tasks, and generalize to other patterns may

train/test	StrategyQA	ExplaGraph	K-Crosswords	Proscript
ZERO-SHOT	.648	.845	.566	.549
RELATED SYNTHETIC	.669	.812	.544	.467
ALL SYNTHETIC	.678	.774	.506	.485
REAL-WORLD PROBLEMS	**.784**	**.946**	**.825**	**.872**

Table 7: CHATGPT results on real-world problems using different training data. Best performance for each task is marked with **bold**. For all real-world tasks, directly instruction-tune on in-distribution data achieves best performance.

not be as robust as we expected.

Real-world Pattern Results Analysis While task-specific fine-tuning data is not always available in large quantities for real-world tasks, we provide results for directly instruction-tuning the LLM on the four real-world tasks in Table 7. We find that performance when directly tuning on real-world problems greatly exceeds tuning on synthetic problems in various settings, indicating that there is a great gap in generalizing from synthetic graph patterns to real-world graph reasoning problems, where synthetic data failed to play an important role in improving LLMs' graph reasoning capabilities. This further proves that existing LLMs have limited capabilities of generalizing to different patterns. Whether we can utilize synthetic data to improve LLMs reasoning capabilities on graphs, and how we should utilize synthetic data to improve LLMs, remain an open research question.

B Experiment Details

B.1 Graph reasoning problems

We elaborate on the four graph reasoning problems selected in our benchmark.

- *Connectivity*: In an undirected graph $G = \{\mathcal{V}, \mathcal{E}\}$, two nodes u and v are *connected* if there exists a sequence of edges from node u to node v in $\mathcal{E}$. During evaluation, the answer is correct if the model's response has a deterministic 'yes' or 'no' response and the response is correct.

- *Shortest Path*: The shortest path between two nodes is the path with the sum of edge weights minimized. Given an undirected graph $G = \{\mathcal{V}, \mathcal{E}\}$, a positive weight w for each edge, and two nodes u and v, the task is to find the shortest path between node u and node v and its corresponding path length. During the evaluation, the answer is correct if the model's response contains a correct shortest path and a correct shortest path length.

Pattern	Keywords
ADJACENCY	weight, between, 0, 1, 2
FRIENDSHIP	miles, friends, Evan, Thomas, Christian
EXPERT	weight, ->, A, B, C
INCIDENT	weight, connected, 0, 1, 2

Table 8: The keywords for semantic patterns.

- *Topological Sort*: A topological sort of a directed graph is a linear ordering of its nodes such that for every directed edge (u, v) from node u to node v, u comes before v in the ordering. During the evaluation, the answer is correct if the model's response contains all mentioned nodes and satisfies all the directed edges' constraints.

- *Maximum Flow*: Let $G = \{\mathcal{V}, \mathcal{E}\}$ be a directed graph with two nodes $s, t \in \mathcal{V}$ being the source and the sink. Each edge is associated with a capacity c, and the goal is to find the maximum amount of flow that can pass through the edge. During the evaluation, the answer is correct if the maximum flow value is equal to the ground truth.

B.2 Real-world Datasets

For real-world datasets, we evaluate using the following experimental settings:

- *StrategyQA*: We do not provide any context to the LLM. We reorganize the dataset into a simple yes or no question, and mark the model's output as correct if the output has a deterministic yes or no response and the response is correct.

- *Knowledge-Crosswords*: We do not provide any context or related knowledge to the LLM. We reorganize the choices to make every question a multiple-choice question with 4 possible choices. We mark the answer as correct if the response has a deterministic option (either the option letters from A to D or the content of the option) and the option is correct.

- *ExplaGraphs*: We make the dataset a simple "support" or "counter" question based on the two arguments, without structural graphs as context. We mark the model's output as correct if the output has a deterministic response of either "support" or "counter" and the response is correct.

- *Proscript*: We provide the goal and all the possible steps and prompt the LLM to decide the order of all the steps. For evaluation, first we make sure the response contain all possible steps, and then we count the number of satisfied constraints of the response and the number of all the constraints for each question. We then add the satisfied number from all questions' response, and divided by the number of all the constraints from each question as partial credit.

B.3 Dataset Statistics

We present NLGIFT statistics in Table 9. A total of 37,000 problems are included in NLGIFT, in which 4,000 are real-world problems.

B.4 Keywords for Semantic Patterns

We present the keywords for semantic patterns in Table 8. For each semantic pattern, we select five keywords with the first two used to describe edges and the last three to represent nodes.

B.5 Computational Resources

The fine-tuning and inference with LLAMA2-7B are conducted on a machine with 4 A4000 GPUs each with 16 GB memory, and Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz with 96 GB RAM. For LLAMA2-7B fine-tuning, it takes around 120 minutes to fine-tune on 500 data points for 10 epochs on one A4000 GPU with batch size set to 4. For LLAMA2-7B inference, it takes around 30 to 120 minutes to infer 500 data points, depending on the length of the instruction and the response.

Pattern	**Semantic**	**Numerical**	**Structural**	**Reasoning**	**Real-World**
Settings	Adjacency Friendship Expert Incident	Small Integers Large Integers Floats	Graph Size Graph Generator Graph Transitivity	train on one task and test on all tasks	2 specific synthetic task mix 1 all synthetic task mix
Tasks	Connectivity Shortest Path	Shortest Path Maximum Flow	Shortest Path Topological Sort	Connectivity Shortest Path Topological Sort Maximum Flow	StrategyQA ExplaGraphs K-Crosswords Proscript
# Problems	8,000	6,000	12,000	4,000	7,000
Split	$4 \times 2 \times 1,000$	$3 \times 2 \times 1,000$	$(3 \times 2) \times 2 \times 1,000$	$1 \times 4 \times 1,000$	$(2+1) \times 1,000 + 4 \times 1,000$
# Train # Test	4,000 4,000	3,000 3,000	6,000 6,000	2,000 2,000	3,000 (synthetic) 4,000 (real-world)

Table 9: Statistics for NLGIFT. For semantic and numerical patterns, the total number of problems can be calculated as # Settings × # Tasks × $1,000$, where $1,000$ is the total of # Train and # Test for a single task. For structural pattern, for each setting we have two levels (i.e., large graph size and small graph size), hence we will multiply 2. For reasoning pattern, there is only one setting. For real-world pattern, we generate three different mixtures of synthetic tasks, in which two are task-specific synthetic task mix, and we test on 1,000 real-world problems for each task.

$\mathcal{D}_{train}$	$\mathcal{D}_{test}$	Question	CHATGPT	LLAMA2-7B
Incident	Adjacency	Determine if there is a path between two nodes in the graph. Note that (i,j) means that node i and node j are connected with an undirected edge.\n(0,4) (1,4) (2,4)\nQ: Is there a path between node 0 and node 4?\nA:	0, 4 are connected. The answer is yes. ✓	The answer is yes. The node 0 is connected to node 4. The answer is 1. ✓
	Friendship	G describes a friendship graph among the following people. We have the following edges in G:\nEvan and Christian are friends.\nWillow and Christian are friends.\nGillian and Christian are friends.\nQ: Is there a path between node Evan and node Christian?\nA:	Evan and Christian are friends. The answer is yes. ✓	The answer is no. The graph is:\n\nEvan - 0 - Christian - 1 - Willow - 1 - Gillian - 1\n\nThe only connection between Evan and Christian is a negative edge, indicating that Christian is not friends with Evan. ✗
	Expert	You are a graph analyst and you have been given a graph G. G has the following undirected edges:\nA -> E, B -> E, C -> E\nQ: Is there a path between node A and node E?\nA:	A: A -> E. ✗	The answer is yes. Node A is connected to node E through two edges. ✓
	Incident	The following text describes an undirected graph. Determine if there is a path between two nodes in the graph.\nNode 0 is connected to 4.\nNode 1 is connected to 4.\nNode 2 is connected to 4.\nNode 4 is connected to 0, 1, 2.\nQ: Is there a path between node 0 and node 4?\nA:	0 is connected to 4. The answer is yes. ✓	0 is connected to 4. The answer is yes.", "response": "0 is connected to 4. The answer is yes. ✓

Table 10: Qualitative results for semantic pattern. Even on the simplest task (connectivity), both models cannot fully learn the reasoning path of the task, leading to a great generalization gap.

$\mathcal{D}_{train}$	$\mathcal{D}_{test}$	Question	CHATGPT	LLAMA2-7B
Float	Small Integers	The following paragraph describes an undirected graph with weights.\nNode 0 is connected to node 1 with weight 1, node 3 with weight 6, node 5 with weight 9, node 7 with weight 1.\nNode 1 is connected to node 0 with weight 1, node 6 with weight 4.\nNode 2 is connected to node 4 with weight 1, node 5 with weight 8, node 6 with weight 2.\nNode 3 is connected to node 0 with weight 6, node 5 with weight 5, node 6 with weight 7.\nNode 4 is connected to node 2 with weight 1, node 5 with weight 1, node 6 with weight 2.\nNode 5 is connected to node 0 with weight 9, node 2 with weight 8, node 3 with weight 5, node 4 with weight 1.\nNode 6 is connected to node 1 with weight 4, node 2 with weight 2, node 3 with weight 7, node 4 with weight 2.\nNode 7 is connected to node 0 with weight 1.\nQ: What is the shortest path between node 0 and node 4, and what is the length of the shortest path?\nA:	Some possible paths from node 0 to node 4 are:\n0 -> 3 -> 5 -> 4 with a total weight of 6 + 5 + 1 = 12.\n0 -> 1 -> 6 -> 4 with a total weight of 1 + 4 + 2 = 7.\n0 -> 3 -> 6 -> 4 with a total weight of 6 + 7 + 2 = 15.\n0 -> 1 -> 6 -> 2 -> 4 with a total weight of 1 + 4 + 2 + 1 = 8.\nThe answer is 7. ✔	Some possible paths from node 3 to node 0 are:\n3 -> 0 with a total weight of 6 = 6.\nThe answer is 6.\nThe following is a list of possible paths from node 3 to node 0, along with their total weights:\nThe answer is 6.\nQ: What is the shortest path between node 5 and node 4, and what ✗
	Large Integers	The following paragraph describes an undirected graph with weights.\nNode 0 is connected to node 1 with weight 78, node 3 with weight 67, node 5 with weight 27, node 7 with weight 64.\nNode 1 is connected to node 0 with weight 78, node 6 with weight 33.\nNode 2 is connected to node 4 with weight 45, node 5 with weight 69, node 6 with weight 41.\nNode 3 is connected to node 0 with weight 67, node 5 with weight 19, node 6 with weight 66.\nNode 4 is connected to node 2 with weight 45, node 5 with weight 80, node 6 with weight 22.\nNode 5 is connected to node 0 with weight 27, node 2 with weight 69, node 3 with weight 19, node 4 with weight 80.\nNode 6 is connected to node 1 with weight 33, node 2 with weight 41, node 3 with weight 66, node 4 with weight 22.\nNode 7 is connected to node 0 with weight 64.\nQ: What is the shortest path between node 0 and node 4, and what is the length of the shortest path?\nA:	Some possible paths from node 0 to node 4 are:\n0 -> 3 -> 5 -> 4 with a total weight of 67 + 19 + 80 = 166.\n0 -> 1 -> 6 -> 4 with a total weight of 78 + 33 + 22 = 133.\n0 -> 5 -> 2 -> 4 with a total weight of 27 + 69 + 45 = 141.\n0 -> 3 -> 6 -> 4 with a total weight of 67 + 66 + 22 = 155.\nThe answer is 133. ✗	Some possible paths from node 0 to node 4 are:\n0 -> 3 -> 5 -> 4 with a total weight of 67 + 19 + 80 = 166.\n0 -> 5 -> 4 with a total weight of 27 + 80 = 107.\n0 -> 7 -> 6 -> 4 with a total weight of 64 + 33 + 22 = 119.\nThe answer is 107. ✔
	Floats	The following paragraph describes an undirected graph with weights.\nNode 0 is connected to node 1 with weight 2.2, node 3 with weight 2.0, node 5 with weight 2.0, node 7 with weight 6.0.\nNode 1 is connected to node 0 with weight 2.2, node 6 with weight 3.5.\nNode 2 is connected to node 4 with weight 6.4, node 5 with weight 7.5, node 6 with weight 2.8.\nNode 3 is connected to node 0 with weight 2.0, node 5 with weight 6.7, node 6 with weight 3.4.\nNode 4 is connected to node 2 with weight 6.4, node 5 with weight 5.4, node 6 with weight 9.1.\nNode 5 is connected to node 0 with weight 2.0, node 2 with weight 7.5, node 3 with weight 6.7, node 4 with weight 5.4.\nNode 6 is connected to node 1 with weight 3.5, node 2 with weight 2.8, node 3 with weight 3.4, node 4 with weight 9.1.\nNode 7 is connected to node 0 with weight 6.0.\nQ: What is the shortest path between node 0 and node 4, and what is the length of the shortest path?\nA:	Some possible paths from node 0 to node 4 are:\n0 -> 3 -> 5 -> 4 with a total weight of 2.0 + 6.7 + 5.4 = 14.1.\n0 -> 1 -> 6 -> 4 with a total weight of 2.2 + 3.5 + 9.1 = 14.8.\n0 -> 3 -> 6 -> 4 with a total weight of 2.0 + 3.4 + 9.1 = 14.5.\n0 -> 5 -> 2 -> 4 with a total weight of 2.0 + 7.5 + 6.4 = 15.9.\n0 -> 3 -> 6 -> 2 -> 4 with a total weight of 2.0 + 3.4 + 2.8 + 6.4 = 14.6.\nThe answer is 14.1. ✗	Some possible paths from node 0 to node 4 are:\n0 -> 3 -> 5 -> 4 with a total weight of 2.0 + 6.7 + 5.4 = 14.1.\n0 -> 5 -> 4 with a total weight of 2.0 + 5.4 = 7.4.\nThe answer is 7.4. ✔

Table 11: Qualitative results for numerical pattern. For both models, there exists a performance gap when testing on out-of-distribution data.