Can Language Models Solve Graph Problems in Natural Language?

- **Sources:** 37th Conference on Neural Information Processing Systems (NeurIPS 2023)
- **Citations:** 98
- **Institutions:** Jiaotong University, University of Washington and University of Notre Dame
- **Implementation:** https://github.com/Arthur-Heng/NLGraph

Saman Khamesian

09/25/2024

INTRODUCTION

- Large language models (LLMs) are increasingly used for tasks with hidden graphical structures, such as robotics planning, multi-hop question answering, knowledge probing, and structured commonsense reasoning. For example:

- In robotics and planning, LLMs are adopted to guide agents through structured environments.

- In multi-hop question answering, LLMs implicitly find connections and paths among a vast network of entities and concepts.

- Together these works demonstrate that LLMs are widely adopted for tasks with implicit graphical structures while achieving preliminary success.

INTRODUCTION

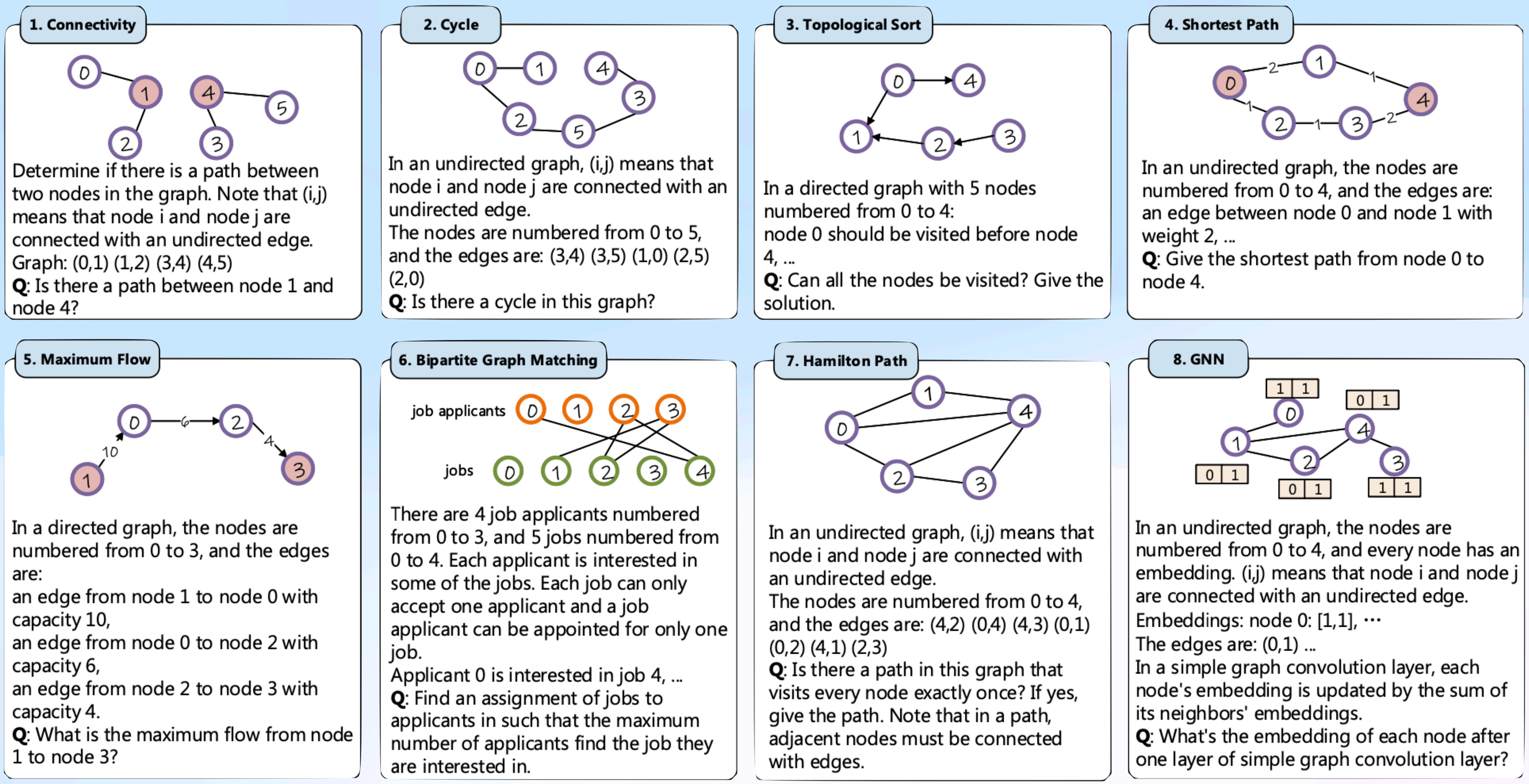
- However, one underlying yet crucial question remains under explored: Can LLMs reason with graphs?

- More concretely, are LLMs capable of mapping textual descriptions of graphs and structures to grounded conceptual spaces and solving graph algorithm problems explicitly with natural language?

METHODOLOGY

- To this end, the authors propose the Natural Language Graph (NLGraph) benchmark, a comprehensive testbed of graph and structured reasoning designed for language models and in natural language.
- NLGraph includes 29,370 problems across eight graph reasoning tasks, ranging from simple tasks such as: connectivity, cycle and shortest path to more complex problems such as topological sort, maximum flow, bipartite graph matching, Hamilton path, and simulating graph neural networks.

METHODOLOGY



Overview of the NLGraph Benchmark, featuring eight tasks with varying complexity. They provide a clear figure for each task, along with example natural language prompts given to the LLMs.

METHODOLOGY

- **Step 1:** They begin by using a random graph generator to create base graphs and structures, adjusting factors like node count and graph sparsity.
- **Step 2:** These graphs are then used to create problems for eight different graph reasoning tasks, each with varying algorithmic complexity.
- **Step 3:** For each task, they control the difficulty by generating easy, medium, and hard subsets, and then adapt the base graphs and design queries accordingly, allowing for scaling and detailed analysis.

METHODOLOGY

- How are these graphs and questions generated? They described each of them, but I'll provide one example here to help understand the concept:

- **Task 1:** Connectivity — In an undirected graph $G = \{V, E\}$, two nodes u and v are *connected* if there exists a sequence of edges from node u to node v in E. They randomly select two nodes in the base graphs $u, v \in V$ to ask whether node u and node v are connected with a true/false question. They retain a balanced set of questions where half of the node pairs are connected and the other half are not connected by discarding additional questions.

METHODOLOGY

Subset	Connect.	Cycle	Topo. Sort	Shortest Path	Max. Flow	Bipartite Graph	Hamilton Path	GNNs
# EASY	352 / 730	150 / 300	180 / 360	180 / 360	150 / 300	300 / 600	150 / 300	100 / 200
SPEC.	n: 5-10	n: 5-10	n: 5-10	n: 5-10	n: 5-10	n: 6-20	n: 5-10	n: 5-8
# MEDIUM	1,200 / 8,580	600 / 1,800	150 / 1,350	/	/	/	/	/
SPEC.	n: 11-25	n: 11-25	n: 11-25	/	/	/	/	/
# HARD	680 / 7,090	400 / 2,000	200 / 1,200	200 / 1,200	200 / 1,200	210 / 1,260	200 / 600	140 / 840
SPEC.	n: 26-35	n: 26-35	n: 26-35	n: 11-20	n: 11-20	n: 17-33	n: 11-20	n: 9-15

- Statistics of the NLGraph benchmark.
- There are two numbers, A and B. They represent two benchmarks: the standard version and the extended version. A is the number of tasks in the standard version, while B indicates the number of tasks in the extended version.
- Totally 5,902 problems in a standard version and 29,370 problems in an extended version.
- **SPEC.** denotes difficulty specifications, and **n** is the number of nodes in the graph (graph size).

EXPERIMENTAL SETUP

- Based on the NLGraph benchmark, they aim to investigate whether language models can solve graph algorithm problems in natural language by evaluating large language models and different prompting approaches.

- They adopt a wide range of prompting approaches as baselines. Specifically, zero-shot prompting, few-shot in-context learning [Brown et al., 2020], chain-of- thought prompting (CoT) [Wei et al., 2022], zero-shot chain-of-thought (0-CoT) [Kojima et al., 2022], least-to-most (LTM) [Zhou et al., 2023], and self-consistency (SC) [Wang et al., 2023] are leveraged to tackle various graph reasoning tasks in the NLGraph benchmark.

- They did not provide descriptions of the approaches and only cited the relevant references. However, out of respect for you as the audience, I have included these brief descriptions of each one:

EXPERIMENTAL SETUP

- **Zero-shot prompting**: The model is asked to perform a task without any examples or prior context, relying solely on its pre-existing knowledge.
- **Few-shot in-context learning**: The model is provided with a few examples (in-context) before being asked to perform the task, helping it learn from the context provided.
- **Chain-of-thought prompting (CoT)**: The model is guided in the prompt to break down its reasoning into intermediate steps, improving its ability to solve complex, multi-step problems.
- **Zero-shot chain-of-thought (0-CoT)**: Similar to CoT, but without any examples. The model is encouraged to reason step-by-step without prior task-specific prompts.
- **Least-to-most (LTM)**: The model starts with simpler subproblems and gradually works its way up to more complex ones, solving the easier tasks first.
- **Self-consistency (SC)**: The model generates multiple answers to the same problem, and then the final answer is determined by selecting the most consistent or common response among them.

EXPERIMENTAL SETUP

- They also adopt a Random baseline:
- For true/false questions such as connectivity and cycle, they use Random to denote a baseline that randomly selects an answer from true and false with an expected accuracy of 50%;
- For the shortest path task, Random denotes a baseline that randomly selects a valid path between the query node pair.
- For the maximum flow task, Random denotes a baseline that randomly selects a value between 0 and the sum of all the edges' capacities.
- The performance comparison between different prompting techniques and the Random baseline could indicate whether LLMs are capable of performing graph reasoning instead of giving randomly generated answers.

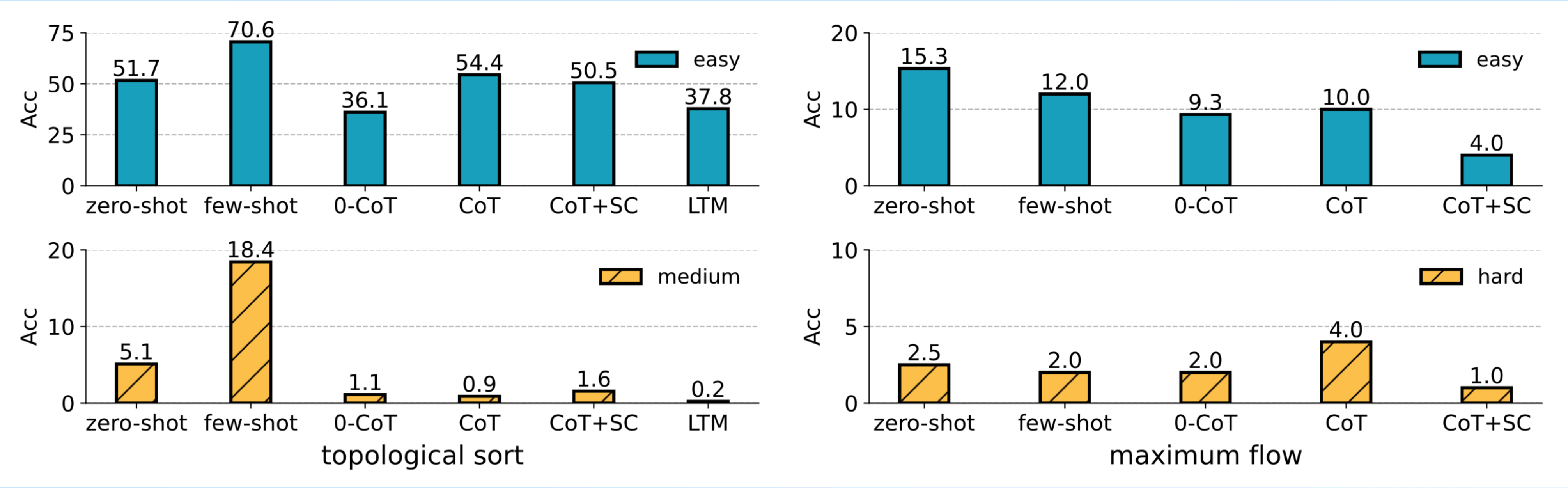
RESULTS

Method	Connectivity				Cycle				Shortest Path				
	Easy	Medium	Hard	Avg.	Easy	Medium	Hard	Avg.	Easy	Hard	Easy (PC)	Hard (PC)	Avg.
RANDOM	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	6.07	6.69	14.73	13.81	17.81
ZERO-SHOT	83.81	72.75	63.38	71.31	50.00	50.00	50.00	50.00	29.40	21.00	46.00	26.76	30.79
FEW-SHOT	93.75	83.83	76.61	84.73	80.00	**70.00**	**61.00**	**70.33**	31.11	26.00	49.19	35.73	35.51
CoT	**94.32**	82.17	77.21	84.57	**84.67**	63.33	53.25	66.75	63.89	**29.50**	76.84	35.79	51.51
0-CoT	79.55	65.83	68.53	71.30	55.33	57.67	49.00	54.00	8.89	7.50	62.39	**43.95**	32.03
CoT+SC	93.18	**84.50**	**82.79**	**86.82**	82.00	63.67	53.50	66.39	**68.89**	29.00	**80.25**	38.47	**54.15**

- Model performance on the connectivity, cycle, and shortest path tasks. PC denotes partial credit. Large language models with CoT or CoT+SC prompting greatly outperforms the random baseline by 37.33% to 57.82%, indicating that LLMs have preliminary graph reasoning abilities.

- What is Partial Credit? Partial credit is a scoring approach that assigns a proportion of full credit based on how close a solution is to the optimal one, rather than an all-or-nothing approach.

RESULTS

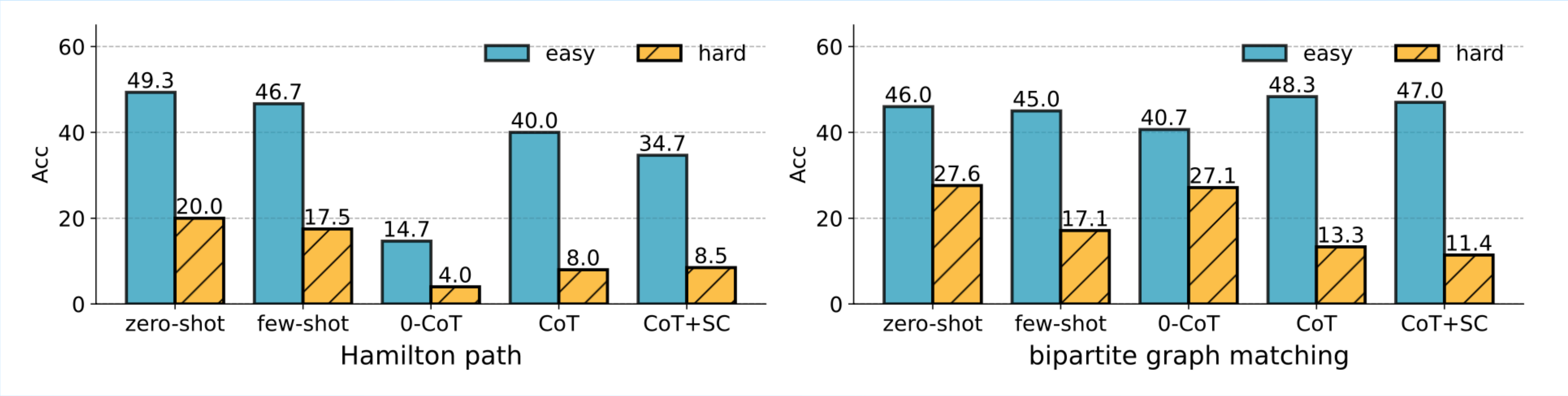


- (left) Model performance on the topological sort task. CoT, LTM, and self-consistency are mostly ineffective on this problem.

- (right) Model performance on the maximum flow task. Few-Shot prompting outperforms CoT+SC prompting on both easy and hard subsets, suggesting that LLMs fall short of generating valid intermediate steps to solve the more complex graph reasoning problem.

- Together these results demonstrate that advanced prompting is ineffective for advanced graph reasoning.

RESULTS



- (Left) Model performance on the Hamilton path task: Zero-shot prompting consistently performs better than other techniques.
- (Right) Model performance on the bipartite graph matching task: In-context learning and advanced prompting have little impact on this complex problem.
- These results suggest that in-context learning may be less effective for advanced graph reasoning tasks.

CONCLUSION

- In this work, they explore whether LLMs can explicitly perform graph reasoning, meaning solving graph algorithm problems using natural language across different problem types and prompting techniques.

- They introduce the NLGraph benchmark, a comprehensive test set with 29,370 problems across eight tasks of varying complexity.

- Their evaluation of LLMs and prompting methods on NLGraph reveals that:

  1. LLMs show some initial graph reasoning abilities

  2. The advantage of advanced prompting and in-context learning decreases with more complex tasks

  3. LLMs are sensitive to unrelated correlations in problem settings.

- Enhancing LLMs' graph reasoning skills for complex tasks is still a challenge, and they encourage future research to build on their NLGraph benchmark.

Thank you for your attention