Can Language Models Solve Graph Problems in Natural Language?

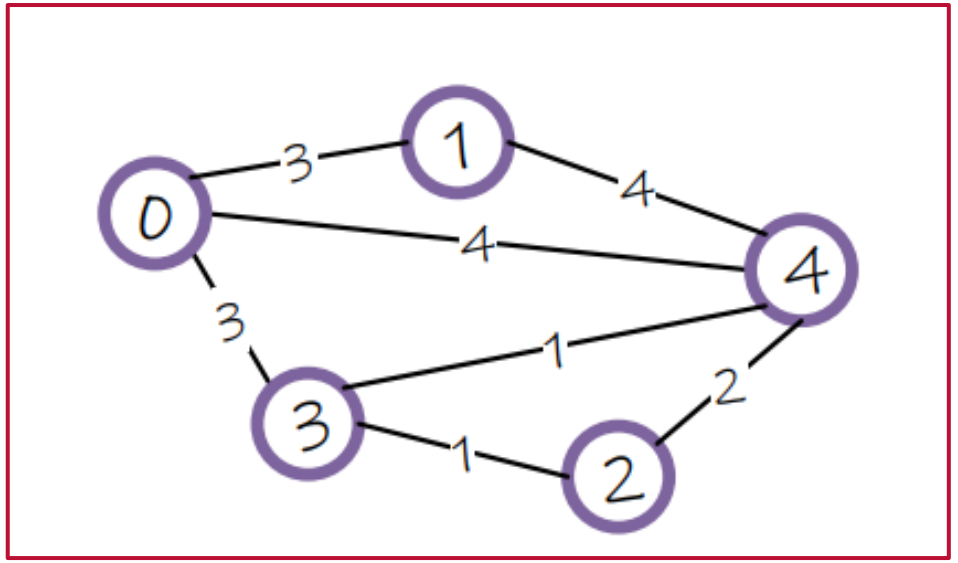
3/19/2025

Arun Cheriakara Joseph,
Faculty Of Math

UNIVERSITY OF WATERLOO

Motivation

- **LLMs** (Large Language Models) excel at many natural language tasks
- Some tasks *implicitly* involve graphs (e.g., multi-hop QA, planning)
- **But** can LLMs handle *explicit* graph problems given as text?



In an undirected graph, the nodes are numbered from 0 to 4, and the edges are:
an edge between node 0 and node 4 with weight 4,
an edge between node 0 and node 3 with weight 3,
an edge between node 0 and node 1 with weight 3,
...
Q: Give the shortest path from node 0 to node 2.

Why Is It Important?

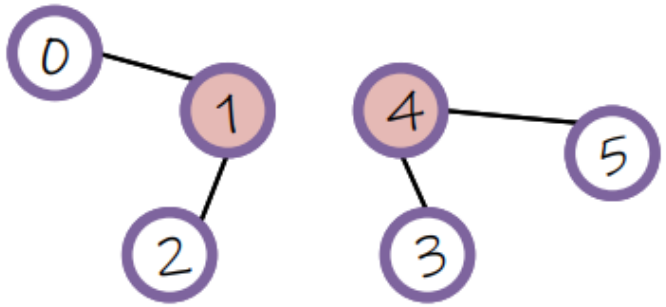
- **Graph Problems** are fundamental in computing (routing, social networks, scheduling, planning)
- Many real-world tasks are essentially **graph-based** under the hood
- If LLMs can solve **graph tasks** from text, it shows potential for more advanced, structured reasoning in **natural language** interfaces

What Is the Solution Proposed by the Authors?

- **NLGraph Benchmark**:
  - 29K+ textual graph problems, covering 8 tasks (connectivity, shortest path, max flow, etc.)
  - Used to evaluate GPT-3/4 with multiple prompting methods
- New Prompting Approaches:
  - **Build-a-Graph Prompting**: Force the LLM to build a graph first then compute.
  - **Algorithmic Prompting**: Outline the steps of a standard graph algorithm for the LLM

Graphs: Connectivity

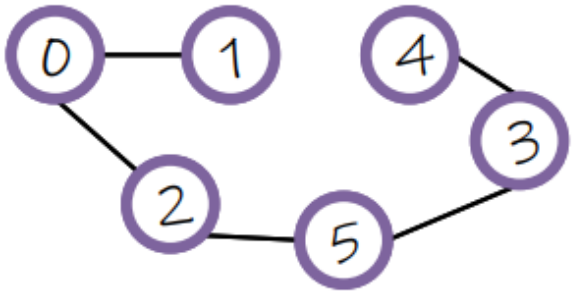
1. Connectivity



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.
Graph: (0,1) (1,2) (3,4) (4,5)
Q: Is there a path between node 1 and node 4?

Graphs : Cycle

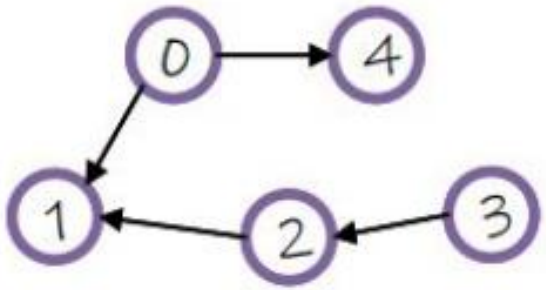
2. Cycle



In an undirected graph, (i,j) means that node i and node j are connected with an undirected edge.
The nodes are numbered from 0 to 5, and the edges are: (3,4) (3,5) (1,0) (2,5) (2,0)
Q: Is there a cycle in this graph?

Graphs : Topological Sort

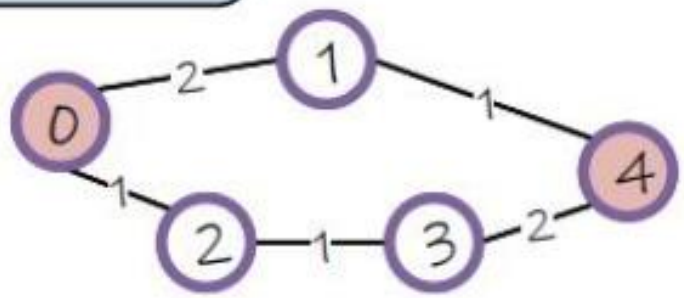
3. Topological Sort



In a directed graph with 5 nodes numbered from 0 to 4:
node 0 should be visited before node 4, ...
Q: Can all the nodes be visited? Give the solution.

Graphs : Shortest Path

4. Shortest Path

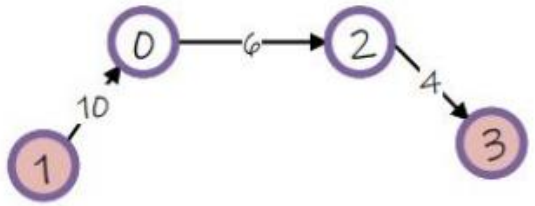


In an undirected graph, the nodes are numbered from 0 to 4, and the edges are: an edge between node 0 and node 1 with weight 2, ...

Q: Give the shortest path from node 0 to node 4.

Graphs : Maximum Flow

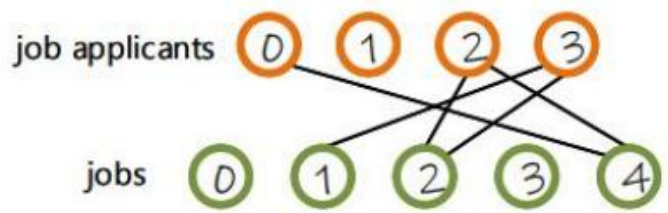
5. Maximum Flow



In a directed graph, the nodes are numbered from 0 to 3, and the edges are:
an edge from node 1 to node 0 with capacity 10,
an edge from node 0 to node 2 with capacity 6,
an edge from node 2 to node 3 with capacity 4.
Q: What is the maximum flow from node 1 to node 3?

Graphs : Bipartite Graph

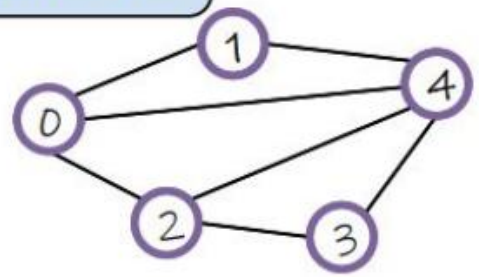
6. Bipartite Graph Matching



There are 4 job applicants numbered from 0 to 3, and 5 jobs numbered from 0 to 4. Each applicant is interested in some of the jobs. Each job can only accept one applicant and a job applicant can be appointed for only one job.
Applicant 0 is interested in job 4, ...
Q: Find an assignment of jobs to applicants in such that the maximum number of applicants find the job they are interested in.

Graphs : Hamilton Path

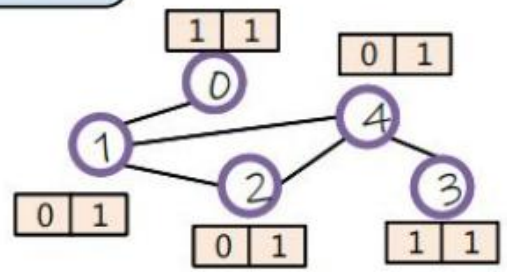
7. Hamilton Path



In an undirected graph, (i,j) means that node i and node j are connected with an undirected edge.
The nodes are numbered from 0 to 4, and the edges are: (4,2) (0,4) (4,3) (0,1) (0,2) (4,1) (2,3)
Q: Is there a path in this graph that visits every node exactly once? If yes, give the path. Note that in a path, adjacent nodes must be connected with edges.

Graphs : GNN

8. GNN

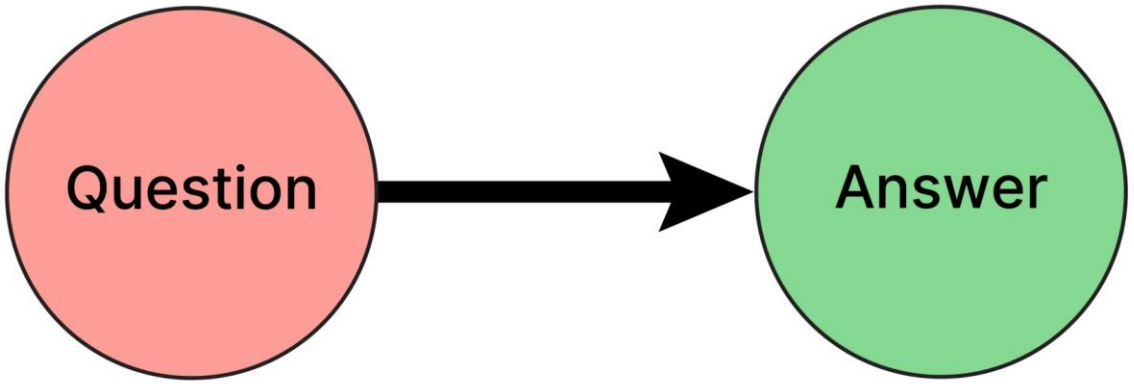


In an undirected graph, the nodes are numbered from 0 to 4, and every node has an embedding. (i,j) means that node i and node j are connected with an undirected edge.
Embeddings: node 0: [1,1], ⋯
The edges are: (0,1) ...
In a simple graph convolution layer, each node's embedding is updated by the sum of its neighbors' embeddings.
Q: What's the embedding of each node after one layer of simple graph convolution layer?

The NLGraph Benchmark

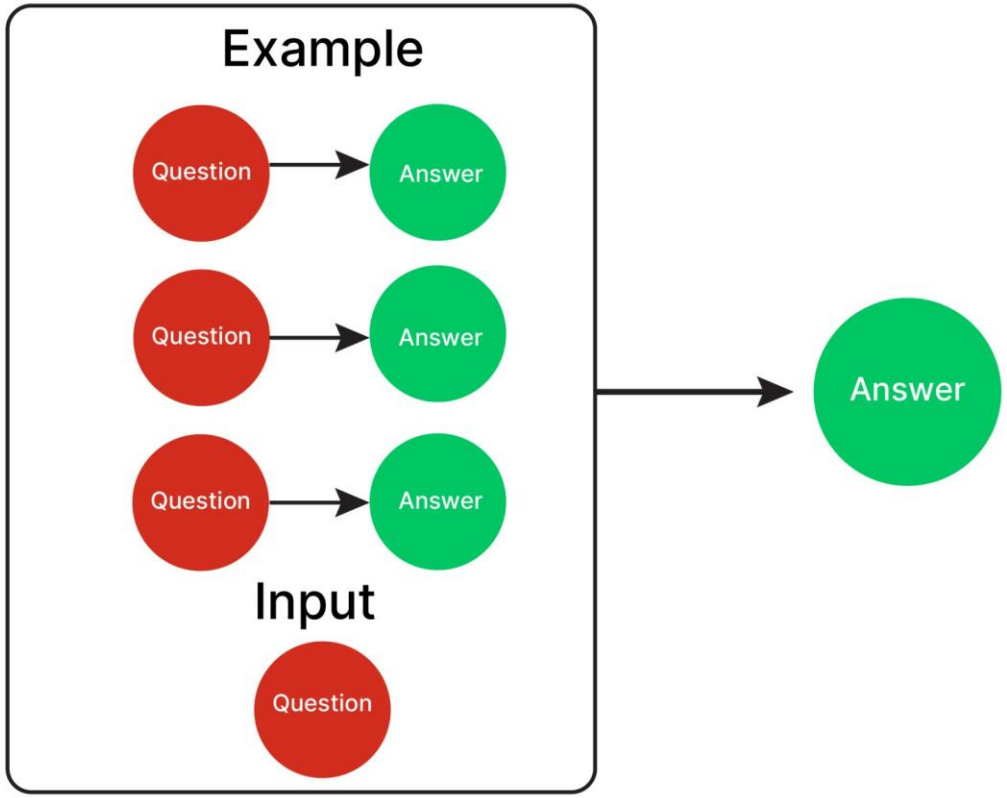
- Employ a random graph generator to generate graphs and structures while controlling for network size, graph sparsity.
- Eight distinct types of graph
  - **Easy, Medium, Hard** for simpler tasks (Connectivity, Cycle, Topo Sort)
  - **Easy, Hard** for advanced tasks (Shortest Path, Max Flow, Bipartite Matching, etc.)
- 5,902 problems in the standard version, 29,370 problems in the deluxe version.
- **Accuracy** (yes/no, or valid solution)
- **Partial credit** for tasks like Shortest Path, Max Flow, GNN if close to optimal

LLM Strategy: Zero-shot prompting

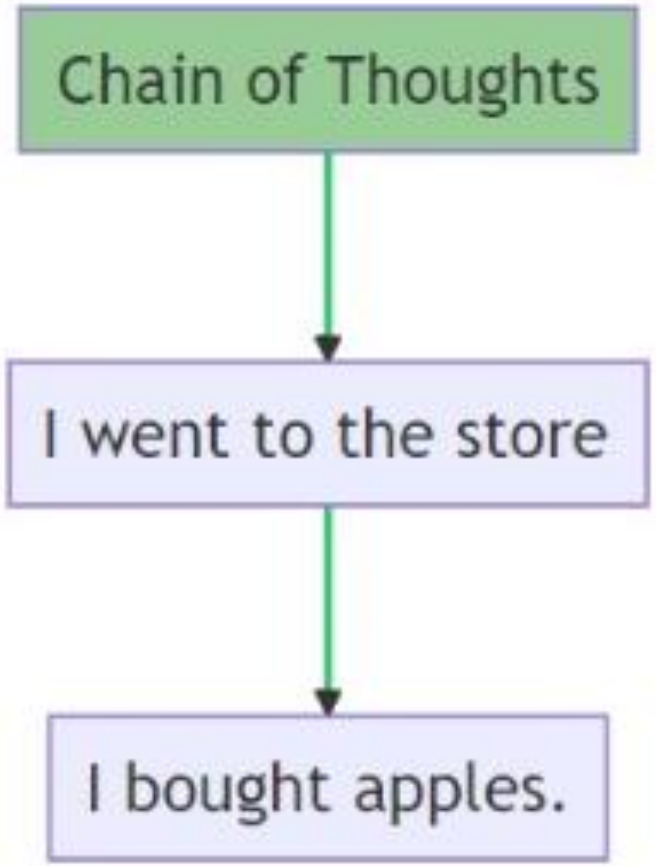


Zero-shot

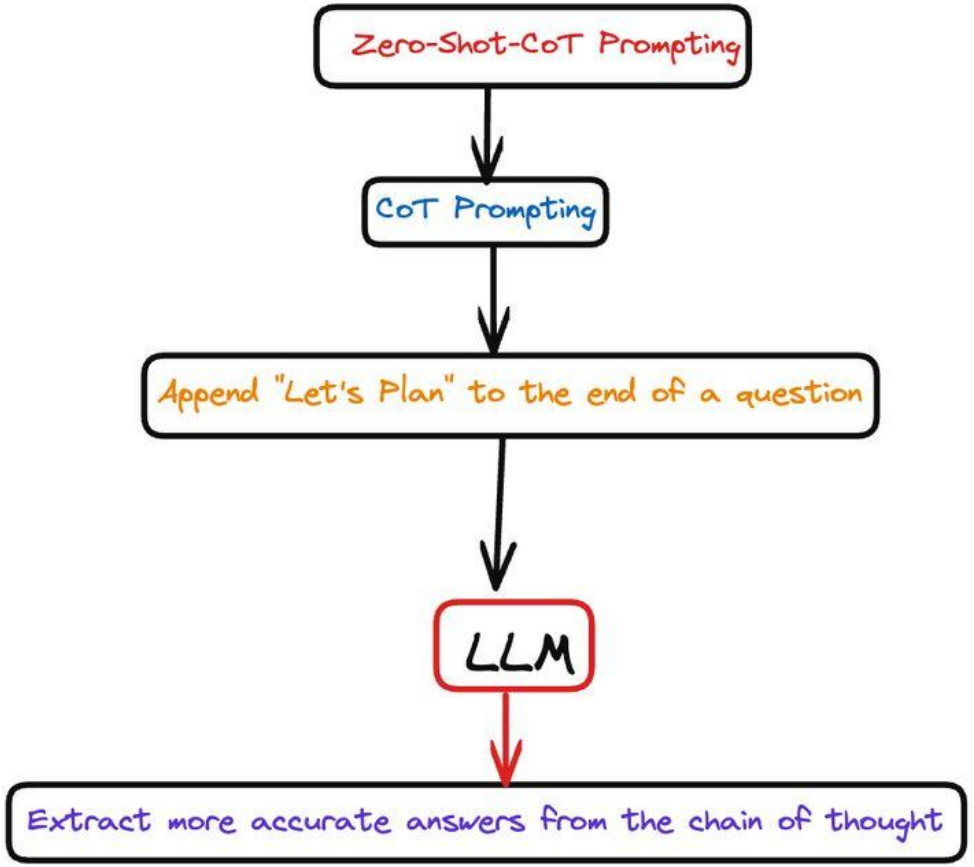
LLM Strategy: Few-shot prompting



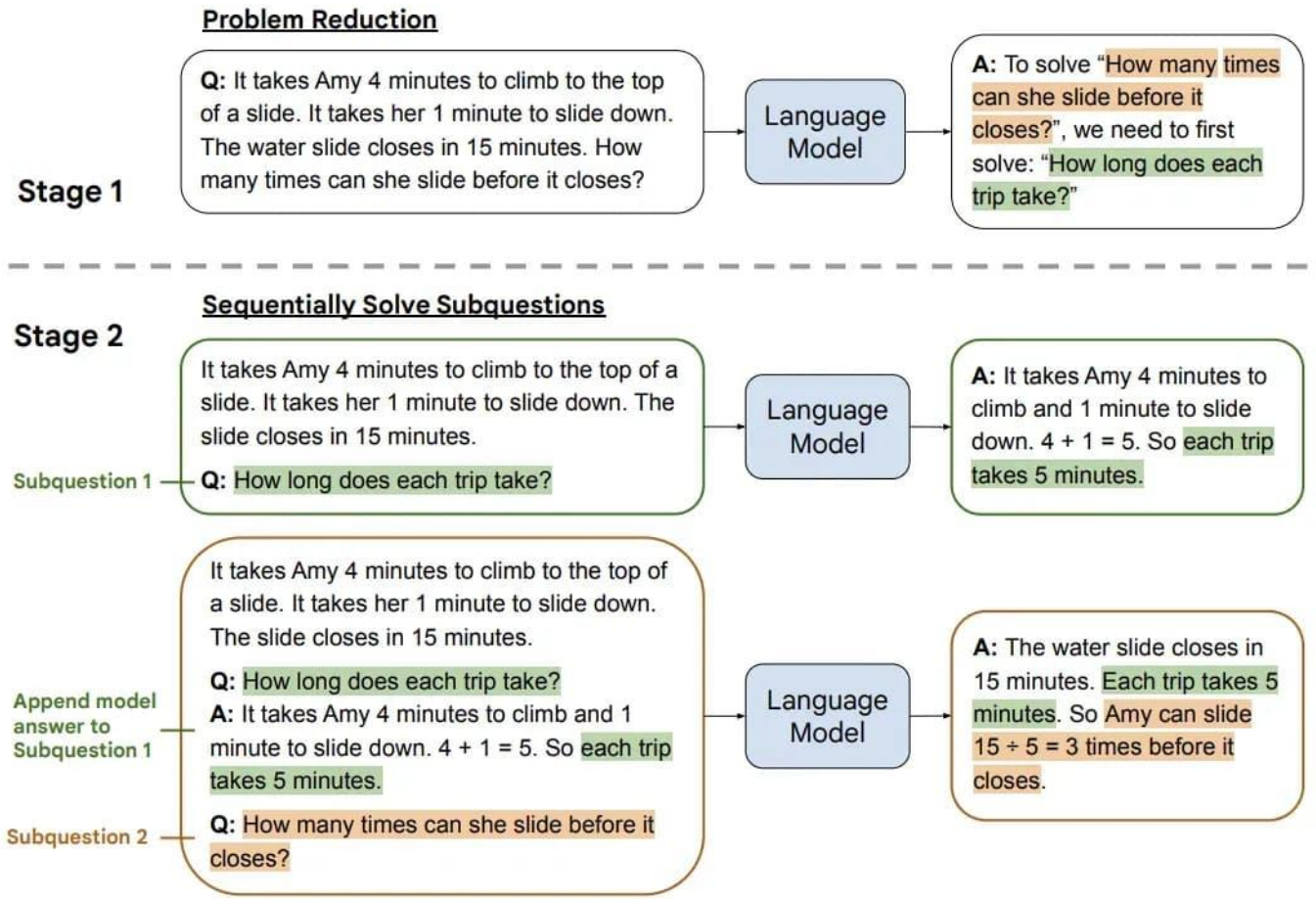
LLM Strategy: Chain-of-thought (COT)



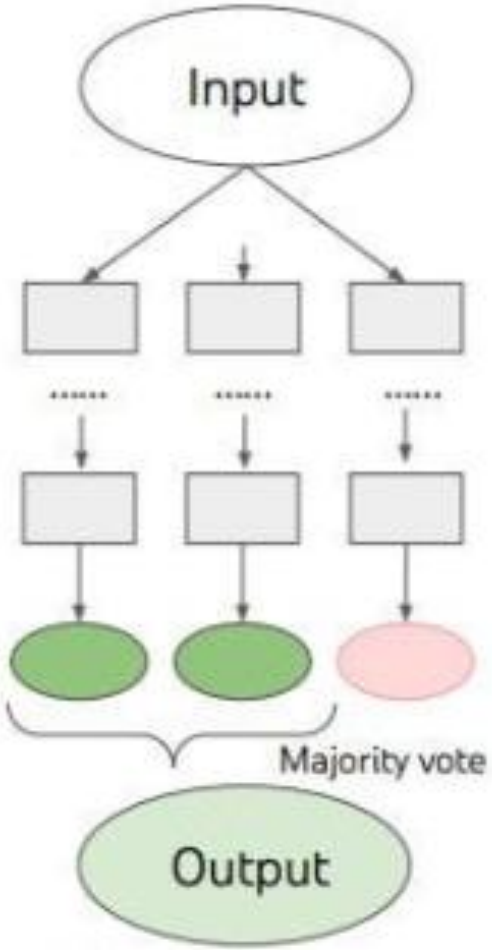
LLM Strategy: Zero-shot chain-of-thought (0-CoT)



LLM Strategy: Least-to-most (LTM)



LLM Strategy: COT self-consistency (COT-SC)



Experiment Setting

- Baselines: zero-shot prompting, few-shot in-context learning, chain-of-thought prompting, zero-shot chain-of-thought, least-to-most, and self-consistency.
- Also adopted random baseline for a fuller comparison.
- Model: text-davinci-003 as the default model, other LLMs (GPT-3.5-turbo, code-davinci-002, and GPT-4) are evaluated on part of the benchmark.

Results

- They first find that on simple graph reasoning tasks, LLMs achieve impressive performance and demonstrate preliminary graph thinking abilities.

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**

Table 2: 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.

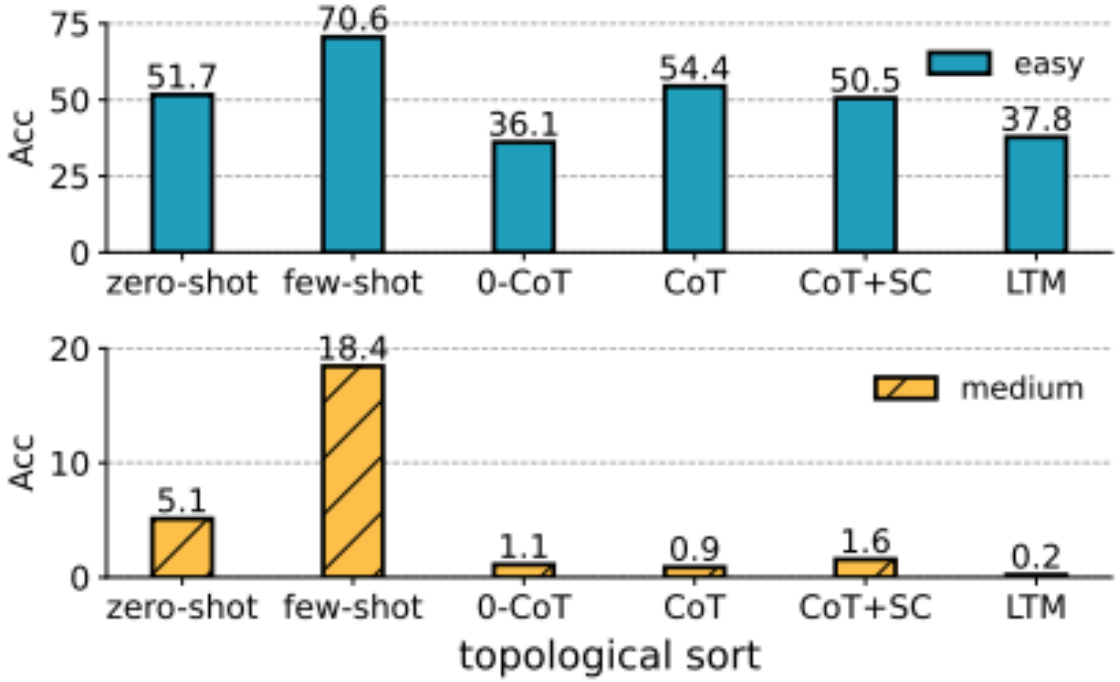
Mixed Results with Advanced Prompting

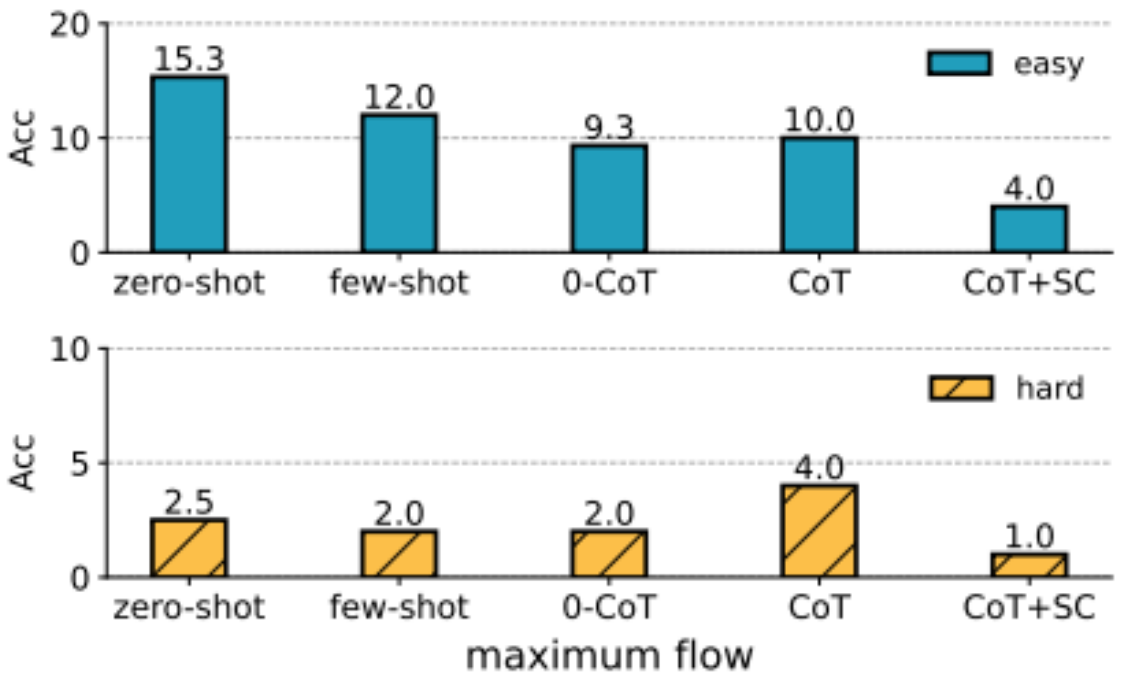
- Advanced prompting methods successfully improve performance on simple graph reasoning tasks. Same for **GNN's**

Method	PC (↑)	Acc (↑)	RE (↓)
ZERO-SHOT	13.61	0.00	20.04
FEW-SHOT	20.04	0.00	37.83
CoT	**64.55**	**31.00**	14.34
0-CoT	13.85	0.00	44.55
CoT+SC	63.92	28.00	**13.28**

Mixed Results with Advanced Prompting

- For topological sort task and maximum flow task, few-shot prompting outperforms advanced prompting techniques.

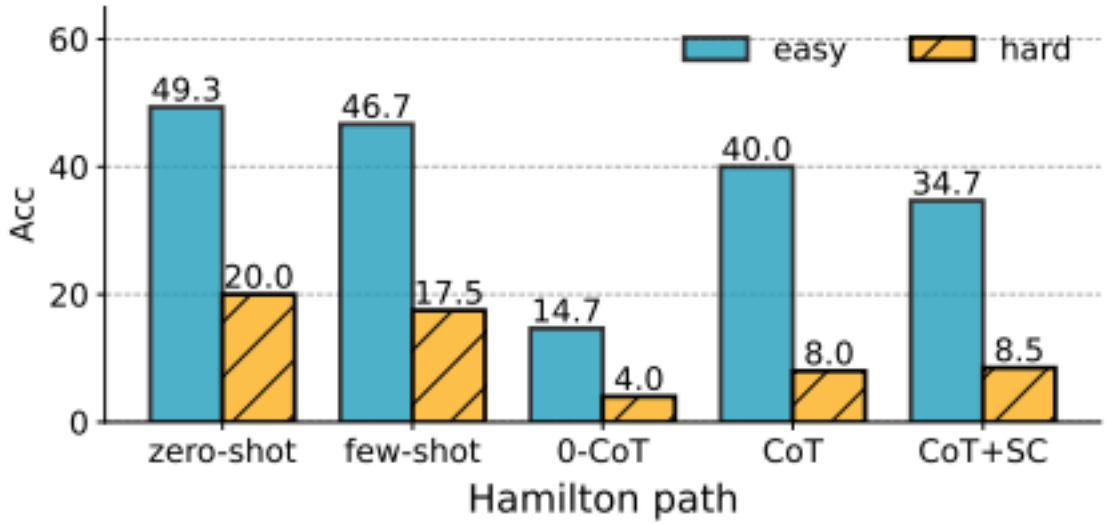


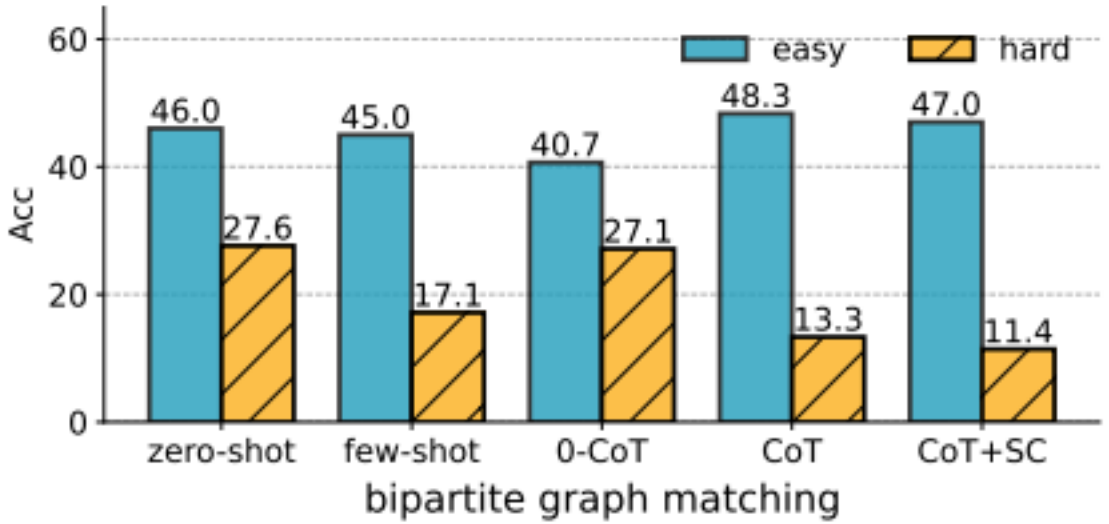


UNIVERSITY OF WATERLOO

In-Context Learning could be Counterproductive

- For Hamilton path task and bipartite graph matching task, zero-shot prompting consistently outperforms all other prompting techniques.

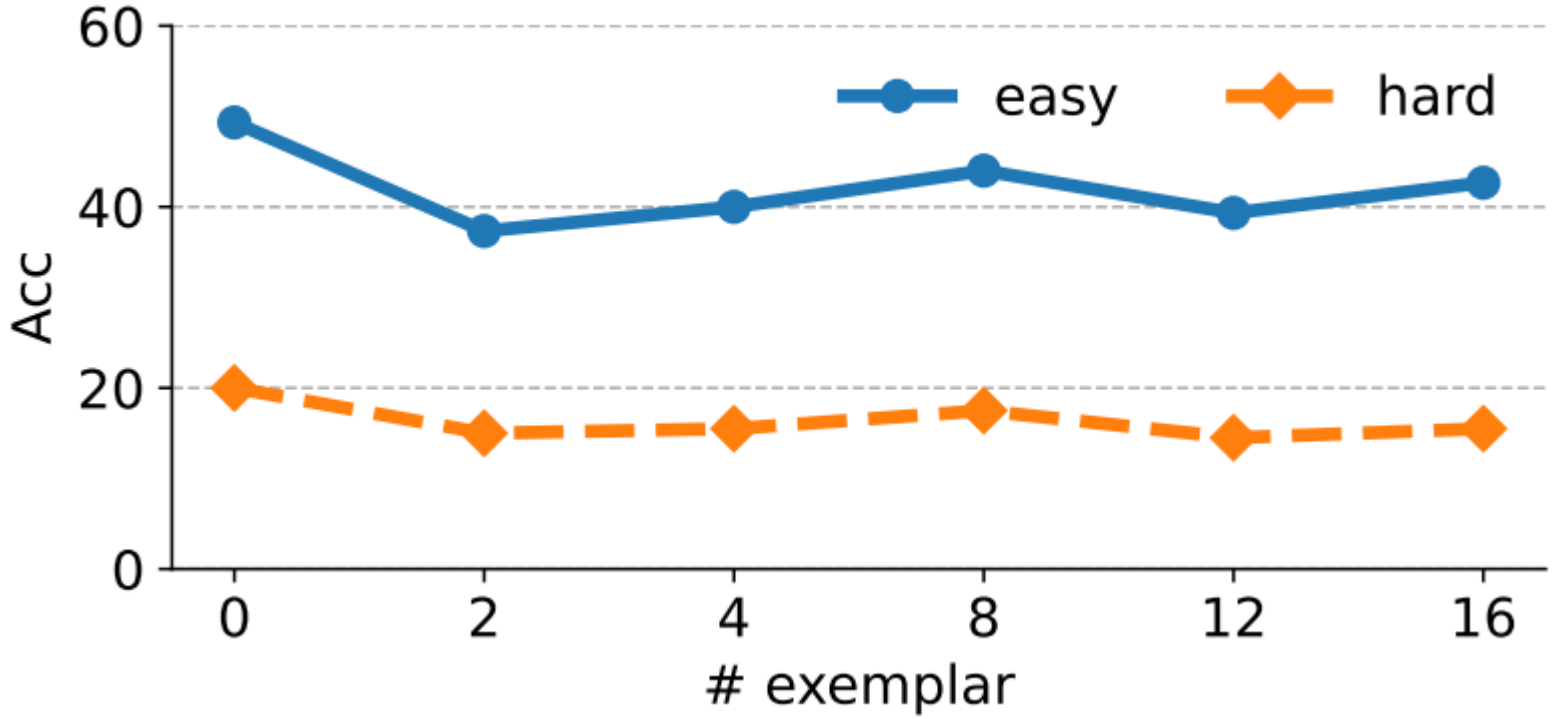




UNIVERSITY OF WATERLOO

In-Context Learning could be Counterproductive

- Performance does not improve with more exemplars, sometimes it even dips.



LLMs are (Un)surprisingly Brittle

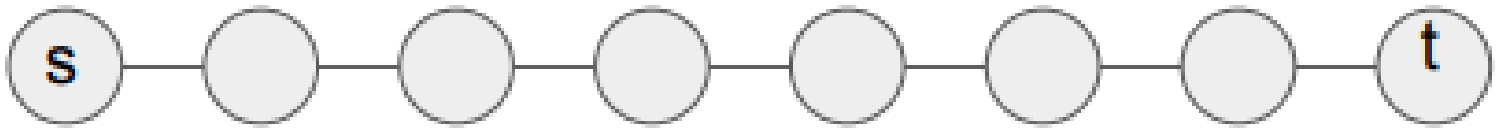
- **Context**: LLMs outperform random on simpler tasks
- **Hypothesis**: They use *spurious correlations* rather than genuine graph traversal
- **Goal**: Show that performance **drops drastically** when these correlations fail

LLMs are (Un)surprisingly Brittle

- **LLMs might count**:
  - Node **degree** (frequent mentions → "must be connected")
- **Reality**: High-degree nodes could be in different subgraphs = **not connected**
- **Chain & Clique** subsets: Designed to **invert** these cues

Chain Dataset

- **Graph Setup**: Multiple chain components (linear paths)
- **Query Pairs**: Endpoints of each chain (degree = 1)
- **Trick**: Nodes look “isolated,” but are connected via the chain
- **120 Examples** in this subset



Clique Dataset

- **Graph Setup**: k densely connected subgraphs
- **Query Pairs**: Nodes from **different** dense subgraphs
- **Trick**: Nodes have **many edges** (high degree) but **no inter-subgraph link**
- **120 Examples** in this subset

Results

Dataset	ZERO-SHOT	FEW-SHOT	CoT	0-CoT	CoT+SC	Avg.
GENERAL	74.67	83.33	85.33	66.00	82.67	78.40
CHAIN	51.67 (-23.00)	45.00 (-35.33)	40.83 (-44.50)	92.50 (+26.50)	44.17 (-38.50)	54.83 (-23.57)
CLIQUE	60.83 (-13.84)	73.33 (-10.00)	85.00 (-0.33)	52.50 (-13.50)	83.33 (+0.66)	71.00 (-7.40)

Making Language Models Better Graph Reasoners

- **Goal**: Improve LLM performance on graph tasks
- Introduce **new prompting** methods
  - **Build-a-Graph Prompting (BAG)**
  - **Algorithmic Prompting**

Build-a-Graph Prompting (BAG)

- Construct a graph with the nodes and edges first.
- Forces LLM to “visualize” or **enumerate** the graph

Build-a-Graph Prompting

<in-context exemplar>
In an undirected graph, the nodes are numbered from 0 to 4, and the edges are: an edge between node 0 and node 4 with weight 4, ⋯

Let's construct a graph with the nodes and edges first.

Q: Give the shortest path from node 0 to node 2.
A: All the paths from node 0 to node 2 are: 0,3,2 with a total weight of 3 + 1 = 4, 0,1,4,2 with a total weight of 3 + 4 + 2 = 9, 0,4,3,2 with a total weight of 4 + 1 + 1 = 6. The weight of path 0,3,2 is the smallest, so the shortest path from node 0 to node 2 is 0,3,2 with a total weight of 4.

Algorithmic Prompting

- LLMs **recall** or **simulate** an algorithm
- We can use a Depth-First Search (DFS) algorithm to…

Algorithmic Prompting

We can use a Depth-First Search (DFS) algorithm to find the shortest path between two given nodes in an undirected graph.
The basic idea is to start at one of the nodes and use DFS to explore all of its adjacent nodes. At each node, you can keep track of the distance it takes to reach that node from the starting node.
Once you have explored all the adjacent nodes, you can backtrack and pick the node which has the shortest distance to reach the destination node.
<in-context exemplar>
In an undirected graph, the nodes are numbered from 0 to 4, and the edges are:
an edge between node 0 and node 4 with weight 4, ...
Q: Give the shortest path from node 0 to node 2.
A: All the paths from node 0 to node 2 are:
0,3,2 with a total weight of 3 + 1 = 4,
0,1,4,2 with a total weight of 3 + 4 + 2 = 9,
0,4,3,2 with a total weight of 4 + 1 + 1 = 6.
The weight of path 0,3,2 is the smallest, so the shortest path from node 0 to node 2 is 0,3,2 with a total weight of 4.

Results

Method	Cycle				Shortest Path					Hamilton Path		
	Easy	Medium	Hard	Avg.	Easy	Hard	Easy (PC)	Hard (PC)	Avg.	Easy	Hard	Avg.
CoT	84.67	63.33	53.25	66.75	63.89	29.50	76.84	35.79	51.51	**40.00**	**8.00**	**24.00**
CoT+BAG	**86.00**	69.33	62.00	**72.44**	**67.78**	**33.50**	**79.20**	**42.56**	**55.76**	38.67	6.00	22.34
CoT+ALGORITHM	77.33	**74.00**	**64.00**	71.78	63.89	28.00	76.06	38.70	51.66	36.67	7.50	22.09

Future Work & Limitations

- Incomplete Tasks
  - Only eight graph tasks (connectivity, cycle, etc.).
- Limited LLM Coverage
- Dataset Constraints
- Prompting Methods

Thank You!