Learn to Think: Bootstrapping LLM Logic Through Graph Representation Learning

Hang Gao[1,2], **Chenhao Zhang**[1,2,3], **Tie Wang**[4*], **Junsuo Zhao**[1,2,3], **Fengge Wu**[1,2,3*], **Changwen Zheng**[1,2,3] and **Huaping Liu**[5]

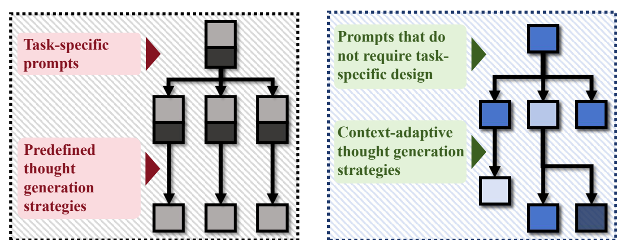
[1] Institute of Software, Chinese Academy of Sciences.
[2] National Key Laboratory of Space Integrated Information System.
[3] University of Chinese Academy of Sciences.
[4] Peking University.
[5] Tsinghua University.
{gaohang, zhangchenhao2024, fengge, changwen, junsuo}@iscas.ac.cn,
wangtie2021@stu.pku.edu.cn, hpliu@tsinghua.edu.cn

Abstract

Large Language Models (LLMs) have achieved remarkable success across various domains. However, they still face significant challenges, including high computational costs for training and limitations in solving complex reasoning problems. Although existing methods have extended the reasoning capabilities of LLMs through structured paradigms, these approaches often rely on task-specific prompts and predefined reasoning processes, which constrain their flexibility and generalizability. To address these limitations, we propose a novel framework that leverages graph learning to enable more flexible and adaptive reasoning capabilities for LLMs. Specifically, this approach models the reasoning process of a problem as a graph and employs LLM-based graph learning to guide the adaptive generation of each reasoning step. To further enhance the adaptability of the model, we introduce a Graph Neural Network (GNN) module to perform representation learning on the generated reasoning process, enabling real-time adjustments to both the model and the prompt. Experimental results demonstrate that this method significantly improves reasoning performance across multiple tasks without requiring additional training or task-specific prompt design. *Code can be found in https://github.com/zch65458525/L2T.*

1 Introduction

In recent years, LLMs [Radford *et al.*, 2018] have achieved remarkable success in fields such as natural language processing [Brown *et al.*, 2022], machine translation [Jiao *et al.*, 2022], and code generation [Ni *et al.*, 2022]. However, training these models requires substantial computational resources and energy, resulting in high costs and environmental impacts [Patterson *et al.*, 2022]. As a result, efficiently utilizing LLMs has become a key research focus, with prompt engineering emerging as a critical technique [Liu *et al.*, 2023; Zhou *et al.*, 2022; Sun *et al.*, 2022]. By designing effective prompts, it is possible to optimize model performance without additional training, making it a cost-effective and straightforward approach. Notably, the Chain-of-Thought (CoT) method [Wei *et al.*, 2022] has demonstrated significant improvements in tasks such as mathematical reasoning and logical inference by guiding models through step-by-step reasoning processes. CoT works by crafting prompts that break down complex problems into logical steps, enabling the model to solve them incrementally.



(a) Conventional Method. (b) Our Proposed Method.

Figure 1: A comparison between our method and conventional methods.

Based on Chain of Thoughts, numerous related methods have been proposed in recent years, including Tree of Thoughts (ToT)[Chu *et al.*, 2024], Graph of Thoughts (GoT)[Besta *et al.*, 2024], and Thread of Thoughts (ThoT)[Zhou *et al.*, 2023b]. These methods introduce more complex thinking paradigms, such as tree structures, graph structures, and thread-based reasoning, thereby further extending the reasoning capabilities of LLMs. Compared to chain-based reasoning structures, these approaches have significantly enhanced the breadth and depth of the cognition of LLMs [Qiao *et al.*, 2023]. They have played an active role in optimizing the performance of LLMs [Hadi *et al.*, 2024].

*Corresponding authors.

However, these methods still face several critical challenges.

First, they lack adaptability to different contexts. Existing approaches are often unable to make real-time adjustments to models and prompts in response to dynamic changes in scenarios, resulting in limited flexibility and robustness when addressing diverse tasks [Chu *et al.*, 2024]. Once the reasoning process begins, LLMs typically follow a predefined prompt and execute reasoning in a relatively fixed manner. This leads to a second issue: these methods often require task-specific prompt design to handle different tasks effectively, particularly for those involving more complex reasoning processes. This reliance is, to some extent, inevitable, as more intricate reasoning demands highly precise and specific prompts to effectively guide the model. Only with carefully crafted prompts can the models fully exploit their extended reasoning frameworks. Without sufficiently targeted prompts, their reasoning performance may degrade greatly, failing to achieve the desired cognitive outcomes. This heavy dependence on task-specific prompts poses a major limitation, severely undermining the generalizability of such methods. One possible solution is to collect task-specific data and use fine-tuning methods to train the LLM. However, this approach incurs significant costs in industrial scenarios and is not feasible for cases where only API access is available. Figure 1(a) provides a visual summary of these challenges.

Thus, a key question emerges: **is there a way to address different types of problems in a unified manner without requiring LLM training or additional prompt design, while also allowing the model to flexibly adjust based on the problem and reasoning process?** To achieve this, it is essential to establish a suitable unified framework to model the entire reasoning process of LLMs, enabling them to adopt different modes of thinking at appropriate moments, much like humans do.

To this end, we propose the ***Learn to Think*** (L2T) method, which guides the LLM to "think" based on graph learning. This method employs graphs to unify the representation of the reasoning process of LLMs across different tasks. These graphs are annotatable, enabling more effective representation and accurate prediction of reasoning strategies. Subsequently, L2T utilizes a graph learning approach based on LLMs to adaptively guide reasoning strategies for various scenarios. By combining such an approach with the automatic extraction of reasoning process formats and evaluation criteria from task descriptions, L2T effectively handles diverse tasks without relying on task-specific prompts. Then, L2T introduces a GNN-based reasoning mode selection module to perform relatively lightweight representation learning on the graph, facilitating the switch between different reasoning modes for LLMs. This enables real-time adjustments during the reasoning process, and the GNN-based reasoning mode selection module is further refined within a reinforcement learning framework. Figure 1(b) illustrates the advantages of the proposed method. In summary, our contributions are as follows:

- We propose an LLM reasoning framework that can adapt to different problems and develop reasoning pathways without requiring task-specific prompts.
- By integrating a GNN-based reasoning mode selection module, we enable real-time adjustment of the LLM reasoning strategies. Furthermore, the module can be continuously optimized through reinforcement learning.
- Extensive experiments are conducted to thoroughly validate and analyze the proposed method.

2 Related Works

Prompt engineering. Prompt engineering for LLMs has seen significant advancements, introducing innovative techniques aimed at enhancing reasoning and reliability. Methods such as CoT [Wei *et al.*, 2022] improve reasoning capabilities by incorporating intermediate steps, while self-consistency [Wang *et al.*, 2023] enhances reliability by aggregating consistent outputs. Interactive question answering further enables dynamic interactions with the model, facilitating adaptive reasoning processes [Yao *et al.*, 2023b; Masson *et al.*, 2024]. To mitigate hallucinations, Retrieval-Augmented Generation (RAG) [Lewis *et al.*, 2020] integrates external retrieval mechanisms to ensure factual accuracy. Additionally, methods like Chain-of-Verification (CoVe) [Dhuliawala *et al.*, 2024], Chain-of-Note (CoN) [Yu *et al.*, 2023], and Chain-of-Knowledge (CoK) focus on step-by-step validation for robust reasoning. Furthermore, prompt engineering research has also explored areas such as user intent understanding [Diao *et al.*, 2024], autonomous prompt selection [Zhou *et al.*, 2023a], external tool integration [Paranjape *et al.*, 2023], and emotional control in responses [Li *et al.*, 2023].

Logic and reasoning within LLM prompting. Efforts to enhance logic and reasoning in LLM prompting have introduced various innovative methods. Auto-CoT [Zhang *et al.*, 2023] automates the generation of reasoning chains, while Logical CoT (LogiCoT) [Zhao *et al.*, 2024] leverages symbolic logic for step-by-step verification. Prompt Sketching [Beurer-Kellner *et al.*, 2024] constrains outputs to predefined logical structures, ensuring coherence and adherence to logical frameworks. Topological frameworks have also been explored, such as ToT [Yao *et al.*, 2023a] and GoT [Besta *et al.*, 2024], which utilize hierarchical and graph-based structures, respectively, to model complex reasoning processes. Algorithm of Thoughts (AoT) [Sel *et al.*, 2024] employs in-context algorithmic examples to guide LLMs through structured reasoning pathways, while ThoT [Zhou *et al.*, 2023b] generates structured thought threads to decompose and address complex problems. Although these methods have made significant contributions, they typically follow predefined reasoning processes and depend heavily on task-specific prompts, limiting their adaptability and generalizability. In contrast, our method addresses these limitations by enabling more flexible and adaptive reasoning capabilities for LLMs. Additional details on related work are available in **Appendix A**.

3 Method

Our method consists of the following parts: first, representing the complete logical reasoning process of the LLM as a specifically designed graph. Second, automatically generate

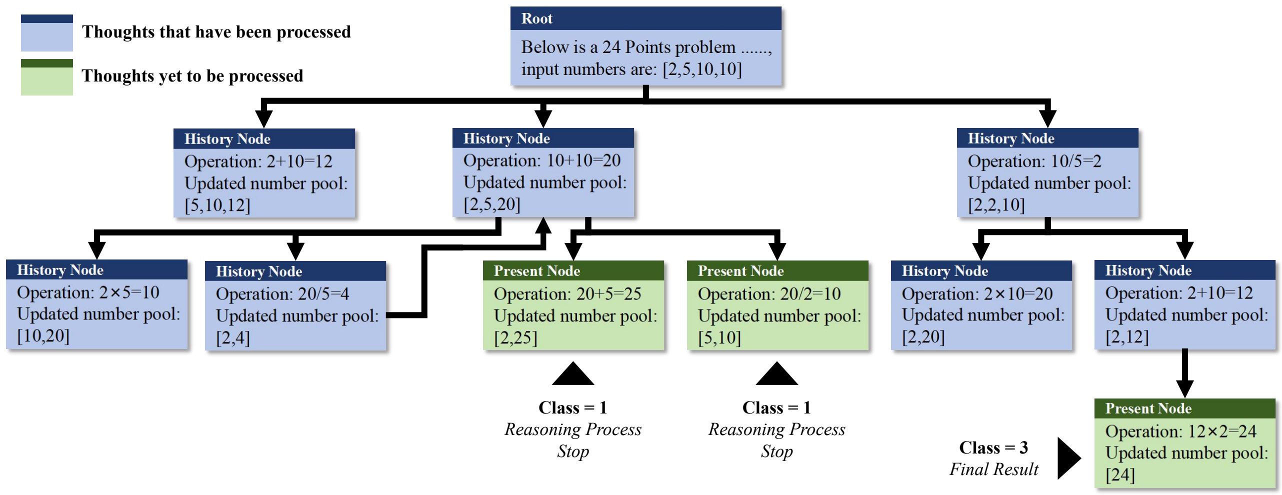


Figure 2: An example of the reasoning process graph. Each box contains a thought generated by the LLM, representing a node in the reasoning process graph. The green boxes in the graph indicate the nodes currently being processed. We classify these nodes and used their categories to guide the LLM's next steps.

the format and evaluation criteria of the reasoning process, then employ a graph learning framework to process the reasoning process graph, thereby facilitating flexible and adaptive multi-step problem-solving that does not require task-specific prompts. Finally, iteratively refining the proposed reasoning model through reinforcement learning. We will elaborate on them in detail.

3.1 Reasoning Process Graph

The conversation with the LLM consists of user messages (prompts) and the LLM's responses (thoughts). Extensive research has been conducted on how to organize such prompts and thoughts to optimize LLM performance [Liu *et al.*, 2023], leading to the proposal of various structures of thoughts, such as chain structures [Wei *et al.*, 2022], tree structures [Yao *et al.*, 2023a], graph structures [Besta *et al.*, 2024], etc. Among these, graphs are particularly effective for representing the reasoning frameworks of most existing models, as trees, chains, and other structures can be viewed as special cases of graphs. Building on this, our approach employs a specifically designed graph to represent logical reasoning, which we refer to as the *reasoning process graph*.

Particularly, we represent the entire reasoning process of an LLM as a reasoning process graph $G = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the set of nodes, with each node $v \in \mathcal{V}$ representing a thought generated by the LLM. Similarly, $\mathcal{E}$ denotes the set of edges, where each edge $e \in \mathcal{E}$ represents a connection from one thought to its subsequent thought.

The set $\mathcal{V}$ can be partitioned into two subsets: $\mathcal{V}^{\text{pres}}$ and $\mathcal{V}^{\text{hist}}$. Here, $\mathcal{V}^{\text{pres}}$ represents the nodes corresponding to unprocessed thoughts and will serve as the basis for generating subsequent thoughts. In contrast, $\mathcal{V}^{\text{hist}}$ represents the nodes that have already been processed and will no longer be revisited.

Each node v in $\mathcal{V}^{\text{pres}}$ is assigned a category label Y_v, where $Y_v \in \{1, 2, 3, 4\}$. The meaning of each label is as follows:

- **Label 1:** Reasoning should not proceed based on node v.
- **Label 2:** Reasoning should continue based on node v.
- **Label 3:** Node v should be output as the final result.
- **Label 4:** A backtracking operation should be performed on node v, meaning that reasoning should continue based on its parent node.

To assign specific labels to each node in $\mathcal{V}^{\text{pres}}$, we utilize LLM-based graph learning for node classification. These labels are subsequently employed to guide the thought generation process. By leveraging this approach, L2T eliminates the need for task-specific prompts to direct the reasoning process. Instead, the labels effectively determine how the reasoning proceeds. In the following sections, we will elaborate on this process in detail. Figure 2 gives an illustration example for the reasoning process graph.

3.2 Thought Generation Framework

Next, we introduce our thought generation framework. Since the reasoning process is carried out step by step, we will explain in detail how reasoning is performed at the first step, the intermediate k-th step, and the final step, respectively. The overall framework is given in Figure 3.

First Step

In the first step, we begin by obtaining an initial state. Using the LLM, we generate three components: the initial reasoning process graph $G^{(1)}$, the constraint format and examples for the process, and the evaluation criteria for the generated thoughts. Specifically, the initial reasoning process graph is defined as $G^{(1)} = \{\mathcal{V}^{(1)}, \mathcal{E}^{(1)}\}$. In $G^{(1)}$, the subscript "1" in parentheses corresponds to the iteration step one. $\mathcal{V}^{(1)}$ and $\mathcal{E}^{(1)}$ represent the sets of nodes and edges in $G^{(1)}$, respectively. At this stage, $|\mathcal{V}^{(1)}| = 1$ and $\mathcal{E}^{(1)} = \varnothing$, as $G^{(1)}$ contains only a single initial node. The node in $G^{(1)}$ is assigned to $\mathcal{V}^{\text{pres}(1)}$. The attribute of this node is the task description.

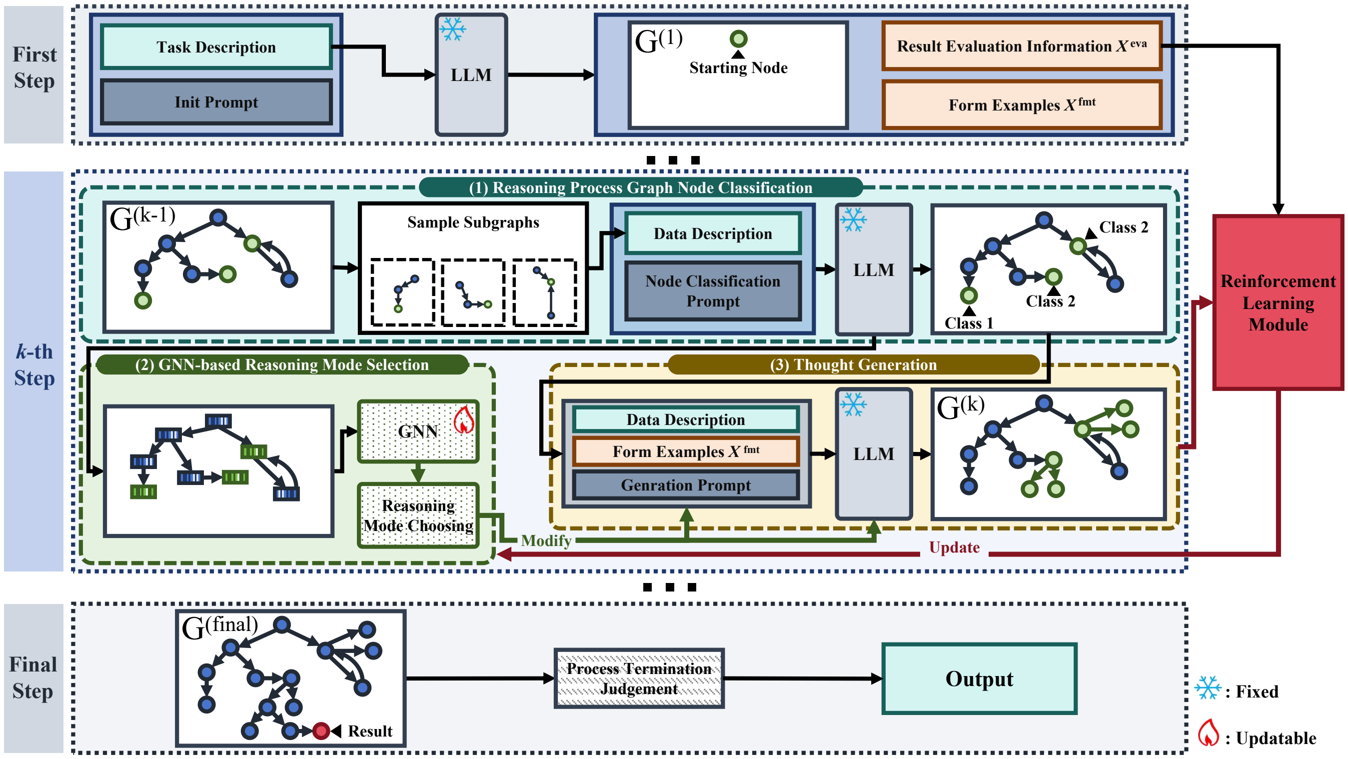


Figure 3: The framework of the proposed method. All LLM modules uniformly utilize the same LLM.

Additionally, we utilize the LLM to directly produce X^{fmt} that includes format descriptions and a set of corresponding example answers for new thought generation. Furthermore, based on the LLM, we extract relevant information X^{eva} from the task description that pertains to the criteria for evaluating and scoring the quality of the model's output. The above design ensures that L2T can perform step-by-step reasoning for complex problems in a unified format without relying on task-specific prompts, while also providing a reasonable evaluation of task execution. Details of all L2T prompts can be found in **Appendix B.5**.

k-th Step

For the k-th step, we generate the subsequent thoughts to construct $G^{(k)}$ based on $G^{(k-1)}$. L2T first conducts reasoning process graph node classification, then achieves GNN-based reasoning mode selection. Based on the classification information and the selected reasoning mode, L2T finally generates the thoughts. We will elaborate on the details in the following content.

(1) Reasoning process graph node classification. The node classification is performed for all nodes within $\mathcal{V}^{\text{pres}(k-1)}$. For each node $v \in \mathcal{V}^{\text{pres}(k-1)}$, we extract its corresponding subgraph $\widetilde{G}_v^{(k-1)}$, where $\widetilde{G}_v^{(k-1)} = \{\widetilde{\mathcal{V}}_v^{(k-1)}, \widetilde{\mathcal{E}}_v^{(k-1)}\}$. Here, $\widetilde{\mathcal{V}}_v^{(k-1)}$ represents the set of all nodes in $G^{(k-1)}$ that have paths pointing to node v with a path length less than β, where β is a predefined hyperparameter. The edge set $\widetilde{\mathcal{E}}_v^{(k-1)}$ is defined as:

$$\widetilde{\mathcal{E}}_v^{(k-1)} = \{(u,w) \in \mathcal{E}^{(k-1)} \mid u \in \widetilde{\mathcal{V}}_v^{(k-1)}, w \in \widetilde{\mathcal{V}}_v^{(k-1)}\}. \quad (1)$$

Thus, $\widetilde{G}_v^{(k-1)}$ is the induced subgraph of $G^{(k-1)}$ whose vertex set is $\widetilde{\mathcal{V}}_v^{(k-1)}$. The provided information will be utilized as input to the LLM to perform node classification. Beyond the node attributes, the topological relationships between the target node v and other nodes will be annotated and expressed in textual form. This annotation process is straightforward, as the neighboring nodes primarily represent historical or backtracking information. The overall node classification process can be mathematically expressed as follows:

$$\dot{Y}_v^{(k)} = f\left(S^{\text{node}}, \tau\left(\{x_u \mid u \in \widetilde{\mathcal{V}}_v^{(k-1)}\}, \widetilde{G}_v^{(k-1)}\right)\right), \quad (2)$$

where S^{node} represents the prompts designed for node classification, $\dot{Y}_v^{(k)}$ denotes the estimated label, x_u denotes the textual representation of the reasoning thought associated with node u, $f(\cdot)$ denotes the LLM, and $\tau(\cdot)$ is the function that converts graph-related information into a descriptive textual format. Further details regarding the implementation of $\tau(\cdot)$ are provided in **Appendix B.3**.

(2) GNN-based reasoning mode selection. Our GNN-based reasoning mode selection module processes the graph $G^{(k-1)}$ using a GNN [Kipf and Welling, 2017; Wu *et al.*, 2020] $g(\cdot)$, which is a deep learning model designed to process and analyze graph-structured data by leveraging the relationships between nodes and edges. The GNN takes an attributed graph as input and outputs feature vectors for each node. For the implementation of $g(\cdot)$, we utilize a one-layer Graph Convolutional Network (GCN) [Kipf and Welling, 2017] followed by a two-layer Multi-Layer Perceptron (MLP). During the aforementioned node classification, each node v in $G^{(k-1)}$ save the final-layer representation h_v generated by the LLM as the node feature vector for this stage. Here, h_v is the representation corresponding to the last output token in the answer sequence. These output representations are subsequently transformed into vectors denoted as $\mathbf{a}$. Specifically, each reasoning node $v^{(k)}$ in $\mathcal{V}^{\text{pres}(k)}$ is associated with a vector $\mathbf{a}_v^{(k)}$. The vector $\mathbf{a}_v^{(k)}$ consists of a set of parameters, including adjustable prompt-related parameters (e.g., the number of generated branches) as well as LLM hyperparameters (e.g., the temperature parameter). Formally, $\mathbf{a}_v^{(k)}$ is defined as:

$$\mathbf{a}_v^{(k)} = A\big(g_{[v]}(G^{(k-1)})\big), \tag{3}$$

where $A(\cdot)$ denotes the function that outputs $\mathbf{a}_v^{(k)}$ based on $g_{[v]}(G^{(k-1)})$, $g_{[v]}(G^{(k-1)})$ is the GNN output representation of node v at the $(k-1)$-th step. In fact, $A(\cdot)$ implements the Actor mechanism in the Actor-Critic algorithm [Konda and Tsitsiklis, 1999], and the implementation details of this function will be elaborated in Section 3.3. We treat $\mathbf{a}_v^{(k)}$ as an action of choosing a mode of reasoning, the model will be iteratively updated to optimize the selection of modes. Further details regarding $\mathbf{a}_v^{(k)}$ can be found in **Appendix B.4**.

(3) Thought generation. Finally, we carry out thought generation. According to the classes described in section 3.1, for a given node v, new nodes need to be generated only when the label of v is 2, and the newly generated nodes are all child nodes of v. For other types of nodes, only deletion, modification, and adjustment of set membership are required, which can be directly addressed through standardized processing on $G^{(k-1)}$. The standardized processing can be implemented through straightforward code development. Subsequently, we input the prompts, pre-generated template examples, and the textual description of node attributes into an LLM, enabling it to generate the subsequent thought nodes when the label of v is 2. The process of generating the textual features of a child node u based on the content of its parent node v can be formalized as follows:

$$x_u = f(S^{\text{gen}}, x_v^{(k-1)}, X^{\text{fmt}}, \mathbf{a}_v^{(k)}), \tag{4}$$

where x_u represents the textual features of the node u, and S^{gen} denotes the prompt used for data generation. Note that a portion of the prompt is determined by $\mathbf{a}_v^{(k)}$, which also influences the hyperparameters of the LLM. Based on x_u, along with other standardized processing, the graph $G^{(k)}$ can then be constructed. The newly generated child nodes, together with the backtracked parent nodes, will form $\mathcal{V}^{\text{pres}(k)}$.

Final Step

The reasoning process concludes when the final result emerges. This occurs when the current set, denoted as $\mathcal{V}^{\text{pres}}$, contains a node labeled as 3, signifying the appearance of the final result. At this point, all intermediate steps and iterations cease, and the process terminates.

Additionally, if all nodes in $\mathcal{V}^{\text{pres}}$ have their corresponding thoughts labeled as 1 (indicating that reasoning stops at the current thought), these thoughts will then be regenerated. If they are still labeled as 1, the process will also terminate.

3.3 Update

We employ the Actor-Critic algorithm from reinforcement learning to optimize and update the GNN-based reasoning mode selection module, which comprises $g(\cdot)$ and $A(\cdot)$. These components work together to produce the output $\mathbf{a}_v^{(k)}$. The Actor-Critic algorithm uses two models: the Actor, which selects actions based on the policy, and the Critic, which evaluates the actions by estimating the value function to improve the policy. Please refer to **Appendix E.2** and **E.3** for detailed introductions. Assuming we are at the k-th step, we first consider the case where there is only one node in $G^{(k-1)}$ that needs to be processed, i.e., $|\mathcal{V}^{\text{pres}(k)}| = 1$, and the pending node is v. As mentioned in the previous section, at step k, we regard $\mathbf{a}_v^{(k)}$ as an action of choosing the mode of reasoning. At this point, $g_{[v]}(G^{(k-1)})$, i.e., the GNN output representation of node v at the $(k-1)$-th step, is treated as the input state.

The Actor, which is represented as $A(\cdot)$, is used to generate the action $\mathbf{a}_v^{(k)}$, which represents the selected reasoning mode. At the k-th step, we calculate an action distribution $\pi(\mathbf{a}_v^{(k)}|g_{[v]}(G^{(k-1)});\theta_{\text{actor}})$ based on a single-layer MLP with θ_{actor} as the parameters, and action $\mathbf{a}_v^{(k)}$ is sampled from this distribution. The process can be formulated as:

$$\mathbf{a}_v^{(k)} \sim \pi(\mathbf{a}_v^{(k)}|g_{[v]}(G^{(k-1)});\theta_{\text{actor}}), \tag{5}$$

π denotes the strategy distribution. The parameters of π is output with the MLP, which takes $g_{[v]}(G^{(k-1)})$ as its input. Next, we acquire an immediate reward r_k and the next state $g_{[v]}(G^{(k)})$. The reward r_k is set to 100 if the generated thought represents the final result. Otherwise, it is an integer between 0 and 10, determined by the LLM based on $G^{(k)}$ and X^{eva}. The detailed prompt used for this process is provided in **Appendix B.5**.

The Critic evaluates the performance of the current strategy by estimating the state value function $V\big(g_{[v]}(G^{(k-1)})\big)$, which is also implemented using a single-layer MLP with θ_{critic} as the parameters.

We adopt the widely used PPO framework [Schulman *et al.*, 2017] for LLM training as the specific implementation of the Actor-Critic algorithm, optimizing and updating the Actor and Critic that we have constructed. Through collaborative optimization, the policy network gradually learns a better strategy for selecting reasoning modes, enabling the model to dynamically optimize inference efficiency and performance under different graph states.

For graphs with multiple pending nodes, i.e., $|\mathcal{V}^{\text{pres}(k)}| > 1$, each node is processed sequentially as different steps, with optimization and updates performed individually.

Method	3×3 Sudoku			4×4 Sudoku			5×5 Sudoku			4×4 Sudoku w/o TSP		
	Average	Min	Max	Average	Min	Max	Average	Min	Max	Average	Min	Max
IO	43.85±10.44	4/13	8/13	24.62±10.50	0/13	4/13	10.77±6.41	0/13	3/13	*24.62±10.50*	*0/13*	*4/13*
CoT (zero-shot)	61.54±9.87	5/13	9/13	33.08±9.47	3/13	7/13	13.85±8.70	0/13	4/13	10.77±9.54	0/13	5/13
CoT (few-shot)	80.77±9.54	9/13	12/13	57.69±10.92	5/13	9/13	46.92±10.32	4/13	8/13	30.00±12.91	1/13	7/13
ToT	92.31±4.39	12/13	13/13	72.31±5.99	8/13	12/13	63.85±10.44	5/13	10/13	34.62±13.91	1/13	9/13
GoT	95.38±5.19	12/13	13/13	72.35±11.47	8/13	13/13	67.69±10.92	5/13	11/13	37.69±15.89	2/13	9/13
AoT	97.65±4.37	12/13	13/13	77.69±7.25	8/13	12/13	69.41±9.66	8/13	12/13	36.47±13.67	2/13	9/13
L2T w/o GNN	98.46±3.61	11/13	13/13	93.08±9.47	9/13	13/13	**89.46±9.87**	9/13	**13/13**	*93.08±9.47*	*9/13*	*13/13*
L2T	**100.00±0.00**	**13/13**	**13/13**	**98.46±3.76**	**12/13**	**13/13**	89.23±6.41	**10/13**	13/13	***98.46±3.76***	***12/13***	***13/13***

Table 1: Results for performance on Sudoku. **Bold** denotes the best result, and underline denotes the second best. For tied results in either first or second place, the performance is determined by comparing other relevant results within the same group. Min and Max represent the best and worst performances achieved by a method, respectively, in terms of the number of correct solutions out of 13 total puzzle sets. Results for 4×4 Sudoku w/o TSP (without task-specific prompts) reflect the performance of models when task-specific prompts are removed. Since IO, L2T w/o GNN, and L2T do not use task-specific prompts by design, their results are directly copied from the corresponding problem and are shown in *italic* to indicate this.

Method	Game of 24	Game of 24 w/o TSP
IO	15.92±1.89	*15.92±1.89*
CoT (zero-shot)	28.63±0.86	25.82±2.01
CoT (few-shot)	30.34±2.21	26.12±2.23
ToT	70.52±3.26	48.12±1.18
GoT	72.30±1.55	*48.15±1.28*
AoT	74.23±1.59	*27.54±7.76*
L2T w/o GNN	77.45±1.17	*77.45±1.17*
L2T	**80.42±2.98**	***80.42±2.98***

Table 2: Results for performance on Game of 24. **Bold** denotes the best result, and underline denotes the second best. Results for Game of 24 w/o TSP reflect the performance of models when task-specific prompts are removed. As Table 1, *italic* denotes the results that are directly copied from the corresponding problem, as the corresponding method do not use task-specific prompts by design.

Method	3 Characters	4 Characters	5 Characters	3 Characters w/o TSP
IO	36.83±1.57	35.06±6.73	6.57±2.04	*36.83±1.57*
CoT (zero-shot)	40.92±0.71	38.91±1.09	10.52±0.85	37.85±0.96
CoT (few-shot)	45.24±1.47	39.43±1.24	13.42±2.25	42.05±1.24
ToT	53.68±2.65	47.82±0.81	17.58±0.34	49.16±1.37
GoT	51.42±1.80	47.95±1.24	16.72±0.28	48.85±1.06
AoT	53.15±1.34	46.69±1.80	16.41±1.01	41.94±1.14
L2T w/o GNN	67.74±1.09	54.84±3.01	25.81±2.58	*67.74±1.09*
L2T	**69.31±0.64**	**59.75±0.99**	**27.93±0.05**	***69.31±0.64***

Table 3: Results for performance on TruthQuest. **Bold** denotes the best result, and underline denotes the second best. Results for 3 Characters w/o TSP reflect the performance of models upon 3 Characters TruthQuest when task-specific prompts are removed. As Table 1, *italic* denotes the results that are directly copied from the corresponding problem, as the corresponding method do not use task-specific prompts by design.

4 Experiments

4.1 Comparison with State-of-the-Art Methods

Baselines

For our experiments, we utilized GPT-4o as the base model. First, we directly compared our proposed L2T method with the original output of GPT-4o[OpenAI, 2023] (denoted as IO). Subsequently, we compared L2T with several advanced LLM reasoning methods, including CoT [Wei *et al.*, 2022], ToT [Yao *et al.*, 2023a], GoT [Besta *et al.*, 2024], and AoT [Sel *et al.*, 2024]. Among these, we specifically analyzed both the zero-shot and few-shot versions of CoT.

Tasks

We evaluated our method on four distinct tasks: Sudoku, the Game of 24, TruthQuest [Mondorf and Plank, 2024], and Creative Writing. These tasks were chosen as they are commonly used in the evaluation of similar methods [Yao *et al.*, 2023a; Besta *et al.*, 2024].

The Sudoku task is a logic-based puzzle involving the placement of numbers within a grid according to specific rules, we adopted 3 sizes, 3×3, 4×4, and 5×5. Game of 24 is a mathematical puzzle where players use four given numbers and basic arithmetic operations to reach a total of 24. TruthQuest [Mondorf and Plank, 2024] is a recently introduced benchmark for evaluating the reasoning and verification abilities of LLMs. Creative Writing task consisted of a series of diverse writing challenges (designed to avoid redundancy in task definitions) to assess the logical and conceptual abilities of LLMs in generating coherent and creative text. More details can be found in **Appendix C**.

For all tasks, the L2T method was tested using identical prompts, ensuring a consistent evaluation framework.

Settings

We utilized the GPT-4o API to conduct all the experiments, including those for the baselines. We also present the performance of L2T w/o GNN, which refers to L2T without the GNN-based reasoning mode selection module and, as a result, does not require any training.

Furthermore, we conducted additional experiments (marked in orange) that removed the task-specific components of methods including CoT, ToT, GoT, and AoT. Further details regarding the experimental settings and hyperparameter configurations can be found in **Appendix B**.

Results

Next, we analyze the results across different tasks. Tables 1, 2, and 3 summarize the results for Sudoku, Game of 24,

Method	Sentence Formation (Less Hints)				Sentence Formation (More Hints)				Text Expansion			
	Higher	Same	Lower	Std.	Higher	Same	Lower	Std.	Higher	Same	Lower	Std.
IO	93.06	6.93	0.00	±3.92	82.67	17.33	0.00	±3.76	51.93	38.91	9.16	±2.24
CoT	62.87	36.14	0.00	±3.22	61.39	38.61	0.00	±3.08	42.28	41.78	15.94	±1.24
ToT	48.27	50.24	1.49	±2.90	50.74	47.02	2.23	±2.53	41.98	36.83	21.19	±2.83
GoT	47.77	49.99	2.23	±2.56	49.75	48.02	2.23	±3.33	41.88	35.64	22.48	±2.65
AoT	48.82	49.06	2.11	±1.88	48.12	49.05	2.82	±2.18	44.24	36.82	18.94	±2.44
L2T w/o GNN	15.05	50.84	34.11	±3.38	15.38	39.13	45.48	±3.84	15.88	64.08	20.04	±4.10

Table 4: Comparison of method performance on the Creative Writing task. All data represent the performance of L2T comparisons to other methods. *Higher* indicates cases where L2T achieved a better score compared to the corresponding method. *Same* represents cases where L2T achieved the same score as the corresponding method. *Lower* indicates cases where the L2T scored worse compared to the corresponding method.

Method	Accuracy (%)	Generated Nodes
L2T	80.42±2.98	36.14±9.29
L2T w MLP	78.20±1.36	40.29±9.87
L2T w/o RL	78.85±1.42	43.13±8.61
L2T w/o GNN	77.45±1.17	46.56±21.11

Table 5: Comparison of accuracy and number of generated nodes for different methods.

Method	Prompt Tokens per Thought	Generate Tokens per Thought	Tokens per Case
IO	0.18k	0.56k	0.56k
CoT	0.23k	1.86k	1.86k
AoT	0.55k	1.74k	1.74k
ToT	0.48k	0.20k	11.60k
GoT	0.48k	0.21k	7.56k
L2T	0.49k	0.18k	4.68k

Table 6: Comparison of prompt tokens per thought, generate tokens per thought, and tokens per case for different methods.

and TruthQuest. Our method consistently outperforms others, showing significant improvements, particularly without task-specific prompts, where its efficiency advantage is more pronounced. Even without the GNN-based reasoning mode selection module (L2T w/o GNN), performance remains superior, highlighting the effectiveness of our approach.

Table 4 presents results on Creative Writing, focusing on relative scores to L2T. Evaluations via an LLM reduce fluctuations. L2T achieves higher or equivalent scores in over 80% of cases, with less than 20% lower, outperforming baselines. L2T w/o GNN performs comparably, supporting conclusions from prior results.

4.2 In-Depth Analysis

Ablation Study

To further delve into the analysis of our algorithm, we conducted ablation experiments. These experiments were performed on the Game of 24 task to evaluate the contribution of each component in our proposed method. We implemented three variations of the method with specific components ablated: (1) L2T w MLP, which replaces the GNN with an MLP; (2) L2T w/o RL, which removes the reinforcement learning mechanism for updating the GNN and instead directly trains the GNN-based reasoning mode selection module based on the scores of individual nodes; and (3) L2T w/o GNN, which completely eliminates the GNN-based reasoning mode selection module.

The experimental results are shown in Table 5. We not only evaluated the accuracy of each variant but also analyzed the number of nodes generated by each method. This provides an indication of the number of reasoning steps required to arrive at the final result. The results demonstrate that the GNN-based reasoning mode selection module does contribute to the performance of the L2T method. However, its primary benefit lies in reducing the number of reasoning steps needed. Clearly, methods incorporating the GNN-based reasoning mode selection module require significantly fewer reasoning steps compared to those without it.

Computational Consumption Analysis

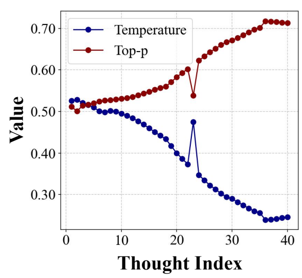
We also analyzed the computational consumption of L2T, using the number of tokens as a metric to measure computational cost. The experimental results are presented in Table 6. As shown, the computational resources consumed by L2T are comparable to those of other methods and outperform GoT. This demonstrates that L2T can accomplish complex reasoning tasks and achieve favorable results without requiring excessive computational resources. We also provide a detailed breakdown of the computational overhead in Table **??**.

Category	L2T	ToT	GoT
24 Points	**26**	48	30
3 × 3 Sudoku	**22**	32	28
TruthQuest	**14**	26	18
Creative Writing	21	32	**20**

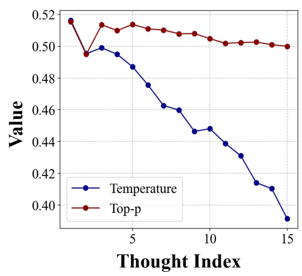
Table 7: Comparison of LLM access counts for different methods. **Bold** denotes the minimum value.

Process Analysis

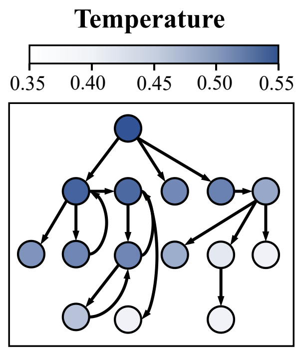
In order to conduct a more in-depth analysis of the working process of L2T, we recorded the temperature and top-p values output by the GNN-based reasoning mode selection module during its operation. The results are shown in Figure 4. An interesting observation is that temperature and top-p exhibit a significant correlation. For the Creative Writing task, the values display an inverse relationship—when one value is relatively high, the other tends to be relatively low. In contrast,



(a) Results of Creative Writing.



(b) Results of Game of 24.



(c) Visualization.

Figure 4: The temperature and top-p value within the reasoning process.

for the Game of 24 task, the values show a direct relationship—when one value is high, the other is also high. This indicates that the trained GNN-based reasoning mode selection module adopts distinct strategies tailored to different tasks. To further clarify this, we provide a concrete visualization of this strategy in Figure 4(c), offering a more explicit visualization of the parameter variations during the inference process is provided.

5 Conclusion

This paper proposes a novel LLM reasoning method, L2T. This method utilizes a graph-based framework to represent the reasoning process of LLMs and applies graph learning techniques to learn and analyze this reasoning graph, subsequently generating corresponding reasoning strategies. L2T incorporates two types of graph learning approaches: one based on LLMs and the other based on GNNs. It eliminates the need for specifically designed prompts for different problems and can integrate reinforcement learning methods to continuously self-optimize during successive problem-solving processes. Extensive experiments demonstrate the effectiveness of L2T.

Acknowledgments

We would like to express our sincere gratitude to the reviewers of this paper, as well as the Program Committee and Area Chairs, for their valuable comments and suggestions. This work is supported by the CAS Project for Young Scientists in Basic Research, Grant No. YSBR-040.

Contribution Statement

Hang Gao and Chenhao Zhang contributed equally to this work.

References

[Besta *et al.*, 2024] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press, 2024.

[Beurer-Kellner *et al.*, 2024] Luca Beurer-Kellner, Mark Niklas Müller, Marc Fischer, and Martin T. Vechev. Prompt sketching for large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[Brown *et al.*, 2022] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2022.

[Chu *et al.*, 2024] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. Navigate through enigmatic labyrinth A survey of chain of thought reasoning: Advances, frontiers and future. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 1173–1203. Association for Computational Linguistics, 2024.

[Dhuliawala *et al.*, 2024] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3563–3578. Association for Computational Linguistics, 2024.

[Diao *et al.*, 2024] Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. Active prompting with chain-of-thought for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 1330–1350. Association for Computational Linguistics, 2024.

[Hadi *et al.*, 2024] Muhammad Usman Hadi, Qasem Al Tashi, Abbas Shah, Rizwan Qureshi, Amgad Muneer, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, et al. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*, 2024.

[Jiao *et al.*, 2022] Wei Jiao, Yingce Xia, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Recent advances in neural machine translation. *AI Open*, 3:36–45, 2022.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[Konda and Tsitsiklis, 1999] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 1008–1014. The MIT Press, 1999.

[Lewis *et al.*, 2020] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[Li *et al.*, 2023] Cheng Li, Jindong Wang, Yixuan Zhang, Kaijie Zhu, Wenxin Hou, Jianxun Lian, Fang Luo, Qiang Yang, and Xing Xie. Large language models understand and can be enhanced by emotional stimuli, 2023.

[Liu *et al.*, 2023] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[Masson *et al.*, 2024] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. Directgpt: A direct manipulation interface to interact with large language models. In Florian 'Floyd' Mueller, Penny Kyburz, Julie R. Williamson, Corina Sas, Max L. Wilson, Phoebe O. Toups Dugas, and Irina Shklovski, editors, *Proceedings of the CHI Conference on Human Factors in Computing Systems, CHI 2024, Honolulu, HI, USA, May 11-16, 2024*, pages 975:1–975:16. ACM, 2024.

[Mondorf and Plank, 2024] Philipp Mondorf and Barbara Plank. Liar, liar, logical mire: A benchmark for suppositional reasoning in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 7114–7137. Association for Computational Linguistics, 2024.

[Ni *et al.*, 2022] Vincent Ni, Adrian Lee, Shruti Kumar, Saikrishna Chalamalasetti, Aditi Singh, Nadjet Tazi, Dhruva Patil, et al. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

[OpenAI, 2023] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[Paranjape *et al.*, 2023] Bhargavi Paranjape, Scott M. Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Túlio Ribeiro. ART: automatic multi-step reasoning and tool-use for large language models. *CoRR*, abs/2303.09014, 2023.

[Patterson *et al.*, 2022] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis Munguia, Daniel Rothchild, David So, Marc Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2204.05149*, 2022.

[Qiao *et al.*, 2023] Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. Reasoning with language model prompting: A survey. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5368–5393. Association for Computational Linguistics, 2023.

[Radford *et al.*, 2018] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 1(8):1–12, 2018.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[Sel *et al.*, 2024] Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[Sun *et al.*, 2022] Kaiyuan Sun, Cheng Zhou, Deng Cai, and Ming Ding. Black-box tuning for language-model-as-a-service. *arXiv preprint arXiv:2201.03514*, 2022.

[Wang *et al.*, 2023] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[Yao *et al.*, 2023a] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[Yao *et al.*, 2023b] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Yu *et al.*, 2023] Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *CoRR*, abs/2311.09210, 2023.

[Zhang *et al.*, 2023] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Zhao *et al.*, 2024] Xufeng Zhao, Mengdi Li, Wenhao Lu, Cornelius Weber, Jae Hee Lee, Kun Chu, and Stefan Wermter. Enhancing zero-shot chain-of-thought reasoning in large language models through logic. In Nicoletta Calzolari, Min-Yen Kan, Véronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/-COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 6144–6166. ELRA and ICCL, 2024.

[Zhou *et al.*, 2022] Kevin Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[Zhou *et al.*, 2023a] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Zhou *et al.*, 2023b] Yucheng Zhou, Xiubo Geng, Tao Shen, Chongyang Tao, Guodong Long, Jian-Guang Lou, and Jianbing Shen. Thread of thought unraveling chaotic contexts. *CoRR*, abs/2311.08734, 2023.