The Role of Deductive and Inductive Reasoning in Large Language Models

Chengkun Cai[1*] Xu Zhao[1*] Haoliang Liu[2*] Zhongyu Jiang[3]
Tianfang Zhang[4] Zongkai Wu[5] Jenq-Neng Hwang[3] Lei Li[3,6†]

[1]University of Edinburgh [2]University of Manchester [3]University of Washington
[4]Tsinghua University [5]Skai Intelligence [6]University of Copenhagen

Abstract

Large Language Models (LLMs) have demonstrated impressive capabilities in reasoning tasks, yet their reliance on static prompt structures and limited adaptability to complex scenarios remains a major challenge. In this paper, we propose the **D**eductive and **I**n**D**uctive(**DID**) method, a novel framework that enhances LLM reasoning by dynamically integrating both deductive and inductive reasoning approaches. Drawing from cognitive science principles, DID implements a dual-metric complexity evaluation system that combines Littlestone dimension and information entropy to precisely assess task difficulty and guide decomposition strategies. DID enables the model to progressively adapt its reasoning pathways based on problem complexity, mirroring human cognitive processes. We evaluate DID's effectiveness across multiple benchmarks, including the AIW, MR-GSM8K, and our custom Holiday Puzzle dataset for temporal reasoning. Our results demonstrate great improvements in reasoning quality and solution accuracy - achieving 70.3% accuracy on AIW (compared to 62.2% for Tree of Thought), while maintaining lower computational costs.

1 Introduction

Large Language Models (LLMs), such as GPT-4, have transformed natural language processing by excelling in tasks such as language translation, summarization, and question-answering (OpenAI, 2023), particularly in reasoning tasks and few-shot learning. However, their reliability in problem-solving remains debatable. While Zhou et al. (2024) notes that scaling and fine-tuning can introduce unpredictable errors even in simple tasks, recent methodologies like Chain of Thought (CoT) (Wei et al., 2022) have shown substantial improvements in arithmetic and symbolic reasoning tasks (Li et al., 2024). Studies have demonstrated that LLMs can achieve high accuracy in multi-step reasoning when guided by structured approaches like CoT and self-consistency (Bubeck et al., 2023; Wang et al., 2022). Additionally, techniques such as reinforcement learning from human feedback (RLHF) have proven effective in reducing harmful or inaccurate outputs (Ouyang et al., 2022; Christiano et al., 2017).

Despite these advances, LLMs face substantial challenges with complex and evolving tasks due to their reliance on static prompt structures and pre-learned patterns. This limitation manifests in tasks requiring logical reasoning, such as calculating family relationships or performing numerical comparisons (Nezhurina et al., 2024). Unlike human problem-solving, which dynamically adjusts strategies based on task complexity through inductive and deductive reasoning (Sloman, 2009), LLMs often struggle to adapt their reasoning processes to novel situations (Marcus, 2020; Hendrycks et al., 2020).

This adaptability gap becomes particularly evident in tasks requiring dynamic adjustment or incremental problem-solving. While existing approaches like CoT (Wei et al., 2022), Tree-of-Thought (ToT) (Yao et al., 2024), Temperature-Tree-of-Thought (T^2oT) (Cai et al., 2024), and Graph-of-Thought (GoT) (Besta et al., 2024) have made progress through extensive output exploration, they often incur considerable computational costs. For instance, ToT achieves 62.2% accuracy on the AIW benchmark but requires substantial output token generation for exploring multiple reasoning paths, resulting in higher computational overhead ($0.0038 per case compared to $0.0022 for CoT).

To address these challenges, we propose the De-In-Ductive (DID) method, a novel approach that enhances LLM reasoning by integrating both inductive and deductive reasoning processes within

[*] Equal contribution.

[†] Corresponding author: lilei@di.ku.dk

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16780–16790
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

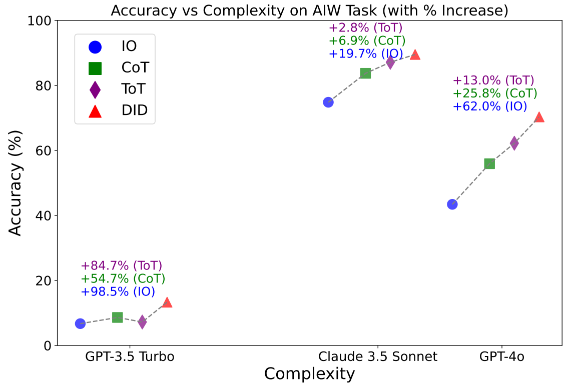


Figure 1: Performance comparison of different reasoning approaches (IO, CoT, ToT, and DID) across model complexity. The x-axis represents an estimated measure of complexity that considers both model size (following public estimates from Abacha et al. (2024)) and reasoning token cost. The y-axis shows accuracy on the AIW reasoning benchmark. The relative positioning of models on the complexity axis is based on their approximate parameter counts and the computational overhead required for inference.

the prompt construction framework. Unlike previous methods that focus on expanding output exploration, DID takes an input-centric approach inspired by Test-Time Training techniques, strategically investing in input structuring to enable more efficient reasoning. The DID framework incorporates two key innovations: problem complexity evaluation and dynamic reasoning adjustment. For problem complexity evaluation, we introduce a dual-metric system that considers both the Littlestone dimension (measuring structural problem complexity) and information entropy (quantifying instance problem complexity) of problems, enabling precise assessment of task difficulty and guiding decomposition strategy.

Grounded in cognitive science models of human reasoning, DID implements a hybrid approach that mirrors human cognitive strategies. The method operates in two phases: first, it employs inductive reasoning to derive general rules from specific instances, progressively increasing problem complexity while maintaining similar Littlestone dimension; then, it applies deductive reasoning to solve particular problems, where the dynamic reasoning adjustment mechanism leverages problem complexity assessment to adaptively control the reasoning chain length and decomposition granularity.

We validate DID's effectiveness on established benchmarks including AIW and MR-GSM8K (Nezhurina et al., 2024; Zeng et al., 2023), as well as our custom Holiday Puzzle dataset focusing on holiday date calculations. As shown in Figure 1, our empirical results demonstrate notable improvements in both solution accuracy and reasoning quality, achieving 70.3% accuracy on AIW (compared to 62.2% for ToT) while maintaining lower computational costs ($0.0031 vs $0.0038 per case). This work makes the following key contributions:

- We propose an innovative input-centric approach to LLM reasoning through the De-InDuctive (DID) framework, which differs from existing output-exploration methods by strategically investing in input structuring. This approach fundamentally changes how we enhance LLM reasoning capabilities, offering a more efficient alternative to traditional methods.

- We develop a theoretically grounded complexity evaluation system that combines Littlestone dimension and information entropy, enabling precise assessment of task difficulty and guiding the dynamic integration of inductive and deductive reasoning processes.

- Through extensive empirical evaluations across diverse reasoning tasks, we demonstrate that DID not only achieves superior accuracy but also maintains lower computational costs through efficient input utilization, establishing a new direction for efficient LLM reasoning enhancement.

2 Related Works

Cognitive Science and Deductive-Inductive Reasoning Deductive and inductive reasoning are essential in cognitive science, with deductive reasoning applying general principles to specific cases, and inductive reasoning generalizing from observations. Cognitive models view these approaches as complementary: inductive reasoning generates hypotheses, while deductive reasoning tests them (Wason, 1960). This combination enhances problem-solving, especially in uncertain domains where balancing exploration and validation is key (Johnson-Laird, 1983; Tversky and Kahneman, 1974). Well-structured problems typically favor deductive reasoning, whereas ill-structured problems benefit from inductive reasoning (Funke, 2013). Cognitive science insights have been integrated into neural networks, improving generalization (L Griffiths et al., 2008; Tenenbaum et al., 2011).

LLMs for Reasoning and Prompting Techniques While LLMs like GPT-4 excel at tasks such as text generation, they struggle with logical reasoning and complex deduction (OpenAI, 2023; Nezhurina et al., 2024). Techniques like CoT (Wei et al., 2022), ToT (Yao et al., 2024), and GoT (Besta et al., 2024) improve reasoning by structuring problems, but they require extensive prompt engineering and lack real-time adaptability. The DID framework addresses these limitations by dynamically integrating inductive and deductive reasoning, improving adaptability and consistency in complex tasks (Marcus, 2020; Gershman et al., 2015).

Recent comprehensive surveys by Giadikiaroglou et al. (2024) and Liu et al. (2023) provide thorough analyses of LLM reasoning capabilities in puzzle-solving and mathematical domains, highlighting both progress and persistent challenges in structured reasoning tasks. Recent advances in LLM reasoning capabilities have explored different approaches to enhancing model performance at test time. Models like DeepSeek-R1 (Guo et al., 2025) and o1-ioi (El-Kishky et al., 2025) achieve impressive results by leveraging reinforcement learning during pre-training and extending reasoning paths during inference - with DeepSeek-R1 reaching 79.8% accuracy on AIME 2024 through naturally emerged reasoning behaviors, and o1-ioi employing sophisticated test-time compute strategies to evaluate multiple solution candidates.

Recent advancements in improving LLM reasoning have explored diverse strategies. While multi-agent frameworks like Jin et al. (2025) demonstrate superior performance through collaborative exploration of reasoning paths, they incur increased computational overhead due to multiple model calls. Other approaches, such as "learning from teaching regularization" (Jin et al., 2024) and Self-Explore (Hwang et al., 2024), enhance reasoning by incorporating structured examples or fine-grained rewards during training. However, these methods necessitate model fine-tuning or multiple model instances, whereas our DID framework distinguishes itself by improving reasoning through structured prompting of a single model instance without additional training.

Inductive Inference and Online Learning Recent work links inductive inference to online learning theory. Lu (2024) demonstrate that inductive inference is possible for hypothesis classes decomposable into countable unions with finite Littlestone dimension. This result extends classical induction models, such as Solomonoff's (Solomonoff, 1964). The connection between Littlestone dimension and learning complexity informs the DID framework, suggesting that decomposing tasks into simpler components can enhance learning and generalization.

3 Methodology

3.1 Problem Formalization and Complexity Evaluation

Most reasoning tasks encountered by LLMs can be characterized as sequential learning problems with finite Littlestone dimension. According to recent theoretical work, a hypothesis class is learnable through inductive inference if and only if it can be decomposed into a countable union of classes with finite Littlestone dimension.

3.1.1 Littlestone Dimension and Beyond

For traditional online learning problems, the Littlestone dimension d alone sufficiently characterizes problem difficulty. This dimension quantifies the intrinsic sequential learning complexity by measuring:

- The minimal depth of decision trees needed for solving the problem
- The number of key decision points in the reasoning process

However, when dealing with Large Language Models (LLMs), we observe that problems with identical Littlestone dimensions can exhibit significantly different difficulty levels. For example:

Example 1 *Alice has 0 brothers and 1 sister. How many sisters does Alice's brother have?*

Example 2 *Alice has 3 brothers and 6 sisters. How many sisters does Alice's brother have?*

Both problems share the same Littlestone dimension, as they follow identical reasoning patterns. However, LLMs consistently perform better on Example 1. This discrepancy arises from several theoretical considerations:

1. **Feature Vector Differences:** In LLM's internal representations, simpler numerical relationships create clearer, more distinguishable feature vectors

2. **Distribution Shift:** Larger numbers and more complex relationships often represent a shift from the training distribution

3. **Information Bottleneck Theory:** With increasing problem scale, the extraction of relevant information becomes more challenging due to the constrained capacity of intermediate representations

3.1.2 Information Entropy Component

To account for these LLM-specific challenges, we introduce an information entropy component H that quantifies:

- The complexity of numerical relationships
- The density of relevant information that needs to be extracted
- The scale of variables involved in the problem

For a problem instance p with n variables $\{x_1, ..., x_n\}$, we define its entropy as:

$$H(p) = \log_2 \left(\prod_{i=1}^{n} (1 + |x_i|) \right) \tag{1}$$

where $|x_i|$ represents the absolute value of each numerical variable in the problem. Importantly, only variables that are directly relevant to solving the problem are included in this calculation. For instance, in the problem "Alice has 3 brothers and 6 sisters. How many sisters does Alice's brother have?", only the number of brothers (3) and sisters (6) would be considered in the entropy calculation. This formulation:

- Grows logarithmically with problem scale
- Remains bounded for reasonable problem sizes
- Captures the intuition that larger numbers and more variables increase processing difficulty

3.1.3 Problem Complexity Evaluation

The overall complexity of a problem p is then defined as:

$$C(p) = d \cdot H(p) \tag{2}$$

This combined measure allows us to:

1. Distinguish between problems of equal Littlestone dimension but different scale complexity
2. Better predict LLM performance on reasoning tasks
3. Guide the decomposition of complex problems into manageable subproblems

Algorithm 1 Problem Decomposition in DID

```
Require:
  Problem $p$ with Littlestone dimension $d$
  Problem complexity $C(p) = d \cdot H$
  Step size parameter $a \in [1, C(p)]$ controlling decomposition granularity
Ensure:
  Sequence of subproblems with increasing complexity
function DecomposeProblem($p, d$)
    $N \leftarrow \left\lceil \frac{C(p)}{a} \right\rceil$
    Initialize subproblems $\leftarrow \emptyset$
    $p_{\text{base}} \leftarrow$ CreateBaseCase$(p, d-2)$
    subproblems.append$(p_{\text{base}})$
    for $i \leftarrow 1$ to $N$ do
        if $i < N/2$ then
            $d_{\text{current}} \leftarrow d - 1$
        else
            $d_{\text{current}} \leftarrow d$
        end if
        $p_{\text{next}} \leftarrow$ IncreaseCplx$(p_{\text{base}}, d_{\text{current}})$
        subproblems.append$(p_{\text{next}})$
        $p_{\text{base}} \leftarrow p_{\text{next}}$
    end for
    return subproblems
end function
```

Dynamic Reasoning Based on Problem Complexity The Algorithm 1 formalizes our DID framework's problem decomposition approach. At its core, the algorithm dynamically decomposes complex problems into a sequence of progressively challenging subproblems while managing both structural complexity (Littlestone dimension) and information density.

The decomposition process starts by creating a base case with reduced dimension $(d-2)$, achieved by setting certain variables to zero. Specifically, this means eliminating key decision points in the reasoning chain by simplifying the problem structure, for example, changing "Alice has 3 brothers and 6 sisters" to "Alice has 0 sisters and 1 brother" to reduce the problem's Littlestone dimension. This simplification maintains the essential reasoning structure while reducing the problem's complexity to its most basic form. From this foundation,

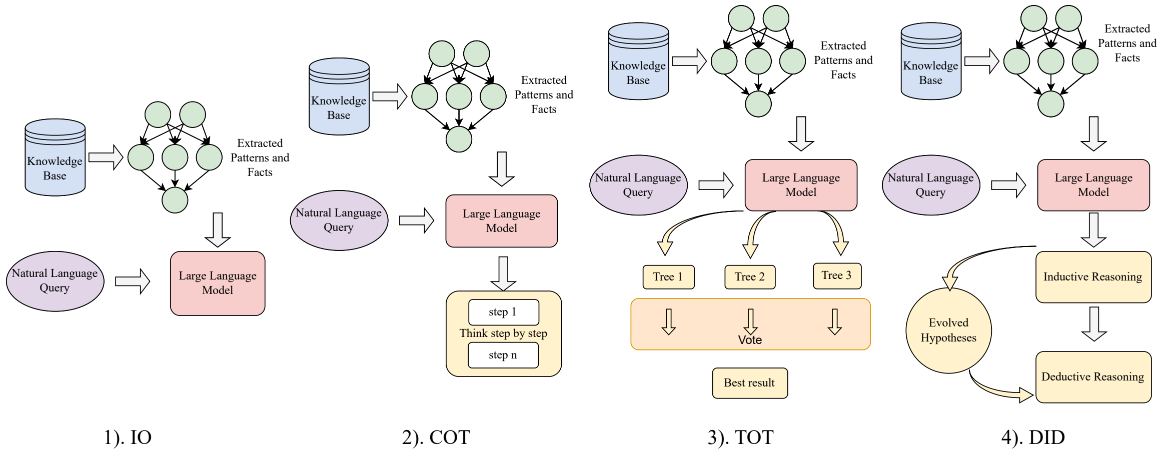


Figure 2: Comparison of reasoning approaches in LLMs including the IO method, Chain of Thought (CoT) prompting, Tree of Thought (ToT) prompting, and the De-In-Ductive (DID) framework, highlighting the progression from direct output generation to dynamic inductive and deductive reasoning for more adaptive problem-solving.

the algorithm iteratively constructs $N = \lceil C(p) \rceil$ subproblems of increasing complexity, where $C(p)$ represents the overall problem complexity.

The algorithm employs a two-phase strategy in complexity progression:

- In the first phase ($i < N/2$), it maintains a reduced dimension ($d - 1$), allowing the model to establish fundamental patterns and relationships with minimal complexity

- In the second phase ($i \geq N/2$), it restores the full dimension (d), gradually introducing complete problem complexity while building upon previously established patterns

This progressive approach mirrors human cognitive processes in problem-solving: starting with simplified versions, identifying core patterns, and systematically applying these insights to more complex cases. The `IncreaseCplx` function implements this gradual progression by introducing additional variables and relationships while maintaining the problem's fundamental structure.

The algorithm's dynamic dimension management ensures that the model can effectively balance between pattern recognition (inductive reasoning) in simpler cases and rigorous application (deductive reasoning) in more complex scenarios. This balance is crucial for maintaining both learning efficiency and solution accuracy across varying problem complexities.

3.2 De-In-Ductive (DID) Framework

Figure 2 illustrates the comparison between the IO, CoT, and DID frameworks. The IO (Input-Output) Method processes natural language queries by retrieving patterns and facts without engaging in iterative reasoning. The Chain of Thought (CoT) Method improves logical reasoning by breaking down complex problems into sequential steps. Our proposed De-In-Ductive (DID) Method goes further by dynamically integrating inductive and deductive reasoning. By iteratively generating and testing hypotheses, DID adapts to problem complexities more effectively than static methods like CoT, optimizing problem-solving by balancing reasoning modes in response to task difficulty.

Dynamic Reasoning Based on Problem Complexity Based on the problem complexity C(p), DID framework adaptively decomposes the problem and adjusts its reasoning process. For a typical problem with Littlestone dimension d (usually 3-5), we decompose it into subproblems:

- Dimension Reduction: We maintain subproblems with dimension d or reduce to d-1 by fixing certain variables to 0, preserving the essential reasoning structure while reducing complexity

- Progressive Complexity: Starting from simple cases with minimal information density, we gradually increase complexity by adding variables and relationships

- Hierarchical Solution: Each subproblem (K)

is solved using insights from solutions to the previous subproblem ($K-1$), enabling progressive knowledge accumulation

Inductive Reasoning The inductive component enables pattern discovery and generalization from specific instances:

- **Pattern Recognition**: Starting with simplified problem instances (reduced Littlestone dimension $d-2$ or $d-1$), the model identifies fundamental patterns and relationships. This aligns with the theoretical basis that inductive inference is possible when hypothesis classes have a finite Littlestone dimension.

- **Hypothesis Generation**: Through progressive exposure to increasingly complex examples, the model generates and refines hypotheses about the underlying structure of the problem. Each subproblem serves as a training instance for pattern recognition.

- **Complexity-Guided Learning**: The inductive process is guided by the complexity measure $C(p) = d \cdot H$, ensuring that pattern recognition proceeds from simpler to more complex cases while maintaining manageable Littlestone dimensions.

Deductive Reasoning The deductive component enables the systematic application of discovered patterns:

- **Rule Application**: Once patterns are identified through induction, the model applies these rules deductively to solve more complex instances. This leverages the theoretical guarantee that hypothesis classes with finite Littlestone dimensions are learnable.

- **Verification Process**: Each deductive step serves as a verification mechanism for inductively derived patterns, helping to refine and validate the model's understanding.

- **Hierarchical Problem Solving**: The deductive process follows the complexity hierarchy established during induction, ensuring that solutions are built systematically on previously verified patterns.

The results from deductive applications inform and refine the inductive pattern recognition process, creating a continuous learning cycle that enhances the model's problem-solving capabilities. A complete step-by-step example of the DID framework is provided in Appendix B.

Integration with Existing Models The DID method seamlessly integrates with various LLM architectures and existing techniques such as CoT prompting (Wei et al., 2022). Through its structured framework combining inductive and deductive reasoning, DID enhances these methods by providing dynamic reasoning strategies and guided incremental reasoning, while maintaining computational efficiency. This approach creates a more flexible framework for LLMs to address complex problems without notable overhead.

4 Experiments

4.1 Experimental Setup

Baseline Methods and Models We compare DID against three baseline prompting methods:

- Input-Output (IO): directly utilizes the LLM without structured prompting

- Chain of Thought (CoT): breaks down problems into sequential reasoning steps

- Tree of Thought (ToT): explores multiple reasoning paths in a tree structure (T=3)

All methods are evaluated using three representative models:

- GPT-4o and Claude 3.5 Sonnet: selected as two leading LLMs from different providers to demonstrate robustness across model architectures

- GPT-3.5-turbo: included to evaluate method robustness across different model scales (in terms of parameter count)

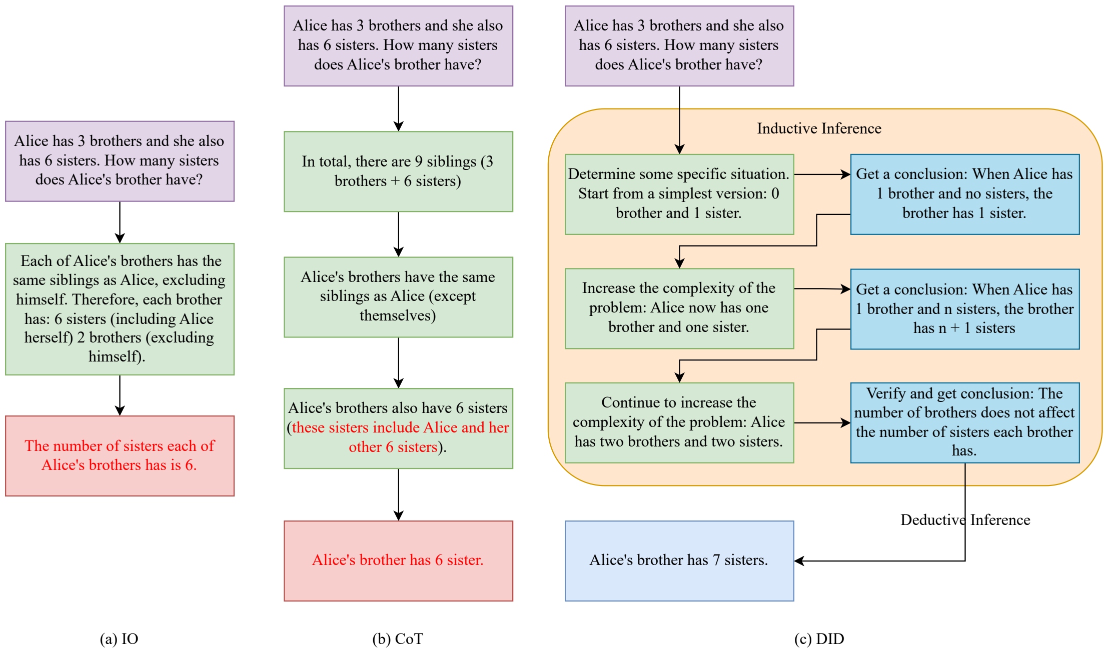
For fair comparison, all model parameters (temperature, top-k sampling, etc.) are maintained at their default values. Evaluations are conducted in a zero-shot setting across all methods and models.

4.2 Tasks and Results

Alice Problems The AIW dataset focuses on evaluating logical reasoning and deduction abilities through family relationship problems (Nezhurina et al., 2024). We manually curated 113 unique problems after removing duplicates and existing prompts, with results averaging over 20 runs. In this task, DID demonstrates consistent superiority across all models:

Model\Method	Alice Problem				MR-GSM8K			Holiday Puzzle			
	IO (%)	CoT (%)	ToT (%)	DID (%)	CoT (%)	ToT (%)	DID (%)	IO (%)	CoT (%)	ToT (%)	DID (%)
GPT-3.5 Turbo	6.7	8.6	7.2	**13.3**	68.1	**74.0**	73.3	0.2	1.4	2.0	**5.6**
GPT-4o	43.4	55.9	62.2	**70.3**	85.0	**89.1**	87.7	7.8	5.2	7.5	**15.4**
Claude 3.5 Sonnet	74.8	83.7	87.1	**89.5**	91.3	**92.0**	**92.0**	17.4	17.8	24.0	**24.5**

Table 1: Merged Results for GPT-3.5 Turbo, GPT-4o, and Claude 3.5 Sonnet across Different Tasks (Alice Problem, MR-GSM8K, Holiday Puzzle)



Alice has 3 brothers and she also has 6 sisters. How many sisters does Alice's brother have?
Each of Alice's brothers has the same siblings as Alice, excluding himself. Therefore, each brother has: 6 sisters (including Alice herself) 2 brothers (excluding himself).
The number of sisters each of Alice's brothers has is 6.
Alice has 3 brothers and she also has 6 sisters. How many sisters does Alice's brother have?
In total, there are 9 siblings (3 brothers + 6 sisters)
Alice's brothers have the same siblings as Alice (except themselves)
Alice's brothers also have 6 sisters (these sisters include Alice and her other 6 sisters).
Alice's brother has 6 sister.
Alice has 3 brothers and she also has 6 sisters. How many sisters does Alice's brother have?
Inductive Inference
Determine some specific situation. Start from a simplest version: 0 brother and 1 sister.
Get a conclusion: When Alice has 1 brother and no sisters, the brother has 1 sister.
Increase the complexity of the problem: Alice now has one brother and one sister.
Get a conclusion: When Alice has 1 brother and n sisters, the brother has n + 1 sisters
Continue to increase the complexity of the problem: Alice has two brothers and two sisters.
Verify and get conclusion: The number of brothers does not affect the number of sisters each brother has.
Deductive Inference
Alice's brother has 7 sisters.

(a) IO

(b) CoT

(c) DID

Figure 3: Comparison of reasoning approaches in LLMs including the IO method, CoT prompting, and the DID framework, highlighting the progression from direct output generation to dynamic inductive and deductive reasoning for more adaptive problem-solving.

Task	Method	Input/Output tokens	Cost per case	Accuracy (%)
Alice Problem	IO	37/55	$0.0007	43.4
	CoT	45/210	$0.0022	55.9
	ToT	56/370	$0.0038	62.2
	DID	90/290	$0.0031	70.3
MR-GSM8K	CoT	86/1017	$0.0104	85.0
	ToT	91/1920	$0.0194	89.1
	DID	190/1230	$0.0128	87.7
Holiday Puzzle	IO	87/570	$0.0059	7.8
	CoT	96/1330	$0.0135	5.2
	ToT	110/2590	$0.0262	7.5
	DID	260/1740	$0.0181	15.4

Table 2: GPT-4o Token Usage and Cost Comparison

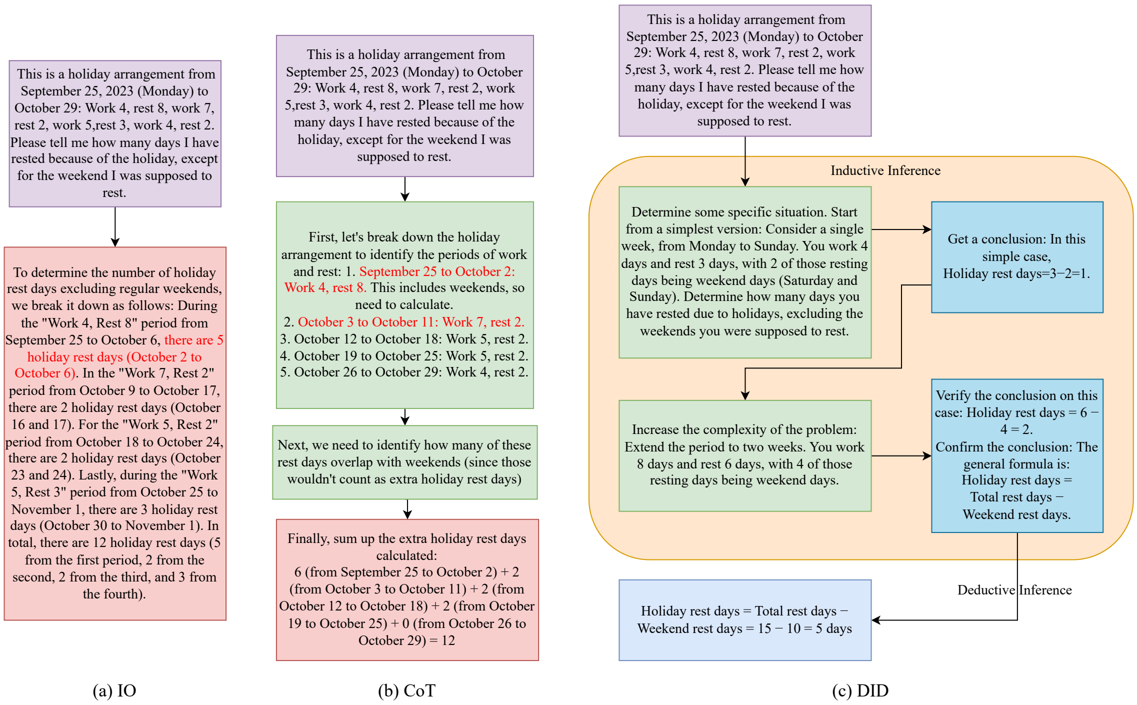
- GPT-3.5 Turbo: DID (13.3%) significantly outperforms IO (6.7%), CoT (8.6%), and ToT (7.2%)

- GPT-4o: DID achieves 70.3% accuracy, surpassing IO (43.4%), CoT (55.9%), and ToT (62.2%)

- Claude 3.5 Sonnet: DID reaches 89.5%, extending the lead over IO (74.8%), CoT (83.7%), and ToT (87.1%)

As illustrated in Figure 3, DID progressively guides LLMs through increasingly complex reasoning steps for family relationship problems. While traditional methods often fail by attempting to solve complex problems directly, DID breaks down the reasoning process into simpler subproblems, helping the model maintain logical consistency and avoid common errors. This structured approach enables LLMs to effectively handle complex relationship inference tasks.

As shown in Table 2, while DID requires slightly more input tokens (90 vs 56 for ToT), it maintains lower total computational costs ($0.0031 vs $0.0038) through more efficient output generation. This demonstrates the effectiveness of our input-centric approach in balancing performance and efficiency.



This is a holiday arrangement from September 25, 2023 (Monday) to October 29: Work 4, rest 8, work 7, rest 2, work 5,rest 3, work 4, rest 2. Please tell me how many days I have rested because of the holiday, except for the weekend I was supposed to rest.

To determine the number of holiday rest days excluding regular weekends, we break it down as follows: During the "Work 4, Rest 8" period from September 25 to October 6, there are 5 holiday rest days (October 2 to October 6). In the "Work 7, Rest 2" period from October 9 to October 17, there are 2 holiday rest days (October 16 and 17). For the "Work 5, Rest 2" period from October 18 to October 24, there are 2 holiday rest days (October 23 and 24). Lastly, during the "Work 5, Rest 3" period from October 25 to November 1, there are 3 holiday rest days (October 30 to November 1). In total, there are 12 holiday rest days (5 from the first period, 2 from the second, 2 from the third, and 3 from the fourth).

(a) IO

This is a holiday arrangement from September 25, 2023 (Monday) to October 29: Work 4, rest 8, work 7, rest 2, work 5,rest 3, work 4, rest 2. Please tell me how many days I have rested because of the holiday, except for the weekend I was supposed to rest.

First, let's break down the holiday arrangement to identify the periods of work and rest: 1. September 25 to October 2: Work 4, rest 8. This includes weekends, so need to calculate.
2. October 3 to October 11: Work 7, rest 2.
3. October 12 to October 18: Work 5, rest 2.
4. October 19 to October 25: Work 5, rest 2.
5. October 26 to October 29: Work 4, rest 2.

Next, we need to identify how many of these rest days overlap with weekends (since those wouldn't count as extra holiday rest days)

Finally, sum up the extra holiday rest days calculated:
6 (from September 25 to October 2) + 2 (from October 3 to October 11) + 2 (from October 12 to October 18) + 2 (from October 19 to October 25) + 0 (from October 26 to October 29) = 12

(b) CoT

This is a holiday arrangement from September 25, 2023 (Monday) to October 29: Work 4, rest 8, work 7, rest 2, work 5,rest 3, work 4, rest 2. Please tell me how many days I have rested because of the holiday, except for the weekend I was supposed to rest.

Determine some specific situation. Start from a simplest version: Consider a single week, from Monday to Sunday. You work 4 days and rest 3 days, with 2 of those resting days being weekend days (Saturday and Sunday). Determine how many days you have rested due to holidays, excluding the weekends you were supposed to rest.

Get a conclusion: In this simple case,
Holiday rest days=3−2=1.

Increase the complexity of the problem: Extend the period to two weeks. You work 8 days and rest 6 days, with 4 of those resting days being weekend days.

Verify the conclusion on this case: Holiday rest days = 6 − 4 = 2.
Confirm the conclusion: The general formula is:
Holiday rest days = Total rest days − Weekend rest days.

Holiday rest days = Total rest days − Weekend rest days = 15 − 10 = 5 days

(c) DID

Figure 4: Comparison of reasoning approaches in LLMs including the IO method, CoT prompting, and the DID framework, highlighting the progression from direct output generation to dynamic inductive and deductive reasoning for more adaptive problem-solving.

MR-GSM8K Math Problems MR-GSM8K extends the GSM8K benchmark with meta-reasoning tasks (Zeng et al., 2023), requiring models to identify and explain errors in provided solutions. Results show consistent performance across models:

- GPT-3.5 Turbo: DID (73.3%) maintains competitive performance against CoT (68.1%) and ToT (74.0%)
- GPT-4o: DID (87.7%) performs comparably to CoT (85.0%) and ToT (89.1%)
- Claude 3.5 Sonnet: DID (92.0%) matches the strong performance of CoT (91.3%) and ToT (92.0%)

As shown in Table 2, DID achieves this performance with lower computational overhead than ToT ($0.0128 vs $0.0194), despite using more input tokens (190 vs 91). This efficiency gain comes from reduced output exploration needs.

Holiday Puzzle This custom dataset comprises 20 holiday arrangement problems, testing models' ability to calculate actual holiday days while accounting for weekends and compensatory workdays. Detailed information about the dataset construction and representative examples are provided in Appendix A. Results demonstrate:

- GPT-3.5 Turbo: DID (5.6%) outperforms IO (0.2%), CoT (1.4%), and ToT (2.0%)
- GPT-4o: DID shows marked improvement (15.4%) over IO (7.8%), CoT (5.2%), and ToT (7.5%)
- Claude 3.5 Sonnet: DID (24.5%) maintains advantage over IO (17.4%), CoT (17.8%), and ToT (24.0%)

The key to success in this task lies in discovering and applying the fundamental relationship *Holiday rest days = Total rest days - Weekend rest days*. As shown in Figure 4, baseline methods struggle with this pattern.

As shown in Table 2, while DID uses more input tokens (260 vs 110 for ToT), its efficient output generation results in lower total costs ($0.0181 vs $0.0262), demonstrating the scalability of our input-centric approach even in complex temporal reasoning tasks.

5 Conclusion

In this work, we introduced the De-In-Ductive (DID) method, a novel framework that dynamically integrates inductive and deductive reasoning to enhance the adaptability and reasoning capabilities of LLMs. By leveraging cognitive science principles, the DID framework allows LLMs to evolve their problem-solving strategies in response to task complexity, overcoming the rigidity of static prompt structures. Through extensive empirical validation on both standard benchmarks and our custom Holiday Puzzle dataset, we demonstrated substantial improvements in accuracy and reasoning quality, achieved without excessive computational costs. The success of DID in improving LLM reasoning while maintaining computational efficiency suggests promising directions for future research in making language models more cognitively aligned and capable of sophisticated reasoning.

6 Limitations

Despite the advances demonstrated by the DID framework, several important limitations and challenges remain to be addressed:

Fundamental Architecture Constraints A key limitation lies in the fundamental architecture of LLMs. These models, based on next-token prediction, struggle to maintain coherent internal representations across multiple reasoning steps. While attention mechanisms allow reference to previous tokens, they lack robust cognitive structures for ensuring logical integrity throughout the reasoning process. This often leads to unexpected errors even in seemingly straightforward tasks.

Generalization Challenges While DID shows strong performance on our evaluated tasks, ensuring consistent generalization to completely unseen problems remains challenging. The framework's effectiveness may vary depending on the nature and complexity of new tasks, particularly those requiring novel forms of reasoning not encountered during development.

7 Acknowledgements

This work was supported in part by the Pioneer Centre for AI, DNRF grant number P1.

References

Asma Ben Abacha, Wen-wai Yim, Yujuan Fu, Zhaoyi Sun, Meliha Yetisgen, Fei Xia, and Thomas Lin. 2024. Medec: A benchmark for medical error detection and correction in clinical notes. *arXiv preprint arXiv:2412.19260*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Chengkun Cai, Xu Zhao, Yucheng Du, Haoliang Liu, and Lei Li. 2024. T^2 of thoughts: Temperature tree elicits reasoning in large language models. *arXiv preprint arXiv:2405.14075*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, et al. 2025. Competitive programming with large reasoning models. *arXiv preprint arXiv:2502.06807*.

Joachim Funke. 2013. Complex problem solving: A case for complex cognition? In *Complex problem solving: Principles and mechanisms*, pages 25–47. Psychology Press.

Samuel J Gershman, Eric J Horvitz, and Joshua B Tenenbaum. 2015. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278.

Panagiotis Giadikiaroglou, Maria Lymperaiou, Giorgos Filandrianos, and Giorgos Stamou. 2024. Puzzle solving using reasoning of large language models: A survey. *arXiv preprint arXiv:2402.11291*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Hyeonbin Hwang, Doyoung Kim, Seungone Kim, Seonghyeon Ye, and Minjoon Seo. 2024. Self-explore: Enhancing mathematical reasoning in language models with fine-grained rewards. *arXiv preprint arXiv:2404.10346*.

Can Jin, Tong Che, Hongwu Peng, Yiyuan Li, Dimitris Metaxas, and Marco Pavone. 2024. Learning from teaching regularization: Generalizable correlations should be easy to imitate. In *Advances in Neural Information Processing Systems*, volume 37, pages 966–994. Curran Associates, Inc.

Can Jin, Hongwu Peng, Qixin Zhang, Yujin Tang, Dimitris N Metaxas, and Tong Che. 2025. Two heads are better than one: Test-time scaling of multi-agent collaborative reasoning. *arXiv preprint arXiv:2504.09772*.

Philip Nicholas Johnson-Laird. 1983. *Mental models: Towards a cognitive science of language, inference, and consciousness*. 6. Harvard University Press.

Thomas L Griffiths, Charles Kemp, and Joshua B Tenenbaum. 2008. Bayesian models of cognition.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*.

Wentao Liu, Hanglei Hu, Jie Zhou, Yuyang Ding, Junsong Li, Jiayi Zeng, Mengliang He, Qin Chen, Bo Jiang, Aimin Zhou, et al. 2023. Mathematical language models: A survey. *arXiv preprint arXiv:2312.07622*.

Zhou Lu. 2024. When is inductive inference possible? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Gary Marcus. 2020. The next decade in ai: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.

Marianna Nezhurina, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. 2024. Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models. *arXiv preprint arXiv:2406.02061*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

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.

S Sloman. 2009. *Causal models: How people think about the world and its alternatives*. Oxford University Press.

Ray J Solomonoff. 1964. A formal theory of inductive inference. part i. *Information and control*, 7(1):1–22.

Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. 2011. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285.

Amos Tversky and Daniel Kahneman. 1974. Judgment under uncertainty: Heuristics and biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *science*, 185(4157):1124–1131.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Peter C. Wason. 1960. On the failure to eliminate hypotheses in a conceptual task. *Quarterly Journal of Experimental Psychology*, 12(3):129–140.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Zhongshen Zeng, Pengguang Chen, Shu Liu, Haiyun Jiang, and Jiaya Jia. 2023. Mr-gsm8k: A meta-reasoning benchmark for large language model evaluation. *arXiv preprint arXiv:2312.17080*.

Lexin Zhou, Wout Schellaert, Fernando Martínez-Plumed, Yael Moros-Daval, César Ferri, and José Hernández-Orallo. 2024. Larger and more instructable language models become less reliable. *Nature*, pages 1–8.

A Holiday Puzzle Dataset Details

The Holiday Puzzle dataset was created based on holiday arrangements in China over the past 10 years, specifically focusing on how special holidays (National Day, Spring Festival, Labor Day, Mid-Autumn Festival, etc.) are rescheduled. In China, the government employs a unique "work day adjustment" system to create longer consecutive holiday periods by rearranging working days and weekends. This practice often involves designating certain weekends as working days while extending official holidays, creating complex patterns where regular weekends are shifted, and compensatory workdays are inserted before or after holidays. This arrangement, while allowing for longer holiday periods, makes it challenging to calculate the actual number of holiday days versus regular weekend days.

A.1 Representative Examples

Prompt: This is a holiday arrangement from April 23, 2022 (Saturday) to May 15: rest 1, work 6, rest 5, work 3, rest 1, work 5, rest 2. Please tell me how many days I have rested because of the holiday, except for the weekend I was supposed to rest.

Right Answer: 1

Prompt: This is a holiday arrangement from January 1, 2022 (Saturday) to February 8: rest 3, work 4, rest 2, work 5, rest 2, work 5, rest 2, work 7, rest 7, rest 2. Please tell me how many days I have rested because of the holiday, except for the weekend I was supposed to rest.

Right Answer: 4

B Detailed DID Framework Example

We provide a complete example of the DID framework in action:

Problem: "Alice has 3 brothers and 6 sisters. How many sisters does Alice's brother have?"

Step 1: Complexity Evaluation The LLM first analyzes the problem structure and identifies:

- Littlestone dimension (d): 3 (requiring three key inferential steps)
- Information entropy: $H(p) = \log_2((1+3)(1+6)) = \log_2(28) \approx 4.8$
- Overall complexity: $C(p) = d \cdot H(p) = 3 \cdot 4.8 \approx 14.4$

Step 2: Problem Decomposition Following Algorithm 1, the problem is decomposed into a sequence of progressively complex subproblems:

Inductive Reasoning Phase:

- *Example 1 (d-1 Dimension):* "Let's start with the simplest version of the problem: Alice has 0 sisters and 1 brother. In this case, Alice's brother has 1 sister (Alice)."
- *Example 2 (d-1 Dimension):* "Now, let's consider a slightly more complex scenario: Alice has 1 sister and 1 brother. In this case, Alice's brother has 2 sisters (Alice and her sister)."
- *Example 3 (d Dimension):* "Let's examine one more scenario: Alice has 2 sisters and 1 brother. In this case, Alice's brother has 3 sisters (Alice and her two sisters)."
- *Pattern Recognition:* "From these examples, we can deduce a general rule: The number of sisters Alice's brother has = X + 1, where X = the number of sisters Alice has. This rule holds true regardless of the number of brothers Alice has (Y), because we're only concerned about the number of sisters from the brother's perspective."

Step 3: Deductive Reasoning "Now I'll apply this formula to our original problem: Alice has 6 sisters (X = 6). Therefore, Alice's brother has X + 1 = 6 + 1 = 7 sisters."

Step 4: Solution Verification "To verify: Alice's brother has the same siblings as Alice, except himself.

- Alice has 3 brothers and 6 sisters
- From her brother's perspective, he has 2 brothers (the other brothers, excluding himself) and 7 sisters (the 6 original sisters plus Alice)

Therefore, Alice's brother has 7 sisters."

Final Answer: 7