Rethinking the Role of Prompting Strategies in LLM Test-Time Scaling: A Perspective of Probability Theory

Yexiang Liu[1,2], **Zekun Li**[3], **Zhi Fang**[1,2], **Nan Xu**[1,4], **Ran He**[1,2*], **Tieniu Tan**[1,2,5]
[1]MAIS, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]University of California, Santa Barbara
[4]Beijing Wenge Technology Co., Ltd [5]Nanjing University
{liuyexiang2023, fangzhi2023, xunan2015, ran.he, tieniu.tan}@ia.ac.cn
zekunli@cs.ucsb.edu

Abstract

Recently, scaling test-time compute on Large Language Models (LLM) has garnered wide attention. However, there has been limited investigation of how various reasoning prompting strategies perform as scaling. In this paper, we focus on a standard and realistic scaling setting: majority voting. We systematically conduct experiments on 6 LLMs $\times$ 8 prompting strategies $\times$ 6 benchmarks. Experiment results consistently show that as the sampling time and computational overhead increase, complicated prompting strategies with superior initial performance gradually fall behind simple Chain-of-Thought. We analyze this phenomenon and provide theoretical proofs. Additionally, we propose a method according to probability theory to quickly and accurately predict the scaling performance and select the best strategy under large sampling times without extra resource-intensive inference in practice. It can serve as the test-time scaling law for majority voting. Furthermore, we introduce two ways derived from our theoretical analysis to significantly improve the scaling performance. We hope that our research can promote to re-examine the role of complicated prompting, unleash the potential of simple prompting strategies, and provide new insights for enhancing test-time scaling performance.

1 Introduction

Over the past few years, how to enhance the reasoning abilities of large language models (LLMs) has been a topic of widespread interest (Dubey et al., 2024; Anil et al., 2023; Touvron et al., 2023; Open AI, 2024a; Team et al., 2024). Researchers have introduced various prompting strategies to improve the reasoning capacity of LLMs, such as Chain of Thought (CoT) (Wei et al., 2022) and so on (Zheng et al., 2024; Yasunaga et al., 2024; Madaan et al., 2023). Recently, many studies have shown that scaling LLM test-time compute can also effectively improve reasoning (Snell et al., 2025; Open AI, 2024b; Ji et al., 2025; Bi et al., 2024).

However, how different prompting strategies behave when scaling test-time compute is less explored. In this paper, we focus on a standard and effective scaling setting: majority voting.We comprehensively evaluate the performance of 8 mainstream prompting strategies under equivalent sampling time or computation overhead. We test 4 open-sourced and 2 closed-sourced LLMs on 6 reasoning benchmarks, finding that simple CoT consistently performs best on all LLMs across benchmarks with given budgets as scaling increases, even if it falls behind at the beginning. This indicates that current LLMs can achieve remarkable reasoning capabilities by only relying on simple CoT without other complicated prompting strategies. It also reminds us to reflect on the necessity of complicated prompting for scaling and fairly compare different strategies under the same budget.

We systematically analyze this phenomenon and provide theoretical and experimental proofs. We conclude that this is caused by two reasons. One is that there are more easy questions and fewer hard questions for CoT compared to other strategies. Easy questions are more likely to get right solutions, and the error possibility decreases until 0% as scaling. In comparison, hard questions are the opposite. The other is that CoT is less likely to be affected by wrong answers. Although CoT sometimes has lower pass@1 accuracy, its probability of obtaining the correct answer is more prominent in the result distribution. In contrast, other strategies have higher disturbed peaks in the distribution of incorrect answers. These two reasons enable CoT to improve reasoning performance more rapidly and gradually dominate as scaling.

What's more, we propose a method with the complexity $O(1)$ according to probability theory to quickly predict the scaling performance, which

*Corresponding author.

can serve as the test-time scaling law for majority voting. Experiments show that our method can accurately estimate the scaling performance and select the best strategy with arbitrary sampling time.

Furthermore, we explore two ways to significantly improve scaling performance with our theories. (1) Adaptively scaling according to the question difficulty. (2) Dynamically selecting the optimal prompting strategy. Extensive experiments verify their general effectiveness and superiority, *e.g.*, improving Majority@10 accuracy from 86.0% to 97.4% and 15.2% to 61.0% for LLaMA-3-8B-Instruct (Dubey et al., 2024) on GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021b) by combining (1) and (2), respectively.

Our contributions can be summarized as follows:

- We comprehensively study the test-time scaling performance on 6 LLMs $\times$ 8 prompting strategies $\times$ 6 benchmarks. (Section 2)
- We find that CoT consistently performs best under the equivalent sampling time and computation overhead. (Section 3)
- We analyze this phenomenon and provide theoretical and experimental proofs. (Section 4)
- We propose a method to quickly predict the scaling performance and the best strategy under given sampling times. (Section 5)
- Based on the above analysis, we introduce two ways to significantly improve the scaling performance. (Section 6)

2 Scaling System Designs

We focus on a straight and effective setting of test-time scaling, majority voting, *i.e.*, Self-Consistency (Wang et al., 2023b), which selects the most consistent answer among several samples. Our goal is to study what prompting strategy performs best under the equivalent scaling overhead, particularly when largely increasing the scaling extent.

2.1 Models

We conduct experiments on 4 open-sourced LLMs including Qwen2.5-7B-Instruct (Yang et al., 2024a), LLaMA-3-8B-Instruct (Dubey et al., 2024), GLM-4-9B-Chat (GLM et al., 2024) and Phi-3.5-mini-Instruct, and 2 close-sourced LLMs including Gemini-1.5-Flash (Team et al., 2024) and GPT-4o-mini (Open AI, 2024a).

2.2 Prompting Strategies

We mainly focus on generalizable reasoning prompting strategies, excluding those individually designed for specific tasks or involving fine-tuning, training auxiliary models, or incorporating other models, tools, or human assistance. In this setting, the model's performance is only related to the input prompt, thus making it fairly compare the scaling performance of those prompting strategies. The prompting strategies we test are listed as follows.

Direct Prompting (DiP): Directly input the question to the model, without any additional instruction or restrictions to the output.

Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022): Use the prompt "Let's think step by step." to solve the problem step by step.

Least-to-Most (L2M) (Zhou et al., 2023): Break down the question into progressive sub-questions. Answer the sub-questions and get the final result according to them and their answers.

Tree-of-Thoughts (ToT) (Yao et al., 2023): Explore multiple reasoning paths to get several solutions, then analyze each solution and decide which one is the most promising.

Self-Refine (S-RF) (Madaan et al., 2023): First, answer the question to get an initial answer. Next, evaluate the previous answer and get feedback. Finally, refine the previous answer according to feedback. This will last for several rounds.

Step-Back Prompting (SBP) (Zheng et al., 2024): First, extract the discipline concepts and principles involved in solving the problem. Then, solve the problem step by step by following the principles.

Analogous Prompting (AnP) (Yasunaga et al., 2024): Recall relevant problems as examples. Afterward, solve the analogous problems and proceed to solve the initial problem according to them.

Multi-Agent Debate (MAD) (Du et al., 2024): Set three model instances as different agents to debate for several rounds, and select the most consistent result among them.

2.3 Benchmarks

We evaluate across 6 reasoning benchmarks used in the original papers of the above prompting strategies, including GSM8K (Cobbe et al., 2021), GSM-Hard (Gao et al., 2023), MATH-500 (Hendrycks et al., 2021b; Lightman et al., 2024), MMLU-high-school-biology, chemistry and physics (Hendrycks et al., 2021a).

2.4 Formal Expression

We divided the prompting strategies into two groups: iterative methods (S-RF, MAD, and ToT) and the other non-iterative methods. For S-RF and MAD, we run them N rounds and get the final result in the Nth round. For ToT, we set the model to explore and evaluate N different reasoning paths to get the best one. For others, we parallel sample N generations and get their most consistent answer with majority voting. For convenience, we refer to all of the above processes as sampling N times. Therefore, we can categorize those iterative strategies that require multiple rounds or reasoning paths as $\mathcal{P}_2 = \{\text{S-RF, MAD, ToT}\}$, and other non-iterative ones as $\mathcal{P}_1 = \{\text{DiP, CoT, L2M, SBP, AnP}\}$.

Formally, assuming that we have n prompting strategies $\{\mathbf{P}_i \mid i = 1, 2, ..., n\}$, when using the prompt strategy $\mathbf{P}_i$ to answer a text question x on a model $\mathcal{M}$, we can get the answered result of one sample with an answer extractor ϕ, which extracts the answer in the output sentence using regular expressions. Then we can formalize the process of getting the final answer when sampling N times as $\phi[\mathcal{M}(x \mid \mathbf{P}_i); N] =$

$$\begin{cases} \text{mode}\{\phi[\mathcal{M}(x \mid \mathbf{P}_i)]\}_1^N, \mathbf{P}_i \in \mathcal{P}_1 \\ \phi[\mathcal{M}(x \mid \mathbf{P}_i; N)], \quad \mathbf{P}_i \in \mathcal{P}_2 \end{cases} \tag{1}$$

With a fixed sampling time N, the best prompting strategy $\mathbf{P}_N^*$ on the dataset $\mathfrak{D}$ is

$$\mathbf{P}_N^* = \underset{\mathbf{P}_i}{\text{argmax}}\ \mathbb{E}_{x\in\mathfrak{D}}\ \mathbb{1}\{\phi[\mathcal{M}(x \mid \mathbf{P}_i); N] = y\}, \tag{2}$$

where y is the ground truth answer for x. However, sampling with distinct $\mathbf{P}_i$ may cause different computation overhead. It would be fairer to compare them with a fixed overhead O. To calculate the overhead of using a model $\mathcal{M}$ to answer a question x by sampling N times with the prompting strategy $\mathbf{P}_i$, we can consider it as a function of $x, \mathcal{M}, \mathbf{P}_i, N$, noted as $\mathcal{C}(x \mid \mathcal{M}; \mathbf{P}_i; N)$. Under a fixed overhead O, the best prompting strategy $\mathbf{P}_O^*$ on the dataset $\mathfrak{D}$ is

$$\begin{aligned} \mathbf{P}_O^* = \underset{\mathbf{P}_i}{\text{argmax}} \max_N \mathbb{E}_{x\in\mathfrak{D}} \mathbb{1}\{\phi[\mathcal{M}(x \mid \mathbf{P}_i); N] = y\}, \\ s.t. \sum_{x\in\mathfrak{D}} \mathcal{C}(x \mid \mathcal{M}; \mathbf{P}_i; N) \le O. \end{aligned} \tag{3}$$

Given that completion tokens are more computationally expensive than prompt tokens, we define the overhead as the weighted sum of prompt tokens and completion tokens (Cost). For the models Gemini-1.5-Flash and GPT-4o-mini, we utilize their respective pricing metrics.[1] For other open-sourced models, we adopt the pricing of GPT-4o-mini as a proxy.

3 CoT Dominates as Test-Time Scaling

Under each sampling time N, we test five times to obtain the average performance of majority voting. We evaluate under two kinds of budget constraints: (1) a fixed sampling time budget N, and (2) a fixed inference cost budget O. Figure 1 and 2 summarize the average performances across benchmarks of different $\mathbf{P}_i$ under constrained sampling time N and cost O on each model, and display the best prompting strategy $\mathbf{P}_N^*$ under different values of N and $\mathbf{P}_O^*$ under different values of O, respectively.[2] We can see that when scaling test-time compute, CoT performs best among all prompting strategies under a constrained N and O most of the time. Although some complicated prompting strategies perform best under lower N and O, CoT dominates without exception on all models when largely scaling. We theoretically and experimentally analyze this phenomenon, whose reasons come from two aspects. We explain these in detail in Section 4.

What's more, we find that about 80% of the results conform to this trend on each model and each benchmark. On certain datasets and LLMs, DiP also performs best as largely scaling. This is particularly evident on powerful models, such as Gemini-1.5-Flash and GPT-4o-mini. More detailed results can be found in Appendix C. These indicate that simple CoT is more efficient and has the potential to surpass other complicated prompting strategies under the same scaling setting. Current LLMs can achieve remarkable reasoning capabilities by only relying on simple prompting strategies. Complicated prompting with superior pass@1 accuracy may not always be better as test-time scaling.

[1] The price of Gemini-1.5-Flash: \$0.075/1M prompt tokens, \$0.3/1M completion tokens. The price of GPT-4o-mini: \$0.15/1M prompt tokens, \$0.6/1M completion tokens.

[2] We don't test the performance with very large N for $\mathbf{P}_i \in \mathcal{P}_2$, as this will lead to extremely long context, large cost and computation time, and marginally increased or even decreased performance, which is no better than Self-Consistency (Smit et al., 2024). S-RF performs poorly even with multiple rounds. This is consistent with the results of (Huang et al., 2024), which points out the limitations of S-RF.

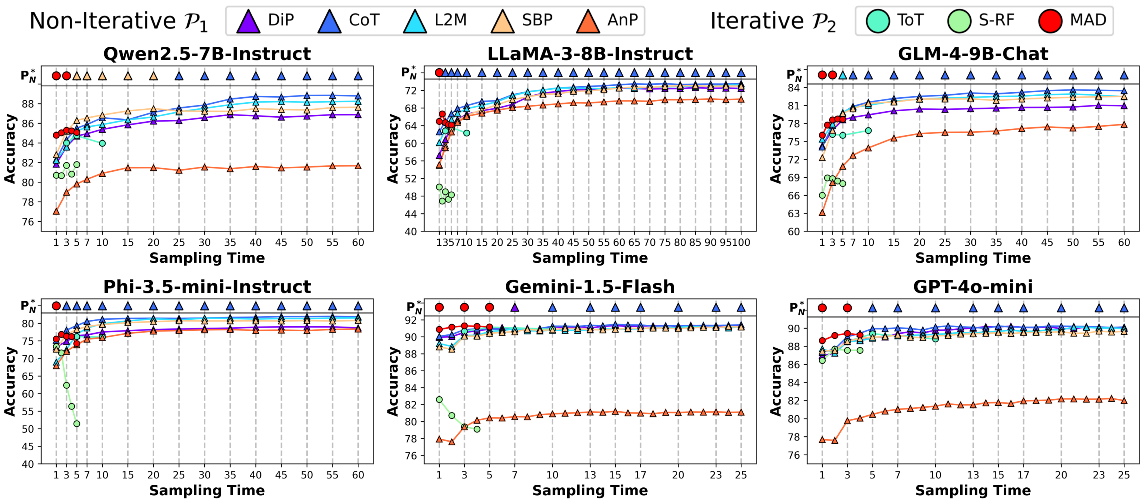

Figure 1: Average performances of distinct prompting strategies and the best one $\mathbf{P}_N^*$ across benchmarks on each LLM under constrained sampling time N. As increasing the sample time N, the accuracy of CoT grows rapidly and it dominates on all models when N is large enough.

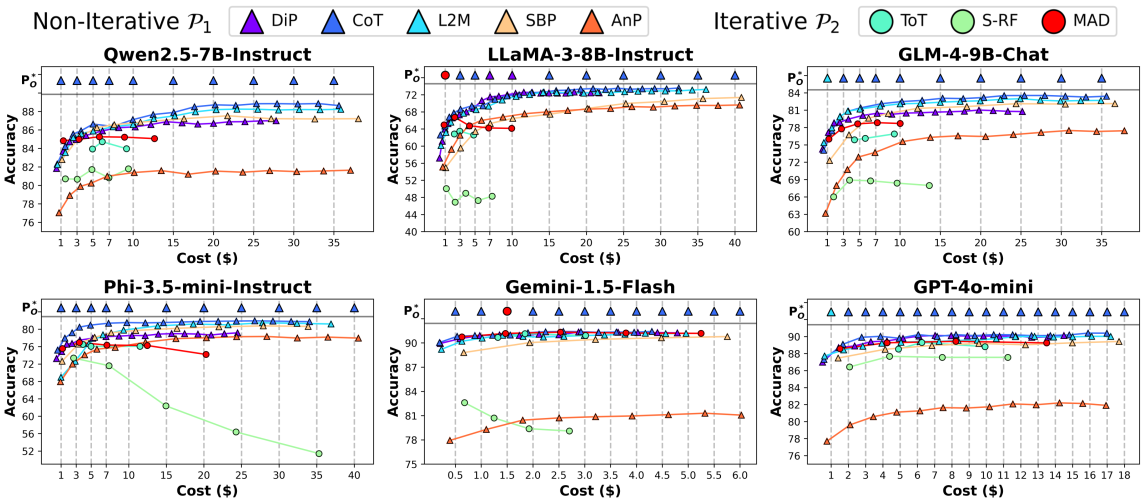

Figure 2: Average performances of distinct prompting strategies and the best one $\mathbf{P}_O^*$ across benchmarks on each LLM under constrained cost O. Under the equal cost O, CoT performs best most of the time. When O grows larger, CoT gradually becomes the best prompt strategy $\mathbf{P}_O^*$ on all models.

4 Why CoT Performs worse with Lower N while better with Larger N?

Let us consider a specific input question x, note the answer space $\mathcal{A} = \{a_1, a_2, \dots, a_m\}$ as the set of all probable values of $\phi[\mathcal{M}(x \,|\, \mathbf{P}_i)]$ for all $\mathbf{P}_i$, *i.e.*, $\phi[\mathcal{M}(x \,|\, \mathbf{P}_i)] \in \mathcal{A}\ for\ \forall\ \mathbf{P}_i$. We omit $N = 1$ in $\phi[\mathcal{M}(x \,|\, \mathbf{P}_i)]$ for brevity. $\{p_{i,1}, p_{i,2}, \dots, p_{i,m}\}$ denotes the corresponding probabilities, *i.e.*, $p_{i,j} = \Pr\left(\phi[\mathcal{M}(x \,|\, \mathbf{P}_i)] = a_j\right)$. Note a_i^* as the final result of $\mathbf{P}_i$ by scaling sampling N times, *i.e.*, $a_i^* = \phi[\mathcal{M}(x \,|\, \mathbf{P}_i); N]$. Then the occurrence number $\mathbf{X}_i = (\mathbf{x}_{i,1}, \dots, \mathbf{x}_{i,m})$ of each probable answer for $\mathbf{P}_i$ follows a multinomial distribution, *i.e.*, $\mathbf{X}_i \sim Mult(N, p_{i,1}, p_{i,2}, \dots, p_{i,m})$. The process of getting the final result a_i^* of $\mathbf{P}_i$ by sampling N times can be formalized as:

$$\begin{aligned} \mathcal{X}_{ir} &= \{\mathbf{x}_{i,j} \,|\, \mathbf{x}_{i,j} = \max\{\mathbf{X}_i\}\} \\ k &\sim \text{Uniform}(\mathcal{X}_{ir}), \quad a_i^* = a_k \end{aligned} \tag{4}$$

Next, we will introduce several lemmas and theorems to explain the two reasons why CoT sometimes performs worse with lower N while better with larger N. In the following proof, we omit the input x, assume a_1 is the correct answer, and note the probability of getting a_1 when sampling N times with $\mathbf{P}_i$ as $\Pr(a_1|\mathbf{P}_i; N)$, which can be regarded as the expectation of the accuracy $\mathbb{1}\{\phi[\mathcal{M}\,(x \,|\, \mathbf{P}_i); N] = y\}$. Details about the proof process can be found in Appendix B.

Definition 1. *Note* $p_{max} = \max\{p_{i,1}, \dots, p_{i,m}\}$, $\mathcal{S} = \{a_j \,|\, p_{i,j} = p_{max}\}$*, we can define the difficulty of the input question* x *for* $\mathbf{P}_i$*. If* $a_1 \in$

$\mathcal{S}$ *and* $|\mathcal{S}| = 1$*, we call* x *an easy question for* $\mathbf{P}_i$*. If* $a_1 \in \mathcal{S}$ *and* $|\mathcal{S}| > 1$*, we call* x *a moderate question for* $\mathbf{P}_i$*. If* $a_1 \notin \mathcal{S}$*, we call* x *a hard question for* $\mathbf{P}_i$*.*

Theorem 1. *If* x *is an easy question for* $\mathbf{P}_i$*,* $Pr(a_1|\mathbf{P}_i; N)$ *is non-decreasing w.r.t.* N*,* $\lim_{N\to+\infty} Pr(a_1|\mathbf{P}_i; N) = 1$*.*

Theorem 2. *If* x *is a moderate question for* $\mathbf{P}_i$*,* $Pr(a_1|\mathbf{P}_i; N)$ *is non-decreasing w.r.t.* N*,* $\lim_{N\to+\infty} Pr(a_1|\mathbf{P}_i; N) = 1/|\mathcal{S}|$*.*

Theorem 3. *If* x *is a hard question for* $\mathbf{P}_i$*,* $Pr(a_1|\mathbf{P}_i; N)$ *exhibits a general declining trend w.r.t.* N*,* $\lim_{N\to+\infty} Pr(a_1|\mathbf{P}_i; N) = 0$*.*

Lemma 1. *Consider a specific condition with answer space* $|\mathcal{A}| = 3$*. For* $N = 3$*,* $Pr(a_1|\mathbf{P}_i; N) = 3p_{i,1}^2 - 2p_{i,1}^3 + 2p_{i,1}p_{i,2}p_{i,3}$*. For* $N = 5$*,* $Pr(a_1|\mathbf{P}_i; N) = 6p_{i,1}^5 - 15p_{i,1}^4 + 10p_{i,1}^3 + 15p_{i,1}^2 p_{i,2}p_{i,3}(p_{i,2} + p_{i,3})$*.*

Theorem 4. *For two prompting strategies* $\mathbf{P}_i$ *and* $\mathbf{P}_{i'}$*, note* $p_{i,q} = \max\{p_{i,2}, \ldots, p_{i,m}\}$*,* $p_{i',q'} = \max\{p_{i',2}, \ldots, p_{i',m}\}$*, if* $p_{i,1} - p_{i,q} < p_{i',1} - p_{i',q'}$ *and* $p_{i,1} + p_{i,q} - p_{i,1}^2 - p_{i,q}^2 > p_{i',1} + p_{i',q'} - p_{i',1}^2 - p_{i',q'}^2$*, there exits a sufficiently large* N_0 *such that for* $N > N_0$*,* $Pr(a_1|\mathbf{P}_i; N) < Pr(a_1|\mathbf{P}_{i'}; N)$*.*

4.1 CoT Has More Easy Questions and Fewer Hard Questions

We identify two primary reasons why CoT sometimes performs worse with lower sample sizes (N) but achieves better performance among these prompting approaches with larger N. The first reason relates to the distribution of question difficulty for CoT. CoT has more easy questions and fewer hard questions. When sampling with lower N, $\mathbf{P}_i$ still has a small probability of obtaining the right answer for hard questions, while the probability diminishes to zero as increasing N. This is the opposite of easy questions. The prompting strategy with fewer hard questions and more easy questions will improve performance more rapidly when scaling. According to Theorem 1 to 3, we can calculate the extreme performance of $\mathbf{P}_i$ according to the difficulty proportion of questions, *i.e.*, $\sum_{x\in\mathfrak{D}} \lim_{N\to+\infty} \Pr(a_1|\mathbf{P}_i; N)$. Table 1 summarizes the difficulty proportion of the questions and extreme performance for each $\mathbf{P}_i$ on each model. It can be observed that CoT has more easy questions and fewer hard questions, and can reach the best extreme performance on all models, thus making CoT gradually dominate as increasing N even if it has a lower pass@1 accuracy.

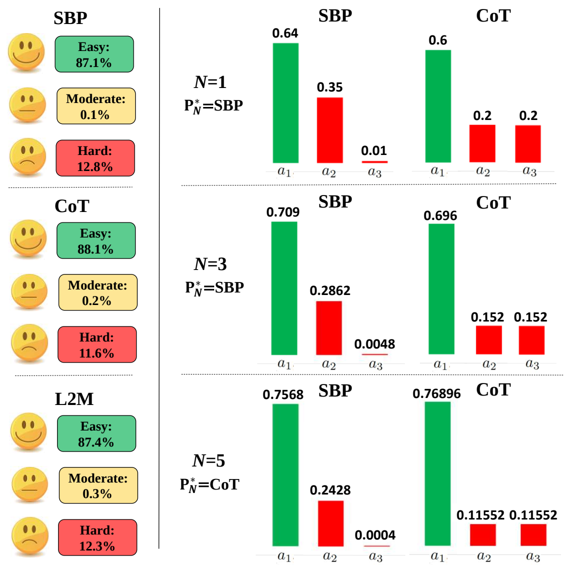

Figure 3: **Illustration of the two reasons why CoT sometimes performs worse with lower N while better with larger N.** *Left:* CoT has more easy questions and fewer hard questions. For example, the probability distribution of L2M is $\{0.4, 0.5, 0.1, 0.0, 0.0\}$ (hard question), and $\{0.3, 0.2, 0.2, 0.2, 0.1\}$ (easy question) for CoT. Although L2M has higher pass@1 accuracy, its accuracy reduces until 0% as scaling while CoT increases until 100%. *Right:* CoT is less likely to be affected by wrong answers due to their relatively uniform distribution. The probability of obtaining the right answer a_1 grows more rapidly as increasing N.

4.2 CoT is Less Likely to be Affected by Wrong Answers

The second reason for this phenomenon is that CoT is less likely to be affected by wrong answers. $\Pr(a_1|\mathbf{P}_i; N)$ is not a function of only the probability $p_{i,1}$ of the right answer a_1, but also related to the probability distribution of other wrong answers. According to Theorem 4, even if $p_{i,1} > p_{i',1}$, *i.e.*, $\Pr(a_1|\mathbf{P}_i; N = 1) > \Pr(a_1|\mathbf{P}_{i'}; N = 1)$, there still may exist an N_0 that $\Pr(a_1|\mathbf{P}_i; N = N_0) > \Pr(a_1|\mathbf{P}_{i'}; N = N_0)$. Considering a question x in GSM8K as an example and a_1 is the correct answer, the result probability distribution of $\mathbf{P}_i$ = SBP is $\{0.64, 0.35, 0.01\}$, and $\{0.6, 0.2, 0.2\}$ for $\mathbf{P}_{i'}$ = CoT, which satisfies the condition in Theorem 4. According to Lemma 1, $\Pr(a_1|\mathbf{P}_i; N = 1) = 0.640 > \Pr(a_1|\mathbf{P}_{i'}; N = 1) = 0.600$, $\Pr(a_1|\mathbf{P}_i; N = 3) = 0.709 > \Pr(a_1|\mathbf{P}_{i'}; N = 3) = 0.696$, while $\Pr(a_1|\mathbf{P}_i; N = 5) = 0.757 < \Pr(a_1|\mathbf{P}_{i'}; N = 5) = 0.769$. This means, although complicated prompting strategies may have higher

Table 1: **Difficulty proportion of questions and extreme peformance (denote by "Acc") for each $\mathbf{P}_i$ and LLM across benchmarks.** CoT has more easy questions and fewer hard questions, and can reach the best extreme performance on all LLMs. More results on other models are shown in Table 4.

$\mathbf{P}_i$	Easy	Moderate	Hard	Acc	Easy	Moderate	Hard	Acc	Easy	Moderate	Hard	Acc
	Qwen2.5-7B-Instruct				**LLaMA-3-8B-Instruct**				**GLM-4-9B-Chat**			
DiP	86.3%	0.3%	13.4%	86.4	69.7%	1.0%	29.3%	70.2	79.8%	0.6%	19.6%	80.1
CoT	**88.1%**	0.2%	**11.6%**	**88.2**	**70.9%**	0.9%	28.2%	**71.3**	**82.8%**	0.8%	**16.5%**	**83.1**
L2M	87.4%	0.3%	12.3%	87.6	70.3%	1.6%	**28.1%**	71.0	81.9%	0.4%	17.7%	82.1
SBP	87.1%	0.1%	12.8%	87.2	67.3%	1.3%	31.3%	68.0	81.4%	0.9%	17.6%	81.9
AnP	81.1%	0.5%	18.4%	81.3	67.5%	1.4%	31.1%	68.2	76.4%	1.2%	22.4%	77.0

Table 2: **Quantity of questions described in Section 4.2.** Results are displayed as "x/y", where x is for Qwen and y for LLaMA. The value v_{ij} in the ith row and jth column represents the quantity of data that satisfies Theorem 4. Results prove that CoT has greater potential to significantly increase performance as scaling.

$\mathbf{P}_i$ \ $\mathbf{P}_{i'}$	DiP↓	CoT↓	L2M↓	SBP↓	AnP↓	Sum ↓
DiP ↑	-	447/816	414/794	457/459	393/513	1711/2582
CoT ↑	423/620	-	374/646	416/382	361/408	**1574/2046**
L2M ↑	505/639	510/677	-	494/432	403/393	1912/2141
SBP ↑	599/1316	601/1433	564/1398	-	429/923	2193/5070
AnP ↑	800/1243	817/1381	799/1380	776/871	-	3192/4875
Sum ↑	2327/3818	**2375/4307**	2151/4218	2143/2144	1586/2237	-

pass@1 accuracy, they are easier to be affected by wrong answers. In contrast, simple CoT has a relatively flat distribution on wrong answers, thus making it focus more on the correct answer, which makes it more rapidly improve performance in easy questions and more slowly reduce accuracy in hard questions as increasing N, as shown in Figure 3. We record the quantity of such questions for each two prompting strategies and display the results of Qwen2.5-7B-Instruct and LLaMA-3-8B-Instruct in Table 2. If CoT is $\mathbf{P}_{i'}$, there are the most data satisfying Theorem 4. If CoT is $\mathbf{P}_i$, there are the least such questions. These demonstrate that CoT has greater potential to significantly increase scaling performance compared with other strategies.

5 Predicting Scaling Performance and $\mathbf{P}_N^*$

In practice, evaluating the test-time scaling performance requires significantly intensive resource consumption, especially with very large sampling time N. For pretraining, it is feasible to predict the train-time scaling performance based on the scaling law (Kaplan et al., 2020) through a series of low-cost experiments, while maintaining the model architecture largely unchanged and minimizing the risks associated with large-scale training. Similarly, we can also use the sample results of $\mathbf{P}_i$ with fewer N to approximately get the distribution $\{p_{i,1}, p_{i,2}, \ldots, p_{i,m}\}$ to predict the test-time scaling performance with larger N. Directly, one can utilize the multinomial distribution probability calculation formula (Equation 13 and 14) to calculate $\Pr(a_1|\mathbf{P}_i; N)$ with enumeration or leverage numerical simulation to estimate. However, their computational complexities are both $O(N)$, and the former needs to traverse all situations and is difficult to operate. Therefore, we propose a method with the computational complexity $O(1)$ to quickly predict the scaling performance of majority voting for arbitrary $\mathbf{P}_i$, which can serve as the test-time scaling law. It can also select the best prompting strategy $\mathbf{P}_N^*$ according to the predicted performances of each $\mathbf{P}_i$.

Here we omit the prompting index i and input question x, and assume a_1 is the correct answer in the following. According to Khinchin's Law of Large Numbers and Lindeberg-Levy Central Limit Theorem, when N is large enough, each $\mathbf{x}_j$ can be approximated by a normal distribution. Specifically, for $\mathbf{x}_1$, we have

$$\mathbf{x}_1 \sim \mathcal{N}(Np_1, Np_1(1-p_1)), \tag{5}$$

i.e., a normal distribution with mean Np_1 and variance $Np_1(1-p_1)$. Considering the maximum value among all other $\mathbf{x}_j$ ($j \neq 1$), denoted as $M = \max(\mathbf{x}_2, ..., \mathbf{x}_m)$, when N is large enough, the distribution of M can be approximated by

$$M \sim \mathcal{N}(Np_{max}, Np_{max}(1-p_{max})), \tag{6}$$

where p_{max} is the second highest probability excluding p_1. We now need to calculate $P(\mathbf{x}_1 > M)$, which can be approximated by comparing two normal distributions. Let $Z = \mathbf{x}_1 - M$, then the distribution of Z is

$$Z \sim \mathcal{N}(N(p_1 - p_{max}), N(p_1(1-p_1) + p_{max}(1-p_{max})). \tag{7}$$

Therefore,

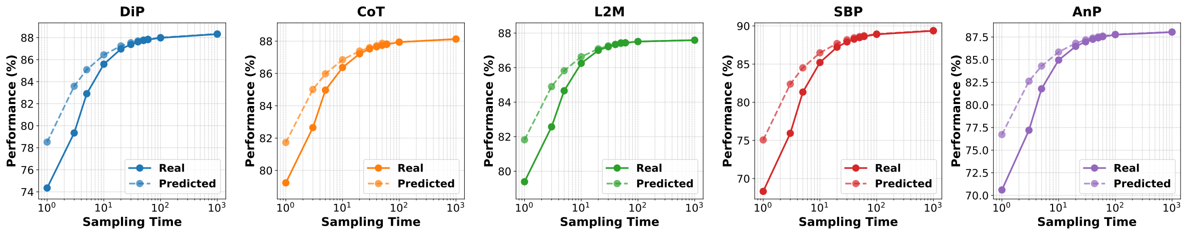

Figure 4: Real and predicted performance using our method of different $\mathbf{P}_i$ under various sampling time constraints. Our method can accurately estimate the scaling performance of arbitrary $\mathbf{P}_i$, especially with large N.

Table 3: The true best prompting strategy $\mathbf{P}_N^*$ and the predicted $\mathbf{P}_N^*$ using our method under various sampling time constraints. Our method can correctly predict the best prompting strategy under any constraints evaluated.

$\mathbf{P}_N^*$	**Sampling Time** N							
	1	**3**	**5**	**10**	**20**	**50**	**100**	**1000**
Real	L2M	CoT	CoT	CoT	SBP	SBP	SBP	SBP
Predicted	L2M	CoT	CoT	CoT	SBP	SBP	SBP	SBP
Correct	✓	✓	✓	✓	✓	✓	✓	✓

$$\Pr(a^* = a_1) \approx \Pr(Z > 0), \tag{8}$$

where a^* is the final sample result. Using properties of the standard normal distribution, we can write

$$\begin{aligned} \Pr(Z > 0) &= \Pr\left(\frac{Z - E[Z]}{\sqrt{\mathrm{Var}[Z]}} > \frac{-E[Z]}{\sqrt{\mathrm{Var}[Z]}}\right) \\ &= 1 - \Phi\left(\frac{-E[Z]}{\sqrt{\mathrm{Var}[Z]}}\right), \end{aligned} \tag{9}$$

$$E[Z] = N(p_1 - p_{max}),$$

$$Var[Z] = N(p_1(1 - p_1) + p_{max}(1 - p_{max})),$$

where Φ is the standard normal cumulative distribution function. Thus, we can quickly predict the scaling performance and select the best prompting strategy $\mathbf{P}_N^*$ with given N by

$$\Pr(a_1|\mathbf{P}_i; N) \approx 1-\Phi\left(\frac{-(p_1 - p_{max})}{\sqrt{\frac{p_1(1-p_1)+p_{max}(1-p_{max})}{N}}}\right), \tag{10}$$

$$\text{Accuracy}(\mathbf{P}_i, N) = \mathbb{E}_{x\in\mathfrak{D}} \Pr(a_1|\mathbf{P}_i, N), \tag{11}$$

$$\mathbf{P}_N^* = \underset{\mathbf{P}_i}{\text{argmax}} \ \text{Accuracy}(\mathbf{P}_i, N). \tag{12}$$

Experiment. We verify our method on LLaMA-3-8B-Instruct on GSM8K, only using 40 samples to estimate $p_{i,j}$. Results are shown in Figure 4. We can see that our method can accurately estimate the scaling performance, and the error decreases until 0% as scaling sampling time. When $N \geq 10$, the error is already less than 1%. This makes sense as our method is based on the assumption that N is large enough. Although the prediction accuracy is not very high when N is small, the difference in predicted performances between distinct $\mathbf{P}_i$ is similar to that in true performances, so our method can correctly select the best prompting strategy $\mathbf{P}_N^*$ with arbitrary N, as shown in Table 3.

6 Improving Scaling Peformance

According to the analysis in Section 4, we can further improve the scaling performance in two ways. Extensive experiments confirm their effectiveness, leading to significant improvements. We will further explore them in the future. All following results are conducted on Qwen-2.5-7B-Instruct on GSM8K. Please refer to Appendix D for more results on other LLMs and benchmarks.

6.1 Adaptively Scaling Based on the Difficulty

According to Theorem 1 to 4, it will lead to decreased performance when scaling sampling time on hard questions. Performances only continuously improve on easy questions. **Therefore, when facing a hard question, we can force LLMs to only answer it once without scaling more. If the question is a moderate or easy question, LLMs scale sampling time as usual.** We evaluate the performance both when forcing the LLM to determine the question difficulty itself (noted as "Adaptive") and providing the difficulty oracle to the LLM as an upper bound reference (noted as "Oracle"), as shown in Figure 5. "Adaptive" performance is almost equal to the usual scaling performance (noted as "Vanilla"), which is because the LLM is more inclined to believe a question is easy, especially on more complicated $\mathbf{P}_i$ such as SBP and AnP. Nevertheless, all $\mathbf{P}_i$ can significantly improve their scaling performances with question difficulty oracles, proving the potential of this method.

6.2 Dynamically Choosing the Optimal $\mathbf{P}_i$

For a question x, it may be a hard question for a prompting strategy $\mathbf{P}_i$ with higher accuracy, while

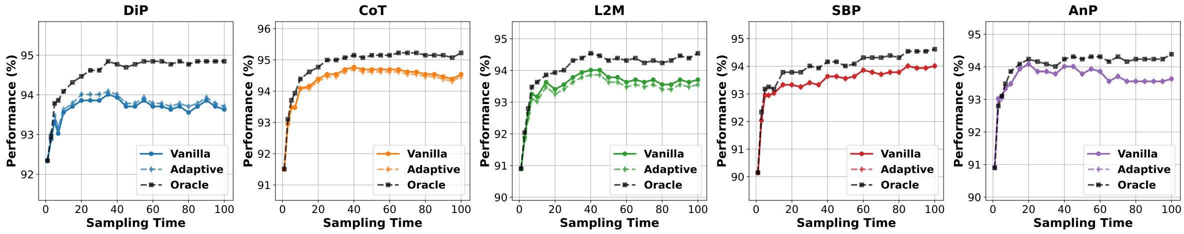

Figure 5: Results of adaptively scaling for each $\mathbf{P}_i \in \mathcal{P}_1$ based on oracle and predicted question difficulty.

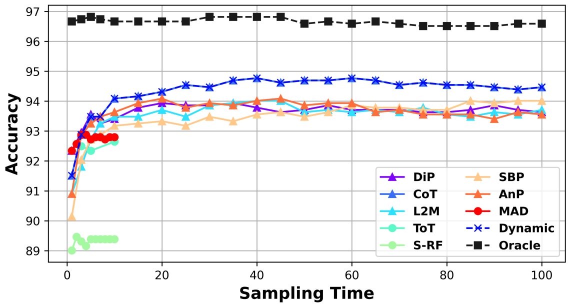

Figure 6: Results of dynamically choosing the optimal $\mathbf{P}_i$.

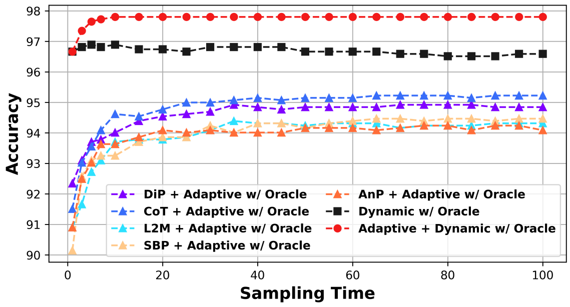

Figure 7: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ with oracles.

an easy question for another strategy $\mathbf{P}_{i'}$ with lower accuracy. **So if we can choose the optimal prompting strategy for each question, it will largely improve the performance.** We test the scaling performance both when forcing the LLM to choose the most suitable $\mathbf{P}_i$ (noted as "Dynamic") and providing the oracles as an upper bound (noted as "Oracle"), *i.e.*, telling the LLM which $\mathbf{P}_i$ maximizes $\Pr(a_1|\mathbf{P}_i; N)$, as shown in Figure 6. "Dynamic" performance is almost equal to CoT. This is because Qwen believes CoT is the best $\mathbf{P}_i$ among 8 prompting strategies in 99.7% of the questions. However, it can achieve significant improvement with oracles. This means that selecting the best $\mathbf{P}_i$ for each question is more effective than majority voting, as "Oracle" performance with only $N = 1$ is much higher than $\forall \mathbf{P}_i$ with even $N \to +\infty$. However, "Oracle" performance does not increase with scaling. This is because there are questions that are hard for all $\mathbf{P}_i$. Even if we select the best $\mathbf{P}_i$ on a question, its accuracy still reduces with scaling. So if we can combine the two methods in Section 6.1 and 6.2, it would lead to much more improvement.

6.3 Combining Adaptively Scaling and Dynamically Choosing the Optimal $\mathbf{P}_i$

Figure 7 reports the peformance upper bounds of each $\mathbf{P}_i \in \mathcal{P}_1$ + "Adaptive", "Dynamic", and combining "Adaptive" and "Dynamic". Experiment results demonstrate the powerful potential of the combined method. We will explore more feasible methods to reach this upper bound in future work.

7 Related Work

Reasoning Prompting Strategies. CoT series carefully design exemplars or 0-shot prompts to unleash the potential of step-by-step solving (Wei et al., 2022; Kojima et al., 2022; Zhang et al., 2023; Fu et al., 2023). (Zhou et al., 2023; Dua et al., 2022; Khot et al., 2023) break down the question into smaller, more manageable subproblems. (Madaan et al., 2023; Kim et al., 2023) force LLMs to self-evaluate and correct. (Du et al., 2024; Liang et al., 2024; Chan et al., 2024; Liu et al., 2025) utilize multi-agent debate to collaborate reasoning. (Yasunaga et al., 2024; Yu et al., 2024) guide LLMs to draw experience from analogous problems. (Zheng et al., 2024; Gao et al., 2025) promote LLMs on abstract reasoning.

Scaling Test-Time Compute. Self-Consistency is a simple but effective scaling method (Wang et al., 2023b). (Li et al., 2023; Hosseini et al., 2024) train a verifier to evaluate samples and select the best solution. Some use iterative refinement (Madaan et al., 2023) or multiple rounds of debate(Du et al., 2024). Others leverage the theory of tree search (Yao et al., 2023; Ding et al., 2024; Zhang et al., 2024a) and graph search (Besta et al., 2024a; Jin et al., 2024a) to expand and aggregate reasoning paths (Besta et al., 2024b).

8 Conclusion

We comprehensively study the behavior of various prompting strategies when scaling majority voting. Our experiments on 6 LLMs $\times$ 8 prompting strategies $\times$ 6 benchmarks consistently show that CoT has the potential to perform best as scaling. Theoretical analyses reveal that CoT benefits from fewer hard questions, more easy questions, and less susceptibility to incorrect answers, enabling rapid performance gains with increased sampling. Additionally, our proposed method for predicting scaling performance offers a practical tool to select the optimal prompting strategy under given budgets. What's more, we introduce two effective methods to further improve scaling performance. We hope our research can provide new insights to understand, predict, and improve LLM test-time scaling performance.

Limitations

In this paper, we mainly focus on majority voting, which is a simple but effective scaling approach. However, we don't test on other more complex scaling approaches such as Monte Carlo Tree Search. Our finding that CoT dominates as scaling most of the time does not always hold for every LLM on every dataset, *e.g.*, Table 3. Nevertheless, 80% of the results conform to this rule. In fact, it depends on the composition of the dataset. If we specifically collect hard questions for $\mathbf{P}_i$ as a dataset, it will lead to a continuous decline performance of $\mathbf{P}_i$. Our experiments and analysis indicate that, even though some $\mathbf{P}_i$ may perform poorly with lower sampling time, they hold the potential to exhibit superior performance than other prompting strategies as test-time scaling. We propose two superior methods according to rigorous theories, which can significantly improve scaling performance on each model and each benchmark we test, and we are confident in the universality of our methods. However, our experiment results indicate that LLMs alone cannot readily achieve the intended effects, pushing us to explore more practicable and effective methods in our future work.

Ethical Considerations

There are many potential societal consequences of our work, none which we feel must be specifically highlighted here. The sole potential risk we acknowledge is that scaling compute may result in substantial electricity consumption and carbon dioxide emissions.

Acknowledgment

We thank Jie Cao and Huaibo Huang for their insightful discussions. The work is supported by National Natural Science Foundation of China (Grant No. 62425606, 32341009, U21B2045) and the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDA0480302).

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, et al. 2023. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

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

Maciej Besta, Florim Memedi, Zhenyu Zhang, Robert Gerstenberger, Nils Blach, Piotr Nyczyk, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Lukas Gianinazzi, et al. 2024b. Topologies of reasoning: Demystifying chains, trees, and graphs of thoughts. *arXiv preprint arXiv:2401.14295*.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2024. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better llm-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024a. Are more llm calls all you need? towards the scaling properties of compound ai systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2024b. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*.

Qiguang Chen, Libo Qin, WANG Jiaqi, Jingxuan Zhou, and Wanxiang Che. 2024c. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024d. A simple and provable scaling law for the test-time compute of large language models. *arXiv preprint arXiv:2411.19477*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yingqian Cui, Pengfei He, Xianfeng Tang, Qi He, Chen Luo, Jiliang Tang, and Yue Xing. 2024. A theoretical understanding of chain-of-thought: Coherent reasoning and error-aware demonstration. In *The 28th International Conference on Artificial Intelligence and Statistics*.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. 2025. Learning how hard to think: Input-adaptive allocation of lm computation. In *The Thirteenth International Conference on Learning Representations*.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. Everything of thoughts: Defying the law of penrose triangle for thought generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1638–1662. Association for Computational Linguistics.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1251–1265.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2024. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10764–10799.

Silin Gao, Jane Dwivedi-Yu, Ping Yu, Xiaoqing Ellen Tan, Ramakanth Pasunuru, Olga Golovneva, Koustuv Sinha, Asli Celikyilmaz, Antoine Bosselut, and Tianlu Wang. 2025. Efficient tool use with chain-of-abstraction reasoning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2727–2743.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. In *Conference On Language Modeling*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Jia Xu, Linjian Mo, and Min Zhang. 2025. Test-time computing: from system-1 thinking to system-2 thinking. *arXiv preprint arXiv:2501.02497*.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024a. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 163–184.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024b. The impact of reasoning step length on large language models. In *Findings of* *the Association for Computational Linguistics: ACL 2024*, pages 1830–1842.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. In *Advances in Neural Information Processing Systems*, volume 36, pages 39648–39677.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in neural information processing systems*, volume 35, pages 22199–22213.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations*.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Yexiang Liu, Jie Cao, Zekun Li, Ran He, and Tieniu Tan. 2025. Breaking mental set to improve reasoning through diverse multi-agent debate. In *The Thirteenth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 46534–46594.

Open AI. 2024a. GPT-4o-mini. https://openai.com/ja-JP/index/gpt-4o-mini-advancing-cost-efficient-intelligence.

Open AI. 2024b. Introducing openai o1. https://openai.com/o1/.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2025. Mutual reasoning makes smaller llms stronger problem-solvers. In *The Thirteenth International Conference on Learning Representations*.

Marija Šakota, Maxime Peyrard, and Robert West. 2024. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 606–615.

Andries Petrus Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D Barrett, and Arnu Pretorius. 2024. Should we be going mad? a look at multi-agent debate strategies for llms. In *Forty-first International Conference on Machine Learning*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling llm test-time compute optimally can be more effective than scaling model parameters. In *The Thirteenth International Conference on Learning Representations*.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

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

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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. In *Advances in neural information processing systems*, volume 35, pages 24824–24837.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Chenxiao Yang, Zhiyuan Li, and David Wipf. 2024b. An in-context learning theoretic analysis of chain-of-thought. In *ICML 2024 Workshop on In-Context Learning*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 11809–11822.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H Chi, and Denny Zhou. 2024. Large language models as analogical reasoners. In *The Twelfth International Conference on Learning Representations*.

Junchi Yu, Ran He, and Zhitao Ying. 2024. Thought propagation: An analogical approach to complex reasoning with large language models. In *The Twelfth International Conference on Learning Representations*.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2024. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *The Twelfth International Conference on Learning Representations*.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024a. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*.

Kexun Zhang, Shang Zhou, Danqing Wang, William Yang Wang, and Lei Li. 2024b. Scaling llm inference with optimized sample compute allocation. *arXiv preprint arXiv:2410.22480*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Broader Related Work

Efficient Reasoning. Several studies have shown that scaling test-time compute can be more effective than scaling model parameters (Snell et al., 2025; Open AI, 2024b). (Aggarwal et al., 2023; Li et al., 2024; Chen et al., 2024a) improve the reasoning efficiency with majority voting by adjusting the sampling time. (Damani et al., 2025; Zhang et al., 2024b) learn to dynamically allocate resources under limited sampling time budgets. (Chen et al., 2024b; Yue et al., 2024; Šakota et al., 2024) leverage multiple models with different prices to reduce cost while maintaining performance.

Role and Mechanism of CoT and Test-Time Scaling. (Jin et al., 2024b) studies the impact of reasoning step length of CoT. (Wang et al., 2023a) studies what makes CoT prompting effective, indicating that being relevant to the query and correctly ordering the reasoning steps are more important. (Feng et al., 2024; Cui et al., 2024) analyze the mechanism of CoT from a theoretical perspective. (Sprague et al., 2025) points out that CoT helps mainly on math and symbolic reasoning by sorting and analyzing a large number of experimental results. (Chen et al., 2024c) proposes a framework to quantify the reasoning boundary of CoT. (Yang et al., 2024b) provides an in-context learning analysis of CoT. (Chen et al., 2024a) investigates and analyzes the performance changes with more LLM calls. (Chen et al., 2024d) proves that the failure probability of test-time scaling decays to zero exponentially or by a power law.

B Proofs

Theorem 1. If x is an easy question for $\mathbf{P}_i$, $\Pr(a_1|\mathbf{P}_i;N)$ is non-decreasing *w.r.t.* N, $\lim_{N\to+\infty}\Pr(a_1|\mathbf{P}_i;N)=1$.

Theorem 2. If x is a moderate question for $\mathbf{P}_i$, $\Pr(a_1|\mathbf{P}_i;N)$ is non-decreasing *w.r.t.* N, $\lim_{N\to+\infty}\Pr(a_1|\mathbf{P}_i;N)=1/|\mathcal{S}|$.

Theorem 3. If x is a hard question for $\mathbf{P}_i$, $\Pr(a_1|\mathbf{P}_i;N)$ exhibits a general declining trend *w.r.t.* N, $\lim_{N\to+\infty}\Pr(a_1|\mathbf{P}_i;N)=0$.

Proof. The occurrence number $\mathbf{X}_i=(\mathbf{x}_{i,1},\ldots,\mathbf{x}_{i,m})$ of each probable answer for $\mathbf{P}_i$ follows a multinomial distribution, *i.e.*, $\mathbf{X}_i \sim Mult(N,p_{i,1},p_{i,2},\ldots,p_{i,m})$. When sampling N times, the specific probability of a certain occurrence number can be calculated with Equation 13. For brevity, we omit the input x, sampling time N, and the prompting index i in $\mathbf{x}_{i,j}$ and $p_{i,j}$ in the following equations.

$$\Pr(\mathbf{x}_1=k_1,\mathbf{x}_2=k_2,\ldots,\mathbf{x}_m=k_m)$$

$$=\underbrace{\underbrace{\frac{N!}{k_1!k_2!\cdots k_m!}}_{coefficient}\underbrace{p_1^{k_1}p_2^{k_2}\cdots p_m^{k_m}}_{probability\ term}}_{a\ term\ in\ \Pr(a_1|\mathbf{P}_i;N)} \tag{13}$$

$$s.t.\quad \textstyle\sum_{j=1}^m k_j=N,\ \sum_{j=1}^m p_j=1$$

Assuming the correct answer is a_1, $M=\max(k_2,\ldots,k_m)$, the probability of obtaining the right answer by sampling N times with $\mathbf{P}_i$ is

$$\Pr(a_1|\mathbf{P}_i)=\Pr(\mathbf{x}_1>M)+\sum_{|\mathcal{J}|=1}^{m-1}$$
$$\frac{\Pr(\mathbf{x}_1=\mathbf{x}_j>\mathbf{x}_q\ for\ all\ j\in\mathcal{J},q\notin\{1\}\cup\mathcal{J})}{|\mathcal{J}|+1}, \tag{14}$$

where $\mathcal{J}$ is the set of all indexes j ($j\neq 1$) of $\mathbf{x}_j$ that satisfies $\mathbf{x}_j=\mathbf{x}_1$.

$$\mathrm{Pr}_1=\Pr(\mathbf{x}_1>M)=$$
$$\sum_{\sum_{j=1}^m k_j=N}\frac{N!}{k_1!\cdots k_m!}p_1^{k_1}\prod_{j=2}^m p_j\mathbb{1}(k_j<k_1), \tag{15}$$

$$\mathrm{Pr}_2=\Pr(\mathbf{x}_1=\mathbf{x}_j>\mathbf{x}_q\ for\ all\ j\in\mathcal{J},q\notin\{1\}\cup\mathcal{J}))$$
$$=\sum_{\sum_{j=1}^m k_j=N}\frac{N!}{k_1!^{|\mathcal{J}|}\prod_{q\notin\{1\}\cup J}k_k!}p_1\prod_{j\in\mathcal{J}}p_j\prod_{q\notin\{1\}\cup\mathcal{J}}p_k\mathbb{1}(k_k<k_1), \tag{16}$$

Pr_1 represents the probability that $\mathbf{x}_1$ is the only maximum number in $\mathbf{X}_i$. Pr_2 denotes the probability that there exists more than one maximum number and correctly obtains a_1 by randomly choosing from them.

Here we present a generalized representation. As shown in Equation 14, $\Pr(a_1|\mathbf{P}_i)$ only includes the cases where $\mathbf{X}_{i,1}$ is the maximum value (maybe not the only one). Therefore, for a certain occurrence number $\mathbf{X}_i=(\mathbf{x}_1,\ldots,\mathbf{x}_m)$ of each probable answer $\{a_1,\ldots,a_m\}$, we can reorder $a_2,\ldots,a_m$ to obtain $\mathbf{x}_1=\mathbf{x}_2=\cdots=\mathbf{x}_l>\mathbf{x}_{l+1},\mathbf{x}_{l+2},\ldots,\mathbf{x}_m$, where $1\le l\le m$. When $l=1$, $\mathbf{x}_1$ is the only maximum value. So each term in $\Pr(a_1|\mathbf{P}_i;N)$ can be written as

$$\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^m k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^m(k_j!)}p_1^kp_2^k\cdots p_l^kp_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}, \tag{17}$$

where $k_1=k_2=\cdots=k_l=k>k_j$, $j=l+1,...,m$ and $lk+\sum_{j=l+1}^m k_j=N$.

Now we prove Theorem 1 and 2. We aim to prove that given the set of answers $\{a_1,a_2,\ldots,a_m\}$ with associated probabilities $\{p_1,p_2,\ldots,p_m\}$ from $\mathbf{P}_i$, we have $\Pr(a_1|\mathbf{P}_i;N+1)\ge\Pr(a_1|\mathbf{P}_i;N)$ for any $N\in\mathbb{N}^+$. Due to $\sum_{j=1}^m p_j=1$, the given proposition can be restated as

$$\Pr(a_1|\mathbf{P}_i;N+1)-(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)\ge 0. \tag{18}$$

We consider the probability term $p_1^kp_2^k\cdots p_l^kp_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}$ in each term in $\Pr(a_1|\mathbf{P}_i;N)$, *i.e*, Equation 17. When it times $\sum_{j=1}^m p_j$, there will be three cases.

Case 1: When it times p_1, $\mathbf{x}_1$ is the only maximum value. The probability term becomes
$p_1^{k+1}p_2^k\cdots p_l^kp_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}$,

Case 2: When it times p_s, where $s\in\{2,...,l\}$, $\mathbf{x}_s$ become the only maximum value. In this situation, its final result would be an incorrect answer. If $l=1$, case 2 will not exist. The probability term becomes
$p_1^kp_2^k\cdots p_{s-1}^kp_s^{k+1}p_{s+1}^k\cdots p_l^kp_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}$

Case 3: When it times p_t, where $t\in\{l+1,...,m\}$, the value of $\mathbf{x}_t$ changes from k_t to k_t+1. If $k_t=k-1$, $\mathbf{x}_t$ becomes a new maximum value. If $l=m$, case 3 will not exist. The probability term becomes
$p_1^kp_2^k\cdots p_l^kp_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t+1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}$.
It can be seen that case 1 and case 3 are also

present in $\Pr(a_1|\mathbf{P}_i;N+1)$, whereas case 2 does not. We begin by considering case 1 and case 2. The terms in $\Pr(a_1|\mathbf{P}_i;N+1)$ corresponding to case 1 $p_1^{k+1}p_2^k\cdots p_l^k p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}$ are shown in Equation 19, and no term in $\Pr(a_1|\mathbf{P}_i;N+1)$ involves case 2. The terms in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$ involving case 1 are shown in Equation 20. The terms in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$ involving case 2 $p_1^k p_2^k\cdots p_{s-1}^k p_s^{k+1}p_{s+1}^k\cdots p_l^k p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}$ are shown in Equation 21. Based on the fact of Equation 22, we can establish the inequality Equation 23, *i.e.*, the terms corresponding to case 1 and case 2 in $\Pr(a_1|\mathbf{P}_i;N+1)$ are greater than or equal to those in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$.

Now we consider case 3 $p_1^k p_2^k\cdots p_l^k p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t+1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}$, which can be analyzed by splitting it into two distinct scenarios: $k_t+1<k$ and $k_t+1=k$.

For scenario $k_t+1<k$, the terms in $\Pr(a_1|\mathbf{P}_i;N+1)$ corresponding to case 3 are shown in Equation 24. The terms in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$ corresponding to case 3 are shown in Equation 25. Evidently, we can obtain Equation 26 similar to Equation 22, and then we can get Equation 27, which proves the terms corresponding to the scenario $k_t+1<k$ in $\Pr(a_1|\mathbf{P}_i;N+1)$ are equal to those in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$.

For scenario $k_t+1=k$, in a similar manner, according to the Equation 28, we obtain the same result as the above scenario, *i.e.*, the terms corresponding to the scenario $k_t+1=k$ in $\Pr(a_1|\mathbf{P}_i;N+1)$ are equal to those in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$.

Thus far, let us revisit the proof steps. In the first step, we expand the expression $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$ and divide it into three cases, where case 1 and case 3 are present in $\Pr(a_1|\mathbf{P}_i;N+1)$ whereas case 2 does not. It has been proven that the coefficients of the terms in $\Pr(a_1|\mathbf{P}_i;N+1)$, where case 3 appears as the probability term part, are identical to those in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$. Consequently, these terms cancel out in the expression $\Pr(a_1|\mathbf{P}_i;N+1)-(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$. However, case 2 is not present in $\Pr(a_1|\mathbf{P}_i;N+1)$, which implies that the terms in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$, where case 2 appears as the probability term part, cannot be combined with any terms in $\Pr(a_1|\mathbf{P}_i;N+1)$ by extracting the exponent and performing subtraction on the coefficients like the terms containing case 3. It is fortunate that the terms in $\Pr(a_1|\mathbf{P}_i;N+1)$, where case 1 appears as the probability term part, cancel out with the corresponding terms in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$ which share the same probability terms and the remaining terms have coefficients identical to those of the terms in $(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)$, where case 2 appears as the probability terms. Therefore, these terms can be combined by factoring out the shared coefficients and partial probability terms. The remaining part after factoring out the common factor is $\sum_{s=2}^l(p_1-p_s)$. It is undeniable that $p_1\geq p_s$ when x is an easy question or moderate question for $\mathbf{P}_i$, therefore $\Pr(a_1|\mathbf{P}_i;N+1)-(\sum_{j=1}^m p_j)\cdot\Pr(a_1|\mathbf{P}_i;N)\geq 0$. Namely, $\Pr(a_1|\mathbf{P}_i;N)$ is strictly non-decreasing *w.r.t.* N if x is an easy or moderate question.

For sufficiently large N, the strong law of large numbers implies that $\Pr(\lim\limits_{N\to+\infty}\mathbf{x}_j/N=p_j)=1$. When x is an easy question, $p_1>p_j, \mathbf{x}_1/N>\mathbf{x}_j/N$ for $j=2,3,...,m$. As N is sufficiently large, it is sure that $\mathbf{x}_1$ is the only maximum value, making the final result must be the correct answer a_1. Therefore, if x is an easy question, N, $\lim\limits_{N\to+\infty}\Pr(a_1|\mathbf{P}_i;N)=1$. If x is a moderate question, there are $|\mathcal{S}|$ equivalent answers in the probability sense, whose probabilities are all the maximum value. Therefore, $\lim\limits_{N\to+\infty}\Pr(a_1|\mathbf{P}_i;N)=1/|\mathcal{S}|$. Similarly, if x is a hard question, the maximum probability is not p_1, the final result must be a wrong answer, so $\lim\limits_{N\to+\infty}\Pr(a_1|\mathbf{P}_i;N)=0$. Theorem 1 to 3 is proved.

Lemma 1. Consider a specific condition with answer space $|\mathcal{A}|=3$. For $N=3$, $\Pr(a_1|\mathbf{P}_i;N)=3p_{i,1}^2-2p_{i,1}^3+2p_{i,1}p_{i,2}p_{i,3}$. For $N=5$, $\Pr(a_1|\mathbf{P}_i;N)=6p_{i,1}^5-15p_{i,1}^4+10p_{i,1}^3+15p_{i,1}^2p_{i,2}p_{i,3}(p_{i,2}+p_{i,3})$.

Proof. For $N=3$, we can calculate $\Pr(a_1|\mathbf{P}_i;N)$ with Equation 14 as follows:

$$\begin{aligned}\Pr(a_1|\mathbf{P}_i;N=3)=&\binom{3}{3}p_{i,1}^3+\binom{3}{2}p_{i,1}^2(1-p_{i,1})\\&+A(3,1)\,p_{i,1}p_{i,2}p_{i,3}\\=&\,3p_{i,1}^2-2p_{i,1}^3+2p_{i,1}p_{i,2}p_{i,3}\,,\end{aligned}\tag{29}$$

where $A(n,k)$ is the permutation number formula

$$\frac{\left(lk+1+\sum_{j=l+1}^{m}k_j\right)!}{(k+1)\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k+1}p_2^k\cdots p_l^k p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m} \tag{19}$$

$$\begin{aligned}
&p_1\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^k p_2^k\cdots p_l^k p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}\\
&+\sum_{s=2}^{l}p_s\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k+1}{k}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k+1}p_2^k\cdots p_{s-1}^k p_s^{k-1}p_{s+1}^k\cdots p_l^k p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}\\
&+\sum_{t=l+1}^{m}p_t\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k+1}{k_t}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k+1}p_2^k\cdots p_l^k p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t-1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}
\end{aligned} \tag{20}$$

$$\sum_{s=2}^{l}p_s\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^k p_2^k\cdots p_{s-1}^k p_s^k p_{s+1}^k\cdots p_l^k p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m} \tag{21}$$

$$\begin{aligned}
\frac{\left(lk+1+\sum_{j=l+1}^{m}k_j\right)!}{(k+1)\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)} &= \frac{\left[(k+1)+(l-1)k+\sum_{t=l+1}^{m}k_t\right](lk+\sum_{j=l+1}^{m}k_j\Big)!}{(k+1)\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}\\
&=\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}+(l-1)\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k+1}{k}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}\\
&+\sum_{t=l+1}^{m}\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k+1}{k_t}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}+(l-1)\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}
\end{aligned} \tag{22}$$

$A(n,k)=\frac{n!}{(n-k)!}$. For $N=5$, we can get

$$\begin{aligned}
\Pr(a_1|\mathbf{P}_i;N=5) &= \binom{5}{5}p_{i,1}^5+\binom{5}{4}p_{i,1}^4(1-p_{i,1})+\\
&\binom{5}{3}p_{i,1}^3(1-p_{i,1})^2+\binom{5}{2}p_{i,1}^2\binom{3}{2}(p_{i,2}^2p_{i,3}+p_{i,3}^2p_{i,2})\\
&=6p_{i,1}^5-15p_{i,1}^4+10p_{i,1}^3+15p_{i,1}^2p_{i,2}p_{i,3}(p_{i,2}+p_{i,3}).
\end{aligned} \tag{30}$$

Lemma 1 is proved.

Theorem 4. For two prompting strategies $\mathbf{P}_i$ and $\mathbf{P}_{i'}$, note $p_{i,q}=\max\{p_{i,2},\ldots,p_{i,m}\}$, $p_{i',q'}=\max\{p_{i',2},\ldots,p_{i',m}\}$, if $p_{i,1}-p_{i,q}<p_{i',1}-p_{i',q'}$ and $p_{i,1}+p_{i,q}-p_{i,1}^2-p_{i,q}^2>p_{i',1}+p_{i',q'}-p_{i',1}^2-p_{i',q'}^2$, there exits a sufficiently large N_0 such that for $N>N_0$, $\Pr(a_1|\mathbf{P}_i;N)<\Pr(a_1|\mathbf{P}_{i'};N)$.

Proof. According to Khinchin's Law of Large Numbers and Lindeberg-Levy Central Limit Theorem, when N is sufficiently large, each X_i can be approximated by a normal distribution. Specifically, for each $\mathbf{x}_{i,j}$, we have $\mathbf{x}_{i,j}\sim\mathcal{N}(Np_{i,j},Np_{i,j}(1-p_{i,j}))$. Note $M_i=\max(\mathbf{x}_{i,2},...,\mathbf{x}_{i,m})$ and $p_{i,q}=\max\{p_{i,2},\ldots,p_{i,m}\}$, the distribution of M_i can be approximated by $M_i\sim\mathcal{N}(Np_{i,n},Np_{i,1}(1-p_{i,1}))$. So $\mathbf{x}_{i,j}-M_i$ obey the normal distribution $\mathcal{N}(N(p_{i,1}-p_{i,q}),N(p_{i,1}(1-p_{i,1})+p_{i,q}(1-p_{i,q})))$. Thus, we can get Equation 31:

$$
\begin{aligned}
&\frac{\left(lk+1+\sum_{j=l+1}^{m}k_j\right)!}{(k+1)\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k+1}p_2^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}\\
&-p_1\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k}p_2^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}\\
&-\sum_{s=2}^{l}p_s\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k+1}{k}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k+1}p_2^{k}\cdots p_{s-1}^{k}p_s^{k-1}p_{s+1}^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}\\
&-\sum_{t=l+1}^{m}p_t\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k+1}{k_t}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k+1}p_2^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t-1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}\\
&-\sum_{s=2}^{l}p_s\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k}p_2^{k}\cdots p_{s-1}^{k}p_s^{k}p_{s+1}^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}\\
&=\sum_{s=2}^{l}\left[\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k}p_2^{k}\cdots p_{s-1}^{k}p_s^{k}p_{s+1}^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}\right]\cdot(p_1-p_s)\geq 0
\end{aligned}
\tag{23}
$$

$$
\frac{1}{l}\cdot\frac{\left(lk+1+\sum_{j=l+1}^{m}k_j\right)!}{(k_t+1)\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k}p_2^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t+1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}
\tag{24}
$$

$$
\begin{aligned}
&\sum_{s=2}^{l}p_s\cdot\frac{1}{l-1}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k_t+1}{k}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k}p_2^{k}\cdots p_{s-1}^{k}p_s^{k-1}p_{s+1}^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t+1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}\\
&+\sum_{r=l+1,r\neq t}^{m}p_r\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k_t+1}{k_r}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k}p_2^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{r-1}^{k_{r-1}}p_r^{k_r-1}p_{r+1}^{k_{r+1}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t+1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}\\
&+p_t\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^{k}p_2^{k}\cdots p_l^{k}p_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}
\end{aligned}
\tag{25}
$$

$$
\begin{aligned}
\frac{1}{l}\cdot\frac{\left(lk+1+\sum_{j=l+1}^{m}k_j\right)!}{(k_t+1)\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}=&\sum_{s=2}^{l}\frac{1}{l-1}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k_t+1}{k}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}\\
&+\sum_{r=l+1,r\neq t}^{m}p_r\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k_t+1}{k_r}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}\\
&+\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}
\end{aligned}
\tag{26}
$$

$$
\begin{aligned}
&\frac{1}{l}\cdot\frac{\left(lk+1+\sum_{j=l+1}^{m}k_j\right)!}{(k_t+1)\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^kp_2^k\cdots p_l^kp_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t+1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}\\
&-\sum_{s=2}^{l}p_s\cdot\frac{1}{l-1}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k_t+1}{k}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^kp_2^k\cdots p_{s-1}^kp_s^{k-1}p_{s+1}^k\cdots p_l^kp_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t+1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}\\
&-\sum_{r=l+1,r\neq t}^{m}p_r\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k_t+1}{k_r}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^kp_2^k\cdots p_l^kp_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_{r-1}^{k_{r-1}}p_r^{k_r-1}p_{r+1}^{k_{r+1}}\cdots p_{t-1}^{k_{t-1}}p_t^{k_t+1}p_{t+1}^{k_{t+1}}\cdots p_m^{k_m}\\
&-p_t\cdot\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}p_1^kp_2^k\cdots p_l^kp_{l+1}^{k_{l+1}}p_{l+2}^{k_{l+2}}\cdots p_m^{k_m}=0
\end{aligned}
\tag{27}
$$

$$
\begin{aligned}
\frac{1}{l+1}\cdot\frac{\left(lk+1+\sum_{j=l+1}^{m}k_j\right)!}{(k_t+1)\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}=&\sum_{s=2}^{l}\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}\\
&+\sum_{r=l+1,r\neq t}^{m}p_r\cdot\frac{1}{l+1}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{\frac{k_t+1}{k_r}\cdot(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}\\
&+\frac{1}{l}\cdot\frac{\left(lk+\sum_{j=l+1}^{m}k_j\right)!}{(k!)^l\cdot\prod_{j=l+1}^{m}(k_j!)}
\end{aligned}
\tag{28}
$$

Table 4: **Difficulty proportion of questions and extreme peformance (denote by "Acc") for each P_i and LLM across benchmarks.** CoT has more easy questions and fewer hard questions, and can reach the best extreme performance on all LLMs.

$\mathbf{P}_i$	Easy	Moderate	Hard	Acc	Easy	Moderate	Hard	Acc	Easy	Moderate	Hard	Acc
	Phi-3.5-mini-Instruct				**Gemini-1.5-Flash**				**GPT-4o-mini**			
DiP	78.4%	0.6%	21.1%	78.6	91.0%	0.0%	9.0%	91.0	89.7%	0.4%	**9.9%**	89.9
CoT	**81.2%**	0.4%	**18.4%**	**81.4**	**91.2%**	0.2%	**8.6%**	**91.3**	**89.8%**	0.3%	**9.9%**	**90.0**
L2M	80.2%	0.6%	19.2%	80.5	90.9%	0.2%	89.8%	90.9	**89.8%**	0.3%	10.0%	89.9
SBP	79.0%	0.6%	20.4%	79.3	90.6%	0.4%	9.0%	90.8	81.4%	0.2%	10.4%	89.5
AnP	77.0%	1.2%	21.8%	77.6	80.5%	0.6%	18.8%	90.9	81.4%	1.1%	17.5%	81.9

$$
\begin{aligned}
&\Pr(\mathbf{x}_{i,1}>M_i)\\
&=\Pr(\mathbf{x}_{i,1}-M_i>0)\\
&=1-\Phi(f(p_{i,1},p_{i,q},N)),\\
&\Phi(f(p_{i,1},p_{i,q},N))=\\
&\Phi(\sqrt{\frac{N}{p_{i,1}(1-p_{i,1})+p_{i,q}(1-p_{i,q})}}(p_{i,q}-p_{i,1})),
\end{aligned}
\tag{31}
$$

where Φ is the standard normal cumulative distribution function. This also holds for any other $\Pr(\mathbf{x}_{i',1}>M_i')$. If $p_{i,1}-p_{i,q}<p_{i',1}-p_{i',q'}$ and $p_{i,1}+p_{i,q}-p_{i,1}^2-p_{i,q}^2>p_{i',1}+p_{i',q'}-p_{i',1}^2-p_{i',q'}^2$, we can get $p_{i,q}-p_{i,1}>p_{i',q}-p_{i',1}$ and $p_{i,1}(1-p_{i,1})+p_{i,q}(1-p_{i,q})<p_{i',1}(1-p_{i',1})+p_{i',q'}(1-p_{i',q'})$, and $\Phi(f(p_{i,1},p_{i,q},N))>\Phi(f(p_{i',1},p_{i',q'},N))$. So there exists a large N_0 such that for $N>N_0$, $\Pr(\mathbf{x}_{i',1}>M_i')>\Pr(\mathbf{x}_{i,1}>M_i)$, *i.e.*, $\Pr(a_1|\mathbf{P}_i;N)>\Pr(a_1|\mathbf{P}_{i'};N)$. Theorem 4 is proved.

C Detailed Results

Here we display the scaling performance of different prompting strategies on each LLM and bench-

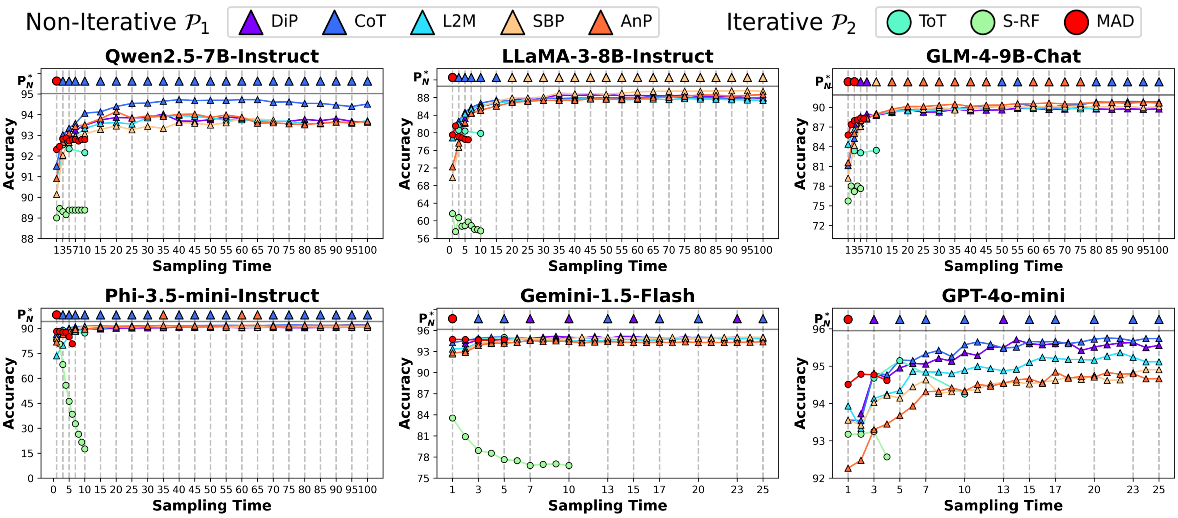

Figure 8: Performance of each prompting strategy under given sampling time N on GSM8K.

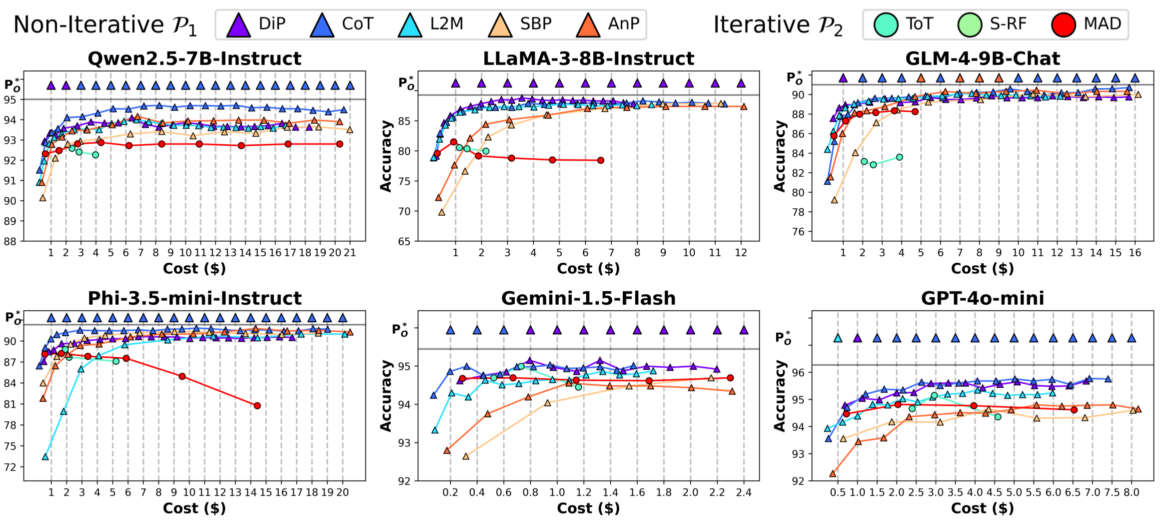

Figure 9: Performance of each prompting strategy under given cost O on GSM8K.

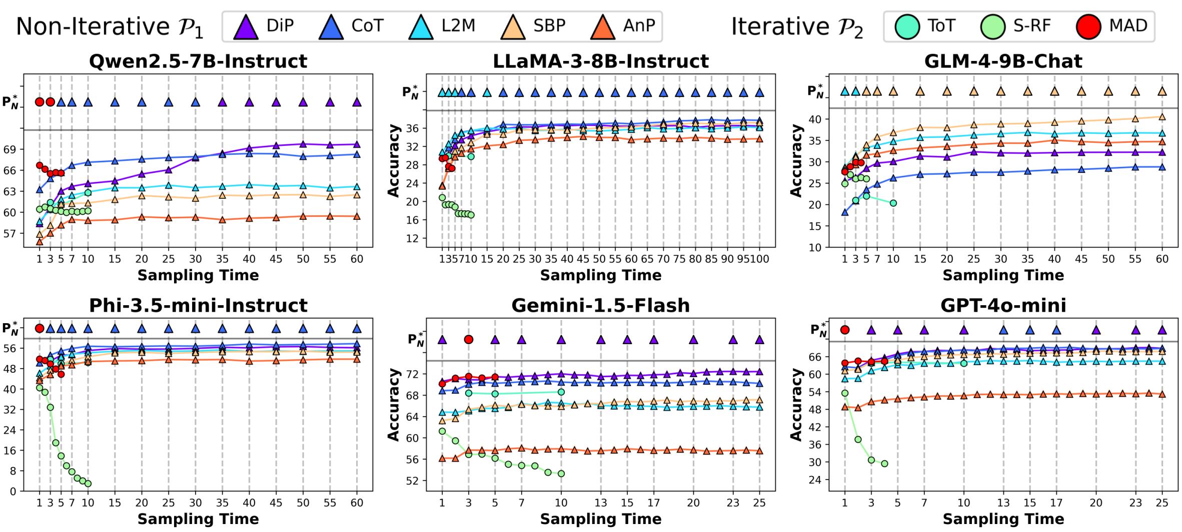

Figure 10: Performance of each prompting strategy under given sampling time N on GSM-Hard.

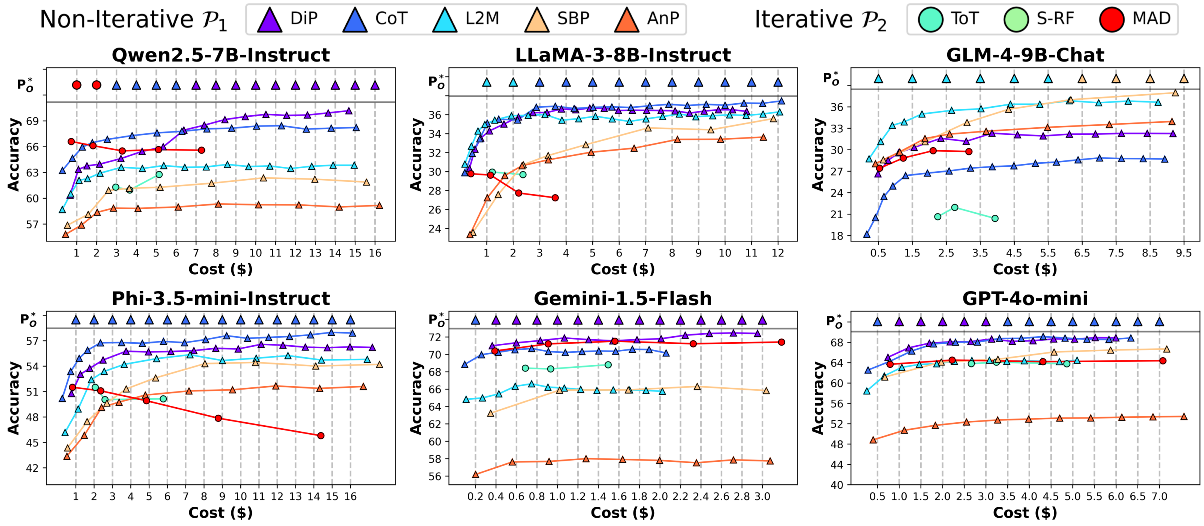

Figure 11: Performance of each prompting strategy under given cost O on GSM-Hard.

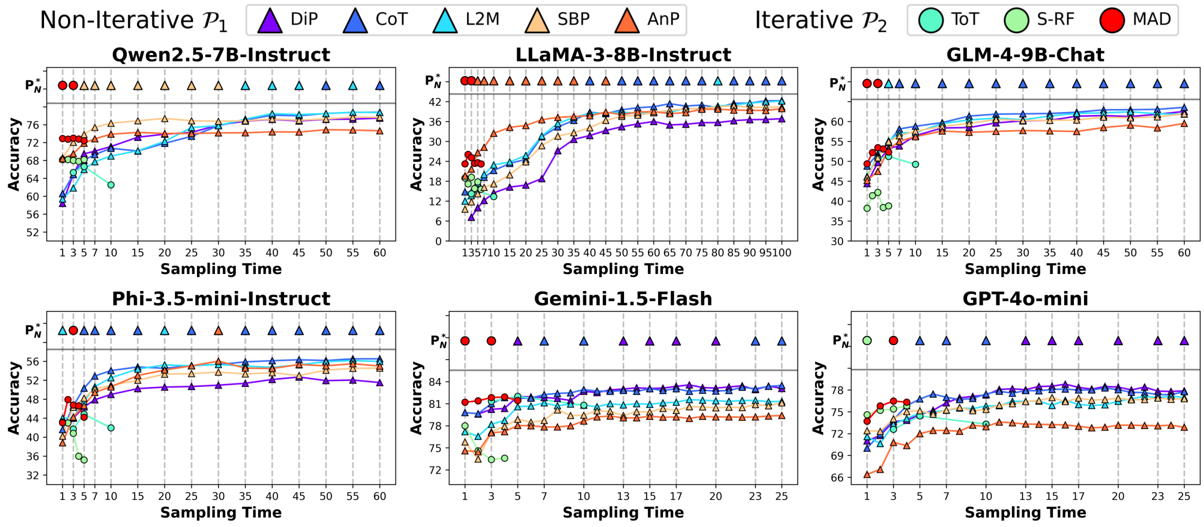

Figure 12: Performance of each prompting strategy under given sampling time N on MATH.

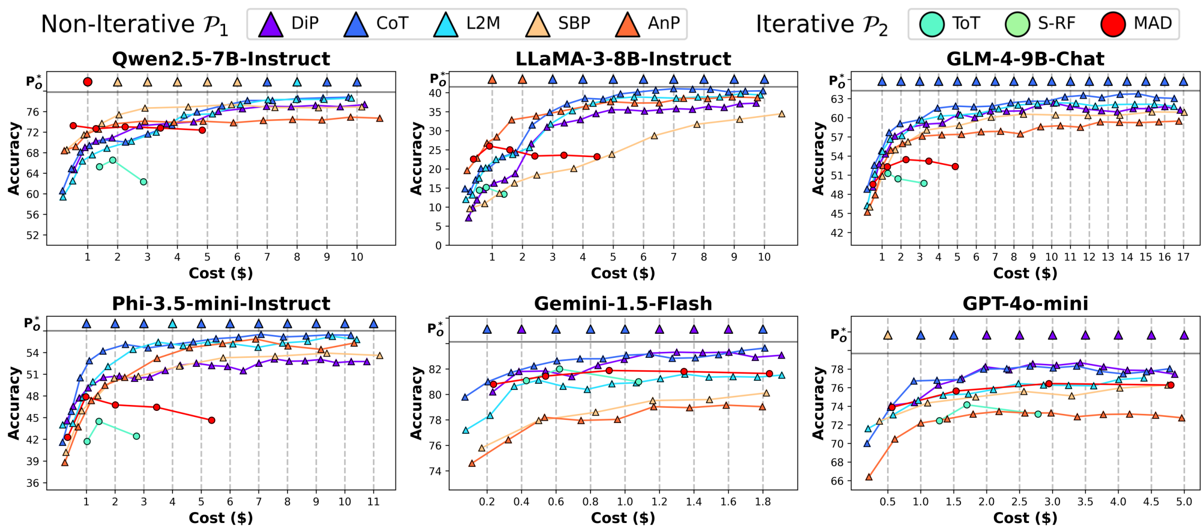

Figure 13: Performance of each prompting strategy under given cost O on MATH.

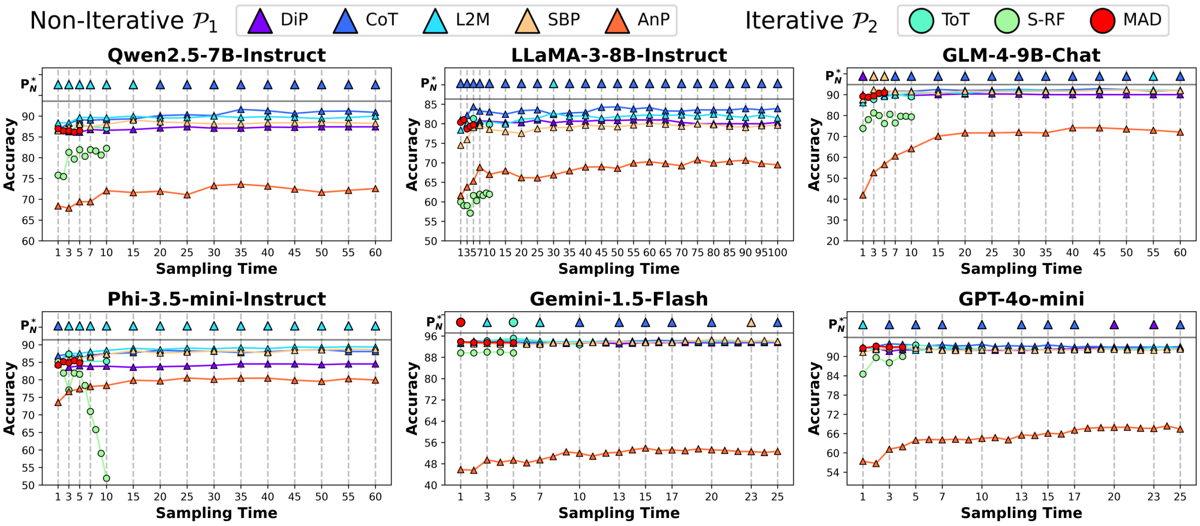

Figure 14: Performance of each prompting strategy under given sampling time N on MMLU.

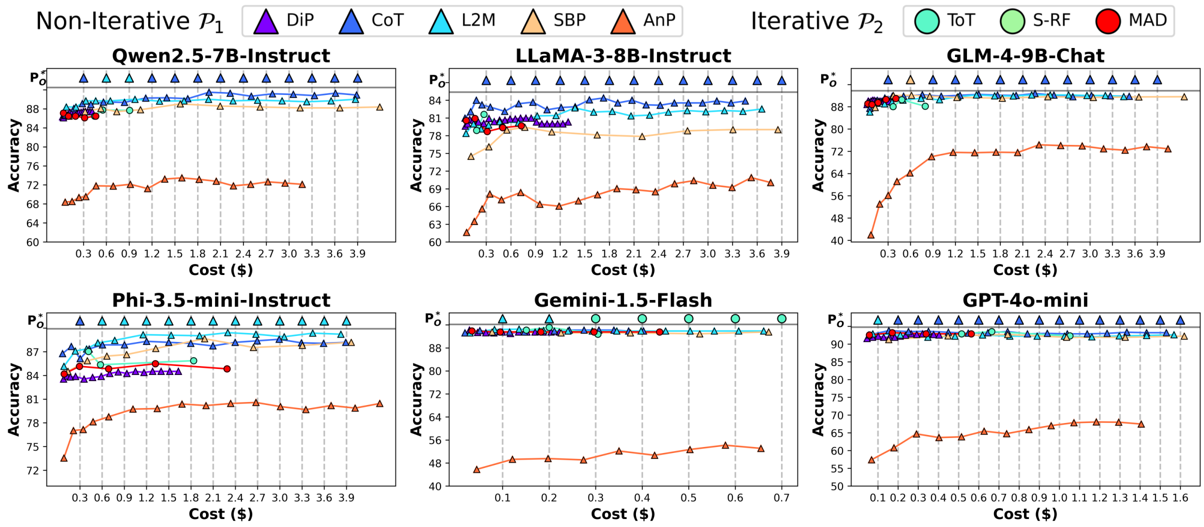

Figure 15: Performance of each prompting strategy under given cost O on MMLU.

mark under given sampling time N and cost O, as shown in Figures 8 to 15. We find that, aside from CoT, DiP also exhibits superior performance compared to other complex prompting strategies on certain models and datasets, *e.g.*, Gemini-1.5-Flash on MATH. This also comes from the two reasons, *i.e.*, DiP has more hard questions and easy questions and a flat probability distribution of wrong answers on the specific dataset. This phenomenon is particularly prominent on powerful LLMs such as Gemini-1.5-Flash on GSM8K and GSM-Hard, where DiP and CoT exhibit comparable performance. Almost 83% of results satisfy that CoT or DiP performs best as significantly scaling. Besides, this trend is also observed on other prompting strategies on few datasets and models. This encourages us to fully unleash the potential of simple prompting strategies, and indicates that the scaling performance does not only depend on the prompting strategies' pass@1 accuracy.

D More Discussions on Improving the Scaling Performance

In this section, we will discuss more about our further exploration on the two ways to improve the scaling performance. We display more results of (1) adaptively scaling based on the question difficulty, (2) dynamically choosing the optimal $\mathbf{P}_i$ and (3) combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ in Section D.1, D.2 and D.3, respectively.

D.1 Adaptively Scaling Based on the Difficulty

We use the following prompt to force the LLM to determine if the question is hard for given $\mathbf{P}_i$.

Question:
{question}

Using the method #{method}# to solve the question:
{description}

If the method is more likely to get the right answer, the question is easy. Otherwise, if the method is more likely to get the wrong answer, the question is hard. Please determine the difficulty of the question for the used method, and answer in the following JSON format.
{"Difficulty": "Easy or Hard", "Reason": ""}

Figures 16 to 20 report the results of each prompting strategy when adaptively scaling based on the question difficulty. Our experiment results show that LLMs cannot accurately judge the difficulty of the input question most of the time, thus even leading to reduced performance. Nevertheless, this method is theoretically capable of enhancing the scaling performance, thereby motivating us to explore other approaches to accurately assess the question difficulty.

D.2 Dynamically Choosing the Optimal $\mathbf{P}_i$

Figures 21 to 25 display the results on GSM8K on LLaMA-3-8B-Instruct, GLM-4-9B-Chat, Phi-3.5-mini-Instruct, Gemini-1.5-Flash and GPT-4o-mini, respectively. It can be observed that all LLMs tend to believe that CoT is the best prompting strategy, while CoT does not excel at every question. With oracles to provide the optimal $\mathbf{P}_i$ labels, all LLMs demonstrate significant performance improvements, even only with one sampling time, proving the enormous potential of this method. we will explore how to approach this upper bound in the future.

D.3 Combining Adaptively Scaling and Dynamically Choosing the Optimal $\mathbf{P}_i$

Figures 26 to 30 display the results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on GSM8K on LLaMA-3-8B-Instruct, GLM-4-9B-Chat, Phi-3.5-mini-Instruct, Gemini-1.5-Flash, and GPT-4o-mini, respectively. Figures 31 to 36 show the results on each LLM on MATH, respectively. Extensive experiments demonstrate the general effectiveness and superiority of this method, which has an extremely high upper bound.

E Implementation Details and Prompts

We use vllm (Kwon et al., 2023) to deploy open-sourced LLMs, with top-p = 0.9 and temperature = 0.7. For close-sourced LLMs, we use their APIs with default settings. We set the content safety detection threshold of Gemini-1.5-Flash to zero to prevent erroneous judgments that may result in null outputs.

Following (Wang et al., 2024; Lightman et al., 2024; Qi et al., 2025), we use MATH-500, a subset of representative problems from the MATH dataset to speed up the evaluation. We use the test split of each dataset. The license for all datasets is CC-BY 4.0 or others for open academic research. The number of samples on each dataset is shown in Table 5. We ensure our use of existing artifacts is aligned with their intended purposes. All of them are public English datasets for academic research. On GSM8K and GSM-Hard, we use the same 1-shot prompt in the original paper of Least-to-Most (Zhou et al., 2023) shown in Figure 39 and Figure 40. On other datasets, we use the 0-shot prompt shown in Figure 41. We use the same prompt in Analogous Prompting (Yasunaga et al., 2024), and guide the LLM to recall one analogous problem. We use the same 1-shot prompt in Step-Back Prompting (Zheng et al., 2024) on MMLU, and apply their prompt designed for reasoning tasks on other datasets. We use the same prompt in 0-shot Chain-of-Thought (Kojima et al., 2022), Multi-Agent Debate (Du et al., 2024) and Self-Refine (Huang et al., 2024) on all datasets. The prompts are shown in Figures 37 to 46.

Table 5: The number of samples in each dataset.

Dataset	Samples
GSM8K	1318
GSM-Hard	1318
MATH-500	500
MMLU-Biology	310
MMLU-Chemistry	203
MMLU-Physics	151

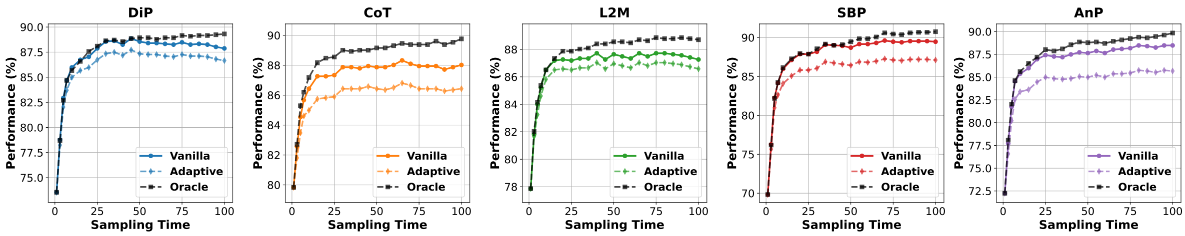

Figure 16: Results of adaptively scaling based on the question difficulty on Llama-3-8B-Instruct on GSM8K.

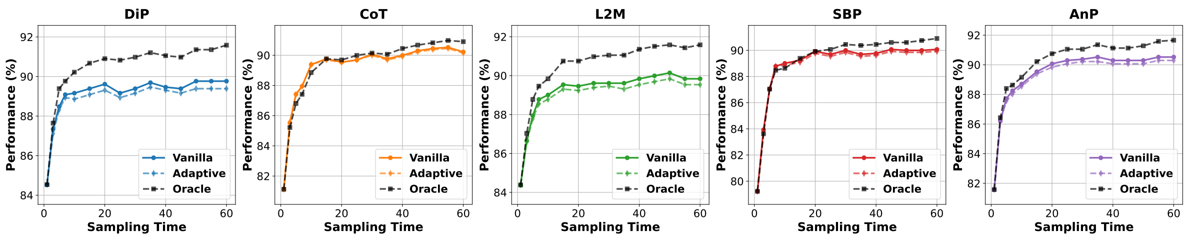

Figure 17: Results of adaptively scaling based on the question difficulty on GLM-4-9B-Chat on GSM8K.

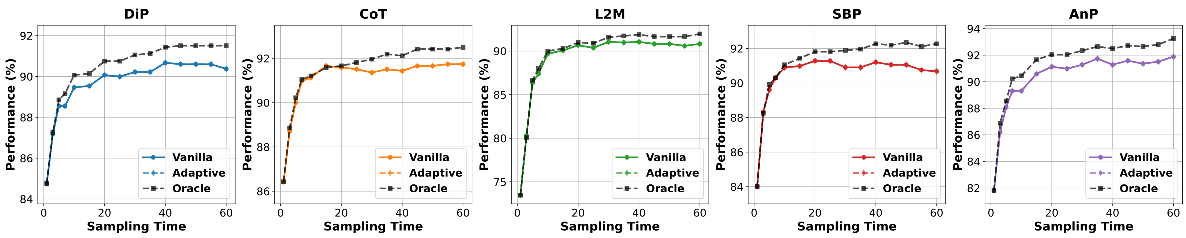

Figure 18: Results of adaptively scaling based on the question difficulty on Phi-3.5-mini-Instruct on GSM8K.

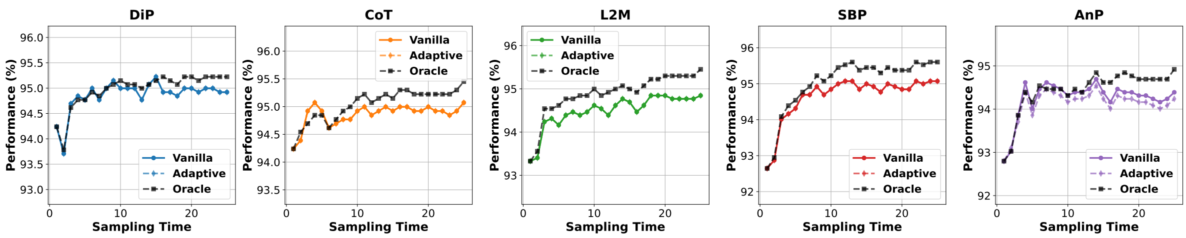

Figure 19: Results of adaptively scaling based on the question difficulty on Gemini-1.5-Flash on GSM8K.

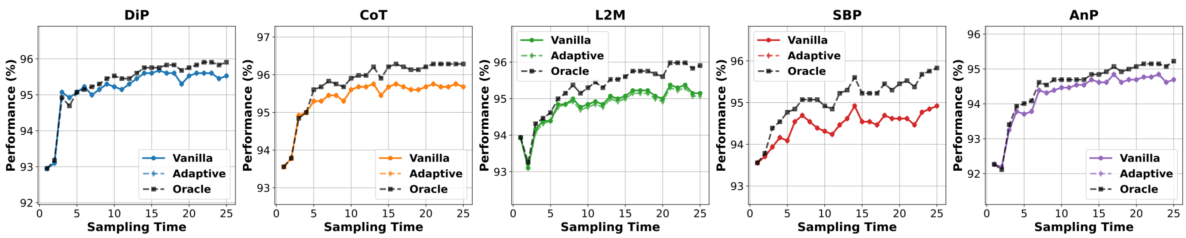

Figure 20: Results of adaptively scaling based on the question difficulty on GPT-4o-mini on GSM8K.

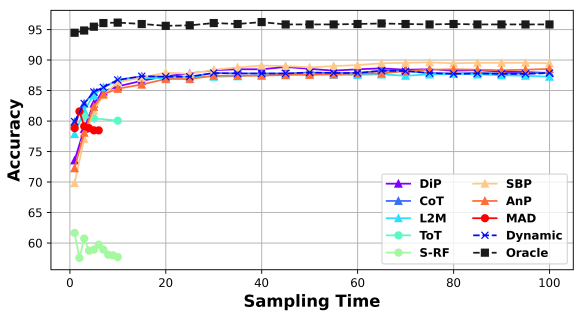

Figure 21: Results of dynamically choosing the optimal $\mathbf{P}_i$ on LLaMA-3-8B-Instruct on GSM8K.

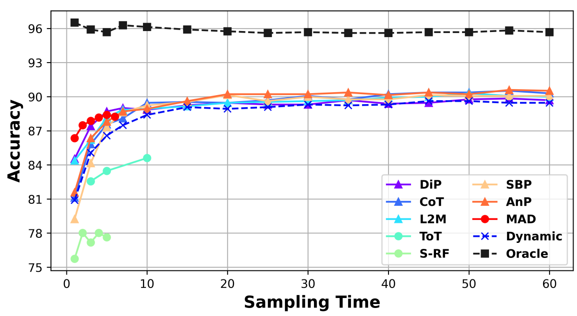

Figure 22: Results of dynamically choosing the optimal $\mathbf{P}_i$ on GLM-9B-Chat on GSM8K.

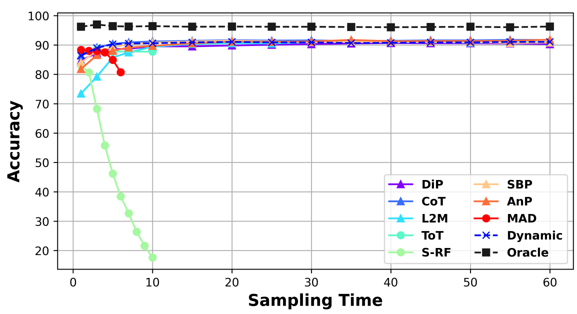

Figure 23: Results of dynamically choosing the optimal $\mathbf{P}_i$ on Phi-3.5-mini-Instruct on GSM8K.

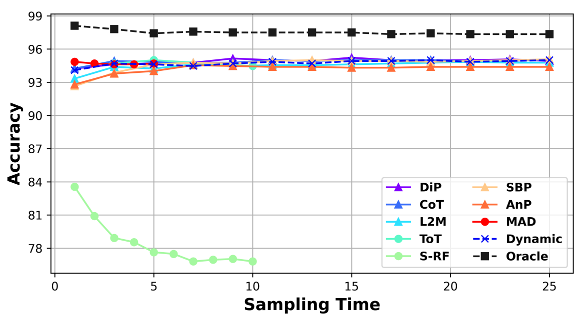

Figure 24: Results of dynamically choosing the optimal $\mathbf{P}_i$ on Gemini-1.5-Flash on GSM8K.

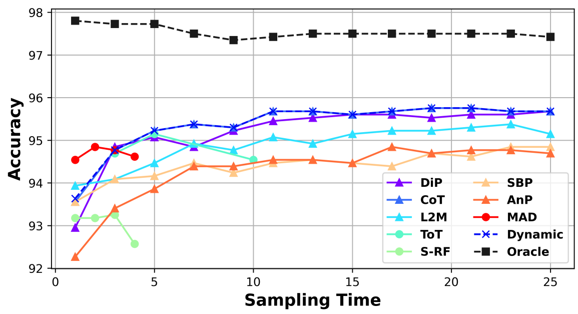

Figure 25: Results of dynamically choosing the optimal $\mathbf{P}_i$ on GPT-4o-mini on GSM8K.

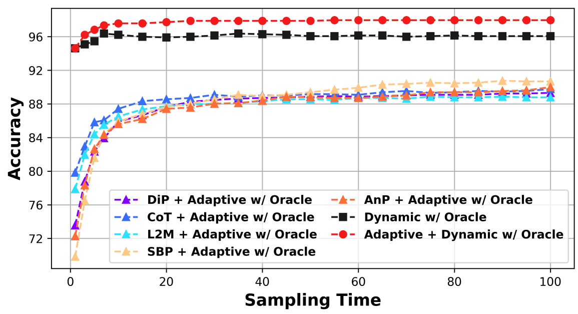

Figure 26: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on LLaMA-3-8B-Instruct on GSM8K.

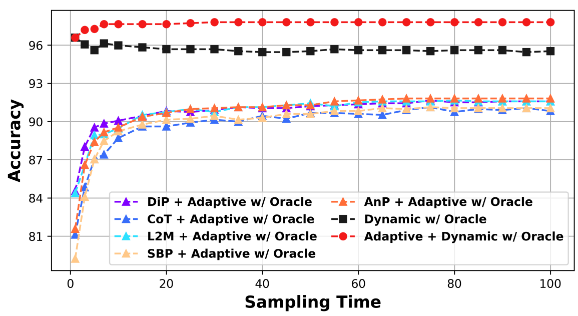

Figure 27: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on GLM4-9B-Chat on GSM8K.

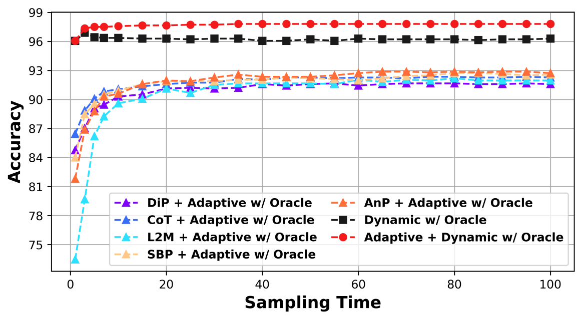

Figure 28: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on Phi-3.5-mini-Instruct on GSM8K.

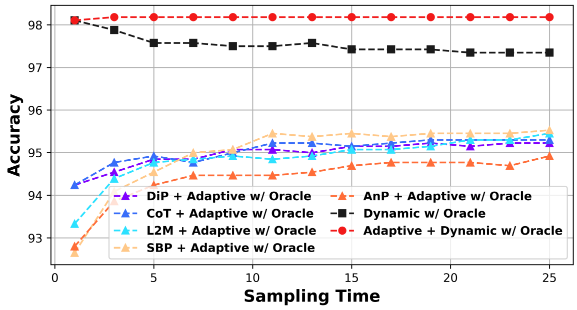

Figure 29: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on Gemini-1.5-Flash on GSM8K.

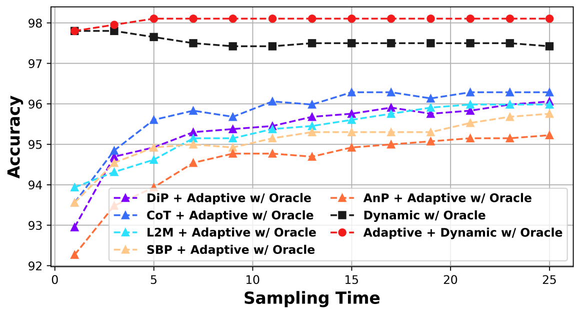

Figure 30: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on GPT-4o-mini on GSM8K.

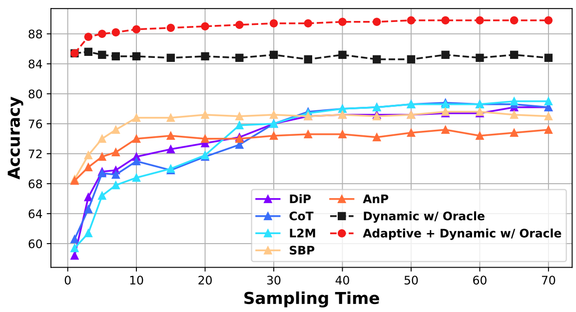

Figure 31: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on Qwen2.5-7B-Instruct on MATH.

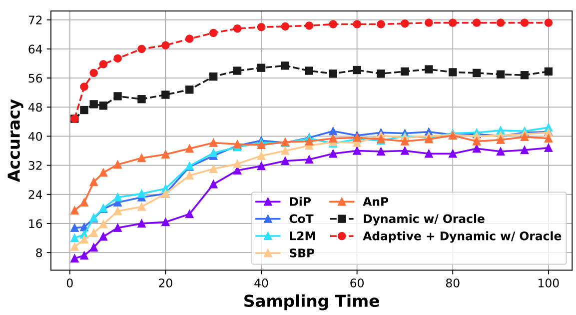

Figure 32: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on LLaMA-3-8B-Instruct on MATH.

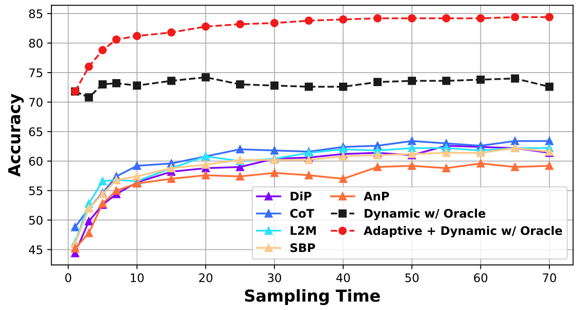

Figure 33: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on GLM-4-9B-Instruct on MATH.

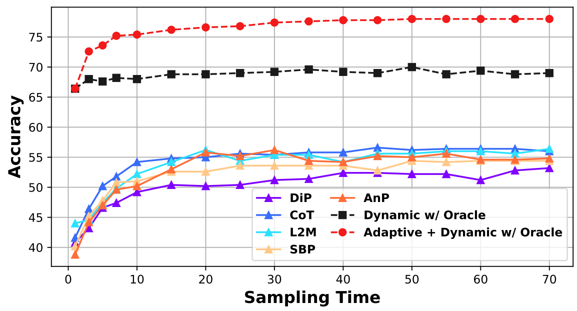

Figure 34: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on Phi-3.5-mini-Instruct on MATH.

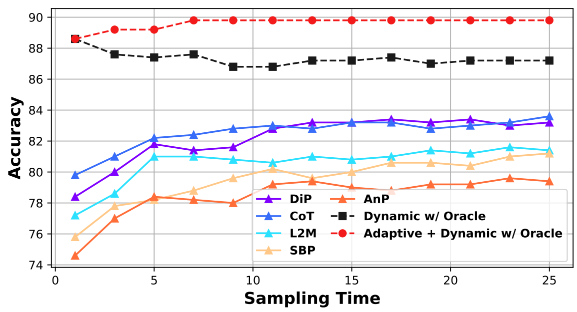

Figure 35: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on Gemini-1.5-Flash on MATH.

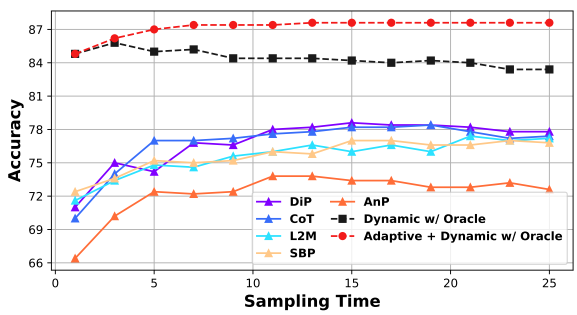

Figure 36: Results of combining adaptively scaling and dynamically choosing the optimal $\mathbf{P}_i$ on GPT-4o-mini on MATH.

Direct Prompting

User:
<question>

Assistant:
<answer>

Figure 37: Prompt of DiP.

Chain-of-Thought prompt

User:
Question:
<question>

Answer:
Let's think step by step.

Assistant:
<answer>

Figure 38: Prompt of CoT.

Least-to-Most prompt on GSM8K

User:
Question: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together?
Answer: Let's break down this problem: 1. How many apples does Anna have? 2. How many apples do they have together?
1. Anna has 2 more apples than Elsa. So Anna has 2 + 5 = 7 apples.
2. Elsa and Anna have 5 + 7 = 12 apples together.
The answer is: \\boxed{12}.

Question:
<question>
Answer:

Assistant:
<answer>

Figure 39: Prompt of L2M on GSM8K.

Least-to-Most prompt on GSM-Hard

User:
Question: Elsa has 524866 apples. Anna has 432343 more apples than Elsa. How many apples do they have together?
Answer: Let's break down this problem: 1. How many apples does Anna have? 2. How many apples do they have together?
1. Anna has 432343 more apples than Elsa. So Anna has 524866 + 432343 = 957209 apples.
2. Elsa and Anna have 524866 + 957209 = 1482075 apples together.
The answer is: \\boxed{1482075}.

Question:
<question>
Answer:

Assistant:
<answer>

Figure 40: Prompt of L2M on GSM-Hard.

Least-to-Most prompt

User:
In order to solve the question more conveniently and efficiently, break down the question into progressive sub-questions. Answer the sub-questions and get the final result according to sub-questions and their answers.

Question:
<question>
Answer:

Assistant:
<answer>

Figure 41: Prompt of L2M on MATH and MMLU.

Tree-of-Thoughts prompt

User:
Question:
<question>

Answer:
Let's think step by step.

Assistant:
<answer>

User:
Given the question and several solutions, decide which solution is the most promising. Analyze each solution in detail, then conclude in the last line "The index of the best solution is x", where x is the index number of the solution.

<Solution 1>
<Solution 2>
......

Figure 42: Prompt of ToT.

Self-Refine Prompt

User:
<question>

Assistant:
<previous answer>

User:
Review your previous answer and find problems with your answer.

Assistant:
<feedback>

User:
Based on the problems of your previous answer, improve your answer.

Assistant:
<revised answer>

Figure 43: Prompt of S-RF.

Step-Back Prompting

User:
You are an expert at <subject>. Your task is to extract the mathematics concepts and principles involved in solving the problem.

Question:
<question>

Principles involved:

Assistant:
<answer>

User:
You are an expert at <subject>. You are given a <subject> problem and a set of principles involved in solving the problem. Solve the problem step by step by following the principles.

Question:
<question>

Principles:
{principles}

Answer:

Assistant:
<answer>

Figure 44: Prompt of SBP.

Analogous Prompting

User:
Your task is to tackle mathematical problems. When presented with a math problem, recall relevant problems as examples. Afterward, proceed to solve the initial problem.

 # Initial Problem:
{question}

 # Instructions:
Relevant Problems:
Recall an example of the math problem that is relevant to the initial problem. Your problem should be distinct from the initial problem (e.g., involving different numbers and names). For the example problem:
- After "Q: ", describe the problem.
- After "A: ", explain the solution and enclose the ultimate answer in \\boxed{}.

Solve the Initial Problem:
Q: Copy and paste the initial problem here.
A: Explain the solution and enclose the ultimate answer in \\boxed{} here.

Assistant:
<answer>

Figure 45: Prompt of AnP.

Multi-Agent Debate

User:
<question>

Assistant:
<solving process>
<answer>

User:
These are the answers to the question from other agents:

One agent answer: ...
One agent answer: ...
...

Using the answers from other agents as additional information, can you provide your answer to the question?
<question>

Assistant:
<answer>

User:
...

Assistant:
...

Figure 46: Prompt of MAD.