AdaptThink: Reasoning Models Can Learn When to Think

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, Juanzi Li
Tsinghua University

Abstract

Recently, large reasoning models have achieved impressive performance on various tasks by employing human-like deep thinking. However, the lengthy thinking process substantially increases inference overhead, making efficiency a critical bottleneck. In this work, we first demonstrate that *NoThinking*, which prompts the reasoning model to skip thinking and directly generate the final solution, is a better choice for relatively simple tasks in terms of both performance and efficiency. Motivated by this, we propose *AdaptThink*, a novel RL algorithm to teach reasoning models to choose the optimal thinking mode adaptively based on problem difficulty. Specifically, *AdaptThink* features two core components: (1) a constrained optimization objective that encourages the model to choose *NoThinking* while maintaining the overall performance; (2) an importance sampling strategy that balances *Thinking* and *NoThinking* samples during on-policy training, thereby enabling cold start and allowing the model to explore and exploit both thinking modes throughout the training process. Our experiments indicate that *AdaptThink* significantly reduces the inference costs while further enhancing performance. Notably, on three math datasets, *AdaptThink* reduces the average response length of DeepSeek-R1-Distill-Qwen-1.5B by 53% and improves its accuracy by 2.4%, highlighting the promise of adaptive thinking-mode selection for optimizing the balance between reasoning quality and efficiency. Our codes and models are available at `https://github.com/THU-KEG/AdaptThink`.

1 Introduction

Recent advancements in large reasoning models, such as OpenAI o1(OpenAI, 2024) and DeepSeek-R1 (DeepSeek-AI, 2025), have demonstrated remarkable capabilities in tackling complex tasks. Given a problem, these models first engage in a long chain of thought—also referred to as *Thinking*—where they iteratively explore different approaches, accompanied by reflection, backtracking, and self-verification. Subsequently, they produce a final solution that contains only the correct steps and the answer to present to the user. While the long-thinking process markedly enhances the model's reasoning capacities, it also substantially increases inference overhead and latency (Qu et al., 2025; Sui et al., 2025). In particular, for some simple queries where users expect fast, near-instant responses, these models often generate excessive thinking with unnecessarily detailed steps or redundant attempts, resulting in a suboptimal user experience (Chen et al., 2024; Shen et al., 2025).

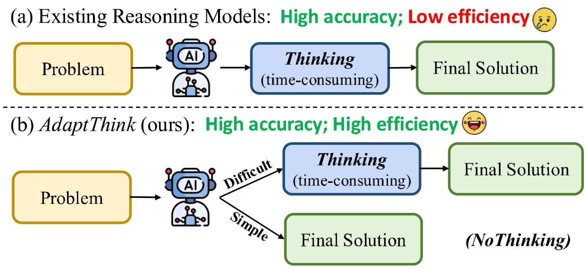

Figure 1: *AdaptThink* enables models to adaptively select between *Thinking* or *NoThinking* mode based on problem difficulty, thereby improving reasoning efficiency while further improving overall performance.

Existing efforts to improve reasoning efficiency primarily focus on reducing the length of model responses, either through incorporating length-based rewards in reinforcement learning (RL) (Arora and Zanette, 2025; Team et al., 2025), finetuning with preference pairs that penalizes longer responses (Chen et al., 2024; Shen et al., 2025; Luo et al., 2025a), or by merging reasoning and non-reasoning models (Wu et al., 2025). Nevertheless, these methods still apply thinking to all instances, regardless of whether thinking itself is necessary for every problem. In this work, we draw inspiration from the recently introduced *NoThinking* approach (Ma et al., 2025), which allows reasoning models to skip the thinking process and directly generate the final solution by prompting with a pseudo-thinking process. Specifically, we further simplify the approach by prompting the model with an empty thinking segment (i.e., "`<think></think>`"). Our pilot study in Section 3 indicates that *NoThinking* achieves comparable or even better performance than *Thinking* on relatively simple problems (up to high-school competition level), while significantly reducing token usage; the benefits of *Thinking* only become pronounced when the problem is difficult enough.

In light of this observation, we are curious: *Can the reasoning model learn to select Thinking or NoThinking mode adaptively based on the difficulty of the input problem, thereby achieving more efficient reasoning without sacrificing or even improving performance?* To this end, we propose *AdaptThink*, a novel RL algorithm to teach reasoning models when to think. Specifically, *AdaptThink* features two core components: (1) a constrained optimization objective that encourages the model to choose *NoThinking* while ensuring overall performance does not degrade; (2) an importance sampling strategy that balances *Thinking* and *NoThinking* samples during on-policy training, thereby overcoming the challenge of cold start and allowing the model to explore and exploit both thinking modes throughout the whole training process.

Our experiments demonstrate that *AdaptThink* effectively enables reasoning models to adaptively select the optimal thinking mode based on problem difficulty, leading to substantial reductions in inference cost compared to prior approaches, while consistently enhancing model accuracy. For instance, on GSM8K, MATH500, and AIME2024, *AdaptThink* reduces the average response length of DeepSeek-R1-Distill-Qwen-1.5B by 50.9%, 63.5%, and 44.7%, and improving its accuracy by 4.1%, 1.4%, and 1.6%, respectively. The remarkable results substantiate the potential of difficulty-adaptive thinking-mode selection as a promising paradigm for advancing the trade-off between reasoning performance and efficiency.

In summary, our key contributions are as follows: (1) We simplify the *NoThinking* approach and demonstrate its advantages over *Thinking* for simpler tasks in terms of both performance and efficiency; (2) We propose *AdaptThink*, a novel RL algorithm that empowers reasoning models to adaptively select the optimal thinking mode adaptively based on problem difficulty, thereby substantially reducing inference costs and further improving performance; (3) We conduct extensive experiments to validate the efficacy of *AdaptThink*.

2 Related Work

Large Reasoning Models. Recent frontier large reasoning models (LRMs), such as OpenAI o1 (OpenAI, 2024), DeepSeek-R1 (DeepSeek-AI, 2025), and QwQ (Qwen Team, 2025), have developed the ability to employ human-like deep thinking in problem solving by generating a long chain of thought before arriving at a final solution. Such advanced ability is typically acquired through large-scale RL with verified rewards or fine-tuning on distilled reasoning traces. Despite promising performance, the lengthy thinking process introduces substantial inference costs and latency. Consequently, a variety of approaches have been proposed for more efficient reasoning.

Efficient Reasoning for LRMs. Most existing methods to improve the efficiency of LRMs focus on reducing the token usage in model responses. Some methods incorporate length-based rewards into RL to incentivize more concise responses (Arora and Zanette, 2025; Team et al., 2025) or enable precise control over response length (Aggarwal and Welleck, 2025). Other approaches finetune models with length-related preference pairs, which are obtained from best-of-N sampling (Luo et al., 2025a; Shen et al., 2025) or through postprocessing (Chen et al., 2024). Additionally, several works pursue training-free methods to decrease response length, employing techniques such as model merging (Team et al., 2025; Wu et al., 2025) or prompting (Han et al., 2024; Muennighoff et al., 2025; Fu et al., 2025; Xu et al., 2025). Nevertheless, these methods still utilize long thinking for all problems, while the recent *NoThinking* approach (Ma et al., 2025) allows reasoning models to bypass long thinking and directly output the final solution via prompting, achieving performance comparable to *Thinking* in low-token-budget settings. In this work, we further demonstrate that even with a sufficient token budget, *NoThinking* can outperform *Thinking* on simple problems while using significantly fewer tokens. This observation motivates us to propose *AdaptThink* to teach reasoning models to adaptively select the optimal thinking mode based on problem difficulty, which is a new direction for efficient reasoning.

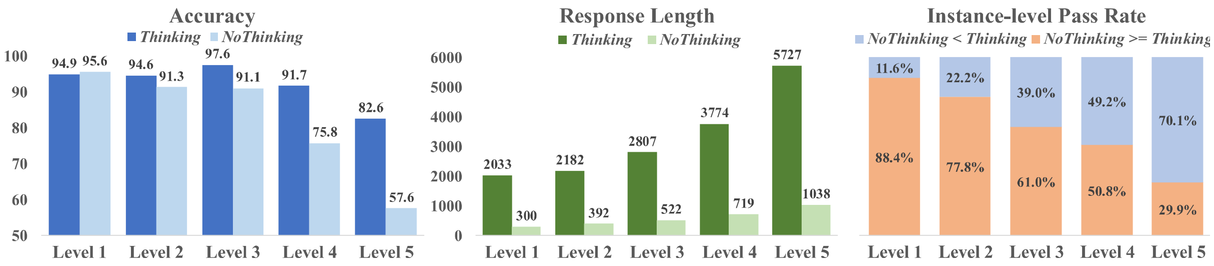

Figure 2: Comparison of DeepSeek-R1-Distill-Qwen-7B using *Thinking* and *NoThinking* mode across different difficulty levels of MATH500 dataset.

3 Motivation

3.1 Preliminary

Consider a reasoning model parameterized by θ and denoted by π_θ. Given a prompt $x = [x_1, \ldots, x_n,$ <think>$]$, where $[x_1, \ldots, x_n]$ represents the problem and <think> is the special token to start the thinking process, the model generates a response $y = [y_1, \ldots, y_l,$ </think>$, y_{l+2}, \ldots, y_m]$. Here, $[y_1, \ldots, y_l]$ corresponds to the thinking, which is a long chain of thought consisting of constant exploration, reflection, and self-verification. The token </think> marks the end of thinking. The remaining sequence, $[y_{l+2}, \ldots, y_m]$, denotes the final solution, which only includes the correct steps to solve the problem and the final answer. From the perspective of probability theory, the response y is a sample drawn from the conditional probability distribution $\pi_\theta(\cdot|x)$. Since y is generated in an auto-regressive way, the conditional probability $\pi_\theta(y|x)$ can be decomposed as:

$$\pi_\theta(y|x) = \prod_{t=1}^{m} \pi_\theta(y_t|x, y_{<t}) \tag{1}$$

3.2 *NoThinking* is Better for Simple Problems

Current reasoning models, such as OpenAI o1 and DeepSeek-R1, apply long thinking across all problems (denoted as *Thinking* mode). Though enhancing models' reasoning capabilities, the lengthy thinking process often leads to unnecessary computation overhead, especially for some simple problems that can also be solved by non-reasoning models (e.g., GPT-4o and Qwen-2.5-Instruct) without thinking. Recently, Ma et al. (2025) proposed *NoThinking* method, which enables reasoning models to bypass long thinking and directly generate the final solution by prompting with a fake thinking process *"Okay, I think I have finished thinking.*</think>*"*, and found it is still effective in low-token-budget settings. In this work, we further simplify *NoThinking* by providing the models with an empty thinking (i.e., enforcing the first generated token $y_1 =$ </think>). Then, we conduct a pilot study to compare *Thinking* and *NoThinking* from the perspective of problem difficulty, with a sufficient token budget (16K).

Specifically, we utilize MATH500 (Lightman et al., 2024) dataset for the pilot study since its have categorized problems into five difficulty levels. For each problem, we employ DeepSeek-R1-Distill-Qwen-7B to generate 16 responses using *Thinking* and *NoThinking*, respectively. Then we analyze the accuracy, response length, and instance-level pass rate across the five difficulty levels. As illustrated in Figure 2, although the model is trained using long-thinking data, *NoThinking* still achieves accuracy comparable to *Thinking* on relatively simple problems (Level 1 to 3), and even slightly outperforms *Thinking* on the easiest Level-1 problems. Meanwhile, the average length of *NoThinking* responses is significantly shorter than *Thinking* ones. Additionally, compared to *Nothinking*, *Thinking* only improves the instance-level pass rate for less than half of the problems from Level 1 to 4. Overall, these findings indicates that *Thinking* only brings notable benefits for challenging problems, whereas *NoThinking* can be a better choice for simpler questions in terms of both accuracy and efficiency. This motivates us to explore efficient reasoning from a new perspective: teaching the reasoning model to adaptively select *Thinking* or *NoThinking* mode based on problem difficulty, thereby reducing inference costs while maintaining or even improving the overall performance. To this end, we propose *AdaptThink*, a novel RL algorithm that teaches reasoning models when to think.

4 AdaptThink

Our *AdaptThink* algorithm consists of two important components: (1) a constrained optimization