

When Is Thinking Enough? Early Exit via Sufficiency Assessment for Efficient Reasoning

Yang Xiang¹, Yixin Ji¹, Ruotao Xu¹, Dan Qiao², Zheming Yang², Juntao Li^{1*}, Min Zhang^{1,3}

¹Soochow University ²ByteDance

³Key Laboratory of General Artificial Intelligence and Large Models, Soochow University
{baldwin021129, jiyixin169}@gmail.com {ljt, minzhang}@suda.edu.cn

Abstract

Large reasoning models (LRMs) have achieved remarkable performance in complex reasoning tasks, driven by their powerful inference-time scaling capability. However, LRMs often suffer from overthinking, which results in substantial computational redundancy and significantly reduces efficiency. Early-exit methods aim to mitigate this issue by terminating reasoning once sufficient evidence has been generated, yet existing approaches mostly rely on handcrafted or empirical indicators that are unreliable and impractical. In this work, we introduce **Dynamic Thought Sufficiency in Reasoning (DTSR)**, a novel framework for efficient reasoning that enables the model to dynamically assess the sufficiency of its chain-of-thought (CoT) and determine the optimal point for early exit. Inspired by human metacognition, DTSR operates in two stages: (1) *Reflection Signal Monitoring*, which identifies reflection signals as potential cues for early exit, and (2) *Thought Sufficiency Check*, which evaluates whether the current CoT is sufficient to derive the final answer. Experimental results on the Qwen3 models show that DTSR reduces reasoning length by 28.9%–34.9% with minimal performance loss, effectively mitigating overthinking. We further discuss overconfidence in LRMs and self-evaluation paradigms, providing valuable insights for early-exit reasoning.

1 Introduction

The success of large reasoning models (LRMs), such as OpenAI’s o1/o4 (OpenAI, 2024, 2025), Gemini 2.5 Pro (Comanici et al., 2025), DeepSeek-R1 (Guo et al., 2025), and Qwen3 (Yang et al., 2025a), represents a significant breakthrough in the evolution of LLMs. LRMs engage in long chain-of-thought (CoT) reasoning before producing answers. The inference-time scaling capability (Chen et al., 2025a; Ji et al., 2025) allows them to achieve remarkable performance in challenging reasoning

tasks (e.g. mathematics, programming, and scientific question answering), and show human-like reasoning patterns, such as problem decomposition, verification and backtracking.

Nevertheless, recent studies (Chen et al., 2025b; Cuadron et al., 2025) have shown that LRMs often suffer from overthinking: even on simple problems, they tend to consume excessive tokens by repeatedly verifying answers or exploring alternative strategies after reaching the correct solution. Such redundant reasoning is unnecessary and leads to inefficiency. To address this, several prior works have proposed early-exit strategies that stop the reasoning process once the LRM has generated sufficient evidence to solve the problem. The main challenge of early-exit strategies is to determine the optimal exit point, where LRMs achieve near-lossless performance with minimal reasoning cost. Dynasor-CoT (Fu et al., 2025) first employs the consistency of intermediate answers to determine when to exit reasoning; however, it still generates additional token intervals after producing the correct answer to verify its correctness, thereby missing the optimal exit point. DEER (Yang et al., 2025b) and CGRS (Huang et al., 2025) employ the entropy of the intermediate result as a confidence measure to determine whether LRMs reach the optimal exit point. Nevertheless, using answer confidence as a criterion for reasoning exit is not always reliable, as reasoning models can exhibit overconfidence even when their answers are incorrect (Mei et al., 2025). Moreover, such methods that rely on probing intermediate answers are limited to tasks with short, fixed-format answers. For tasks involving long answers or open-ended generation without a definitive answer, intermediate answers are difficult to extract, rendering these methods impractical.

To overcome these limitations, this work aims to establish a reliable and generalizable criterion for early-exit. Unlike LLM reasoning, human cognition involves metacognitive (Flavell, 1979; Fleur

* Corresponding author.

et al., 2021) abilities, which allow people to monitor, evaluate, and regulate their thoughts, and to stop thinking once the process is deemed sufficient. Inspired by this, we wonder whether LRMs can similarly determine the optimal point to exit reasoning by continuously assessing the sufficiency of their current thoughts? To answer this, we propose Dynamic Thought Sufficiency in Reasoning (DTSR), a novel framework that enables models to dynamically evaluate whether their current CoT is sufficient to solve the problem. DTSR consists of two stages: 1) Reflection Signal Monitoring: LRMs often exhibit reflective behaviors after producing an answer, typically triggered by specific reflection signals. We identify these signals as potential exit points for reasoning. 2) Thought Sufficiency Check: once a reflection signal is detected, the model evaluates the sufficiency of its current reasoning. If the sufficiency score is high, reasoning stops and the solution is output; otherwise, the model continues along the reasoning path until the next reflection signal occurs.

To evaluate the effectiveness of our proposed method, we conducted extensive experiments across multiple LRMs. The results show that DTSR significantly outperforms baseline approaches, achieving a favorable balance between accuracy and efficiency. This also demonstrates that reasoning models can reliably assess the sufficiency of their thought, providing valuable insights and guidance for advancing efficient reasoning.

Overall, our contributions are as follows:

- We propose a novel framework for efficient reasoning that enables the model to dynamically assess the sufficiency of its CoT and determine the optimal point for early exit.
- Extensive experiments conducted on Qwen3 series models of various sizes demonstrate that DTSR consistently outperforms baseline methods, achieving a 28.9%–34.9% reduction in generation length with almost no loss in performance, thereby significantly mitigating overthinking.
- Further discussion of overconfidence in LRMs and the self-evaluation paradigm offers a fresh perspective and valuable insights for guiding early-exit reasoning.

2 Methodology

2.1 Preliminaries

Generation Patterns of Reasoning Models The responses of LRMs are typically structured in two

stages: a reasoning phase, in which the model engages in slow, human-like thinking, encapsulated within the `<think>` and `</think>` delimiters, followed by a solution phase that summarizes the reasoning and produces the final answer. During the thinking process, the model exhibits complex reasoning behaviors, such as problem decomposition, strategy exploration, output verification, and self-correction (Gandhi et al., 2025; Bogdan et al., 2025). To trigger these reflective behaviors, LRMs often generate tokens like “Wait”, “Alternatively”, or “Let me check” as cues. However, once the correct answer has been reached, further verification becomes redundant, leading to unnecessary reasoning cost (Chen et al., 2025b).

Early Exit in Reasoning Early exit is a promising strategy for mitigating redundant reasoning. Muennighoff et al. (2025) first introduces a simple budget forcing method: when generation reaches the maximum token budget, the model is forced to append the end-of-thinking token and “Final Answer:” to terminate reasoning and output its current answer. Dynasor-CoT (Fu et al., 2025) probes intermediate answers via designed prompts at fixed intervals and terminates reasoning once consecutive answers are consistent. It partially mitigates overthinking but still generates extra token intervals to verify correctness after the correct answer is produced, failing to achieve optimal exit point. In contrast, DEER (Yang et al., 2025b) outputs intermediate answers at reasoning transition points (e.g., “Wait”) and uses entropy to estimate their confidence, terminating the reasoning process when achieving high confidence.

However, previous methods mainly rely on hand-crafted or statistical criteria, which are unreliable for determining whether the current reasoning process is sufficient to yield the correct answer. Mei et al. (2025) find that even state-of-the-art reasoning models are overconfident, particularly for incorrect answers, and this overconfidence grows with the longer reasoning process. Consequently, a substantial gap exists between token probabilities and the actual correctness of answers. From a cognitive perspective, humans rarely produce explicit intermediate answers to decide when to stop thinking. Instead, through metacognitive processes (Fleur et al., 2021), they internally assess whether their thoughts are sufficient to reach a final answer. Inspired by this, enabling reasoning models to self-assess the sufficiency of their thought to determine

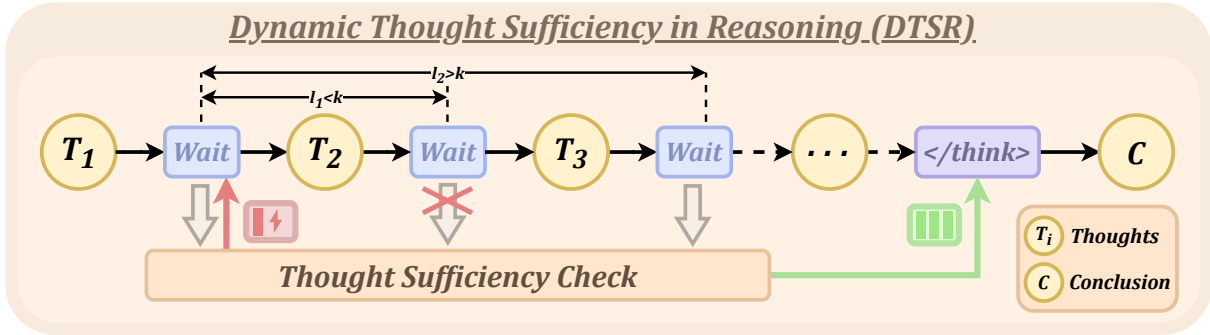


Figure 1: Overview of the Dynamic Thought Sufficiency in Reasoning (DTSR) method. Upon a reflection signal, the model evaluates the sufficiency of its previous thought. If the thought is insufficient, reasoning continues; checks are skipped within k tokens. Once the thought is sufficient, `</think>` is appended and a conclusion is generated.

if it can yield a final answer is highly intuitive. Therefore, we propose Dynamic Thought Sufficiency in Reasoning (DTSR), a method that enables models to assess whether their current thought is sufficient to derive the final answer.

2.2 Our Proposal: Dynamic Thought Sufficiency in Reasoning

As shown in Figure 1, we present DTSR, a method that enables the model to dynamically evaluate the sufficiency of its reasoning process. Our method consists of two stages: reflection signal monitoring and thought sufficiency check.

Reflection Signal Monitoring Our objective is to exit the reasoning process at the optimal point. Since assessing thought sufficiency after every generated token is computationally impractical, we instead seek to identify potential exit signals that indicate when verification should be triggered. To this end, we conduct an empirical analysis of reasoning trajectories generated by Qwen3-32B. For each trajectory, we identify its optimal early-exit point, defined as the earliest position in the reasoning process at which the model is already capable of producing a correct answer (Figure 7). We find that these optimal exit points are often followed by explicit self-reflection or answer-verification behaviors, manifested through cues such as “Wait”, “Alternatively”, and phrases beginning with “But” (e.g., “But wait”, “But let me check”). Based on these observations, we construct a reflection signal set, the complete list of which is shown in Table 4.

Thought Sufficiency Check The LRM continues reasoning until a reflection signal is encountered. Upon detecting such a signal, the model performs a thought sufficiency check on the reasoning trace

generated so far to determine whether it can terminate further thinking and produce a final solution. Specifically, at each checkpoint, the original question together with the current chain-of-thought (CoT) is fed into a dedicated thought sufficiency check prompt template (shown in Figure 8). The model is then prompted to generate a scalar score indicating the sufficiency of the current reasoning. A score exceeding the threshold τ indicates that the model deems its reasoning sufficient; in this case, we append a termination marker `</think>` to halt further reasoning and allow the model to output the final answer. Conversely, if the score falls below the threshold τ , the reasoning is considered insufficient, and the model resumes generation from the last reflection signal, continuing the reasoning process until the next checkpoint. This procedure is repeated until the reasoning process is complete. Unlike prior approaches that rely on handcrafted heuristics or empirical external metrics, our method determines the reasoning exit point through the model’s internal self-evaluation. In this paradigm, the model assesses the sufficiency of its previously generated reasoning from a third-person perspective, treating the reasoning trace as an external object rather than directly evaluating itself. Further discussion of this can be found in Section 4.4.

Moreover, reflection signals frequently occur in close succession, causing repeated sufficiency checks to yield similar outcomes and resulting in redundant computation. To mitigate this issue, we enforce a minimum token interval k between consecutive sufficiency checks. Concretely, a reflection signal triggers a thought sufficiency check only if at least k tokens have been generated since the previous check; otherwise, the current signal is skipped and the process continues (Algorithm 1).

Algorithm 1 Dynamic Thought Sufficiency in Reasoning (DTSR)

```
1: Init: model  $\mathcal{M}$ , question  $Q$ , prompt  $q_{cot}$ ,  
   reflection signals  $\mathcal{S}$ , min interval  $k$ , threshold  
    $\tau$ , max length  $max\_len$   
2:  $L \leftarrow q_{cot} + Q$ , reasoning trace  $T \leftarrow \emptyset$ ,  
   counter  $c \leftarrow 0$   
3: while  $len(L) < max\_len$  do  
4:   Generate token  $t$ ,  $T \leftarrow T + t$ ,  $c \leftarrow c + 1$   
5:   if  $t \in \mathcal{S}$  and  $c \geq k$  then  
6:     Build sufficiency prompt with  $(Q, T)$   
7:     Obtain score  $s \in [0, 100]$  from  $\mathcal{M}$   
8:     if  $s \geq \tau$  then  
9:        $T \leftarrow T + \langle /think \rangle$   
10:    else  
11:      Continue reasoning,  $c \leftarrow 0$   
12:    end if  
13:  end if  
14: end while  
15: return  $T$ 
```

3 Experiments

3.1 Experimental Setup

Benchmarks and Metrics To comprehensively evaluate the reasoning capabilities of LRMs, we select six widely used benchmarks: GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AMC 2023, OlympiadBench (He et al., 2024), GPQA Diamond (Rein et al., 2024) and LiveCodeBench (Jain et al., 2025). Our goal is to significantly reduce the number of tokens generated during the reasoning process while maintaining high reasoning performance. To this end, we employ two primary metrics: Accuracy (Acc) and Token Count (Tok). Acc measures the proportion of correct final answers, calculated as the average pass@1 score, whereas Tok represents the average number of tokens generated per sample, serving as an indicator of reasoning efficiency.

Models We conduct extensive experiments on the Qwen3 (Yang et al., 2025a) series of models. Our selection spans different model sizes (8B, 14B, 32B) of reasoning models to validate the generalizability of our method. Despite their strong reasoning ability, these models often exhibit overthinking during the reasoning process.

Baselines We compare DTSR with the vanilla LRM as well as existing state-of-the-art training-free efficient reasoning methods, including No-

Thinking (Ma et al., 2025a), NOWAIT (Wang et al., 2025), and DEER (Yang et al., 2025b). Vanilla allows the model to generate answers directly without any intervention. NoThinking prompts the model to skip the reasoning stage and answer the question directly. NOWAIT suppresses explicit self-reflection by masking reflective tokens such as “wait” and “hmm” during the reasoning process. DEER makes early-exit decisions based on the confidence scores of intermediate answers. In addition, we compare DTSR with recent training-based methods (Arora and Zanette, 2025a; Dai et al., 2025).

Implementation Details All experiments are conducted using the vLLM framework to ensure high throughput and memory-efficient inference. For the decoding strategy, we set the temperature to 0.6 and top- p to 0.95. The maximum generation length is set to 16k tokens, and each dataset is generated three times, with the final results reported as the average pass@1. In our method, the thought sufficiency threshold τ is set to 100, and the minimum token interval k is set to 64.

3.2 Experimental Results

Overall Performance The results in Table 1 show that DTSR consistently outperforms the baseline methods across three model sizes and five benchmark datasets. Compared with vanilla reasoning models, DTSR achieves almost the same level of accuracy without any degradation, while reducing sequence length by 28.9%–34.9%, leading to more efficient inference. Notably, when applied to Qwen3-14B, DTSR even improves performance on challenging reasoning tasks such as GPQA and OlympiadBench, underscoring its effectiveness.

Comparison with SOTAs We compare DTSR with several recent training-free efficient reasoning methods. As shown in Table 1, DTSR effectively shortens reasoning length without compromising accuracy, whereas other baselines struggle to balance efficiency with performance. For instance, NoThinking enforces models to bypass the reasoning process through prompting, which indeed shortens sequences but deviates from the natural generation paradigm and severely limits reasoning capability. NoWAIT mitigates redundant output by masking frequent reflective tokens, yet this restriction on the model’s inherent self-reflection results in significant performance degradation on complex reasoning tasks. DEER relies on the confidence scores of intermediate answers, yet models often exhibit

Method	GSM8K		MATH-500		AMC		GPQA		Olympiad		Overall	
	Acc \uparrow	Tok \downarrow	Acc \uparrow	Tok \downarrow	Acc \uparrow	Tok \downarrow	Acc \uparrow	Tok \downarrow	Acc \uparrow	Tok \downarrow	Acc \uparrow	Tok \downarrow
Qwen3-8B												
<i>Vanilla</i>	96.0	2208	94.4	5067	92.5	7232	60.1	8861	66.4	9182	81.9	6510
<i>NoThinking</i>	92.6	303	86.0	1067	72.5	2264	53.0	1271	48.3	2385	70.5	1458
<i>NoWAIT</i>	94.9	1307	92.8	3485	90.0	5448	55.6	6920	62.4	6998	79.1	4832
<i>DEER</i>	95.1	963	93.4	2549	87.5	5155	57.1	8306	63.4	5768	79.3	4532
<i>DTSR (ours)</i>	95.1	1152	94.0	2359	90.0	4507	59.6	7942	66.4	6180	81.0 _{-0.9}	4428 _{-32.0%}
Qwen3-14B												
<i>Vanilla</i>	96.2	1672	95.4	4503	97.5	6562	65.7	7513	67.0	8555	84.4	5761
<i>NoThinking</i>	95.1	278	88.2	852	77.5	2065	55.6	1286	51.7	2029	73.6	1302
<i>NoWAIT</i>	95.8	1133	92.8	3307	97.5	5035	64.1	6775	65.6	6670	83.2	4584
<i>DEER</i>	95.3	896	94.4	2601	97.5	4514	62.6	7105	65.1	6717	83.0	4367
<i>DTSR (ours)</i>	96.2	849	95.0	2247	97.5	3975	66.2	5916	68.9	5755	84.8 _{+0.4}	3748 _{-34.9%}
Qwen3-32B												
<i>Vanilla</i>	96.6	1707	94.8	4489	95.0	6807	68.2	6906	69.0	8280	84.7	5638
<i>NoThinking</i>	94.8	283	86.4	958	70.0	1738	54.6	561	52.4	2076	71.6	1123
<i>NoWAIT</i>	95.8	1212	93.4	3285	95.0	4894	66.7	5436	65.8	6598	83.3	4285
<i>DEER</i>	95.5	979	94.0	2881	92.5	4966	67.2	6395	66.4	6370	83.1	4318
<i>DTSR (ours)</i>	96.0	849	95.0	2660	95.0	4617	69.2	5677	67.7	6249	84.6 _{-0.1}	4010 _{-28.9%}

Table 1: Experimental results across various reasoning models. "Acc" denotes accuracy, and "Tok" denotes token count. \uparrow indicates that higher values are better, while \downarrow indicates that lower values are better.

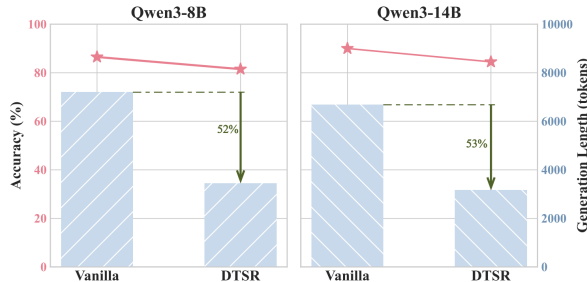


Figure 2: Performance of DTSR on LiveCodeBench.

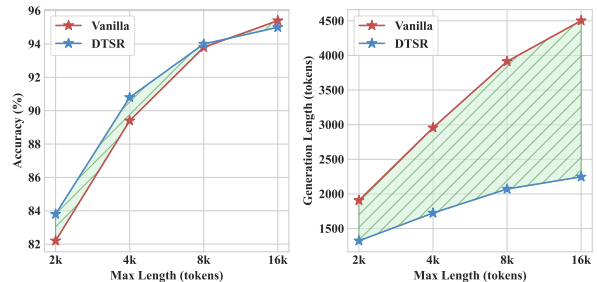


Figure 3: Comparison of Vanilla and DTSR on MATH-500 as the token budget increases.

high confidence even in incorrect ones. Moreover, it focuses only on partial answers while neglecting the global reasoning process, making it unreliable. In contrast, DTSR performs self-assessment over the entire reasoning process, which accounts for global coherence while mitigating overconfidence, thereby achieving superior overall performance.

We also present a comparison between DTSR and training-based methods in Table 5. The results show that DTSR achieves a generation length comparable to training-based approaches. Although DTSR exhibits slightly lower accuracy due to the absence of additional training, these results demonstrate that DTSR remains competitive with training-based methods while requiring no extra training.

Performance on Programming Tasks To evaluate the generalization of DTSR across different domains, we additionally assess its performance on programming tasks. Figure 2 presents the evalua-

tion results of Qwen-8B and Qwen-14B on LiveCodeBench. Compared to mathematical and scientific tasks, the reasoning process in programming tasks tends to be more redundant. DTSR is able to reduce the generation length by over 50% while incurring only minimal performance degradation. Overthinking is particularly pronounced in programming tasks, as they typically involve longer answers, requiring the model to spend more tokens on verification and reflection. In many cases, the model is already capable of generating correct code during the intermediate reasoning steps, repeated verification prolongs this process.

4 Analysis and Discussion

4.1 Ablation Study

Performance with different token budget We expand the maximum generation token budget from

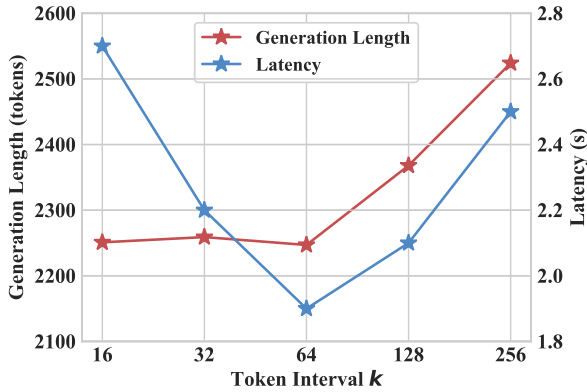


Figure 4: Impact of token interval k .

2k to 16k to compare Vanilla and DTSR in terms of accuracy and generation length. Figure 3 shows that DTSR consistently generates shorter sequences across all tested budgets, with accuracy only showing a slight fluctuation at 16k, while outperforming Vanilla in the other settings. Notably, the difference in sequence length between Vanilla and DTSR grows as the budget increases, suggesting that DTSR achieves greater token savings under larger budgets. In practice, our method can dynamically adjust the token budget to suit diverse application scenarios, thereby improving flexibility and efficiency in real-world deployments.

Impact of token interval k To begin with, varying the token interval k has almost no effect on accuracy, since the criterion for assessing thought sufficiency remains unchanged. Instead, k primarily affects two factors: generation length and inference latency. Figure 4 presents the results of DTSR on MATH-500 under different token intervals. When $k \leq 64$, the generation length remains nearly constant. However, once $k > 64$, the length starts to increase as k grows. This occurs because overly large token intervals may cause the model to miss the optimal exit points for certain problems, thereby delaying exit. Regarding latency, an overall U-shaped trend can be observed. When $k \leq 64$, the reasoning latency increases due to the frequent thought-sufficiency checks required at smaller intervals. In contrast, when $k > 64$, although the number of checks is reduced, the model tends to generate longer reasoning traces after missing optimal exit points, which in turn leads to increased latency. Consequently, $k = 64$ represents a balanced trade-off between generation length and latency, making it a desirable choice in practice.

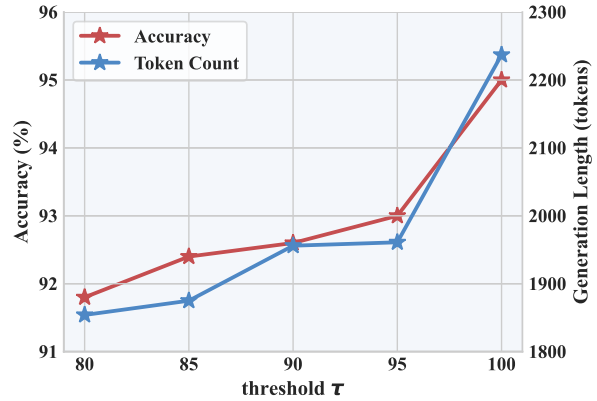


Figure 5: Impact of threshold τ .

Method	MATH-500		AMC	
	Len	Latency	Len	Latency
Vanilla	4480	3.1	6562	8.3
DEER	2601	4.2	4514	15.6
DTSR	2247	1.9	3975	6.2

Table 2: Comparison of Inference Latency (s).

Impact of threshold τ Figure 5 presents the performance of DTSR on the MATH-500 dataset across different threshold values of τ . When the threshold is low, accuracy drops significantly because the model tends to terminate reasoning prematurely, before its thought process is fully developed. This leads to incorrect answers, even though the overall generation length becomes somewhat shorter. These results indicate that the optimal exit point for reasoning is achieved when $\tau = 100$, at which the model shows strong confidence in the sufficiency of its reasoning process.

4.2 Analysis of Inference Latency

Table 2 reports the end-to-end inference latency of DTSR on the MATH-500 and AMC datasets. The results show that our method not only substantially reduces the generation length, but also achieves a significant reduction in inference latency, yielding improvements of approximately 25%–40%. In contrast, although DEER reduces the final generation length, it unexpectedly increases inference latency. This performance gap can be attributed to two key factors. First, we enable the continuous batching strategy in vLLM, which allows different samples to interleave between the generation and evaluation phases. In contrast, DEER adopts a synchronous scheme that requires all samples to complete generation before a collective evaluation, resulting in

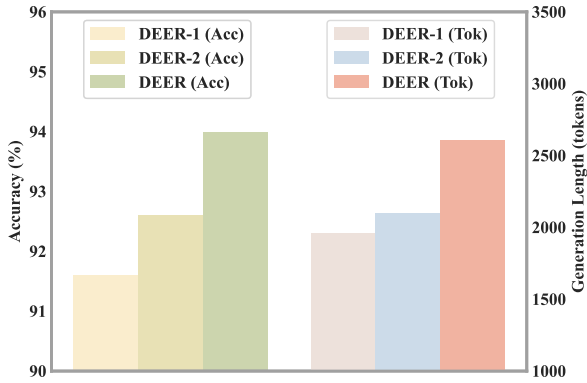


Figure 6: Comparison between DEER and its two variants, DEER-1 and DEER-2.

idle computation and increased latency. Second, we introduce a token interval parameter k , which substantially reduces redundant checks, thereby further improving overall inference efficiency.

4.3 Overconfidence in LRMs

Here, we discuss the issue of overconfidence in LRM. In fact, DEER has observed in experiments that the Qwen-3 series models tend to be overconfident in their confidence predictions. In the original setup, DEER adds an early-exit condition on top of the confidence score derived from intermediate answers: the model is required to generate `</think>` after producing a trial answer. To further investigate this issue, we derive two variants from DEER:

- **DEER-1:** The early-exit decision is based solely on the confidence score of the trial answer.
- **DEER-2:** Whether the model generates `</think>` after producing an intermediate answer is used as the criterion for early exit.

Figure 6 presents the performance of DEER and its two variants, DEER-1 and DEER-2, on MATH-500. DEER-1, which operates without the constraint of generating `</think>`, achieves the lowest accuracy, highlighting the model’s overconfidence in its generated answers. This indicates that relying solely on the confidence score of a trial answer is insufficient for determining when to exit reasoning. In contrast, DEER-2, which uses the simple criterion of whether `</think>` is generated, attains higher accuracy than DEER-1. The generation of `</think>` after producing an answer is a behavior learned during training, serving as a signal of the model’s self-assessment of answer correctness. These results suggest that leveraging the model’s intrinsic self-evaluation is a more reliable approach

Method	Accuracy (\uparrow)	Length (\downarrow)
DTSR	95.0	2247
DTSR-1 ($\tau=100$)	94.2	3254
DTSR-1 ($\tau=95$)	93.8	2457

Table 3: Comparison between DTSR and DTSR-1.

for guiding early-exit reasoning than using externally designed criteria.

4.4 Self-Evaluation Paradigms

In DTSR, self-evaluation is conducted from a third-person perspective rather than by having the model directly score its own thoughts. The two approaches are illustrated in Figure 9. To compare them, we introduce another variant, DTSR-1, in which an additional prompt is added during the intermediate reasoning process to ask the model to directly evaluate the adequacy of its prior reasoning. The results on MATH-500 are shown in Table 3. It is clear that DTSR-1 performs worse than DTSR in both accuracy and generation length. This is primarily because the model is unable to accurately assess the sufficiency of its reasoning during the process. As the saying goes, “the observer sees clearly what the participant cannot”. Only by evaluating from a third-person perspective and separating reasoning generation from the judgment of its adequacy can the model demonstrate its full reasoning capability.

4.5 Case Study

We conduct case studies on three types of problems: mathematical, programming, and open-ended. Figure 10 shows the detailed reasoning process of Qwen3-14B on a math problem. At the beginning, the model generates reasoning normally. When it produces the reflection signal, it starts to evaluate the adequacy of its previous thoughts and gives a score of 75, which is below the early-exit threshold τ . As a result, the model returns to the reflection signal position to continue reasoning. In the third round of reasoning, several reflection signals appear, but the model performs the thought sufficiency check only at the last one. This is because the number of generated tokens has not yet reached the token interval k , and redundant checks are skipped. After several checks, the model finally gives a score of 100 on the fourth attempt, satisfying the early-exit condition. It then appends `</think>` and proceeds to make a conclusion, generating the final answer. Figure 11 shows how

DTSR solves a programming problem, and Figure 12 presents the comparison between Vanilla and DTSR on an open-ended question. Across all cases, DTSR reduces token usage substantially while maintaining performance, improving reasoning efficiency and alleviating excessive thinking.

5 Related work

Recently, Large Reasoning Models (LRMs) have demonstrated remarkable performance on complex reasoning tasks by generating long Chains of Thought (CoT) for deep reasoning (Jaech et al., 2024; Guo et al., 2025). However, recent studies (Chen et al., 2025b; Cuadron et al., 2025) have revealed the issue of overthinking, where models continue to reason even after obtaining the correct answer, leading to redundant computational overhead. To address this, prior research has explored a variety of approaches to enable efficient reasoning (Feng et al., 2025), which can be broadly categorized into three directions: (Input) prompt-guided methods, (Model) post-training methods, and (Output) decoding-intervention methods.

Prompt-guided Methods Such methods leverage carefully designed prompts to guide the model in performing efficient reasoning. Concise CoT (Renze and Guven, 2024) simply adds the instruction “Be concise” to the prompt, whereas TALE (Han et al., 2025) first estimates an optimal token budget and then incorporates it into the prompt to encourage shorter reasoning chains. Chain-of-Draft (Xu et al., 2025a) constrains the model to preserve only the most essential reasoning drafts at each step, thereby reducing token usage in few-shot settings. Although easy to implement, these approaches (Lee et al., 2025) are limited in compressing long reasoning chains, and therefore better suited for use alongside other methods.

Post-training Methods These methods typically employ supervised fine-tuning (SFT) using variable-length CoT data (Yu et al., 2024; Liu et al., 2024; Ma et al., 2025b), or leverage reinforcement learning (RL) with length-based rewards (Team et al., 2025; Arora and Zanette, 2025b; Yeo et al., 2025; Yi et al., 2025). L1 (Aggarwal and Welleck, 2025) and Elastic Reasoning (Xu et al., 2025b) incorporate length constraints into the reward function, training models to flexibly adjust the length of reasoning chains under different token budgets. Recently, numerous studies (Lou et al., 2025; Jiang

et al., 2025b; Fang et al., 2025) have focused on training hybrid reasoning models that adaptively choose between concise responses and long-chain reasoning based on task complexity. While these methods attempt to address the problem at the fundamental level, their high training costs restrict most experiments to smaller models (e.g., 1.5B or 7B), limiting their applicability to larger models.

Decoding-intervention Methods Such methods intervene during decoding to adjust generation, guiding models to produce answers earlier (Qiao et al., 2025; Huang et al., 2025) or exit reasoning sooner (Liu and Wang, 2025; Jiang et al., 2025a; Zhang et al., 2025). NoWAIT (Wang et al., 2025) attempts to suppress explicit self-reflection by blocking tokens like Wait and Hmm during reasoning. However, the model can readily bypass this restriction by generating other tokens. Dynasor-CoT (Fu et al., 2025) probes the model’s intermediate answers at fixed token intervals and stops thinking if three consecutive answers are consistent, whereas DEER (Yang et al., 2025b) tracks intermediate answers at reasoning transition points and determines whether to terminate reasoning based on their confidence scores. However, both approaches rely on probing intermediate answers to determine stopping points, which works well only for problems with definite answers, but is less effective for long-form or open-ended outputs. In contrast, our approach effectively overcomes this limitation.

6 Conclusion

In this work, we alleviate the inefficiency of LRMs by introducing Dynamic Thought Sufficiency in Reasoning (DTSR), a novel framework that enables models to dynamically determine when their CoT is sufficient to yield a reliable answer. By integrating reflection signal monitoring with thought sufficiency checks, DTSR provides a principled mechanism for mitigating overthinking while maintaining strong reasoning performance. Experiments on Qwen3 models demonstrate that DTSR achieves substantial reductions in reasoning length with only minimal performance degradation, highlighting its effectiveness in improving efficiency. We further discuss overconfidence in LRMs and the self-evaluation paradigm, shedding light on how models effectively assess the sufficiency of their CoT. This exploration offers a fresh perspective and valuable insights for guiding early-exit reasoning.

Limitations

While DTSR demonstrates promising results in improving reasoning efficiency, we identify some possible limitations as follows.

- Due to computational resource constraints, we conduct experiments only on LLMs up to 32B.
- This work focuses exclusively on textual reasoning tasks, including mathematics and code. We have not yet extended our method to multimodal reasoning or agent scenarios, which we leave for future exploration.

Acknowledgement

This work was supported by the National Science Foundation of China (NSFC No. 62576232), Key Laboratory of General Artificial Intelligence and Large Models in Provincial Universities, Soochow University.

References

- Pranjal Aggarwal and Sean Welleck. 2025. [L1: Controlling how long a reasoning model thinks with reinforcement learning](#). *Preprint*, arXiv:2503.04697.
- Daman Arora and Andrea Zanette. 2025a. [Training language models to reason efficiently](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Daman Arora and Andrea Zanette. 2025b. [Training language models to reason efficiently](#). *CoRR*, abs/2502.04463.
- Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. 2025. [Thought anchors: Which llm reasoning steps matter?](#) *Preprint*, arXiv:2506.19143.
- Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025a. [Towards reasoning era: A survey of long chain-of-thought for reasoning large language models](#). *CoRR*, abs/2503.09567.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. [Do NOT think that much for 2+3=? on the overthinking of long reasoning models](#). In *Forty-second International Conference on Machine Learning*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. [Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities](#). *Preprint*, arXiv:2507.06261.
- Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. 2025. [The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks](#). *Preprint*, arXiv:2502.08235.
- Mz Dai, Chenxu Yang, and Qingyi Si. 2025. [S-GRPO: Early exit via reinforcement learning in reasoning models](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. [Thinkless: Llm learns when to think](#). *Preprint*, arXiv:2505.13379.
- Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. [Efficient reasoning models: A survey](#). *Preprint*, arXiv:2504.10903.
- John H. Flavell. 1979. [Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry](#). *American Psychologist*, 34:906–911.
- D. S. Fleur, B. Bredeweg, and W. van den Bos. 2021. [Metacognition: Ideas and insights from neuro- and educational sciences](#). *npj Science of Learning*, 6:1–11.
- Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu, Ion Stoica, and Hao Zhang. 2025. [Reasoning without self-doubt: More efficient chain-of-thought through certainty probing](#). In *ICLR 2025 Workshop on Foundation Models in the Wild*.
- Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. 2025. [Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STars](#). In *Second Conference on Language Modeling*.
- D. Guo, D. Yang, H. Zhang, and et al. 2025. [Deepseek-r1 incentivizes reasoning in llms through reinforcement learning](#). *Nature*, 645:633–638.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. [Token-budget-aware LLM reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855, Vienna, Austria. Association for Computational Linguistics.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. [OlympiadBench](#):

- A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Jiameng Huang, Baijiong Lin, Guhao Feng, Jierun Chen, Di He, and Lu Hou. 2025. [Efficient reasoning for large reasoning language models via certainty-guided reflection suppression](#). *Preprint*, arXiv:2508.05337.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helvar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, and 80 others. 2024. [Openai o1 system card](#). *CoRR*, abs/2412.16720.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. [Live-codebench: Holistic and contamination free evaluation of large language models for code](#). In *The Thirteenth International Conference on Learning Representations*.
- Yixin Ji, Juntao Li, Yang Xiang, Hai Ye, Kaixin Wu, Kai Yao, Jia Xu, Linjian Mo, and Min Zhang. 2025. [A survey of test-time compute: From intuitive inference to deliberate reasoning](#). *Preprint*, arXiv:2501.02497.
- Guochao Jiang, Guofeng Quan, Zepeng Ding, Ziqin Luo, Dixuan Wang, and Zheng Hu. 2025a. [Flash-think: An early exit method for efficient reasoning](#). *Preprint*, arXiv:2505.13949.
- Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang, Tengchao Lv, Lei Cui, and Furu Wei. 2025b. [Think only when you need with large hybrid-reasoning models](#). *Preprint*, arXiv:2505.14631.
- Ayeong Lee, Ethan Che, and Tianyi Peng. 2025. [How well do LLMs compress their own chain-of-thought? a token complexity approach](#). In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.
- Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. [Can language models learn to skip steps?](#) In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Xin Liu and Lu Wang. 2025. [Answer convergence as a signal for early stopping in reasoning](#). *Preprint*, arXiv:2506.02536.
- Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. 2025. [Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning](#). *Preprint*, arXiv:2505.11896.
- Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025a. [Reasoning models can be effective without thinking](#). *Preprint*, arXiv:2504.09858.
- Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025b. [CoT-valve: Length-compressible chain-of-thought tuning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6025–6035, Vienna, Austria. Association for Computational Linguistics.
- Zhiting Mei, Christina Zhang, Tenny Yin, Justin Lidard, Ola Shorinwa, and Anirudha Majumdar. 2025. [Reasoning about uncertainty: Do reasoning models know when they don't know?](#) *Preprint*, arXiv:2506.18183.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. [s1: Simple test-time scaling](#). In *Workshop on Reasoning and Planning for Large Language Models*.
- OpenAI. 2024. [Openai o1 system card](#). *Preprint*, arXiv:2412.16720.
- OpenAI. 2025. [Introducing openai o3 and o4-mini](#).
- Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Fandong Meng, Jie Zhou, Ju Ren, and Yaoyue Zhang. 2025. [Concise: Confidence-guided compression in step-by-step efficient reasoning](#). *Preprint*, arXiv:2505.04881.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. [GPQA: A graduate-level google-proof q&a benchmark](#). In *First Conference on Language Modeling*.
- Matthew Renze and Erhan Guven. 2024. [The benefits of a concise chain of thought on problem-solving in large language models](#). In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, page 476–483. IEEE.
- Kimi Team, Angang Du, Bofei Gao, BOWEI XING, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chunying Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 75 others. 2025. [Kimi k1.5: Scaling reinforcement learning with llms](#). *CoRR*, abs/2501.12599.
- Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou.

2025. [Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency.](#) *Preprint*, arXiv:2506.08343.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025a. [Chain of draft: Thinking faster by writing less.](#) *Preprint*, arXiv:2502.18600.

Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Junnan Li, and Caiming Xiong. 2025b. [Scalable chain of thoughts via elastic reasoning.](#) *Preprint*, arXiv:2505.05315.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. [Qwen3 technical report.](#) *Preprint*, arXiv:2505.09388.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. 2025b. [Dynamic early exit in reasoning models.](#) *Preprint*, arXiv:2504.15895.

Edward Yeo, Yuxuan Tong, Xinyao Niu, Graham Neubig, and Xiang Yue. 2025. [Demystifying long chain-of-thought reasoning in LLMs.](#) In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*.

Jingyang Yi, Jiazheng Wang, and Sida Li. 2025. [Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning.](#) *Preprint*, arXiv:2504.21370.

Ping Yu, Jing Xu, Jason E Weston, and Ilya Kulikov. 2024. [Distilling system 2 into system 1.](#) In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. 2025. [Reasoning models know when they're right: Probing hidden states for self-verification.](#) In *Second Conference on Language Modeling*.

A Reflection Signal

In this paper, we take the following keywords as signals for LRM reflection:

“Wait”, “Alternative”, “Alternatively”
“But wait”, “But let me”, “But let’s”

Table 4: Reflection Signal List

Comparison with NoWAIT NoWAIT suppresses all reflection-related tokens, which can disrupt intrinsic reasoning and reduce accuracy. In contrast, DTSR preserves normal reasoning and triggers sufficiency evaluation only upon detecting

reflection signals, achieving significant generation-length reduction without harming performance.

B Prompt Template

Figure 8 presents the prompt template used for thought sufficiency check stage. Figure 9 illustrates the differences between the prompt templates of DTSR and its variant, DTSR-1.

C More Experiments

Comparison with Training-based Methods We further compare DTSR with more baselines. For training-based methods, we include RL + Length Penalty (Arora and Zanette, 2025a) and S-GRPO (Dai et al., 2025). RL + Length Penalty penalizes overly long correct responses by assigning rewards based on their deviation from the mean length. S-GRPO uses rule-based outcome rewards to regulate intermediate reasoning, encouraging LLMs to generate high-quality thoughts earlier and to exit once sufficient reasoning is reached, thereby improving efficiency while maintaining accuracy. As shown in Table 5, DTSR achieves a generation length comparable to training-based methods. Although it exhibits slightly lower accuracy due to the absence of additional training, DTSR remains competitive while requiring no extra training.

Comparison with Training-free Methods We also compare DTSR with two recent efficient reasoning methods, TALE (Han et al., 2025) and Chain-of-Draft (CoD) (Xu et al., 2025a). As shown in Table 6, TALE achieves almost no reduction in generation length, while CoD substantially reduces token usage at the cost of a significant accuracy drop. In contrast, DTSR strikes a better balance between accuracy and generation length.

Performance under Different Decoding Strategies In the main experiments, we adopt a decoding configuration with temperature set to 0.6 and top_p set to 0.95. Table 7 reports the results under greedy decoding. The results show that DTSR maintains strong performance across different decoding strategies, further demonstrating the robustness and applicability of our method.

D Case Study

Figures 10, 11 and 12 present representative examples of Qwen-14B solving mathematical, programming, and open-ended problems under the DTSR framework, respectively.

Question: Let ABC be a triangle inscribed in circle ω . Let the tangents to ω at B and C intersect at point D , and let \overline{AD} intersect ω at P . If $AB=5$, $BC=9$, and $AC=10$, AP can be written as the form $\frac{m}{n}$, where m and n are relatively prime integers. Find $m + n$.

Full Thought:

<think>\nOkay, so I need to find the probability that, for three randomly chosen distinct vertices Q, R, S of a regular icosahedron, the distance $d(Q, R)$ is greater than $d(R, S)$. The answer should be in the form m/n where m and n are coprime, and then find $m + n$.

\n</think>\n\n We are given triangle ABC \n\n Final Answer:\n\n $\boxed{13}$ \n\n **False**

\n\nFirst, I remember that when two tangents are drawn from a point to a circle, they are equal in length. So, $DB = DC$ because D is the intersection of the tangents at B and C . \n\n Wait, but A is on ω , so power of A would be zero. Hmm, not sure.

\n</think>\n\n We are given triangle ABC \n\n Final Answer:\n\n $\boxed{137}$ \n\n **False**

\n\nAlternatively, since D is the intersection of tangents at B and C \n\n Simplify:\n\n First, note that 88 and 22 have common factor 11 : $88 = 8 \cdot 11$, $22 = 2 \cdot 11$. \n\n So $88/22 = 4$. \n\n Therefore, $AP = 4 \cdot (25/13) = 100/13$.

\n</think>\n\n To solve the problem, we are given \n\n Final Answer:\n\n $\boxed{197}$ \n\n **False**

\n\nWait, this is confusing. Maybe I need to use directed lengths or consider the power of point formula with signs. But if P is beyond A from D , then $DA \cdot DP$ would be $DA \cdot (DA + AP)$. \n\n But I need to figure out which is the case.

\n</think>\n\n To solve the problem, we are given \n\n Final Answer:\n\n $\boxed{113}$ \n\n **True**

\n\nWait, that's nice! So $AP = 100/13$. Which is already in reduced form since 100 and 13 are coprime (13 is prime, doesn't divide 100). Therefore, $m = 100$, $n = 13$, $m + n = 113$.

\n</think>\n\n To solve the problem, we are given \n\n Final Answer:\n\n $\boxed{113}$ \n\n **True**

\n\nWait, but let me check my steps again to make sure. \n\n First, we found $t = 88/169$. Then $AP = t \cdot |\text{direction vector}| = (88/169) \cdot (325/22)$. Then simplifying: \n\n Divide 88 by $22 = 4$, and 325 divided by $169 = 25/13$. So $4 \cdot (25/13) = 100/13$. Yes, that seems correct.

\n</think>\n\n To solve the problem, we are given \n\n Final Answer:\n\n $\boxed{113}$ \n\n **True**

Under Thinking

Optimal Exit Point

Over Thinking

Figure 7: An illustrative example showing how we identify the optimal exit point. After each generated sentence, we append </think> to force the model to produce an answer. The correctness of the resulting answer is then used to determine whether the current reasoning state is sufficient and corresponds to the optimal exit point.

Thought Sufficiency Check Prompt Template

```
<|im_start|>users
Assess the confidence that the provided thought is sufficient to solve the question.
Output only in the format: Confidence: <number>, where <number> is a real value
between 0 and 100. Here, 100 means the thought is fully sufficient to reach the
final answer directly without additional reasoning, and 0 means the thought is
completely insufficient to do so.
```

```
### Question
{question}

### Thought
{thought}
<|im_end|>
<|im_start|>assistant
<think>

</think>

Confidence:
```

Figure 8: Thought Sufficiency Check Prompt Template

DTSR

<|im_start|>users

Assess the confidence that the provided thought is sufficient to solve the question. Output only in the format: Confidence: <number>, where <number> is a real value between 0 and 100. Here, 100 means the thought is fully sufficient to reach the final answer directly without additional reasoning, and 0 means the thought is completely insufficient to do so.

Question

{How many positive whole-number divisors does 196 have?}

Thought

{<think>\nOkay, so I need to figure out how many positive whole-number divisors the number 196 has. Hmm, let me think. I remember that to find the number of divisors of a number, you first have to find its prime factorization. Once you have that, you can use exponents to determine the total number of divisors. Let me try to recall exactly how that works.}

<|im_end|>

<|im_start|>assistant

<think>

</think>

Confidence:

DTSR-1

<|im_start|>system

Please reason step by step, and put your final answer within `\\boxed{}`.

<|im_end|>

<|im_start|>users

How many positive whole-number divisors does 196 have?

<|im_end|>

<|im_start|>assistant

<think>\nOkay, so I need to figure out how many positive whole-number divisors the number 196 has. Hmm, let me think. I remember that to find the number of divisors of a number, you first have to find its prime factorization. Once you have that, you can use exponents to determine the total number of divisors. Let me try to recall exactly how that works.

Assess the confidence that the provided thought is sufficient to solve the question. Output only in the format: Confidence: <number>, where <number> is a real value between 0 and 100. Here, 100 means the thought is fully sufficient to reach the final answer directly without additional reasoning, and 0 means the thought is completely insufficient to do so.

Confidence:

Figure 9: Differences between DTSR and its variant DTSR-1.

Method	GSM8K		MATH-500		AMC		GPQA		Overall	
	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓
<i>Qwen3-14B</i>										
<i>Vanilla</i>	96.2	1672	95.4	4503	97.5	6562	65.7	7513	88.7	5062
<i>RL + Length Penalty</i>	95.8	1090	95.8	2866	97.5	5059	65.4	4949	88.6	3491
<i>S-GRPO</i>	96.3	952	96.4	2652	100	4538	67.6	4537	90.1	3170
<i>DTSR</i>	96.2	849	95.0	2247	97.5	3975	66.2	5916	88.7	3246

Table 5: Comparison of DTSR with training-based efficient reasoning methods. "Acc" denotes accuracy, and "Tok" denotes token count. ↑ indicates that higher values are better, while ↓ indicates that lower values are better.

Method	GSM8K		MATH-500		AMC		GPQA		Overall	
	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓
<i>Qwen3-14B</i>										
<i>Vanilla</i>	96.2	1672	95.4	4503	97.5	6562	65.7	7513	88.7	5062
<i>TALE</i>	95.7	1241	94.6	4484	95.0	6623	65.6	7138	87.7	4872
<i>CoD</i>	85.7	648	75.2	2359	72.5	4122	51.0	1177	71.1	2077
<i>DTSR</i>	96.2	849	95.0	2247	97.5	3975	66.2	5916	88.7	3246

Table 6: Comparison of DTSR with TALE and Chain-of-Draft (CoD). "Acc" denotes accuracy, and "Tok" denotes token count. ↑ indicates that higher values are better, while ↓ indicates that lower values are better.

Method	GSM8K		MATH-500		AMC		GPQA		Olympiad		Overall	
	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓
<i>Qwen3-14B</i>												
<i>Vanilla</i>	96.3	1676	94.8	4511	97.5	6998	64.1	7382	65.6	8644	83.7	5842
<i>DTSR</i>	96.1	814	94.6	2296	97.5	3977	65.6	6103	66.8	5967	84.1	3831

Table 7: Experimental results under greedy decoding. "Acc" denotes accuracy, and "Tok" denotes token count. ↑ indicates that higher values are better, while ↓ indicates that lower values are better.

Question

In a table tennis tournament every participant played every other participant exactly once. Although there were twice as many right-handed players as left-handed players, the number of games won by left-handed players was 40% more than the number of games won by right-handed players. (There were no ties and no ambidextrous players.) What is the total number of games played?

Response

<think>\nOkay, so I need to solve this table tennis tournament problem. \n\nFirst, the problem says that every participant played every other participant exactly once. <omitted> \n\nTherefore, total games $T = G_L + G_R = 1.4G_R + G_R = 2.4G_R$. Score: 70 ❌

Alternatively, since T is also equal to $(3L)(3L - 1)/2$, so maybe I can set up equations here. \n\nBut how do I find G_L and G_R ? <omitted> \n\nSo adding them: \n\nConvert all terms to halves: \n\n $(L^2 - L)/2 + (4L^2 - 2L)/2 + (4L^2)/2$ Score: 90 ❌

Wait, $L=3: 3 \bmod 8$. $L=8: 0 \bmod 8$. $L=11: 3 \bmod 8$ (since $11-8=3$). <omitted> **Wait**, maybe my initial thought is not correct. **But wait**, $L=3$ and $L=8$ and $L=11$? \n\n <omitted> So seems that $L \equiv 0 \bmod 8$ or $L \equiv 3 \bmod 8$? Score: 95 ❌

Wait let me check $L = 3 + 8k$. Therefore, $L(L - 3) = (3 + 8k)(8k)$. <omitted> Let me check $L = 3 - 8 =$ negative, which is invalid. \n\nSo seems that $L=3$ is the only possible value. Hence total number of games is 36. Score: 100 ✅

\n</think>\n\nTo determine the total number of games played in the table tennis tournament, we analyze the structure of the competition and the given constraints. \n\n---\n\n<omitted> \n\n---\n\n ✅ **Final Answer** \n\n $\boxed{36}$

Vanilla's response: 10596 tokens ➔ *DTSR's response:* 824 tokens

Figure 10: An example illustrating the reasoning process of DTSR.

Question

You are given a 0-indexed 2D integer array `nums`. Initially, your score is 0. Perform the following operations until the matrix becomes empty:
From each row in the matrix, select the largest number and remove it. In the case of a tie, it does not matter which number is chosen.
Identify the highest number amongst all those removed in step 1. Add that number to your score.
Return the final score.

Response

<think> Okay, let's see. The problem is about processing a 2D matrix in such a way that in each step, we take the largest element from each row, **<omitted>** Then repeat until all rows are empty. Score: 50 ✖

Wait, but each row's length might be different? **Wait**, no. **Wait**, the problem says that the matrix is 0-indexed 2D array. But the rows can have different lengths? **<omitted>** But how do we process this? Score: 90 ✖

Wait, the process continues until the matrix becomes empty. **<omitted>** Then, for each step, we take the first element of each row, find the max among them, which would take the next element in each row (since the first **Score: 100** moved) ✔

```
\n</think>\n\nclass Solution:\n    def matrixSum(self, nums: List[List[int]]) -> int:\n        current_max = max(current_max, row[col])\n        score += current_max\n        return score
```

Vanilla's response: 10596 tokens



DSTR's response: 824 tokens

Figure 11: An example illustrating the reasoning process of DTSR on a programming problem.

Case Example: Open Question

Question:

Suppose gravity on Earth suddenly became half as strong.
How would human architecture, biology, and transportation adapt over time?

Vanilla:

<think>...</think>

If Earth's gravity suddenly became half as strong (i.e., gravitational acceleration reduced from 9.8 m/s² to 4.9 m/s²), the effects on human architecture, biology, and transportation would be profound and multifaceted, with immediate and long-term adaptations across these domains.

1. Architecture

- Reduced Structural Load: Buildings, bridges, and other infrastructure would require significantly less material and reinforcement. For example, skyscrapers could be taller or constructed with lighter materials (e.g., advanced composites instead of steel) due to decreased compressive forces.
- Design Adjustments: Foundations might be smaller or less deep, and load-bearing walls could be thinner. However, engineers would need to account for new challenges, such as wind forces (which are less mitigated by gravity) and seismic activity (potentially altered by changes in Earth's mass or tectonic dynamics).
- Fluid Dynamics: Plumbing systems might require adjustments to handle reduced pressure in water distribution, and drainage systems could become less efficient due to slower fluid flow under lower gravity...

2. Human Biology

- Reduced Physical Strain: Humans would feel lighter, making movement easier and reducing joint stress. People might jump higher, run faster, and experience less fatigue in daily activities.
- Musculoskeletal Changes: Over time, muscle atrophy and bone density loss could occur due to reduced mechanical loading. This would mirror the effects of prolonged microgravity on astronauts, potentially leading to osteoporosis and weakened ligaments.
- Cardiovascular Adaptation: The heart would pump less forcefully, and blood pressure might decrease. Circulatory systems might adjust to reduced gravitational pull, potentially altering the distribution of fluids in the body...

3. Transportation

- Easier Space Travel: Launching spacecraft would require less fuel to escape Earth's gravity, making interplanetary missions more feasible and cost-effective.
- Vehicle Design: Cars and other vehicles might become lighter and more agile, with reduced traction requirements...

DTSR:

<think>...</think>

If Earth's gravity suddenly became half as strong, the following adaptations would occur in architecture, biology, and transportation over time:

1. Architecture

- Structural Design: Buildings would require less material and reinforcement due to reduced gravitational load. Foundations could be shallower, and skyscrapers might reach greater heights since the structural stress is halved.
- Material Use: Lighter materials (e.g., composites, expanded metals) might replace heavier ones like concrete and steel. Open spaces, larger windows, and more cantilevers could become common.
- Wind Considerations: While gravity is lower, wind forces on tall structures might remain a limiting factor, potentially leading to more aerodynamic designs.

2. Biology

- Musculoskeletal System: Humans would likely experience reduced bone density and muscle mass due to less gravitational stress. Over generations, evolutionary adaptations might include taller, more slender bodies with less robust bones and muscles.
- Cardiovascular System: The heart might become smaller, as it would not need to work as hard to pump blood against gravity. However, if atmospheric pressure decreases (due to lower gravity), respiratory adaptations (e.g., larger lungs, higher hemoglobin levels) could emerge to compensate for lower oxygen availability.
- Fluid Distribution: Fluids (e.g., blood, interstitial fluid) would pool more in the upper body, leading to facial puffiness and potential changes in circulatory dynamics.

3. Transportation

- Ground Vehicles: Vehicles would weigh less, improving fuel efficiency and acceleration. Braking systems might need redesigning due to reduced friction (normal force is halved). Heavier cargo could be transported with the same infrastructure.
- Aircraft: If atmospheric pressure remains constant, aircraft would require less lift, enabling smaller, more efficient designs. However, if the atmosphere thins (due to lower gravity), air density would decrease, complicating lift generation. This might necessitate larger wings or more powerful engines.
- Space Travel: Launching to space would require less energy, making interplanetary travel more feasible..

Figure 12: Comparison of Vanilla's and DTSR's answer on an open question.