

PATS: Process-Level Adaptive Thinking Mode Switching

Yi Wang*, Junxiao Liu*, Shimao Zhang*, Jiajun Chen, Shujian Huang†
National Key Laboratory for Novel Software Technology, Nanjing University
{yiw, junxiaoliu, smzhang}@smail.nju.edu.cn
chenjj@nju.edu.cn, huangsj@nju.edu.cn

Abstract

Current large-language models (LLMs) typically adopt a fixed reasoning strategy, either simple or complex, regardless of reasoning complexity. This neglect of variation in task and reasoning process complexity leads to an imbalance between performance and efficiency. Existing methods attempt to implement training-free fast-slow thinking system switching to handle problems of varying difficulty, but are limited by coarse-grained solution-level strategy adjustments. To address this issue, we propose a novel reasoning paradigm: *Process-Level Adaptive Thinking Mode Switching (PATS)*, which enables LLMs to dynamically adjust their thinking mode based on the difficulty of each reasoning step, optimizing the balance between accuracy and computational efficiency. Our approach integrates Process Reward Models (PRMs) with Beam Search, incorporating progressive mode switching and bad-step penalty mechanisms based on thresholds derived from a data-driven methodology. Experiments on diverse mathematical and non-mathematical benchmarks across various model families and scales, with PRMs of varying quality demonstrate that our methodology achieves high accuracy while maintaining moderate token usage. This study emphasizes the significance of process-level, difficulty-aware thinking mode adaptation, offering insights into efficient inference for LLMs.

1 Introduction

In recent years, large language models (LLMs) have progressed from simple direct answering to more sophisticated reasoning paradigms, such as Chain-of-Thought prompting (Wei et al., 2022) and reflection-based methods inspired by o1-like systems (Jaech et al., 2024; Guo et al., 2025), leading

to improved performance on complex reasoning tasks. More sophisticated reasoning paradigms are better suited for solving harder problems. In practice, question difficulty varies, yet current models typically adopt a fixed reasoning strategy regardless of difficulty, leading to an imbalance between performance and computational efficiency.

Beyond the variation in difficulty across tasks, the reasoning steps within the same task also vary in difficulty. For example, even complex problems contain simple parts. Research has shown that in mathematical reasoning, sub-steps requiring intensive computation pose the primary challenge for limited-scale supervised fine-tuned models (Sun et al., 2025). Similarly, in reasoning tasks such as maze navigation, subproblems with varying levels of difficulty exist (Saha et al., 2024). These findings indicate that the difficulty in the reasoning process changes dynamically, necessitating the dynamic allocation of more resources to harder sub-steps and adaptive adjustment of the reasoning strategy accordingly. Fixed strategies fail to account for variations in problem and process difficulty, highlighting the necessity of adaptive adjustments to balance both accuracy and efficiency.

Prior work has attempted to switch thinking modes at the solution level (Yao et al., 2024), while adaptive switching at the finer-grained step level remains largely unexplored.

To address these limitations, we propose Process-Level Adaptive Thinking Mode Switching (PATS), a novel reasoning paradigm that dynamically selects appropriate thinking mode at each reasoning step based on its difficulty, achieving a strong balance between accuracy and efficiency. Our approach is built on the following main designs: (1) thinking modes are categorized based on the number of candidates generated at each step in Beam Search; (2) the difficulty of each step is assessed using the PRM score for the top candidate, and the thinking mode is adjusted accordingly; (3) mech-

*Equal contribution.

†Corresponding author.

¹The project is available at <https://github.com/NJUNLP/PATS>.

anisms such as progressive switching, rapid error recovery, and appropriate rollback for bad steps are incorporated. Compared to previous work, we perform finer-grained and more effective mode adjustments at the process-step level.

We conduct experiments across policy models of diverse families and scales, with PRMs of varying quality, evaluating on multiple math reasoning benchmarks with varying difficulty levels and multiple non-mathematical domains. Experiments demonstrate that our approach consistently maintains high accuracy with moderate token usage, achieving a strong balance between performance and efficiency. Furthermore, analysis of the performance of different thinking modes used at the first reasoning step and the distribution of thinking modes across tasks of varying complexity further enhances our understanding of how thinking modes can be aligned with task difficulty.

Our main contributions are as follows:

- We propose a novel reasoning paradigm that adaptively switches between thinking modes based on the difficulty of each reasoning step, enabling real-time, step-level strategy adjustment.
- Our approach achieves a strong balance between performance and efficiency, consistently delivering high accuracy with moderate token usage, and providing insights into efficient reasoning with LLMs.
- We conduct comprehensive experiments to validate and analyze our paradigm, demonstrating its robust generalization across diverse model families and scales, with PRMs of varying quality, and on both mathematical and non-mathematical domains.

2 Related Work

2.1 System Switching

Inspired by the dual-process theory (Wason and Evans, 1974; Kahneman, 2011), humans can flexibly switch between fast (System 1) and slow (System 2) thinking. Prior work has explored mechanisms for switching between System 1 and System 2 in LLMs, broadly categorized into training-based (Lou et al., 2025; Su et al., 2024; Saha et al., 2024; Cheng et al., 2025) and training-free methods (Yao et al., 2024). In this work, we focus on training-free approaches. HDFLOW (Yao et al., 2024) fixedly employs System 1 to generate an initial solution, and if the solution fails evaluation,

the more complex System 2 is then activated to reconsider the problem. Deciding whether to switch thinking modes only after obtaining a complete solution is overly coarse: it misaligns with the current context of stepwise reasoning for complex problems and lacks adaptability to variations in the difficulty of the reasoning process. Although PATS is mainly compared with training-free methods, to better highlight the practical value of PATS, we also compare PATS with training-based methods in Appendix A.9.

2.2 Test-time Scaling

Test-time scaling improves performance by increasing inference-time computation (Jaech et al., 2024). A representative approach is step-level beam search (Snell et al., 2024), a PRM-guided method (Wang et al., 2023; Zhang et al., 2025a; She et al., 2025) that leverages step-level correctness estimation (Uesato et al., 2022; Lightman et al., 2023; Li et al., 2025).

2.3 Reasoning Process

Studies show that some steps in hard-level problems are computation-intensive, posing challenges for models with limited-scale SFT (Sun et al., 2025). In tasks like maze navigation, subproblems vary in difficulty and require adaptive allocation of computational resources (Saha et al., 2024). These indicate that reasoning difficulty changes dynamically throughout the reasoning process, and thus the reasoning strategy should adapt accordingly. Unlike concurrent works on step-level adaptive reasoning—DREAM (Cui et al., 2025), which allocates budgets across different phases within a single step, and SAT (Huang et al., 2026), which softly modulates per-step reasoning depth relying on the model’s instruction-following ability—PATS deterministically switches thinking modes across steps of varying difficulty by explicitly controlling the beam width, without training any component.

3 Method

This section introduces the PATS pipeline (Figure 1) and its key components.

3.1 Preliminary Study

PRM-guided beam search is a representative test-time scaling method (Snell et al., 2024). It scores candidate steps generated by the policy model at each step and selects the top- k candidates with the highest scores for the next iteration. This paper

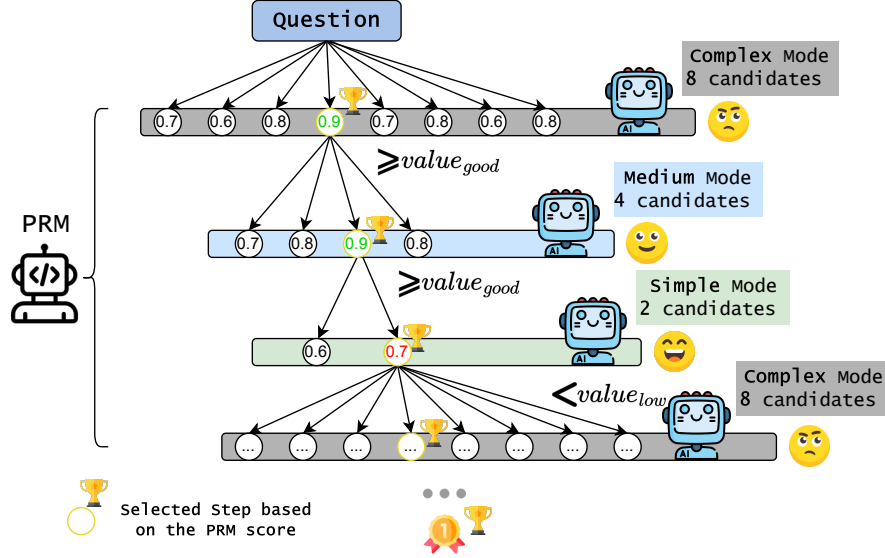


Figure 1: Illustration of our PATS paradigm. For math reasoning tasks, the model performs PRM-guided beam search, where the number of candidate steps (2/4/8) serves as a proxy for thinking mode complexity. At each step, the policy model selects the top-scoring candidate, infers reasoning difficulty, and dynamically switches to the corresponding mode.

focuses on the best-first search (BFS) scenario with $K = 1$ as the reasoning context (Wang et al., 2025).

Given a problem q , the policy model performs step-by-step reasoning, generating a solution path $S = \{s_1, s_2, \dots, s_n\}$, where s_n denotes the n -th reasoning step. BFS consists of two iterative operations: expansion and selection.

In the expansion stage of step i , the model generates w candidate steps:

$$C_i = \{c_{i,1}, c_{i,2}, \dots, c_{i,w}\} \quad (1)$$

Here, C_i denotes all candidate steps at step i , w is a hyperparameter controlling search width, and $c_{i,j}$ is the j -th candidate step at step i .

In the selection phase of step i , BFS chooses the candidate step with the highest PRM score $v(\cdot)$ from C_i as the final selected step s_i , proceeding to the next iteration:

$$s_i = \arg \max_{c \in C_i} v(c) \quad (2)$$

This iteration continues until the final answer is generated. Through iterative expansion and optimal selection, BFS efficiently attains a final solution.

PRM-guided beam search uses a fixed width w at every step. Prior studies (Wang et al., 2025) show that increasing w improves performance but significantly raises computational cost. This trade-off, where increased computational cost leads to

improved performance, resembles a model adopting more complex reasoning modes to solve more challenging problems by expending additional computational resources. Building on this foundation, we use PRM-guided beam search framework to simulate problem-solving scenarios across a range of complexity levels.

3.2 Process-Level Adaptive Thinking Mode Switching

As discussed in §3.1, a larger search width w results in higher accuracy but greater computation. Current PRM-guided beam search typically employs a predefined, fixed w at each step, implying an unchanged reasoning strategy throughout the inference process. Therefore, at each step, a larger search width w can be considered to represent a more complex thinking mode.

Let w_i denote the number of candidates generated at step i , and $mode_i$ represent the reasoning mode employed at that step. We define the reasoning strategy at step i based on the number of candidates w_i as follows:

- **Simple Thinking Mode:** $mode_i = \text{simple}$, corresponding to $w_i = 2$
- **Medium Thinking Mode:** $mode_i = \text{medium}$, corresponding to $w_i = 4$

- **Complex Thinking Mode:** $mode_i = \text{complex}$, corresponding to $w_i = 8$

Considering that mathematical reasoning is characterized by step-by-step execution and generalizability, we choose it as our primary task. While focused on the mathematical domain, we also investigate the generalizability of PATS to diverse non-mathematical fields. The proposed PATS framework works as follows:

The policy model performs step-by-step reasoning under PRM-guided beam search, selecting the candidate step with the highest PRM score at each step. Based on this score, the model assesses the current reasoning difficulty and adaptively switches to an appropriate thinking mode. By default, the model starts reasoning in complex mode ($mode_1 = \text{complex}$).

Remark. PATS is a pioneering exploratory work in the field of process-level thinking-mode switching, for which the dual-process theory serves as motivation rather than a literal implementation: PATS uses beam width as a proxy for thinking mode, thus performing adaptive beam-width scheduling rather than switching between fundamentally different reasoning strategies. At this initial exploration stage, beam width is a simple, direct, and controllable variable for changing the single-step reasoning intensity, allowing us to directly control the single-step thinking mode by controlling the beam width. This setup is highly suitable for preliminarily studying the performance of step-level mode switching, helping us observe its behavior and obtain useful insights, providing valuable empirical experience and foundations for extending this paradigm to more complex scenarios in the future.

3.3 Thinking Mode Switching Mechanism

The core of PATS is to dynamically adjust the thinking mode based on the reasoning difficulty of the current state. As the reasoning state becomes more difficult, a correspondingly more complex reasoning mode is required. A higher PRM score for the current step indicates better reasoning quality, suggesting that the model is in a more favorable reasoning state and is more likely to be on the correct reasoning path (Wang et al., 2023). Studies use PRM score of the final answer to evaluate problem difficulty (Snell et al., 2024), with higher scores indicating easier problems for the model. Additionally, inspired by reward signal can guide re-

source allocation (Sun et al., 2024), Fu et al. (2024) collect the terminal reward scores from each reasoning path and aggregate them, where a higher aggregated reward indicates that less computational resources need to be allocated.

Thus, the PRM score can act as an indicator of current reasoning difficulty. We hypothesize that a higher PRM score at the current step indicates a lower reasoning difficulty for that step, allowing for the allocation of fewer computational resources and the adoption of simpler thinking modes. In structured reasoning tasks like mathematics, reasoning steps are not independent events. There is strong logical and semantic continuity and correlation between adjacent steps. Since the next reasoning step is conditioned on the current state, a low score at the current step indicates higher downstream risk and suggests that stronger inference should be used in the next step. Similarly, if the current step is easy to reason, suggesting that the next step is also more likely to be easy and thus allows switching to a simpler mode.

After selecting the final step s_i at step i , we use $v(s_i)$, the PRM score of the chosen step, to estimate the current reasoning difficulty and adapt the reasoning strategy for the subsequent step accordingly.

- If $v(s_i) \geq \text{value}_{\text{good}}$, the reasoning state is considered favorable, and the thinking mode should switch to a simpler one. Inspired by the approach of employing progressively shorter reasoning trajectories during training (Ma et al., 2025), we implement a smooth mode transition when the reasoning state is favorable. Specifically, if the current step i is in complex thinking mode ($mode_i = \text{complex}$), the next step becomes medium thinking mode ($mode_{i+1} = \text{medium}$). Similarly, if $mode_i = \text{medium}$, then $mode_{i+1} = \text{simple}$; if $mode_i = \text{simple}$, then $mode_{i+1} = \text{simple}$.
- If $v(s_i) < \text{value}_{\text{low}}$, the reasoning state is considered unfavorable, and the thinking mode should switch to the most complex setting to mitigate potential error accumulation. This means that, regardless of the current mode $mode_i$, the reasoning mode for the next step is set to $mode_{i+1} = \text{complex}$.
- If $v(s_i)$ does not satisfy the threshold conditions for mode switching, the current think-

ing mode is retained for the next step; that is, $mode_{i+1} = mode_i$.

In addition to adapting the reasoning strategy for the subsequent step, this mechanism includes immediate strategy adjustments for critically poor steps. If $v(s_i)$ falls significantly below $value_{low}$, it suggests a severe error at the current step, and delaying correction until the next step may be too late to prevent further reasoning failure. Thus, we define a critical threshold $value_{bad}$. If $v(s_i) < value_{bad}$, we penalize the current step by restarting step i in complex mode (regenerating $w_i = 8$ candidates). To avoid infinite loops on unresolvable steps, each step is penalized at most once.

3.4 Thresholds Selection

To ensure our mode-switching thresholds are principled and generalizable, we derive them using a data-driven methodology based on the PRM score distribution, rather than by arbitrary selection. Our method is as follows:

Data Collection. We first create a large pool of reasoning step scores by running all PRMs on a 10% random sample of the training-set data. These steps are then categorized into “correct-trajectory steps” and “error-trajectory steps” based on the final answer’s correctness.

Principled Threshold Selection. We then analyze the score distributions of these two groups to set our thresholds. Our goal is to identify statistical markers that robustly differentiate between confident, problematic, and erroneous steps:

- **value_{good}** = 0.85: This captures most high-quality steps while filtering out “false high-score” steps from error paths. The distribution of correct-trajectory scores shows a strong concentration at very high values. Our choice corresponds to the **15th percentile** of scores from the correct-trajectory steps.
- **value_{low}** = 0.75: This threshold is used to identify steps that may be problematic. The scores of the error trajectories are more spread out. Our choice corresponds to the **35th percentile** of scores from the error-trajectory steps.
- **value_{bad}** = 0.40: This threshold is designed to capture clear, low-scoring errors. Our choice corresponds to the **10th percentile** of scores from the error-trajectory steps.

This statistical approach provides clear guidance for applying PATS to new settings.

4 Experiment

4.1 Experimental Setup

Datasets. We evaluate on widely used mathematical reasoning benchmarks, including **GSM8k** (Cobbe et al., 2021), **MATH500** (Lightman et al., 2023), **Minerva Math** (Lewkowycz et al., 2022), **AIME24** (AI-MO, 2024a), and **AMC23** (AI-MO, 2024b), which collectively span elementary, intermediate, and advanced levels of mathematical reasoning. Additionally, we employ **MMLU-Pro-CoT-Eval** (Zeng et al., 2025), a multi-domain evaluation dataset, to assess generalizability across diverse non-mathematical fields.

Policy Models. To demonstrate broad generalizability, our experiments utilize models from diverse families, including the **Qwen2.5** series (Yang et al., 2024a) and **Llama3** (Dubey et al., 2024). We specifically test on models of varying scales (1.5B, 3B, 7B, and 14B) to evaluate scaling properties. Unless otherwise specified, our primary experiments are conducted using the **Qwen2.5-7B-Instruct**.

Process Reward Models. To ensure PATS’s robustness across domains and quality levels, we utilize both math-specific and multi-domain PRMs. For mathematical tasks, we select three models representing distinct quality levels based on their ProcessBench scores (Zhang et al., 2025b): **Math-Shepherd** (Wang et al., 2023) (weak, 31.5), **Qwen2.5-Math-7B-PRM800K** (Zheng et al., 2024) (medium, 56.5), and **Qwen2.5-Math-PRM-7B** (Yang et al., 2024b) (strong, 73.5). For non-mathematical domains, we employ **VersaPRM** (Zeng et al., 2025), a specialized multi-domain PRM. Unless otherwise specified, the main results are reported using **Qwen2.5-Math-PRM-7B**.

Evaluation Metrics. We evaluate model performance along two dimensions: **accuracy** and **efficiency**. For accuracy, we report the mean answer correctness. For efficiency, we follow (Kang et al., 2024; Wang et al., 2024) and compute the average number of generated output tokens, which is a widely adopted, robust, and representative proxy for computational efficiency. The overhead of PRM

scoring is much smaller than the token-generation cost of the policy model and is thus not incorporated into the main efficiency metric. For reference, the corresponding PRM-call statistics are provided in Appendix A.8.

Hyperparameters and Thresholds. We set the temperature to 0.6 to balance candidate step diversity and generation quality (Zhang et al., 2024). We set reward thresholds as follows: $value_{good} = 0.85$, $value_{low} = 0.75$, and $value_{bad} = 0.4$. The thresholds are derived from a principled, data-driven methodology based on the PRM score distribution, as detailed in §3.4.

The random seed for the main experiments is set to 42, and results of four random-seed runs are provided in Appendix A.7.

4.2 Baselines

The core of PATS is to adaptively adjust the thinking mode based on reasoning difficulty during the process. Therefore, the primary goal of our experiments is to compare our adaptive strategy against strategies that employ a fixed thinking mode during the process. This work differentiates thinking modes by the expansion width at each step of the Beam Search.

To assess the effectiveness of our adaptive approach, we selected the following baselines:

- **All-simple:** It uses a fixed *simple* mode at every step, generating two candidate steps per iteration. Formally, this corresponds to $mode_1 = mode_2 = \dots = mode_n = simple$, resembling direct-answer-style inference and suitable for relatively simple problems.
- **All-medium:** It follows the same structure as All-simple, but adopts a fixed *medium* mode at each step, generating four candidate steps per iteration. This reflects typical chain-of-thought reasoning, suited for intermediate problems.
- **All-complex:** It continues this fixed pattern by using the *complex* mode throughout, with eight candidate steps per iteration. This reflects o1-style slow thinking and is better suited for more challenging problems.
- **Random-mode Switch:** It begins with the *complex* thinking mode and, at each subsequent step, randomly selects one of the three

modes (*simple*, *medium*, or *complex*) with equal probability.

- **Solution-verification Switch:** It first generates a complete solution using *All-simple* and uses the PRM score at the final step as the solution-level score, following (Snell et al., 2024). If the score is greater than or equal to the threshold $value_{low}$, the solution is accepted; otherwise, we retry using *All-medium*. If this fails again, we switch to *All-complex* for final reasoning and answer generation. This approach simulates coarse-grained solution-level switching: fast reasoning is first used to generate a complete solution, and more complex strategies are applied only if it fails verification. This baseline serves as our implementation of the core HDFLOW idea within our experimental framework.

5 Results

5.1 Main Results

We report the performance of PATS and baselines in Table 1, comparing their performance in terms of average accuracy and average output token count. Our findings are as follows:

PATS achieves both high accuracy and efficiency. Across tasks of varying difficulty, PATS achieves an average accuracy close to that of the All-complex setting (only 0.3 points lower), while using just 55.4% of its output tokens. Furthermore, a McNemar’s test (McNemar, 1947) on the aggregated results of all 2161 problems confirms that there is no statistically significant difference in accuracy between PATS and All-complex ($p = 0.54$), reinforcing their comparable performance. Compared to All-medium, it achieves nearly 3 points higher accuracy with comparable token usage. Relative to All-simple, it yields a substantial 5.8-point improvement in accuracy. We note that PATS outperforms All-complex on AIME24, an effect we attribute to our one-time penalty mechanism, which helps correct severe procedural errors (see Appendix A.1 for the full analysis and robustness check). These results demonstrate that PATS maintains high average accuracy while achieving low token usage.

PATS achieves a superior performance-cost balance. The results in Table 1 show that PATS achieves a better balance between accuracy and efficiency. To further investigate the performance-cost dynamics, we evaluate the average accuracy

Setting	GSM8K		MATH500		AMC23		MinervaMATH		AIME24		Average	
	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$
All-simple	93.8	564.8	76.2	1154.3	52.5	1742.5	38.2	1234.3	16.7	2075.7	55.5	1354.3
All-medium	94.4	1001.0	80.2	2204.2	57.5	3337.3	43.0	2431.6	16.7	4113.3	58.4	2617.5
All-complex	94.9	1774.9	81.0	4068.9	67.5	6231.6	44.5	4571.9	20.0	8705.6	61.6	5070.6
Solution-verification Switch	94.5	596.3	79.6	2110.8	55.0	4318.6	38.6	1466.7	16.7	6621.1	56.9	3022.7
Random-mode Switch	94.3	1214.7	79.0	2838.1	52.5	4858.9	43.0	2996.2	13.3	6330.0	56.4	3647.6
★ PATS (Ours)	94.8	855.8	80.6	2067.7	65.0	3365.7	43.0	1929.9	23.3	5821.0	61.3	2808.0

Table 1: Comparison of average accuracy and token usage on five math reasoning benchmarks. Based on the average metrics, our method PATS achieves effective and efficient reasoning with an excellent accuracy-efficiency balance.

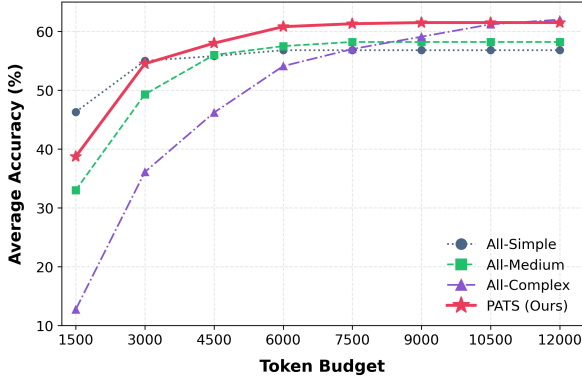


Figure 2: Average accuracy comparison across various token budgets, illustrating the superior Pareto frontier of PATS.

under controlled token budgets across different inference settings, as illustrated in Figure 2. The results show that PATS forms a superior Pareto frontier; its curve consistently lies in the top-left corner, reaching 61.3% accuracy with only 7,500 tokens, which represents a $\sim 30\%$ compute cost reduction compared to All-complex. This highlights the high resource efficiency of our process-level switching.

PATS outperforms coarse-grained switching. As shown in Table 1, PATS surpasses Solution-verification Switch by 4.4 points in average accuracy while using about 7% fewer tokens. These results demonstrate that our fine-grained, process-level thinking mode adjustment outperforms coarse-grained, solution-level switching, underscoring the importance of timely strategy adaptation throughout the reasoning process. Delaying the switch until a complete solution is generated fails to adapt to the varying difficulty of the reasoning process.

PATS’s pipeline is rational and effective. PATS not only outperforms the Random-mode Switch by 4.9 points in average accuracy, but also reduces average token consumption by 23%, as

shown in Table 1. Under the same process-level switching framework, PATS significantly outperforms random switching across all dimensions. Notably, random switching yields performance comparable to coarse-grained solution-level switching, suggesting that random mode integration alone fails to fully exploit the advantages of process-level adaptability. This underscores the rationality and effectiveness of our pipeline’s switching strategy design.

In summary, PATS dynamically adjusts thinking modes based on step-wise difficulty, achieving an excellent accuracy-efficiency balance while enabling effective and efficient reasoning. These results also empirically validate the reasonable assumption underlying our switching mechanism (§3.3): the PRM score of the current step serves as a reliable signal for adjusting the reasoning strength of the next step.

5.2 Analysis and Generalization Experiments

This subsection further analyzes PATS and conducts experiments on its generalization capabilities.

Influence of Task Difficulty. We investigate how task difficulty affects both the optimal initial thinking mode and reasoning behavior. Our analysis reveals that harder tasks benefit more from complex initialization and exhibit a higher proportion of complex modes during the mid-to-late reasoning stages, reflecting the need for greater reasoning effort. Easier tasks, in contrast, favor simpler modes. Detailed results and process distributions are provided in Appendix B.

Necessity and Moderation of Penalty on Bad Steps. In PATS, steps below a threshold $value_{bad}$ trigger a one-time penalty (rethinking in complex mode). Table 9 (in Appendix A.2) compares PATS against No Penalty and Infinite Penalty strategies. PATS strikes the best balance, achieving the highest

Setting	Math-Shepherd		Qwen2.5-Math-7B-PRM800K	
	Avg. Acc $\uparrow$	Avg. Token $\downarrow$	Avg. Acc $\uparrow$	Avg. Token $\downarrow$
All-simple	53.5	1286.8	55.4	1356.5
All-medium	54.1	2270.8	56.3	2743.4
All-complex	54.5	4167.8	58.3	5275.9
★ PATS (Ours)	55.5	2732.1	57.5	2418.0

Table 2: Generalization of PATS across PRMs with varying quality, while keeping the policy model fixed as Qwen2.5-7B-Instruct.

accuracy while maintaining significantly lower token usage than Infinite Penalty. This demonstrates the importance of penalizing suboptimal steps to prevent delayed correction, while underscoring the need for moderation to avoid unnecessary token consumption on unresolvable steps.

Generalization Across Diverse Model Families and Scales. The complete results for policy models of varying scales and families are provided in Appendix A.3. As Table 10 clearly shows, PATS maintains its excellent accuracy-efficiency balance across different model families and scales effectively to larger models. These findings strongly support that our proposed adaptive paradigm scales effectively and is not limited to a specific model architecture.

Generalization Across PRMs of varying quality. To ensure PATS’s robustness to PRMs of varying quality, we evaluate it on three PRMs with large performance gaps on the ProcessBench benchmark: **Strong** (Qwen2.5-Math-PRM-7B, score: 73.5), **Medium** (Qwen2.5-Math-7B-PRM800K, score: 56.5), and **Weak** (Math-Shepherd, score: 31.5). As shown in Table 1 and Table 2, PATS consistently maintains an excellent accuracy-efficiency balance across all three PRMs, proving its effectiveness even with noisy or weaker PRMs. Complete results are provided in Appendix A.3.

Comparison with Other Test-Time Compute Methods. To assess PATS against alternative test-time compute scaling methods, we evaluate its performance against Monte Carlo Tree Search (MCTS) and the Best-of-N baseline. Conventional Tree of Thoughts (ToT) approaches, such as MCTS, predominantly focus on improving search accuracy while often overlooking computational efficiency (Ding et al., 2025). To compare against such tree search methods, we evaluate accuracy on

GSM8K and MATH500 using the Qwen-2.5-7B-Instruct model, with MCTS results derived from Ding et al. (2025). As shown in Table 3, PATS consistently outperforms MCTS by a clear margin, demonstrating that its adaptive strategy is more effective than conventional tree search. Furthermore, we evaluate the Best-of-N baseline, where the solution score is calculated as the product of all step-level PRM scores. As presented in Table 4, PATS continues to maintain high accuracy with moderate token consumption, achieving a superior accuracy-efficiency balance.

Setting	GSM8K	MATH500	Average
MCTS	89.6	75.2	82.4
PATS (Ours)	94.8	80.6	87.7

Table 3: Accuracy comparison between MCTS and PATS.

Setting	Avg. Acc $\uparrow$	Avg. Token $\downarrow$
Best-of-2	58.5	1459.2
Best-of-4	60.5	2921.2
Best-of-8	62.1	5830.4
PATS (Ours)	61.3	2808.0

Table 4: Performance comparison between PATS and Best-of-N baselines.

Comparison with Other Inference Scaling Methods. To evaluate the efficacy of PATS, we compare it against recent training-free test-time compute methods, REBASE (Wu et al., 2025) and AB-MCTS (Inoue et al., 2025). Unlike REBASE, which employs a globally fixed strategy determined prior to execution, and AB-MCTS, which relies on multi-turn tree expansion overhead, PATS introduces fine-grained, step-level adaptive switching driven by process difficulty. As summarized in Table 5, PATS establishes a superior accuracy-efficiency balance, achieving the highest average accuracy of 61.3% while reducing token consumption by over 50% compared to both baselines. Detailed dataset-by-dataset breakdowns are provided in Table 13 of Appendix A.6.

Generalization Across Diverse Domains. To investigate generalizability beyond math, we sampled and evaluated questions across 13 diverse non-math domains (e.g., Law, History, Psychology) from the multi-domain evaluation dataset: MMLU-Pro-CoT-Eval. Sampling details are provided in

Setting	Avg. Acc $\uparrow$	Avg. Token $\downarrow$
REBASE	59.3	5926.2
AB-MCTS	57.8	5647.8
PATS (Ours)	61.3	2808.0

Table 5: Average performance comparison against other inference scaling methods.

Setting	Avg. Acc $\uparrow$	Avg. Token $\downarrow$
All-simple	56.0	1148.5
All-medium	58.3	2364.5
All-complex	61.5	4752.5
★ PATS (Ours)	60.9	1824.3

Table 6: Average performance metrics across 13 diverse non-math domains. PATS maintains a superior accuracy-efficiency balance in non-mathematical domains.

Appendix A.4. Table 6 presents the average performance across these sampled non-math domains using VersaPRM, a multi-domain PRM. The results show that PATS achieves superior accuracy with moderate token usage, confirming its effective transfer to broader non-mathematical domains.

Sensitivity Analysis on Thresholds. We evaluate robustness by perturbing each threshold by ± 0.05 . As shown in Appendix A.5, the impact is minimal, revealing stable performance in both accuracy and token usage. Across all perturbations, PATS consistently maintains a superior accuracy-efficiency balance compared to fixed-mode baselines, demonstrating robustness to the chosen thresholds.

6 Conclusion

In this paper, we propose a novel reasoning paradigm—**Process-Level Adaptive Thinking Mode Switching (PATS)**. This method leverages PRM scores during the reasoning process to estimate the current difficulty and accordingly adjusts the thinking mode, guided by thresholds derived from a principled, data-driven methodology. Experiments show that our approach achieves high accuracy with low computational cost, demonstrating robust generalization across diverse model families, scales, PRMs of varying quality, and non-mathematical domains. This paradigm highlights that fine-grained, real-time adjustment of thinking modes based on process-level difficulty can effectively balance accuracy and efficiency, offering new insights into efficient reasoning with LLMs.

Limitations

Our experiments demonstrate the effectiveness of PATS across multiple model families and scales up to 14B parameters. Due to computational constraints, we do not evaluate substantially larger models (e.g., 70B+). Extending our experiments to such scales could further deepen the understanding of our proposed paradigm. We also verify that PATS remains effective under PRMs of varying quality. Future extensions may incorporate alternative evaluation signals, such as LLM-as-Judge, to expand the scope of our experiments. Furthermore, this paradigm can also be extended to more complex and diverse scenarios.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. Shujian Huang is the corresponding author. This work is supported by National Science Foundation of China (No. 62376116), the Fundamental Research Funds for the Central Universities (No. 2024300507), Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118).

References

- AI-MO. 2024a. Ai-mo validation aime dataset. <https://huggingface.co/datasets/AI-MO/aimo-validation-aime>.
- AI-MO. 2024b. Ai-mo validation amc dataset. <https://huggingface.co/datasets/AI-MO/aimo-validation-amc>.
- Xiaoxue Cheng, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2025. [Think more, hallucinate less: Mitigating hallucinations via dual process of fast and slow thinking](#). *Preprint*, arXiv:2501.01306.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- Yingqian Cui, Zhenwei Dai, Pengfei He, Bing He, Hui Liu, Xianfeng Tang, Jingying Zeng, Suhang Wang, Yue Xing, Jiliang Tang, and Benoit Dumoulin. 2025. [Adaptive test-time reasoning via reward-guided dual-phase search](#). *Preprint*, arXiv:2509.25420.
- Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zeng-mao Wang, Ziwei Liu, Bo Du, Xianglong Liu, and

- Dacheng Tao. 2025. [Dynamic parallel tree search for efficient llm reasoning](#). *Preprint*, arXiv:2502.16235.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. 2024. [Efficiently serving llm reasoning programs with certaintindex](#). *Preprint*, arXiv:2412.20993.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Weiyang Huang, Xuefeng Bai, Kehai Chen, Xinyang Chen, Yibin Chen, Weili Guan, and Min Zhang. 2026. [Sat: Balancing reasoning accuracy and efficiency with stepwise adaptive thinking](#). *Preprint*, arXiv:2604.07922.
- Yuichi Inoue, Kou Misaki, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. 2025. Wider or deeper? scaling llm inference-time compute with adaptive branching tree search. *Advances in Neural Information Processing Systems*, 38:35448–35484.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. [Openai o1 system card](#). *Preprint*, arXiv:2412.16720.
- Daniel Kahneman. 2011. *Thinking, fast and slow*. macmillan.
- Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, Qianyi Sun, Boxing Chen, Dong Li, Xu He, Quan He, Feng Wen, and 1 others. 2024. [Mindstar: Enhancing math reasoning in pre-trained llms at inference time](#). *Preprint*, arXiv:2405.16265.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.
- Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. 2025. [From system 1 to system 2: A survey of reasoning large language models](#). *Preprint*, arXiv:2502.17419.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- 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.
- Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. [Cot-valve: Length-compressible chain-of-thought tuning](#). *Preprint*, arXiv:2502.09601.
- Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.
- Swarnadeep Saha, Archiki Prasad, Justin Chih-Yao Chen, Peter Hase, Elias Stengel-Eskin, and Mohit Bansal. 2024. [System-1.x: Learning to balance fast and slow planning with language models](#). *Preprint*, arXiv:2407.14414.
- Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. [R-prm: Reasoning-driven process reward modeling](#). *Preprint*, arXiv:2503.21295.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. [Scaling llm test-time compute optimally can be more effective than scaling model parameters](#). *Preprint*, arXiv:2408.03314.
- DiJia Su, Sainbayar Sukhbaatar, Michael Rabbat, Yuan-dong Tian, and Qinqing Zheng. 2024. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces. In *The Thirteenth International Conference on Learning Representations*.
- Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. 2024. [Fast best-of-n decoding via speculative rejection](#). *Preprint*, arXiv:2410.20290.
- Yiyi Sun, Georgia Zhou, Hao Wang, Dacheng Li, Nouha Dziri, and Dawn Song. 2025. [Climbing the ladder of reasoning: What llms can- and still can’t solve after sft?](#) *Preprint*, arXiv:2504.11741.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. [Solving math word problems with process- and outcome-based feedback](#). *Preprint*, arXiv:2211.14275.
- Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Dian Yu, Haitao Mi, Jinsong Su, and Dong Yu. 2024. [Lite-search: Efficacious tree search for llm](#). *Preprint*, arXiv:2407.00320.

- Ante Wang, Linfeng Song, Ye Tian, Dian Yu, Haitao Mi, Xiangyu Duan, Zhaopeng Tu, Jinsong Su, and Dong Yu. 2025. [Don't get lost in the trees: Streamlining llm reasoning by overcoming tree search exploration pitfalls](#). *Preprint*, arXiv:2502.11183.
- Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2023. [Math-shepherd: Verify and reinforce llms step-by-step without human annotations](#). *Preprint*, arXiv:2312.08935.
- Peter C Wason and J St BT Evans. 1974. Dual processes in reasoning? *Cognition*, 3(2):141–154.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference on Learning Representations*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024b. [Qwen2.5-math technical report: Toward mathematical expert model via self-improvement](#). *Preprint*, arXiv:2409.12122.
- Wenlin Yao, Haitao Mi, and Dong Yu. 2024. [Hdflow: Enhancing llm complex problem-solving with hybrid thinking and dynamic workflows](#). *Preprint*, arXiv:2409.17433.
- Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, Ying Fan, Jungtaek Kim, Hyung Il Koo, Kannan Ramchandran, Dimitris Papailiopoulos, and Kangwook Lee. 2025. [Versaprm: Multi-domain process reward model via synthetic reasoning data](#). *Preprint*, arXiv:2502.06737.
- Shimao Zhang, Yu Bao, and Shujian Huang. 2024. [Edt: Improving large language models' generation by entropy-based dynamic temperature sampling](#). *Preprint*, arXiv:2403.14541.
- Shimao Zhang, Xiao Liu, Xin Zhang, Junxiao Liu, Zheheng Luo, Shujian Huang, and Yeyun Gong. 2025a. [Process-based self-rewarding language models](#). *Preprint*, arXiv:2503.03746.
- Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. [The lessons of developing process reward models in mathematical reasoning](#). *Preprint*, arXiv:2501.07301.
- Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. [Processbench: Identifying process errors in mathematical reasoning](#). *Preprint*, arXiv:2412.06559.

A Complete Analysis and Generalization Results

A.1 Analysis of Performance on the AIME24 Benchmark

In our main results, PATS shows a noteworthy performance on the challenging AIME24 benchmark, where it outperforms the All-complex strategy. This effect is not due to randomness but is a result of our **one-time penalty** mechanism, which helps correct severe procedural errors. To validate this and demonstrate that the benefits originate from "adaptive switching" rather than solely from the "penalty mechanism," we extended the penalty mechanism to all fixed-mode baselines on the AIME24 dataset, with the comprehensive ablation results presented in Table 7.

Setting	Acc $\uparrow$	Token $\downarrow$
All-simple w/ one-time penalty	16.7	3238.5
All-medium w/ one-time penalty	20.0	6217.8
All-complex w/ one-time penalty	26.7	10017.3
PATS (Ours)	23.3	5821.0

Table 7: Performance on AIME24 with the penalty mechanism uniformly applied to all baseline settings.

As the results show, adding the penalty to All-complex increases its accuracy over PATS. Furthermore, it can be observed that even after incorporating the penalty mechanism into all baselines, PATS consistently maintains a superior accuracy-efficiency balance, demonstrating that the primary gains are derived from the adaptive switching itself. The penalty mechanism is merely a safeguard for exceptional cases during the switching process, triggered only when significant errors occur.

Furthermore, to confirm that the overall advantages of PATS do not solely stem from the AIME24 dataset, we recalculated the average performance across all methods with the AIME24 results excluded. As shown in Table 8, PATS still exhibits an excellent accuracy-efficiency balance, which confirms the robustness of our conclusions.

Setting	Average Acc $\uparrow$	Average Token $\downarrow$
All-simple	65.2	1173.9
All-medium	68.8	2243.5
All-complex	72.0	4161.8
★ PATS (Ours)	70.9	2054.8

Table 8: Comparison of average performance with the AIME24 dataset excluded.

A.2 Complete Results for the Penalty Study

In PATS, steps with PRM scores below a threshold $value_{bad}$ are classified as bad steps. To prevent error propagation, we apply a one-time penalty by rethinking these steps with the complex mode. Table 9 presents a detailed comparison of three strategies across all five benchmarks:

- **No Penalty:** Bad steps are left unprocessed.
- **Infinite Penalty:** Bad steps are rethought repeatedly in complex mode until the score exceeds the threshold.
- **PATS:** Our proposed one-time penalty method.

The results indicate that PATS strikes the best balance, achieving the highest accuracy (+5.4 over No Penalty and +1.3 over Infinite Penalty) while maintaining significantly lower token usage than Infinite Penalty and comparable usage to No Penalty. This demonstrates the importance of penalizing suboptimal reasoning steps to prevent delayed correction, while also underscoring the need for moderation to avoid excessive rethinking on unresolvable steps, which would otherwise lead to unnecessary token consumption.

A.3 Complete Results of the Generalization Experiments

We report the complete results of generalization experiments conducted on policy models of varying scales and families, as well as on different process reward models. The complete results for the policy model generalization experiments are presented in Table 10, and the complete results for the process reward model generalization experiments are presented in Table 11.

Note that PATS targets a better accuracy-efficiency balance in average performance rather than All-complex-level accuracy on every single benchmark: on harder benchmarks (e.g., AMC23) or smaller models (e.g., Qwen2.5-1.5B), it is reasonable if All-complex obtains higher accuracy, since it uses the maximum reasoning intensity at every step. Near-parity is not limited to the 7B setting: on Qwen2.5-14B-Instruct, PATS also approaches All-complex (62.8 vs. 63.2) with only 46.4% of its tokens. We also observe the high All-complex token count on AIME24 with Llama-3.1-8B (34,289.7) and faithfully report this result, which does not affect the main conclusion of PATS.

Setting	GSM8K		MATH500		AMC23		MinervaMATH		AIME24		Average	
	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$
PATS-No-Penalty	93.9	829.4	79.0	1931.3	47.5	2604.0	42.6	1836.8	16.7	4612.9	55.9	2362.9
PATS-Infinite-Penalty	95.0	920.3	81.4	2307.8	60.0	3880.9	43.8	2188.5	20.0	11621.5	60.0	4183.8
★ PATS (Ours)	94.8	855.8	80.6	2067.7	65.0	3365.7	43.0	1929.9	23.3	5821.0	61.3	2808.0

Table 9: Complete results for the analysis on the effect of penalizing bad reasoning steps. Based on the average metrics, PATS achieves the best balance—significantly improving accuracy while avoiding the high token cost on unresolvable steps.

Overall, across all different experimental settings, the conclusion remains consistent: PATS consistently achieves high accuracy with moderate token consumption, realizing an excellent accuracy-efficiency balance.

A.4 Dataset Sampling Details for Domain Generalization

For the cross-domain generalization experiments, we utilized **MMLU-Pro-CoT-Eval**, a multi-domain evaluation dataset. The reasoning process was guided by **VersaPRM**, a multi-domain PRM. To ensure a balanced evaluation across non-mathematical reasoning tasks, we excluded the *Math* category and selected the remaining 13 domains: *Physics, Chemistry, Law, Engineering, Other, Economics, Health, Psychology, Business, Biology, Philosophy, Computer Science, and History*. From each of these 13 domains, we randomly sampled 76 examples, resulting in a total evaluation set of 988 queries. This balanced sampling strategy ensures that the reported average performance metrics reflect PATS’s generalization capabilities in non-mathematical domains.

A.5 Threshold Sensitivity Analysis

We investigate the sensitivity of PATS to the hyperparameters used for mode switching. Specifically, we perturb the three key thresholds: $\text{value}_{\text{good}}$ (default 0.85), $\text{value}_{\text{low}}$ (default 0.75), and $\text{value}_{\text{bad}}$ (default 0.40). We test variations of ± 0.05 for each threshold while keeping the others fixed.

Table 12 reports the detailed performance across all five benchmarks. The results indicate that the performance of PATS is largely insensitive to small threshold perturbations. Overall, the variations remain within acceptable bounds (accuracy: -4.54% to $+0.03\%$, output tokens: -1.59% to $+2.05\%$). Crucially, across all perturbations, PATS consistently maintains a superior accuracy-efficiency balance compared to fixed-mode baselines (All-simple/medium/complex), confirming

that our adaptive paradigm is robust and not dependent on precise hyperparameter tuning.

A.6 Detailed Results for Comparisons with Other Inference Scaling Methods

We provide the comprehensive dataset-by-dataset evaluation breakdown for the comparisons against REBASE (Wu et al., 2025) and AB-MCTS (Inoue et al., 2025). Table 13 details the specific accuracy and token consumption metrics across all five mathematical reasoning benchmarks. The results show that PATS consistently delivers competitive or superior performance across individual tasks while maintaining a highly compact token footprint.

A.7 Multi-Seed Robustness Analysis

The random seed for the main experiments is 42. We further conduct three additional random-seed experiments, and the averaged results across all benchmarks are shown in Table 14, which show that PATS consistently maintains a strong accuracy-efficiency balance.

Across the four random-seed runs, PATS achieves an average accuracy of 61.1 ± 0.5 (variance 0.27) with an average token usage of 2893.3 ± 146.8 . It remains comparable to All-complex in accuracy (61.3 ± 0.5), with an average accuracy gap of only 0.25 ± 0.06 (variance 0.003), while using only 55.3% of All-complex tokens on average.

A.8 PRM Scoring Overhead

We count the average number of PRM calls for each setting in the main experiments, and the results are shown in Table 15.

As shown in Table 15, PATS uses only 51.8% of the PRM calls of All-complex, while achieving near-All-complex accuracy. Therefore, even when PRM scoring overhead is considered, our conclusion remains unchanged: PATS consistently maintains a strong accuracy-efficiency balance.

Setting	GSM8K		MATH500		AMC23		MinervaMATH		AIME24		Average	
	Acc ↑	Token ↓	Acc ↑	Token ↓	Acc ↑	Token ↓	Acc ↑	Token ↓	Acc ↑	Token ↓	Acc ↑	Token ↓
Qwen2.5-1.5B-Instruct												
All-simple	69.0	574.9	57.0	1343.3	32.5	1874.0	16.5	1244.7	6.7	2705.7	36.3	1548.5
All-medium	79.8	1188.8	66.6	2667.2	30.0	4030.1	17.3	2647.8	6.7	5198.1	40.1	3146.4
All-complex	84.8	2345.3	69.4	5519.0	45.0	7772.2	24.3	5507.7	10.0	10744.3	46.7	6377.7
★ PATS (Ours)	83.2	1387.5	66.8	3127.7	37.5	4757.9	22.4	3531.4	3.3	7166.1	42.6	3994.1
Qwen2.5-3B-Instruct												
All-simple	88.6	616.1	69.4	1235.1	50.0	1758.0	33.8	1431.5	6.7	2145.2	49.7	1437.2
All-medium	92.3	1185.2	74.4	2420.1	45.0	3609.1	37.5	2903.2	6.7	4862.0	51.2	2995.9
All-complex	93.7	2203.8	75.8	4819.7	52.5	6939.7	37.1	5951.5	13.3	10718.8	54.5	6126.7
★ PATS (Ours)	92.7	1126.6	75.8	2376.2	55.0	4083.7	34.9	2969.2	6.7	5886.6	53.0	3288.5
Qwen2.5-14B-Instruct												
All-simple	94.9	584.4	79.4	1145.3	62.5	1738.1	47.1	1269.6	13.3	2182.7	59.4	1384.0
All-medium	95.5	1100.6	81.6	2212.0	65.0	3384.0	48.5	2504.2	13.3	4837.4	60.8	2807.6
All-complex	96.0	2036.7	85.2	4322.6	67.5	6693.5	47.4	5052.6	20.0	8742.0	63.2	5369.5
★ PATS (Ours)	96.0	1094.6	82.2	1979.5	67.5	2959.1	48.2	1954.6	20.0	4469.0	62.8	2491.4
Llama-3.1-8B-Instruct												
All-simple	56.0	410.3	40.0	1125.2	22.5	1691.2	16.9	1104.6	10.0	4202.0	29.1	1706.7
All-medium	66.9	913.9	48.8	2480.3	37.5	3752.4	19.9	2480.8	6.7	14247.2	36.0	4774.9
All-complex	75.0	1592.7	53.2	4540.8	32.5	7011.1	25.7	5029.9	6.7	34289.7	38.6	10492.8
★ PATS (Ours)	70.0	797.5	43.0	1996.9	40.0	3059.8	22.8	2146.6	10.0	13616.6	37.2	4323.5

Table 10: Generalization of PATS for policy models across varying scales and families. The average results demonstrate that PATS consistently maintains an excellent accuracy-efficiency balance, indicating that our proposed adaptive paradigm scales effectively and is not limited to a specific model architecture.

Setting	GSM8K		MATH500		AMC23		MinervaMATH		AIME24		Average	
	Acc ↑	Token ↓	Acc ↑	Token ↓	Acc ↑	Token ↓	Acc ↑	Token ↓	Acc ↑	Token ↓	Acc ↑	Token ↓
Qwen2.5-Math-PRM-7B (Strong)												
All-simple	93.8	564.8	76.2	1154.3	52.5	1742.5	38.2	1234.3	16.7	2075.7	55.5	1354.3
All-medium	94.4	1001.0	80.2	2204.2	57.5	3337.3	43.0	2431.6	16.7	4113.3	58.4	2617.5
All-complex	94.9	1774.9	81.0	4068.9	67.5	6231.6	44.5	4571.9	20.0	8705.6	61.6	5070.6
★ PATS (Ours)	94.8	855.8	80.6	2067.7	65.0	3365.7	43.0	1929.9	23.3	5821.0	61.3	2808.0
Qwen2.5-Math-7B-PRM800K (Medium)												
All-simple	93.8	585.5	77.2	1144.5	52.5	1729.5	40.1	1267.2	13.3	2055.7	55.4	1356.5
All-medium	94.1	1062.6	77.4	2244.2	60.0	3546.0	36.8	2527.2	13.3	4336.8	56.3	2743.4
All-complex	94.2	1916.0	79.8	4295.2	62.5	6556.1	41.9	4843.5	13.3	8768.7	58.3	5275.9
★ PATS (Ours)	94.1	899.9	80.0	1954.1	55.0	3223.7	41.5	2071.8	16.7	3940.7	57.5	2418.0
Math-Shepherd (Weak)												
All-simple	92.1	543.8	75.2	1092.6	50.0	1799.9	40.1	1124.9	10.0	1872.6	53.5	1286.8
All-medium	92.8	924.7	77.4	1915.0	47.5	2970.1	39.3	2086.4	13.3	3457.9	54.1	2270.8
All-complex	91.7	1707.5	74.2	3424.8	50.0	5318.1	40.1	3853.3	16.7	6535.5	54.5	4167.8
★ PATS (Ours)	93.0	860.8	76.0	2216.5	57.5	3204.6	40.8	2020.1	10.0	5358.6	55.5	2732.1

Table 11: Complete results for the generalization experiments across Process Reward Models (PRMs) of varying quality, classified as **Strong**, **Medium**, and **Weak** based on the ProcessBench benchmark. The average results demonstrate the robustness of PATS, which consistently maintains a strong balance between accuracy and efficiency regardless of the PRM’s quality.

A.9 Comparison with Training-Based Methods

Although PATS is training-free, we compare it with training-based methods to better highlight its practical value. System-1.x (Saha et al., 2024) targets finite-state maze tasks and is not applicable to our mathematical reasoning scenarios where the state space is infinite, and AdaCoT (Lou et al.,

2025) is difficult to reproduce due to the lack of an open-source repository. Thus, we implement a Dualformer-style (Su et al., 2024) Qwen2.5-7B baseline, with results shown in Table 16.

Dualformer obtains fast/slow modes by switching system prompts after SFT, greatly reducing token usage but substantially degrading task performance, with an average accuracy far below PATS. In contrast, PATS performs step-level adaptive

Setting	GSM8K		MATH500		AMC23		MinervaMATH		AIME24		Average	
	Acc	Token	Acc	Token	Acc	Token	Acc	Token	Acc	Token	Acc	Token
PATS (Ours)	94.80	855.80	80.60	2067.70	65.00	3365.70	43.00	1929.90	23.30	5821.00	61.30	2808.00
PATS ($value_{good} = 0.8$)	94.90	853.20	80.20	2092.30	72.50	3179.50	39.00	2014.80	20.00	5841.40	61.32 ($\uparrow 0.03\%$)	2796.24 ($\downarrow 0.42\%$)
PATS ($value_{good} = 0.9$)	94.10	852.50	81.80	2094.79	62.50	3428.67	37.50	2008.00	23.30	5747.50	59.84 ($\downarrow 2.38\%$)	2826.29 ($\uparrow 0.65\%$)
PATS ($value_{low} = 0.7$)	94.90	844.50	79.20	2243.90	67.50	4252.20	37.10	1960.90	20.00	5026.90	59.74 ($\downarrow 2.54\%$)	2865.68 ($\uparrow 2.05\%$)
PATS ($value_{low} = 0.8$)	94.50	848.30	80.20	2157.10	60.00	3280.60	37.90	2085.10	20.00	5445.20	58.52 ($\downarrow 4.54\%$)	2763.26 ($\downarrow 1.59\%$)
PATS ($value_{bad} = 0.45$)	94.10	849.40	81.40	2189.60	62.50	3426.90	37.10	2005.60	23.30	5532.60	59.68 ($\downarrow 2.64\%$)	2800.82 ($\downarrow 0.26\%$)
PATS ($value_{bad} = 0.35$)	94.50	838.80	81.60	2120.71	67.50	3322.60	37.10	2001.40	20.00	5752.97	60.14 ($\downarrow 1.89\%$)	2807.30 ($\downarrow 0.03\%$)

Table 12: Sensitivity analysis of PATS thresholds (± 0.05 perturbations). The values in parentheses indicate the fluctuation percentage relative to the default PATS settings. **Bold** and underlined values denote the maximum and minimum fluctuation bounds observed. PATS shows minimal fluctuation and consistently outperforms fixed-mode baselines in accuracy-efficiency balance across all perturbations.

	GSM8K		MATH500		AMC23		MinervaMATH		AIME24		Average	
	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$
REBASE	95.4	2324.0	82.6	4954.0	60.0	7865.0	41.9	5283.0	16.7	9205.0	59.3	5926.2
AB-MCTS	93.2	2443.0	80.8	4740.0	60.0	6944.0	41.7	5058.0	13.3	9054.0	57.8	5647.8
★ PATS (Ours)	94.8	855.8	80.6	2067.7	65.0	3365.7	43.0	1929.9	23.3	5821.0	61.3	2808.0

Table 13: Comparison with other inference scaling methods on five math reasoning benchmarks.

Seed	All-simple		All-medium		All-complex		PATS (Ours)	
	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$
42	55.5	1354.3	58.4	2617.5	61.6	5070.6	61.3	2808.0
123	53.6	1335.3	56.3	2905.8	61.8	5217.2	61.6	3112.8
2024	57.2	1369.0	57.9	2675.6	61.2	5099.5	60.9	2835.2
7	56.4	1318.1	57.8	2706.4	60.6	5555.3	60.4	2817.1

Table 14: Averaged experimental results under different random seeds.

Setting	Avg. Acc $\uparrow$	Avg. PRM Calls $\downarrow$
All-simple	55.5	17.0
All-medium	58.4	32.3
All-complex	61.6	60.8
★ PATS (Ours)	61.3	31.5

Table 15: Average accuracy and average number of PRM calls for each setting in the main experiments.

Method	Avg. Acc $\uparrow$	Avg. Token $\downarrow$
Dualformer-fast	14.6	12.3
Dualformer-slow	42.2	553.1
★ PATS (Ours)	61.3	2808.0

Table 16: Average performance comparison between PATS and Dualformer.

switching at inference time without requiring any training.

B Additional Analysis

B.1 Performance of Initial Thinking Modes Across Varying Task Difficulties

We investigate the impact of initiating reasoning with different thinking modes on model performance across tasks of varying difficulty. Based on greedy decoding accuracy, we categorize datasets into three levels of difficulty: easy, medium, and hard. PATS defaults to initiating the first step with the complex thinking mode. Building on this design, we further investigate the effects of starting with simple and medium thinking modes. As shown in Table 17, we analyze the trade-off between accuracy and token efficiency. The results indicate that more difficult tasks derive greater benefit from initiating reasoning with more complex thinking modes. For easy tasks, initiating with the simple mode achieves the lowest token usage with comparable accuracy. For medium tasks, the medium mode achieves the highest accuracy with moderate token usage. For hard tasks, the complex

mode clearly outperforms the others in terms of accuracy. These findings suggest that aligning the initial thinking mode with task difficulty effectively balances accuracy and computational efficiency.

B.2 Reasoning Behavior across Task Difficulties

We compare the reasoning processes across tasks of varying difficulty levels, selecting MATH500 as a representative of easier tasks and AMC23 for harder ones. To enable a meaningful comparison of reasoning trajectories across problems with differing numbers of steps, we normalize the reasoning process to the range $[0, 1]$.

As shown in Figure 3, more complex modes indicate greater reasoning difficulty and effort used. For equally correct solutions, AMC23 exhibits a higher proportion of complex thinking modes during the mid-to-late stages of reasoning compared to MATH500, reflecting increased cognitive effort and aligning with the greater complexity of AMC23 problems. This shows that harder tasks require more reasoning effort to achieve correct solutions, underscoring that greater reasoning difficulty necessitates more computational resources to ensure final answer correctness.

Setting	Easy				Medium				Hard	
	GSM8K		MATH500		AMC23		MinervaMATH		AIME24	
	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$	Acc $\uparrow$	Token $\downarrow$
PATS-first-simple	94.4	600.1	80.2	1582.0	62.5	2400.1	43.0	1648.4	16.7	3998.5
PATS-first-medium	94.7	644.1	80.2	1687.8	65.0	3349.2	44.1	1633.9	16.7	5776.0
★ PATS (Ours)	94.8	855.8	80.6	2067.7	65.0	3365.7	43.0	1929.9	23.3	5821.0

Table 17: Performance of different initial thinking modes across tasks of varying difficulty. Results show that aligning the initial thinking mode with task difficulty leads to better performance.

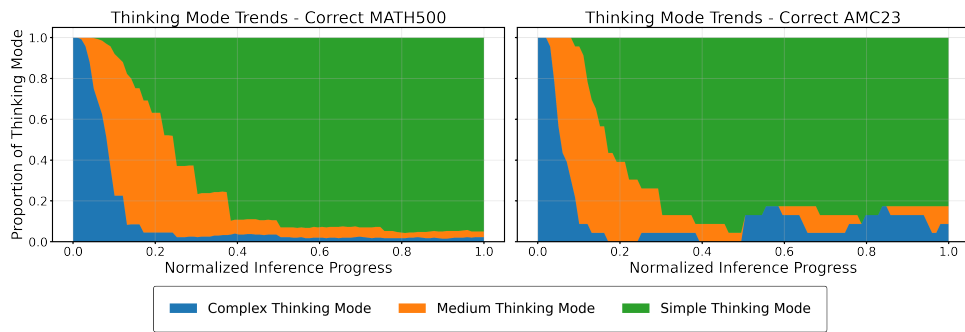


Figure 3: Thinking mode distributions over the reasoning process for MATH500 (easier) and AMC23 (harder). The harder task shows increased use of more complex modes in later stages, reflecting greater reasoning effort.