SmartThinker: Learning High-Density Reasoning via Step-Level Token Allocation

Xingyang He[1,2] **Jie Liu**[1,†] **Xiao Ling**[1]
[1]National Key Laboratory of Intelligent Tracking and Forecasting for Infectious Diseases, Engineering Research Center of Trusted Behavior Intelligence, Ministry of Education, Tianjin Key Laboratory of Intelligent Robotic (tKILIR), Institute of Robotics and Automatic Information System (IRAIS), Nankai University
[2]Shanghai Innovation Institute
{xingyanghe,lingxiao}@mail.nankai.edu.cn, jliu@nankai.edu.cn

Abstract

Efficient reasoning is critical for large reasoning models (LRMs) deployed in agent systems with strict context constraints. Existing methods predominantly rely on global length penalties that overlook step-level structure, inadvertently compressing critical derivations while retaining low-value detours. Hence, we argue for a principled, step-aware alternative that preserves critical steps and compresses low-value detours. However, it raises three fundamental challenges: the difficulty of obtaining step-level supervision, the need for reward signals that induce differentiated length pressure, and the requirement to balance training stability with exploration on complex problems. We propose SmartThinker, a step-level token allocation framework that learns to selectively preserve or compress individual reasoning steps. At its core is Step-Level Length Control Policy Optimization (SCPO), which provides label-free feedback by online estimation of each step's contribution from the change in correct-answer likelihood after removing it. We further introduce reasoning density, defined as contribution per token, to relax length pressure on high-value steps while intensifying it on low-density ones. Finally, SCPO stabilizes training through length-bias-corrected advantages and difficulty-adaptive clipping, achieving superior accuracy-efficiency trade-offs across nine in-domain and out-of-domain benchmarks.

1 Introduction

Efficient reasoning is critical for deploying large reasoning models (LRMs) in agentic frameworks. Although longer reasoning trajectories can improve complex problem solving (Guo et al., 2025; Jaech et al., 2024; Yang et al., 2024; Luo et al., 2024), they often include redundant verification and repetitive reflection (Ye et al., 2025; Kumar et al., 2025; Fu et al., 2024; Sui et al., 2025; Qu et al., 2025), reducing agent throughput and crowding out contextual evidence needed for downstream decisions. Prior works primarily improve reasoning efficiency through global output shortening, either by imposing global length penalties during reinforcement learning (RL) (Aggarwal and Welleck, 2025; Arora and Zanette, 2025; Hou et al., 2025; Zhang and Zuo, 2025) or by routing queries to short- or long-reasoning modes based on estimated difficulty (Fang et al., 2025; Zhang et al., 2025a; Lou et al., 2025; Luo et al., 2025a).

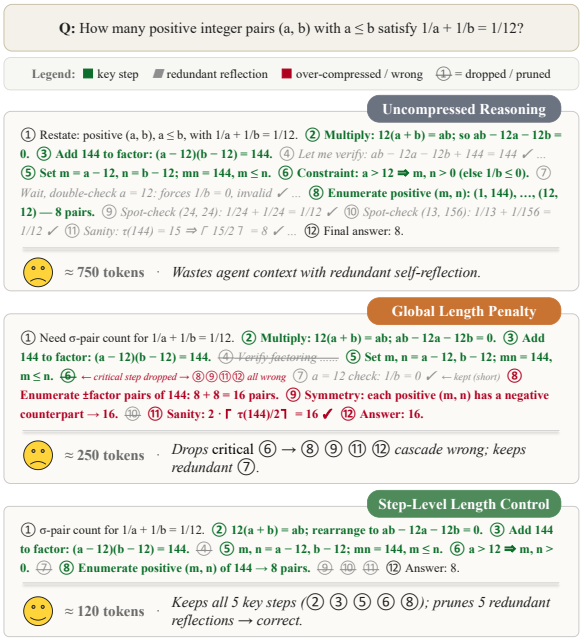


Figure 1: Comparison of reasoning traces under different length-control settings. Uncompressed reasoning wastes context on redundant detours; global length penalties shorten responses uniformly and can remove critical derivations; Step-level length control compresses low-value steps while preserving critical ones.

Despite their effectiveness, global-level methods treat all reasoning steps as equally compressible and ignore step-level differences, which leads to

[†]Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 13409–13428
October 24-29, 2026 ©2026 Association for Computational Linguistics

compressing critical derivations while preserving low-value ones. Figure 1 gives an intuitive example. Without compression, the model spends many tokens on redundant reasoning that consumes context. A global length penalty reduces the overall chain length, but may remove critical derivations together with redundant text. In contrast, step-level length control can preserve the steps that support the final answer while compressing those with little contribution. However, step-level optimization still faces three key challenges. First, step-level supervision is difficult to obtain because human annotations are expensive and external reward models introduce additional training cost and bias. Second, even if a step's contribution can be estimated, it must be converted into differentiated length pressure so that useful steps are protected and redundant ones are compressed. Third, training must remain stable while preserving exploration, since hard problems often require longer verification and correction.

To address these challenges, we propose SmartThinker, a step-level token allocation method that learns which reasoning steps should be preserved or compressed. SmartThinker contains two stages. The first stage uses rejection sampling and supervised fine-tuning (SFT) to warm up the model into a concise reasoning regime, reducing rollout cost and providing a stable initialization for later RL. The second stage introduces Step-Level Length Control Policy Optimization (SCPO), which performs fine-grained length control over individual reasoning steps. To provide label-free process feedback, SCPO estimates each step's answer contribution online by measuring how removing that step changes the likelihood of the correct answer. We further define reasoning density as answer contribution per generated token and uses it to guide the model to impose stronger penalties on low-density steps while relaxing constraints on high-density ones. To stabilize optimization, SCPO further applies length-bias-corrected advantage and difficulty-adaptive clipping, allowing the model to preserve stable reasoning paths on easier questions while maintaining exploration on harder ones.

We evaluate SmartThinker across nine in-domain and out-of-domain benchmarks. The results show that SmartThinker reduces redundant reasoning while outperforming efficient-reasoning baselines. In context-budgeted settings, the saved tokens translate into more useful reasoning opportunities, supporting the practical value of step-level token allocation.

In summary, our contributions are as follows:

- We introduce the perspective of reasoning density, reformulating efficient reasoning as a step-level token allocation problem and estimating each step's answer contribution through its effect on the probability of the correct answer.
- We propose SmartThinker, a two-stage framework for learning high-density reasoning flows, where SCPO enables fine-grained step-level length control without human process labels or external reward models.
- We systematically evaluate SmartThinker across in-domain, context-budgeted, out-of-domain, and ablation settings, showing that it reduces redundant reasoning while improving the accuracy-efficiency trade-off.

2 Related Work

Efficient and Adaptive Reasoning. Long reasoning traces can improve the problem-solving ability of LRMs, but they also introduce substantial redundancy and serving cost (Ye et al., 2025; Kumar et al., 2025; Fu et al., 2024; Sui et al., 2025; Qu et al., 2025). Existing work on efficient reasoning typically controls this cost at the global level. One line of work constructs concise reasoning data or trains models to write shorter intermediate derivations under a target budget (Xia et al., 2025; Han et al., 2024; Xu et al., 2025; Zhang et al., 2025b; Yan et al., 2025). Another line incorporates global length penalties into RL objectives, encouraging the model to reduce the total number of generated tokens (Aggarwal and Welleck, 2025; Arora and Zanette, 2025; Hou et al., 2025; Zhang and Zuo, 2025; Wu et al., 2025b; Huang et al., 2025; Liu et al., 2025a; Tu et al., 2025). A third line learns adaptive short/long reasoning behavior, either by selecting a reasoning mode or by triggering chain-of-thought only when the problem appears sufficiently difficult (Fang et al., 2025; Zhang et al., 2025a; Lou et al., 2025; Luo et al., 2025a). These methods improve efficiency, but their primary control signal is still defined over the whole response or the whole reasoning mode. SmartThinker is complementary: it focuses on how tokens are allocated within a reasoning trajectory, so that high-contribution steps can be preserved while low-contribution steps are selectively compressed.

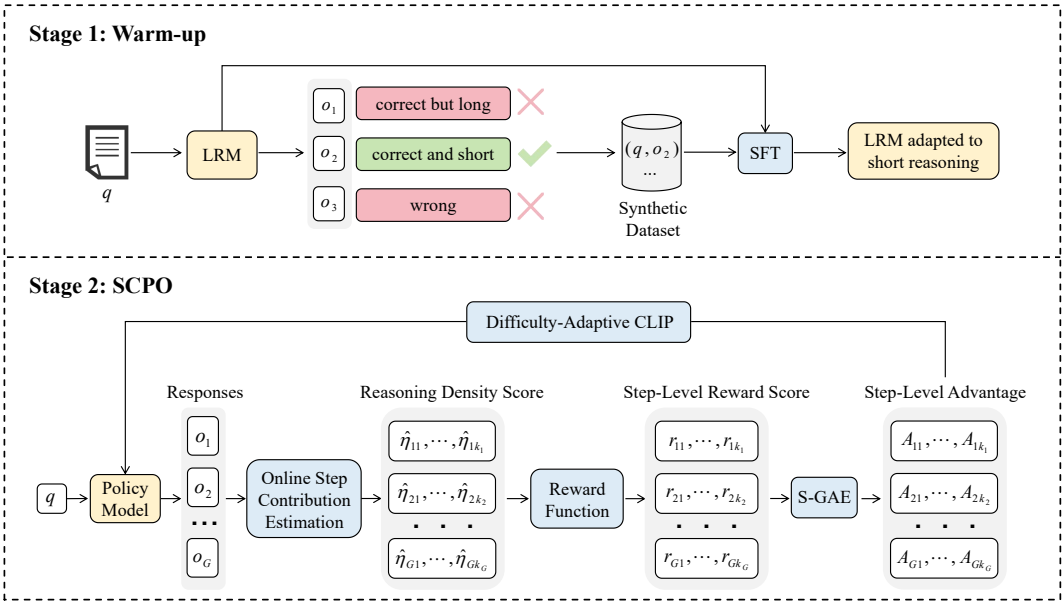


Figure 2: Overview of the proposed SmartThinker.

Process Supervision and Credit Assignment. Process supervision assigns feedback to intermediate steps rather than only to final answers (Lightman et al., 2024; Luo et al., 2024), making it closely related to credit assignment in long-chain reasoning. Existing methods often depend on human step labels, model-generated annotations, or a separate process reward model. SCPO instead estimates each step's answer contribution online by measuring how removing the step changes the probability of the correct answer. It then converts this estimate into step-level length-control rewards, linking process feedback to token efficiency: a step should be preserved only when its answer-relevant contribution justifies its token cost. Concurrent work also moves control signals below the trajectory level, including step-level advantage selection for stabilizing efficient reasoning (Wang et al., 2026), stepwise length penalization (Li et al., 2026a), and step-level credit assignment for RL training (Li et al., 2026b). SCPO differs in that the step score is an online, contribution-aware estimate of how much a step helps the final answer, and it is used to allocate tokens selectively within a single trajectory rather than to penalize steps or select steps globally.

Agent Harness and Context-Efficient Learning. Reasoning models are increasingly deployed in agentic systems, where they interact with tools, search procedures, retrieved memories, and environment feedback (Wu et al., 2025a; Chen et al., 2025). In such settings, reasoning tokens compete with observations, intermediate state, conversation history, and candidate plans within the same context window. Token efficiency is therefore not only a serving concern; it affects how much useful state an agent can retain while planning and acting. SmartThinker addresses this pressure at training time by encouraging denser reasoning trajectories, leaving more context budget for later tool calls, memory updates, and multi-branch search.

3 Method

SmartThinker learns high-density reasoning by dense process reward signals. As illustrated in Figure 2, it first uses a short-reasoning warm-up for stable initialization, and then applies Step-Level Length Control Policy Optimization (SCPO). SCPO does not optimize for uniformly shorter outputs; it learns to spend tokens on reasoning steps with high answer contribution per token. Algorithm 1 provides a compact overview of the SCPO update, and the following subsections unpack its estimation, reward, credit assignment, and clipping components.

3.1 Short-Reasoning Warm-up

Direct RL from a long-reasoning model is inefficient because early rollouts are often lengthy and low-density. We therefore use rejection sampling followed by supervised fine-tuning (SFT) as an initialization step. For each training question, the current model generates multiple candidate responses;

Algorithm 1 Step-Level Length Control Policy Optimization

Require: Question q, answer a, policy π_θ, old policy $\pi_{\theta_{\text{old}}}$, rollout size G

Sample G responses $\{o_i\}_{i=1}^{G}$ from $\pi_{\theta_{\text{old}}}$
Split each o_i into steps $S_i = \{s_{i,1}, \ldots, s_{i,n_i}\}$
Compute answer correctness and difficulty $\rho(q) = 1 - N_c/G$
for each step $s_{i,j}$ **do**
    Estimate raw contribution $m_{i,j}$ by removing $s_{i,j}$
    Compute density $\eta_{i,j}$ and normalized allocation score $\widehat{\eta}_{i,j}$
    Compute the step reward $r_{i,j}$ with density- and difficulty-aware penalties
end for
Normalize step rewards to obtain $\widetilde{r}_{i,j}$
Compute $A_{i,j}^{\text{S-GAE}} = \sum_{m=0}^{n_i-j} \gamma^m \widetilde{r}_{i,j+m}$
Assign $A_{i,j}^{\text{S-GAE}}$ to all tokens in step $s_{i,j}$
Set adaptive clipping bounds ε_{low} and $\varepsilon_{\text{high}}$
Update π_θ with the clipped SCPO objective using bounds $[1 - \varepsilon_{\text{low}}, 1 + \varepsilon_{\text{high}}]$

we retain the shortest correct response and fine-tune on the resulting synthetic data. Since the data come from the model itself, the warm-up stays close to the model distribution while moving the policy into a concise regime. Fine-grained length control is learned later by SCPO.

3.2 Online Step Contribution Estimation

For each training pair (q, a), SCPO samples G responses from the old policy. Each response o_i contains a reasoning trajectory and a final predicted answer:

$$o_i = (S_i, a_i), \quad S_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,n_i}\}, \quad (1)$$

where $s_{i,j}$ is the j-th reasoning step, $l_{i,j}$ is its token length, and n_i is the number of steps. SCPO uses this step structure to preserve useful reasoning while compressing low-value detours.

The sampled group also provides an online estimate of question difficulty:

$$\rho(q) = 1 - \frac{N_c}{G}, \quad (2)$$

where N_c is the number of correct responses. A larger $\rho(q)$ indicates a harder question for the current policy, and is used later to control both step-count penalty and exploration.

SCPO estimates each step's contribution online without human process labels or an external reward model. For a sampled trajectory S_i, let

$$p_1 = P_\theta(a \mid q, S_i), \quad (3)$$
$$p_2 = P_\theta(a \mid q, S_i \setminus \{s_{i,j}\}), \quad (4)$$

where p_1 is the probability of the correct answer given the full trajectory, and p_2 is the probability after removing step $s_{i,j}$. We define the raw contribution score as

$$m_{i,j} = \begin{cases} 0, & \text{if } p_1 \le p_2, \\ \dfrac{p_1^2 - p_2^2}{p_1^2}, & \text{if } p_1 > p_2. \end{cases} \quad (5)$$

The removal-based probability drop estimates the answer contribution of a step on answer correctness, while the squared difference sharpens the contrast between critical and ordinary steps. To account for token cost, we define reasoning density as contribution per token:

$$\eta_{i,j} = \frac{m_{i,j}}{l_{i,j}}, \quad (6)$$

Thus, a step receives relaxed length pressure only when its contribution is high relative to its length.

We add a lightweight safeguard for reflective steps on difficult questions. Steps containing transition markers such as "but" or "however" often indicate verification or correction. To avoid suppressing such behavior too aggressively, we define

$$\widetilde{\eta}_{i,j} = \eta_{i,j} + \rho(q) I(s_{i,j}), \quad (7)$$

where

$$I(s_{i,j}) = \begin{cases} \max_m \eta_{i,m}, & \text{if marker present,} \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

This term protects verification behavior only when the rollout set suggests that the question is difficult. Finally, we min-max normalize $\widetilde{\eta}_{i,j}$ within the same correctness group:

$$\widehat{\eta}_{i,j} = \frac{\widetilde{\eta}_{i,j} - \min(\mathcal{E}_{y_i})}{\max(\mathcal{E}_{y_i}) - \min(\mathcal{E}_{y_i}) + \epsilon}, \quad (9)$$

where $y_i = \mathbf{1}[a_i = a]$ and $\mathcal{E}_{y_i}$ denotes the set of density-based scores with the same correctness label as response i. The resulting $\widehat{\eta}_{i,j}$ is the allocation score used by the reward.

3.3 Step-Level Reward as Token Allocation

SCPO rewards correct responses while penalizing low-density length. The reward for step $s_{i,j}$ is

$$r_{i,j} = \mathbf{1}[a_i = a]\left(1 - k_1\sigma(\widetilde{l}_{i,j})\right) \cdot \left(1 - k_2\sigma(\widetilde{n}_i)\right). \tag{10}$$

Here, $\sigma(\cdot)$ is the sigmoid function and

$$k_1 = k_0(1 - \widehat{\eta}_{i,j}), \tag{11}$$

$$k_2 = k_0(1 - \rho(q)). \tag{12}$$

Here, k_0 is the base penalty coefficient. The density-aware coefficient k_1 gives low-density steps stronger length pressure, while the difficulty-aware coefficient k_2 penalizes excessive step counts more strongly on easier questions.

The normalized length and step-count terms are computed over correct responses:

$$\widetilde{l}_{i,j} = \frac{l_{i,j} - \mathrm{mean}_{\{i,j:a_i=a\}}(l_{i,j})}{\mathrm{std}_{\{i,j:a_i=a\}}(l_{i,j})}, \tag{13}$$

$$\widetilde{n}_i = \frac{n_i - \mathrm{mean}_{\{i:a_i=a\}}(n_i)}{\mathrm{std}_{\{i:a_i=a\}}(n_i)}. \tag{14}$$

Thus, SCPO avoids rewarding short but wrong trajectories, and converts compression into step-level token allocation rather than global shortening.

3.4 S-GAE for Step-Level Credit Assignment

Naively accumulating future step rewards gives longer correct responses larger advantages simply because they contain more rewarded steps (Shao et al., 2024). SCPO avoids this length bias with Step-Level Generalized Advantage Estimation (S-GAE).

We first normalize step rewards across the sampled group:

$$\widetilde{r}_{i,j} = \frac{r_{i,j} - \mathrm{mean}_{i,j}(r_{i,j})}{\mathrm{std}_{i,j}(r_{i,j})}. \tag{15}$$

The S-GAE advantage for step $s_{i,j}$ is then

$$A_{i,j}^{\text{S-GAE}} = \sum_{m=0}^{n_i-j} \gamma^m \widetilde{r}_{i,j+m}, \tag{16}$$

where $\gamma \in (0, 1]$ is a discount factor. The resulting advantage is assigned to all tokens in the corresponding step, providing dense feedback while keeping the step as the unit of credit assignment.

3.5 Difficulty-Adaptive Clipping as Exploration Control

Clip-Higher in DAPO (Yu et al., 2025) expands the upper clipping bound to mitigate entropy collapse and give low-probability tokens more room for exploration. This is useful for difficult problems, but it can be harmful for easy ones: when the model already follows a reliable reasoning path, allowing large probability increases for low-probability tokens may push it away from the correct solution. SCPO therefore keeps the clipped policy update but adapts the clipping range to the online difficulty estimate:

$$\varepsilon_{\text{low}} = \varepsilon - \delta_1(1 - \rho(q)), \tag{17}$$

$$\varepsilon_{\text{high}} = \varepsilon + \delta_2\rho(q), \tag{18}$$

where δ_1 and δ_2 control the adjustment magnitude. Easy questions receive tighter updates to preserve known correct paths, while difficult questions receive a larger upper bound to explore promising low-probability tokens.

4 Experiments

4.1 Models and Datasets

DeepSeek-R1-Distill-Qwen-1.5B, DeepScaleR-1.5B-Preview, and DeepSeek-R1-Distill-Qwen-7B are used as base models. We train these models on a randomly sampled subset of 3,000 QA pairs from the DeepScaleR-Preview dataset (Luo et al., 2025b). For evaluation, we use six in-domain benchmarks: AIME24, AIME25 (Veeraboina, 2023), AMC23, MATH (Lewkowycz et al., 2022), MinervaMATH (Hendrycks et al., 2021), Olympiad-Bench (He et al., 2024) and three out-of-domain benchmarks: Live-Code-Bench (Jain et al., 2024), GPQA-Diamond (Rein et al., 2023), and MuSR-MurderMysteries (MuSR-MM) (Sprague et al., 2024). Please see Appendix A.2 for more details.

We evaluate performance using three main metrics: (i) Pass@k, the probability that at least one of the top k samples is correct under a given decoding strategy, reflecting the upper bound of the model's reasoning ability. (ii) Maj@k, the accuracy obtained through majority voting across k samples, which measures the stability of the model's output. (iii) AvgLen, the average length of all sampled responses, indicating the model's reasoning efficiency. For AIME24, AIME25, and AMC23, we set $k = 10$ due to the small size of their test sets. For all other datasets, we use $k = 5$.

Table 1: Average in-domain accuracy-efficiency trade-off. Averages are computed over the six in-domain benchmarks. Red. denotes token reduction relative to the corresponding original backbone; Dens. denotes average Pass@k per 1K generated tokens.

Models	Pass@k	Maj@k	AvgLen.	Red.	Dens.
DeepSeek-R1-1.5B	61.88	51.77	5768	0.0	10.7
Alpha-0.1-1.5B	65.17	53.08	4406	23.6	14.8
Thinkless-1.5B	63.69	51.93	4362	24.4	14.6
SmartThinker-1.5B	**65.75**	**53.69**	**3651**	**36.7**	**18.0**
DeepScaleR-1.5B	66.53	**57.51**	4988	0.0	13.3
L1-Max-1.5B	64.33	53.98	2841	43.0	22.6
SmartThinker-ScaleR	**66.65**	55.38	**2370**	**52.5**	**28.1**
DeepSeek-R1-7B	71.51	63.08	4987	0.0	14.3
Alpha-0.1-7B	73.83	65.10	4061	18.6	18.2
AdaptThink-7B	73.40	65.04	3833	23.1	19.1
SmartThinker-7B	**75.26**	**66.59**	**3367**	**32.5**	**22.4**

Note that we do not use Pass@1 as our primary evaluation metric, since improvements in Pass@1 mainly reflect better output ranking (Zhu et al., 2025). In other words, the model becomes more skilled at placing the correct answer first rather than solving previously unsolvable problems. In contrast, Pass@k (for $k > 1$) allows multiple attempts, allowing the model to explore potentially correct answers even if the first attempt fails. This setting better reflects the model's true reasoning ability when sufficient exploration is allowed.

4.2 Baselines

The baselines we select can be grouped into two categories: (i) models based on global length penalty, including Alpha-0.1 (Arora and Zanette, 2025) and L1-Max (Aggarwal and Welleck, 2025), and (ii) models that incorporate both long- and short-reasoning modes, such as Thinkless (Fang et al., 2025) and AdaptThink (Zhang et al., 2025a).

4.3 Experiment Settings

The hyperparameters are set to $k_0 = 0.6$, $\gamma = 0.95$, $\delta_1 = 0.03$, and $\delta_2 = 0.08$. Detailed experimental settings are provided in Appendix A.4.

4.4 Main Results

In-Domain Reasoning. Table 1 summarizes the average in-domain trade-off, with full per-benchmark results reported in Appendix Tables 6 and 7. The central pattern is consistent across all backbone families: SmartThinker moves the model toward a better accuracy-efficiency frontier, improving reasoning accuracy while using substantially fewer tokens. On the DeepSeek-R1-Distill-Qwen-1.5B family, SmartThinker achieves the best average Pass@k and Maj@k among the efficient-reasoning baselines while reducing AvgLen by 37%. This gives it higher reasoning density than both Alpha-0.1 and Thinkless, showing that the model is not merely learning to truncate its output, but to remove reasoning segments that contribute little to the final answer.

The same conclusion holds in the more demanding DeepScaleR setting, where the base model is already stronger and L1-Max already applies aggressive length control. SmartThinker keeps the accuracy of the strong base model essentially intact, slightly improving average Pass@k from 66.53% to 66.65%, while cutting AvgLen from 4988 to 2370 tokens. It also improves over L1-Max under a smaller token budget, yielding the highest density. These results indicate that step-level allocation provides a signal that coarse length control misses: useful derivations should be protected even under strong compression, while low-density continuation can be safely pruned. At 7B scale, SmartThinker again improves both Pass@k and Maj@k over the original backbone while reducing average length by 32.5%. Taken together, the in-

Table 2: Context-budgeted AIME24 evaluation. Each entry reports the average number of correctly solved problems before the cumulative chat context reaches budget B.

Models	4K	8K	16K	32K
DeepSeek-R1-7B	0.20	0.80	1.60	4.20
Alpha-0.1-7B	0.20	0.80	3.20	3.60
AdaptThink-7B	**0.60**	0.80	2.80	3.40
SmartThinker-7B	**0.60**	0.80	**3.40**	**5.60**

Table 3: Out-of-domain Pass@1 accuracy (%). LCB, GPQA, and MuSR denote Live-Code-Bench, GPQA-Diamond, and MuSR-MM, respectively.

Models	LCB	GPQA	MuSR	Avg.
DeepSeek-R1-1.5B	22.00	16.67	36.80	25.16
Alpha-0.1-1.5B	28.00	21.72	49.60	33.11
Thinkless-1.5B	22.50	16.16	33.20	23.95
SmartThinker-1.5B	**32.00**	**23.74**	**52.00**	**35.91**
DeepScaleR-1.5B	31.00	22.73	48.40	34.04
L1-Max-1.5B	31.50	34.37	54.00	39.96
SmartThinker-ScaleR	**33.00**	**35.83**	**55.60**	**41.48**
DeepSeek-R1-7B	45.50	34.34	54.80	44.88
Alpha-0.1-7B	49.00	38.89	57.60	48.50
AdaptThink-7B	50.00	39.90	54.00	47.97
SmartThinker-7B	**54.00**	**44.44**	**60.80**	**53.08**

domain results support the main hypothesis of this work: efficient reasoning is best achieved by reallocating tokens within the reasoning trace, rather than by applying uniform pressure to the whole response. Section 5.2 further confirms that these gains cannot be explained by short-context training, the SFT warm-up initialization, or differences in the training recipe.

Context-Budgeted Reasoning. Table 2 tests whether higher-density reasoning translates into higher effective throughput under a fixed context budget. We simulate a harness-style setting in which a model answers a shuffled sequence of AIME24 problems in one growing chat and stops when the cumulative context reaches budget B. The metric is the number of problems solved before the budget is exhausted. Please see Appendix A.9 for further experimental details.

At very small budgets, all models are bottlenecked by the first few attempts, so the differences are limited. Once the budget allows multiple full attempts, the benefit of step-level allocation becomes clear. At $B = 16$K, SmartThinker-7B solves 3.40 problems on average, more than DeepSeek-R1-7B and the efficient-reasoning baselines. At $B = 32$K, the gap widens: SmartThinker-7B solves 5.60 problems, compared with 4.20 for DeepSeek-R1-7B, 3.60 for Alpha-0.1-7B, and 3.40 for AdaptThink-7B. This pattern shows that SmartThinker uses the same context window to create more useful reasoning opportunities while preserving enough reasoning quality for those additional attempts to become correct answers. This is the operational value of reasoning density in agent harnesses: tokens saved from low-contribution detours are converted into additional useful reasoning opportunities, not merely shorter transcripts.

Out-of-Domain Generalization. Table 3 evaluates whether the high-density reasoning policy learned from mathematical training transfers to other reasoning domains. The answer is consistently positive: across three OOD benchmarks, SmartThinker achieves the best average accuracy within each backbone family. The improvements are not marginal. SmartThinker-1.5B raises the average accuracy from 25.16 to 35.91 over the original DeepSeek-R1 backbone and SmartThinker-7B reaches 53.08 average accuracy, outperforming both efficient-reasoning baselines by more than four points. These results suggest that SCPO does not simply fit math-specific answer formats or benchmark-specific shortcuts. Instead, it learns a more general reasoning behavior: allocating more tokens to steps that are consequential for the final answer while compressing steps that are not. Because this principle is domain-agnostic, the resulting compact reasoning policy remains useful when the task shifts from mathematical derivation to program synthesis, scientific knowledge reasoning, and long-context commonsense inference. The OOD gains therefore strengthen the main claim of SmartThinker: high-density reasoning improves not only efficiency, but also the reliability of reasoning across domains.

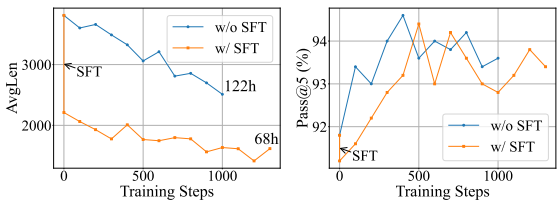


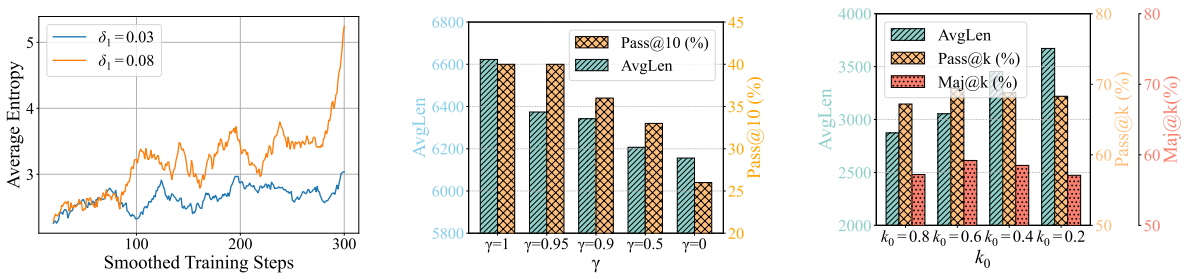
Figure 3: Pass@5 accuracy and AvgLen at different training steps on the MATH dataset.

Training Overhead. The step-contribution estimator is implemented efficiently: all leave-one-step-out sequences are constructed in parallel, dynamically packed across rollouts into a single batch, and scored with one batched prefill-only forward pass using vLLM (Kwon et al., 2023), completely avoiding autoregressive decoding (implementation details and full timings are given in Appendix A.8). One complete contribution evaluation over eight rollouts of up to 4,096 tokens each takes only 0.60s on the 1.5B model and 4.06s on the 7B model, and end-to-end SCPO training adds less than 8% per-step overhead over plain GRPO. Since the estimator runs only during training and leaves the decoding path untouched, SmartThinker incurs zero inference-time cost: this one-time training overhead is small and practically acceptable, and is repaid many times over by the sustained inference-efficiency gains reported above.

5 Ablation Study

5.1 Module Ablation Analysis

Table 4 isolates the contribution of each SmartThinker component on DeepSeek-R1-Distill-Qwen-7B. The SFT warm-up primarily shifts the policy into a shorter-response regime. Although it substantially improves reasoning density, it does



(a) (b) (c)

Figure 4: Hyperparameter selection.

Table 4: Cumulative module-level ablation on DeepSeek-R1-Distill-Qwen-7B. Results are averaged over the six in-domain benchmarks. Each "+" row adds the corresponding component to the previous setting. Dens. denotes average Pass@k per 1K generated tokens.

Setting	Pass@k	AvgLen	Dens.
DeepSeek-R1-7B	71.51	4987	14.3
+ SFT warm-up	71.13	4001	17.8
+ Step contribution estimator	73.38	3673	20.0
+ Lightweight safeguard	74.92	3861	19.4
+ S-GAE	74.83	3452	21.7
+ Difficulty-adaptive clipping	**75.26**	**3367**	**22.4**

not improve accuracy on its own. Figure 3 further corroborates this interpretation: with SFT warm-up, SCPO attains comparable accuracy at a substantially lower training cost, completing 1,300 RL steps in 68 hours, as opposed to 1,000 steps in 122 hours without SFT. These findings indicate that the warm-up is best viewed as an initialization procedure that reduces rollout cost and better prepares the model for subsequent RL.

The online step contribution estimator produces the first clear accuracy improvement while further reducing response length. This result supports the central design choice of SCPO: reasoning quality improves only when the reward signal can distinguish high-density steps from low-value detours. The lightweight safeguard then raises accuracy at the cost of a small length increase, suggesting that some verification and self-correction steps are worth preserving, especially on difficult problems. Appendix A.10 further analyzes this behavior and shows that the lightweight safeguard prevents useful reflective steps from being removed too aggressively. This supports a more nuanced view of efficiency: not every verification and reflection step is redundant; the key question is whether the step supports answer-changing reasoning.

S-GAE restores a stronger efficiency gain while maintaining nearly the same accuracy, indicating that step-level credit assignment helps remove the length bias introduced by accumulating rewards over many steps. Finally, difficulty-adaptive clipping yields the best overall trade-off, showing that exploration should be relaxed on hard questions but constrained on easier ones. Appendix A.11 further supports this interpretation by showing that the model's entropy tracks problem difficulty during training. Taken together, the cumulative ablation shows that SmartThinker does not work through indiscriminate shortening. Its advantage comes from progressively turning length control into step-level token allocation, which improves both accuracy and reasoning density.

5.2 Controlled Comparisons under Identical Warm-up and Recipe

Our gains could in principle come from the warm-up or the 4K recipe rather than SCPO itself. To isolate the RL objective, we retrain every baseline from the same SFT checkpoint with identical data, context, optimization, and training steps; only the RL objective differs. Table 5 shows the pattern: the shortest-answer warm-up compresses every method, but most baselines trade accuracy for brevity, whereas SCPO dominates on both axes. The LAPO pair further shows that adding global length learning on top of this prior hurts accuracy: LAPO trails the released LAPO-I (Wu et al., 2025b) by 2.69 Pass@k points.

Shortest-correct SFT is a strong compression prior: it reduces redundant unrolling and RL exploration cost, but it also narrows the set of correct long-form trajectories. On hard problems that require search, backtracking, or verification, global length penalties or fixed-budget objectives can fur-

Table 5: Controlled comparisons under identical SFT warm-up, training data, 4K context, and training steps. Only the RL objective differs across rows; results are averaged over the six in-domain benchmarks. †LAPO-I is the publicly released checkpoint trained without our SFT warm-up.

Models	Pass@k	AvgLen.	Red.	Dens.
SFT+Alpha-0.1-1.5B	65.47	3991	30.8	16.4
SFT+Thinkless-1.5B	60.59	3897	32.4	15.6
SmartThinker-1.5B	**65.75**	**3651**	**36.7**	**18.0**
SFT+GRPO-ScaleR	64.07	3030	39.3	21.1
SFT+L1-Max-1.5B	63.59	2460	50.7	25.8
LAPO-I-ScaleR†	**68.94**	3710	25.6	18.6
SFT+LAPO-ScaleR	66.25	2896	41.9	22.9
SmartThinker-ScaleR	66.65	**2370**	**52.5**	**28.1**
SFT+GRPO-7B	73.65	3687	26.1	20.0
SmartThinker-7B	**75.26**	**3367**	**32.5**	**22.4**

ther suppress key derivations. SCPO recovers accuracy by separating redundant from critical steps with step-level rewards, while its lightweight safeguard and Difficulty-Adaptive Clipping preserve the exploration needed on difficult problems.

5.3 Entropy-Stable Clipping

Following DAPO (Yu et al., 2025), we set δ_2 to 0.08 and compare the effects of a larger δ_1 with the smaller value used in this work. As shown in Figure 4a, a larger δ_1 raises entropy but eventually causes entropy collapse, as excessive exploration during moderately complex tasks drives the model off the correct reasoning path. As training continues, this drift escalates, destabilizing training. In contrast, a smaller δ_1 curbs entropy growth, maintaining a healthier balance between exploration and exploitation across problem difficulties.

5.4 S-GAE Reduces Length Bias

As shown in Figure 4b, when $\gamma = 1$, the model tends to generate excessively long sequences. As γ decreases, the average sequence length also shortens. However, setting γ too low greatly reduces the contribution of later steps to the overall advantage, leading to less accurate advantage estimates and potentially harming model performance. Balancing these trade-offs, we set $\gamma = 0.95$, which effectively mitigates length bias while maintaining stable performance.

5.5 Balanced Length Pressure

Figure 4c studies the effect of the base penalty coefficient k_0, which scales both the density-aware coefficient k_1 and the difficulty-aware coefficient k_2 in the SCPO reward. A large k_0 produces shorter responses, but the pressure becomes too aggressive and can remove useful derivations steps along with redundant self-reflection. A small k_0 has the opposite limitation: because both k_1 and k_2 are weakened, the reward has insufficient leverage to prune low-density steps or discourage excessive step counts on easy questions. We therefore set $k_0 = 0.6$, which gives enough pressure to compress low-value reasoning while preserving steps whose contribution justifies their token cost. This supports the central design of SmartThinker: efficiency gains should come from selective token allocation rather than uniform shortening.

6 Conclusion

SmartThinker treats efficient reasoning as allocating tokens rather than simply using fewer of them: reasoning density is uneven, with a few steps carrying most of the evidence and many tokens adding little. Using this signal, SmartThinker compresses low-value detours while preserving correctness-critical derivations. This matters especially in agent harnesses, where reasoning shares a bounded context window with tools, memory, and state. Efficient reasoning should therefore be judged by whether the model spends its budget on the steps that matter.

7 Limitations

Our method assumes that a reasoning trajectory can be decomposed into discrete steps, which are then used to estimate contribution scores and assign rewards. In this study, we use a fixed delimiter based on `"\n\n"` to define these steps. Although this strategy is simple and consistent with the formatting of many generated reasoning traces, we have not systematically examined how alternative granularities, such as sentence level units, semantic chunks, or fixed token windows, affect training dynamics and final performance. A more adaptive definition of reasoning steps may further improve the stability and generality of step based length control, and we leave this direction for future work.

Acknowledgments

This research is supported by the National Key Research and Development Program of China under the grant No. 2023YFF0725003, the National Natural Science Foundation of China un-

der the grant No. 62376129, the Tianjin Science and Technology Plan Project under the grant No. 25ZGZNGX00100, the Hebei Province Science and Technology Plan Project under the grant No. 26241703D, the Tianjin Science and Technology Plan Project under the grant No. 24JCYBJC01950, the Fundamental Research Funds for the Central Universities under the grant No. 63261005, and the Nankai University-AlphaESS Smart Energy Storage Joint Research Center Project.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, and 1 others. 2025. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Thinkless: Llm learns when to think. *arXiv preprint arXiv:2505.13379*.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. 2024. Efficiently serving llm reasoning programs with certaindex. *arXiv 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. *arXiv preprint arXiv:2501.12948*.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

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. *arXiv preprint arXiv:2103.03874*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Chengyu Huang, Zhengxin Zhang, and Claire Cardie. 2025. HAPO: Training language models to reason concisely via history-aware policy optimization. *arXiv preprint arXiv:2505.11225*.

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. *arXiv preprint arXiv: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. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. 2025. OverThink: Slowdown attacks on reasoning LLMs. *arXiv preprint arXiv:2502.02542*.

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

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.

Xintong Li, Sha Li, Rongmei Lin, Hongye Jin, Linwei Li, Hejie Cui, Sarah Zhang, Chia-Yuan Chang, Kewei Cheng, Besnik Fetahu, Priyanka Nigam, Jingbo Shang, and Bing Yin. 2026a. Stepwise penalization for length-efficient chain-of-thought reasoning. *arXiv preprint arXiv:2603.00296*.

Yugu Li, Zehong Cao, Jianglin Qiao, and Siyi Hu. 2026b. SSVPO: Effective step-level credit assignment for RL training of language models. In *International Conference on Learning Representations*.

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

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. 2025a. Learn to reason efficiently with adaptive length-based reward shaping. *arXiv preprint arXiv:2505.15612*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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. *arXiv preprint arXiv:2505.11896*.

Haotian Luo, Haiying He, Yibo Wang, Jinluan Yang, Rui Liu, Naiqiang Tan, Xiaochun Cao, Dacheng Tao, and Li Shen. 2025a. Ada-r1: Hybrid-cot via bi-level adaptive reasoning optimization. *arXiv preprint arXiv:2504.21659*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, and 1 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, and 1 others. 2025b. DeepScaleR: Surpassing O1-Preview with a 1.5B model by scaling RL. Project blog post.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, and 1 others. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. MuSR: Testing the limits of chain-of-thought with multistep soft reasoning. In *International Conference on Learning Representations*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Songjun Tu, Jiahao Lin, Qichao Zhang, Xiangyu Tian, Linjing Li, Xiangyuan Lan, and Dongbin Zhao. 2025. Learning when to think: Shaping adaptive reasoning in R1-style models via multi-stage RL. *arXiv preprint arXiv:2505.10832*.

Hemish Veeraboina. 2023. AIME problem set 1983–2024. Kaggle dataset.

Han Wang, Xiaodong Yu, Jialian Wu, Jiang Liu, Ximeng Sun, Mohit Bansal, and Zicheng Liu. 2026. Stabilizing efficient reasoning with step-level advantage selection. *arXiv preprint arXiv:2604.24003*.

Junde Wu, Jiayuan Zhu, and Yuyuan Liu. 2025a. Agentic reasoning: Reasoning llms with tools for the deep research. *arXiv preprint arXiv:2502.04644*.

Xingyu Wu, Yuchen Yan, Shangke Lyu, Linjuan Wu, Yiwen Qiu, Yongliang Shen, Weiming Lu, Jian Shao, Jun Xiao, and Yueting Zhuang. 2025b. LAPO: Internalizing reasoning efficiency via length-adaptive policy optimization. *arXiv preprint arXiv:2507.15758*.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. 2025. Inftythink: Breaking the length limits of long-context reasoning in large language models. *arXiv preprint arXiv:2503.06692*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025a. Adaptthink: Reasoning models can learn when to think. *arXiv preprint arXiv:2505.13417*.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. 2025b. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*.

Jixiao Zhang and Chunsheng Zuo. 2025. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. *arXiv preprint arXiv:2504.09696*.

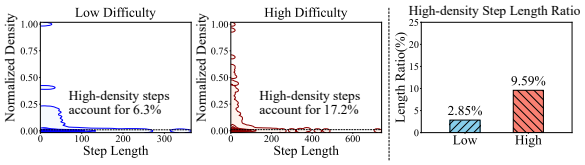
Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. 2025. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*.

A Experimental Appendix

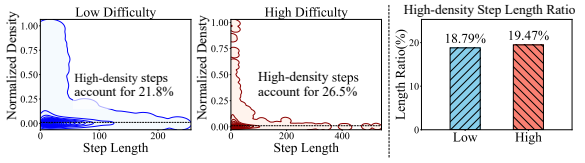
A.1 Reasoning Density Distribution under Global-Level and Step-Level Training

Figure 5 compares reasoning-density distributions under global-level and step-level training. Reasoning density measures the estimated answer contribution of a reasoning step per generated token. Under global-level length-penalty training, substantial reasoning remains allocated to low-density steps, indicating redundant token use. Under SmartThinker's step-level training, the distribution shifts toward higher-density reasoning: low-value detours are compressed, while answer-relevant derivations receive more of the reasoning budget. This shows that SmartThinker learns denser reasoning trajectories rather than merely shortening outputs.

Regarding the experimental setup shown in Figure 5, we randomly sample 20 questions from the DeepScaleR-Preview-Dataset and generate 8 responses for each question. Problem difficulty is computed using Eq.2 in the main text: questions with difficulty scores no greater than 0.5 are categorized as low-difficulty, while those with scores greater than 0.5 are categorized as high-difficulty. We instruct the model to use `\n\n` to separate reasoning steps. For each step $s_{i,j}$, we compute the correct-answer probability with the full trajectory and after removing that step, obtain the raw contribution score using Eq.5, and divide it by the step length according to Eq.6. For visualization, we apply the same min–max normalization used in SCPO within each correctness group and aggregate the resulting density scores by training method and problem difficulty.



(a) Global-level length-penalty training.



(b) SmartThinker step-level training.

Figure 5: Reasoning-density distribution under global-level and step-level training. Higher density indicates greater answer contribution per generated token. Compared with global-level length-penalty training, SmartThinker step-level training shifts reasoning toward high-density steps and compresses low-density detours across both easy and hard questions.

A.2 Benchmark Datasets

In this section, we will describe in detail all the datasets used in our experiments. These datasets are divided into in-domain datasets and out-of-domain datasets.

In-domain datasets are as follows:

- AIME24–25 comprises 30 questions from the 2024 and 2025 American Invitational Mathematics Examination, with 15 fill-in-the-blank questions per exam. These questions are more difficult than AMC and span number theory, combinatorics, geometry, and algebra.
- AMC10/12 consists of 25 multiple-choice questions each for the AMC10 (up to 10th grade) and AMC12 (up to 12th grade). Each competition consists of 25 multiple-choice questions, totaling 975 questions across 39 tests. Questions progress from basic algebra and geometry to introductory probability and counting, covering various tasks for LLM reasoning evaluation.
- MinervaMATH comprises 12,500 high-school–level contest questions. Each includes detailed solution steps and spans prealgebra through precalculus.
- MATH is a challenging dataset used to evaluate models' complex multi-step reasoning abilities in mathematics, featuring problems

that require precise logical reasoning where a single error can invalidate an entire solution. It has automatically checkable answers, allowing for outcome supervision without human input, and is employed in comparing outcome and process supervision methods.

- OlympiadBench is an Olympiad-level bilingual multimodal scientific benchmark, featuring 8,476 problems from Olympiad-level mathematics and physics competitions, including the Chinese college entrance exam. Each problem is detailed with expert-level annotations for step-by-step reasoning. Notably, the best-performing model, GPT-4V, attains an average score of 17.97% on OlympiadBench, with a mere 10.74% in physics, highlighting the benchmark rigor and the intricacy of physical reasoning.

Out-of-domain datasets are as follows:

- Live-Code-Bench evaluates code generation on recent competitive programming problems collected from platforms such as LeetCode, AtCoder, and Codeforces. Each instance contains a problem statement, input/output examples, and hidden test cases, and models must generate a complete executable solution. We evaluate 200 sampled problems from the release-latest split.

- GPQA-Diamond is the highest-quality subset of GPQA, a graduate-level multiple-choice benchmark written by domain experts in biology, chemistry, and physics. The questions are designed to be difficult to answer through web search alone and require expert-level scientific reasoning. We evaluate the full 198-question GPQA-Diamond subset.

- MuSR evaluates multistep soft reasoning over natural-language narratives. We use the MuSR-MurderMysteries subset, where each example requires identifying the correct answer from a long story through chained commonsense and evidential reasoning. We evaluate 250 examples from this subset.

A.3 Baselines

In this section, we will describe in detail all the baselines used in our experiment.

- Alpha-0.1 introduces a global length penalty mechanism into the PPO algorithm to encourage the model to produce more concise reasoning processes. This approach is fully fine-tuned on both the DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B models, achieving a substantial reduction in redundant reasoning while maintaining high accuracy.

- L1-Max employs a Length Controlled Policy Optimization (LCPO) mechanism to precisely regulate the length of chain-of-thought reasoning. The method first trains the DeepScaleR-1.5B-Preview model with LCPO, enabling it to follow user-specified token budgets and yielding the base model L1-Exact. Building on this foundation, L1-Max is further trained to generate responses that do not exceed the target length.

- Thinkless adopts a two-stage training strategy to guide the model in dynamically switching between short and long reasoning modes based on problem difficulty. In the first stage, supervised fine-tuning (SFT) serves as a warm-up, equipping the model with the ability to generate both concise and elaborate reasoning styles. In the second stage, Decoupled Group Relative Policy Optimization (DeGRPO) is introduced to reinforce training, enabling the model to learn when to output the ⟨short⟩ or ⟨think⟩ control tokens to manage reasoning length. Ultimately, Thinkless is trained on the DeepSeek-R1-Distill-Qwen-1.5B model.

- AdaptThink leverages RL to encourage the model to skip the reasoning process whenever possible, while incorporating importance sampling techniques to balance the proportion of reasoning and non-reasoning samples during training. The final AdaptThink model is trained on DeepSeek-R1-Distill-Qwen-7B.

A.4 Experimental Setting Details

In the first stage, we sample five responses for each training question and retain only those that are both correct and the shortest, discarding any samples exceeding 4096 tokens. This process yields approximately 2,000 samples per model for constructing the synthetic dataset. Sampling uses a temperature of 1 and a top-p of 0.95. During supervised

Table 6: Detailed Pass@k, Maj@k, and AvgLen results for 1.5B model families on the six in-domain reasoning benchmarks.

Models	AIME25	AIME24	AMC23	MinervaMATH	MATH	Olympiad-Bench	Avg.
Pass@k (%)							
DeepSeek-R1-1.5B	43.33	50.00	87.50	44.85	91.20	54.37	61.88
Alpha-0.1-1.5B	40.00	63.33	90.00	49.26	92.00	56.44	65.17
Thinkless-1.5B	40.00	50.00	92.50	45.59	92.00	**62.07**	63.69
SmartThinker-1.5B	**43.33**	**63.33**	**95.00**	45.22	91.00	56.59	**65.75**
DeepScaleR-1.5B-Preview	**43.33**	60.00	**95.00**	**48.53**	**92.60**	**59.70**	66.53
L1-Max-1.5B	**43.33**	50.00	**95.00**	47.43	**92.60**	57.63	64.33
SmartThinker-ScaleR	**43.33**	**66.67**	92.50	46.32	91.80	59.26	**66.65**
Maj@k (%)							
DeepSeek-R1-1.5B	33.33	36.67	77.50	32.72	84.60	45.78	51.77
Alpha-0.1-1.5B	30.00	**40.00**	80.00	**35.29**	**86.80**	46.37	53.08
Thinkless-1.5B	30.00	30.00	80.00	**35.29**	85.20	**51.11**	51.93
SmartThinker-1.5B	**40.00**	33.33	**82.50**	33.46	83.80	49.04	**53.69**
DeepScaleR-1.5B-Preview	**36.67**	**40.00**	85.00	**38.60**	**90.00**	**54.81**	**57.51**
L1-Max-1.5B	33.33	30.00	85.00	37.50	87.40	50.67	53.98
SmartThinker-ScaleR	30.00	**40.00**	**90.00**	33.46	88.60	50.22	55.38
AvgLen							
DeepSeek-R1-1.5B	7307	7314	5335	4844	3821	5985	5768
Alpha-0.1-1.5B	6459	6486	3771	2822	2300	4596	4406
Thinkless-1.5B	6475	6565	3888	3062	1930	4249	4362
SmartThinker-1.5B	**5653**	**5310**	**3105**	**2495**	**1643**	**3700**	**3651**
DeepScaleR-1.5B-Preview	6534	6833	4421	4369	2888	4881	4988
L1-Max-1.5B	3076	3255	2440	3103	2221	2948	2841
SmartThinker-ScaleR	**2748**	**3004**	**2217**	**2072**	**1885**	**2291**	**2370**

Table 7: Detailed Pass@k, Maj@k, and AvgLen results for 7B models on the six in-domain reasoning benchmarks.

Models	AIME25	AIME24	AMC23	MinervaMATH	MATH	Olympiad-Bench	Avg.
Pass@k (%)							
DeepSeek-R1-7B	50.00	73.33	95.00	54.04	95.20	61.48	71.51
Alpha-0.1-7B	60.00	73.33	95.00	**54.78**	**95.60**	**64.30**	73.83
AdaptThink-7B	60.00	73.33	95.00	53.31	95.20	63.56	73.40
SmartThinker-7B	**66.67**	**80.00**	95.00	50.37	95.40	64.15	**75.26**
Maj@k (%)							
DeepSeek-R1-7B	43.33	56.67	90.00	43.38	91.00	54.07	63.08
Alpha-0.1-7B	43.33	60.00	**95.00**	**44.49**	**92.40**	55.41	65.10
AdaptThink-7B	40.00	**70.00**	90.00	42.28	91.80	56.15	65.04
SmartThinker-7B	**53.33**	63.33	92.50	40.81	**92.40**	**57.19**	**66.59**
AvgLen							
DeepSeek-R1-7B	6820	6791	4600	3589	2815	5308	4987
Alpha-0.1-7B	6265	6161	3466	2322	1887	4267	4061
AdaptThink-7B	6332	5962	3185	2002	**1505**	4014	3833
SmartThinker-7B	**5296**	**5416**	**2777**	**1667**	1570	**3475**	**3367**

fine-tuning (SFT), we set the batch size to 1 with a gradient accumulation step of 8. The learning rate is 1×10^{-5}, following a cosine schedule with a warm-up ratio of 0.1, and we train for three epochs.

In the second stage, we train the models using the TRL framework with a batch size of 8, generating 8 rollouts per question for SCPO reward computation. We omit the KL penalty, as previous studies show that it suppresses exploration (Yu et al., 2025; Liu et al., 2025b). The maximum context length is set to 4096 tokens during training and extended to 8192 tokens for evaluation. We carry out training with ZeRO Stage 2 (Rajbhandari et al., 2020) across two A6000 GPUs. We set the temperature to 1 and top-p to 0.95 during training. For in-domain evaluation, we use temperature 0.6. For the OOD evaluation in Table 3, we use EvalScope with temperature 0.6, top-p 0.95, and a maximum generation length of 8192 tokens. To maximize data utilization, we update the policy model two times per batch. The hyperparameters are configured as $k_0 = 0.6$, $\gamma = 0.95$, $\delta_1 = 0.03$, and $\delta_2 = 0.08$. We optimize using the AdamW (Loshchilov and Hutter, 2017) optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a learning rate of 2×10^{-6}, following a cosine schedule.

A.5 Detailed Results for the 1.5B Models

Table 6 provides the full per-benchmark results for the 1.5B model families. We report the DeepSeek-R1-Distill-Qwen and DeepScaleR groups separately because they start from different base models, so the most meaningful comparisons are within each group. The table includes Pass@k, Maj@k, and AvgLen, allowing us to examine not only whether a method improves accuracy, but also whether it preserves useful reasoning under a shorter token budget.

For the DeepSeek-R1-Distill-Qwen-1.5B family, SmartThinker-1.5B achieves the strongest average Pass@k and Maj@k among the efficient-reasoning methods, while also producing the shortest responses. Compared with Alpha-0.1-1.5B, it improves the average Pass@k from 65.17% to 65.75% and Maj@k from 53.08% to 53.69%, while reducing AvgLen by 755 tokens. Compared with Thinkless-1.5B, SmartThinker gains 2.06 Pass@k points and 1.76 Maj@k points with 711 fewer tokens. These results suggest that the improvement is not obtained by uniformly truncating reasoning, but by removing low-contribution steps while preserving the derivations that support correct answers.

The DeepScaleR group further tests whether the same idea remains useful when the starting model is already stronger and when the baseline applies aggressive length control. DeepScaleR-1.5B-Preview remains a strong starting point, achieving 66.53% average Pass@k and the highest average Maj@k in this group. SmartThinker-ScaleR slightly raises average Pass@k to 66.65% while reducing AvgLen from 4988 to 2370 tokens. Compared with L1-Max-1.5B, it also improves average Pass@k and Maj@k by 2.32 and 1.40 points under a shorter token budget. This supports the central motivation of SCPO: effective compression should depend on step importance rather than a single global length pressure.

A.6 Detailed Results for the 7B Model

Table 7 reports the detailed 7B results across the six in-domain reasoning benchmarks. This setting tests whether the proposed step-level control continues to scale when the base model already has stronger reasoning ability.

SmartThinker-7B obtains the best average Pass@k and Maj@k, improving over DeepSeek-R1-7B by 3.75 and 3.51 points while reducing average response length to 68% of the original. The gains are especially clear on the most difficult competition-style benchmarks: SmartThinker-7B improves Pass@k by 16.67 points on AIME25 and 6.67 points on AIME24, indicating that its shorter traces do not come at the expense of hard-problem exploration. At the same time, it consistently shortens responses across benchmarks, with the largest reductions on MinervaMATH, AMC23, and Olympiad-Bench.

Compared with Alpha-0.1-7B and AdaptThink-7B, SmartThinker-7B achieves higher average accuracy with fewer tokens. This contrast is informative: global length penalties can shorten responses but may compress important reasoning along with redundant text, while mode-switching methods can reduce cost without directly optimizing the allocation of reasoning effort inside a chain. SmartThinker instead assigns pressure at the step level, which keeps high-contribution derivations available while removing low-value continuation and verification. The 7B results therefore show that SCPO improves the accuracy-efficiency trade-off even when the backbone already has strong reasoning capacity.

A.7 Controlled Comparison Details

Table 8 reports the full per-benchmark results behind the controlled comparisons in Section 5.2. All methods are retrained from the same SFT warm-up checkpoint of each backbone family, with identical training data, 4K training context, optimization configuration, and number of training steps; only the RL objective differs. The publicly released LAPO-I checkpoint (trained without our warm-up) is included as a reference point on the accuracy-efficiency frontier. The table shows that the additional accuracy and compression gains of SmartThinker persist under this strictly controlled setting, so they cannot be attributed to short-context training, initialization, or recipe differences.

A.8 Training Cost of the Step-Contribution Estimator

This section gives the implementation details and full timings behind the training-overhead paragraph in the main results. Tables 9 and 10 quantify the training overhead of the online step-contribution estimator. For each training step, all leave-one-step-out sequences are constructed in parallel, dynamically packed across rollouts into a single batch, and scored by one batched prefill-only forward pass with vLLM (Kwon et al., 2023), so that no autore-

Table 8: Per-benchmark Pass@k (%) and average response length of the controlled comparisons described in Section 5.2. Each cell pair reports Pass@k and average token count. †LAPO-I is the publicly released checkpoint trained without our SFT warm-up and is shown only as a frontier reference.

Models	AIME25		AIME24		AMC23		MinervaMATH		MATH		Olympiad-Bench		Avg.	
	P@k	Tok.	P@k	Tok.	P@k	Tok.	P@k	Tok.	P@k	Tok.	P@k	Tok.	P@k	Tok.
DeepSeek-R1-Distill-Qwen-1.5B backbone														
SFT+Alpha-0.1-1.5B	**43.33**	5925	**63.33**	5964	90.00	3405	**47.79**	**2491**	**92.20**	1987	56.15	4172	65.47	3991
SFT+Thinkless-1.5B	36.67	5948	43.33	6073	87.50	3281	45.22	2541	89.60	1711	**61.19**	3825	60.59	3897
SmartThinker-1.5B	**43.33**	**5653**	**63.33**	**5310**	**95.00**	**3105**	45.22	2495	91.00	**1643**	56.59	**3700**	**65.75**	**3651**
DeepScaleR-1.5B-Preview backbone														
SFT+GRPO-ScaleR	40.00	3268	53.33	3481	**95.00**	2553	45.59	3296	92.00	2415	58.52	3164	64.07	3030
SFT+L1-Max-1.5B	40.00	2813	53.33	**2976**	92.50	2359	**47.06**	2215	92.20	2001	56.44	2397	63.59	2460
LAPO-I-ScaleR†	**50.00**	5257	**73.33**	5474	92.50	3295	44.49	2905	**94.20**	2165	59.11	3164	**68.94**	3710
SFT+LAPO-ScaleR	46.67	3769	66.67	3891	90.00	2614	43.38	2458	93.00	1952	57.78	2693	66.25	2896
SmartThinker-ScaleR	43.33	**2748**	66.67	3004	92.50	**2217**	46.32	**2072**	91.80	**1885**	**59.26**	**2291**	66.65	**2370**
DeepSeek-R1-Distill-Qwen-7B backbone														
SFT+GRPO-7B	63.33	6048	73.33	5923	**95.00**	2987	**51.46**	1809	95.20	1654	63.56	3701	73.65	3687
SmartThinker-7B	**66.67**	**5296**	**80.00**	**5416**	**95.00**	**2777**	50.37	**1667**	**95.40**	**1570**	**64.15**	**3475**	**75.26**	**3366**

Table 9: Wall-clock time (seconds) of one online step-contribution evaluation over 8 rollouts (up to 4,096 tokens each), measured over three runs.

Model	Run 1	Run 2	Run 3	Avg.
DeepSeek-R1-1.5B	0.62	0.58	0.59	0.60
DeepSeek-R1-7B	4.08	4.06	4.05	4.06

Table 10: Average wall-clock time (seconds) per training step under identical hardware and configuration. SCPO adds less than 8% end-to-end overhead over plain GRPO on both model scales.

Model	GRPO	SCPO	SCPO/GRPO
DeepSeek-R1-1.5B	9.35	9.81	104.92%
DeepSeek-R1-7B	34.92	37.66	107.80%

gressive decoding is involved. Table 9 reports the wall-clock time of one complete contribution evaluation over 8 rollouts of up to 4,096 tokens each, averaged over three independent runs.

Table 10 further compares the end-to-end per-step training time of SCPO and plain GRPO under identical hardware and training configuration. The removal-based estimation adds less than 8% overhead on both model scales. Since the estimator runs only during training and leaves decoding untouched, SmartThinker incurs no inference-time cost, and this one-time training overhead is repaid by the sustained inference-efficiency gains reported in the main results.

Table 11: Average number of attempted AIME24 problems under each context budget. Attempts include correctly solved, incorrectly answered, and truncated problems before the run stops.

Models	$B = 4$K	$B = 8$K	$B = 16$K	$B = 32$K
DeepSeek-R1-7B	1.20	1.60	4.20	16.00
Alpha-0.1-7B	1.20	2.00	6.60	5.60
AdaptThink-7B	1.60	2.00	8.40	11.80
SmartThinker-7B	**1.60**	**2.60**	**12.20**	**20.80**

A.9 Context-Budgeted Evaluation Details

We evaluate context-budgeted reasoning on the 30-problem AIME24 test set. For each run, we shuffle the problems with a fixed seed, append them one by one to a cumulative chat history, and let the model answer the current problem before moving to the next one. The run stops when the accumulated prompt and completion tokens reach the context budget B, or when all 30 problems have been attempted. The main metric in Table 2 is the number of correctly solved problems before stopping. We use budgets $B \in \{4096, 8192, 16384, 32768\}$. For each budget, we evaluate five shuffled problem orders and use the same seed set for all models under that budget. Decoding uses temperature 0.6 and top-p 0.95. At each turn, the maximum generation length is set to the remaining context budget after formatting the current chat prompt. Answers are extracted with the same answer parser and mathematical equivalence checker used in the main evaluation.

Table 11 shows how many AIME24 problems each model can attempt before exhausting the same

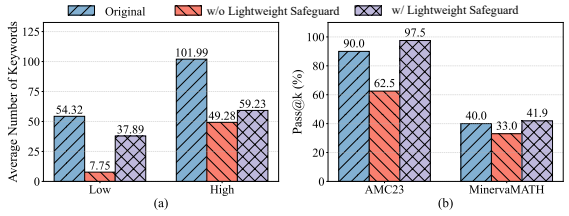


Figure 6: Effect of lightweight safeguard on reasoning behavior and Pass@k performance across different difficulty levels.

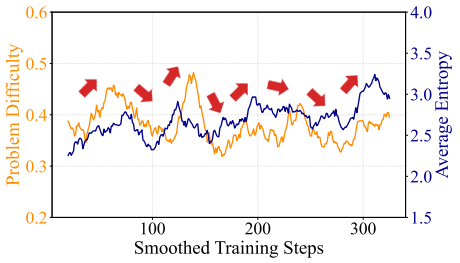


Figure 7: Entropy trends with problem difficulty during training. Because each sample batch updates the policy model for four iterations, both entropy and problem difficulty are averaged over every four steps.

cumulative context budget. This auxiliary metric isolates the throughput side of the context-budgeted evaluation. At $B = 4$K, all models can only make one or two attempts, so the margin is small. As the budget grows, the difference becomes much clearer: SmartThinker-7B attempts 12.20 problems at $B = 16$K, compared with 4.20 for DeepSeek-R1-7B, 6.60 for Alpha-0.1-7B, and 8.40 for AdaptThink-7B. At $B = 32$K, SmartThinker-7B reaches 20.80 attempted problems, exceeding the strongest baseline by 4.80 attempts.

Crucially, these additional attempts are not obtained by collapsing reasoning quality. When combined with the solved-problem results in Table 2, Table 11 shows that SmartThinker converts saved tokens into productive additional opportunities: it attempts the most problems and also solves the most problems at larger budgets. This distinguishes step-level length control from coarse shortening. Rather than simply forcing every response to be brief, SmartThinker removes low-contribution reasoning segments while preserving enough high-value derivation for later attempts to remain useful.

A.10 Lightweight Safeguard Preserves Useful Reflection

Figure 6 examines whether the lightweight safeguard preserves useful reflective behavior during compression. Without the safeguard, the model sharply reduces reasoning transition markers, which weakens its ability to revise or verify intermediate conclusions. This failure mode is especially important because reflection is often sparse: many reflective steps are short and may not immediately increase the estimated contribution score, yet they can prevent the model from committing to an early but fragile solution path.

The safeguard addresses this issue by selectively protecting reflective steps instead of relaxing length pressure globally. As a result, the model keeps the verification behavior needed for reliable exploration while still compressing low-value continuation. This explains why the safeguard improves the module-level ablation in Table 4: it does not simply make responses longer, but preserves a small class of steps whose value is easy to underestimate with removal-based scoring alone. Appendix D lists the transition markers used by this auxiliary signal.

A.11 Difficulty-Adaptive Clipping Tracks Exploration Need

Figure 7 shows that output entropy rises with problem difficulty and falls when the sampled questions become easier. This pattern is desirable: difficult problems require broader exploration over possible reasoning paths, whereas easy problems benefit from exploiting a stable solution pattern. A fixed clipping range cannot express this distinction and may either under-explore hard problems or destabilize easy ones.

Difficulty-adaptive clipping provides this control by tying the update range to the online difficulty estimate. The resulting entropy trajectory indicates that SmartThinker does not rely on uniformly high exploration to improve accuracy. Instead, it allocates exploration where the rollout set suggests uncertainty and constrains updates where the current policy is already reliable. This behavior explains why the final module in Table 4 improves the overall accuracy-efficiency trade-off rather than merely changing response length.

B Case Studies

We conduct case studies on different baseline models to provide an intuitive demonstration of SmartThinker's effectiveness. As shown in Fig. 8, for a question that allows a direct answer, the original

model generates a lengthy textual reasoning and continues to elaborate even after arriving at the correct answer. This unnecessarily complicates simple problems and results in substantial token redundancy. Although Alpha-0.1 mitigates this redundancy to some extent, it still repeats portions of the reasoning process when producing the final answer. Although Thinkless is capable of switching to short-reasoning modes, its reasoning remains relatively complex. In contrast, SmartThinker provides the final answer directly with a single concise line of reasoning, entirely avoiding such redundancy.

As shown in Fig. 9, when faced with more challenging questions, the original model often engages in extensive purely textual reasoning, followed by multiple rounds of reflection and verification. This significantly increases computational resources required for reasoning without contributing meaningfully to producing the final answer. Although Alpha-0.1 and Thinkless are able to arrive at the correct answers, they still perform redundant self-reflection and verification afterward, leading to unnecessary token consumption. In contrast, SmartThinker relies more on concise and effective statements interwoven with mathematical formulas, outputting only the mathematical derivations and essential textual explanations that are genuinely helpful for producing the final answer. This enables a more efficient reasoning process overall.

C Theoretical Analysis

In this section, we provide a theoretical proof that step-level length control can achieve a better trade-off between reasoning quality and computational efficiency compared to global length control.

Typically, the step-level length control algorithm aims to maximize the following reward:

$$R_{\text{step}} = \mathbb{1}(\text{correct}) - \sum_{i=1}^{n} \lambda_i \frac{l_i}{T}, \tag{19}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function, λ_i is the length penalty coefficient for the i-th step, l_i is the length of the i-th step, and $T = \sum_{i=1}^{n} l_i$ denotes the total length of the response.

The reward for the i-th step can therefore be defined as:

$$r_i^{\text{step}} = m_i - \lambda_i \frac{l_i}{T}, \tag{20}$$

where m_i represents the answer contribution of the i-th step to the overall correctness of the answer.

In the GRPO algorithm, the advantage for the i-th step is defined as:

$$A_i = \frac{r_i - \overline{r}}{\sigma_r}, \tag{21}$$

where $\overline{r}$ and σ_r denote the mean and standard deviation of the rewards, respectively.

Let the current length of the i-th step be l_i. After an update to the policy gradient, it becomes l_i^{step}. For simplicity, we assume that the change in step length is proportional to the policy gradient in reinforcement learning, yielding:

$$\Delta l_i \propto \mathbb{E}\left[\nabla_\theta \frac{\pi_\theta(s_i \mid q)}{\pi_{\text{old}}(s_i \mid q)} A_i\right]. \tag{22}$$

Considering only the variability introduced by advantage estimation, we have:

$$l_i^{\text{step}} = l_i + \eta \cdot A_i. \tag{23}$$

Substituting Eq.20 and 21 into Eq.23:

$$l_i^{\text{step}} = l_i \cdot \left(1 - \frac{\alpha \lambda_i}{T}\right) + \alpha \cdot \delta_i, \tag{24}$$

where $\alpha = \frac{\eta}{\sigma_r}$ and $\delta_i = m_i - \overline{r}$.

If we focus solely on differences in penalty coefficients, the Eq. 24 can be approximated as:

$$l_i^{\text{step}} \approx l_i \cdot \left(1 - \frac{\alpha \lambda_i}{T}\right). \tag{25}$$

Similarly, for the global length control algorithm, after one update, the length of the i-th step can be expressed as:

$$l_i^{\text{global}} \approx l_i \cdot \left(1 - \frac{\alpha \lambda}{T}\right). \tag{26}$$

Let v_i denote the contribution density of the i-th step to producing the correct answer. Then, the total contribution under step-level length control is:

$$I_{\text{step}} = \sum_i v_i \cdot l_i^{\text{step}} \tag{27}$$

$$= \sum_i v_i l_i - \frac{\alpha}{T} \sum_i v_i \lambda_i l_i. \tag{28}$$

Similarly, under global length control, the total contribution is:

$$I_{\text{global}} = \sum_i v_i \cdot l_i^{\text{global}} \tag{29}$$

$$= \sum_i v_i l_i - \frac{\alpha \lambda}{T} \sum_i v_i l_i. \tag{30}$$

Define $S = \sum_i v_i l_i$. Then:

$$I_{\text{step}} - I_{\text{global}} = \frac{\alpha}{T}\left(\lambda S - \sum_i v_i \lambda_i l_i\right). \quad (31)$$

Assuming that both methods yield the same total token budget after the update, we have:

$$\sum_i l_i^{\text{step}} = \sum_i l_i^{\text{global}}, \quad (32)$$

and substitute Eq.25 and 26 into Eq. 32. we can obtain:

$$\sum_i \lambda_i l_i = \lambda \sum_i l_i. \quad (33)$$

In the step-level length control algorithm, λ_i should be inversely proportional to the contribution density of the step. Thus, we define $\lambda_i = 1 - v_i$, yielding:

$$I_{\text{step}} - I_{\text{global}} = \sum_i v_i^2 l_i - \frac{S^2}{T}. \quad (34)$$

By the Cauchy–Schwarz inequality:

$$\left(\sum_i v_i^2 l_i\right) \cdot \left(\sum_i l_i\right) \geq \left(\sum_i v_i l_i\right)^2. \quad (35)$$

Therefore, we finally obtain:

$$I_{\text{step}} \geq I_{\text{global}}. \quad (36)$$

This result demonstrates that, under the same token budget, step-level length control preserves a higher total contribution toward producing the correct answer compared to global length penalty methods.

D Reasoning Transition Words

In this section, we list a curated set of commonly used reasoning transition words and phrases that frequently appear in the LLM's chain-of-thought. We group them by their *reasoning function* to make them easy to read and to use as a keyword set in our method.

Contrast, Correction, and Turning Points These markers indicate a change of direction, contrast, or self-correction (e.g., catching an error and revising the reasoning).

but, however, nevertheless, nonetheless, yet, still, though, although, even though, instead, rather, on the other hand, on the contrary, by contrast, in contrast, in reality, in fact, that said, despite this, despite that, conversely. wait, hold on, that is not correct I made a mistake, I miscalculated, I misread the problem, let me correct that, let me fix this, let me reconsider, let me re-evaluate this.

Checking, Verification, and Self-reflection These phrases indicate that the model is explicitly checking or verifying its previous steps, which often correspond to important reflective behavior.

let me check, let me check this, let me double-check, let me verify, let me verify the calculation, let me make sure, let us check, let us verify, let us confirm, to verify this, to check this, this does not make sense, this seems wrong, this seems incorrect, I now see that, I see a mistake here, there is an error here, let me try again, let me start over, let me try a different approach, let me try another method, let me take a different approach, instead, I will try.

Logical Progression and Deduction These words and phrases express logical consequence and progression from one step to the next.

therefore, thus, hence, so, consequently, as a result, it follows that, this implies that, we deduce that, we conclude that, from this we get.

Alternative Approaches and Reformulation These expressions introduce alternative methods or rephrase the same idea in a different way.

alternatively, another way is to, another approach is to, a different approach is to, equivalently, in other words.

Assumptions, Conditions, and Setup These phrases are used when the model sets up assumptions, conditions, or notations, which often structure the reasoning that follows.

suppose that, assume that, let us suppose, let us assume, let us consider, given that, note that, we note that, observe that, we observe that.

Meta-reasoning Cues and Heuristics These expressions comment on the difficulty, correctness, or reliability of the current reasoning path.

this looks promising, this approach may work, this might be wrong, I may have missed something, I need to be careful

Lightweight Discourse Markers These short discourse markers are common in free-form chain-of-thought and often signal local transitions between micro-steps.

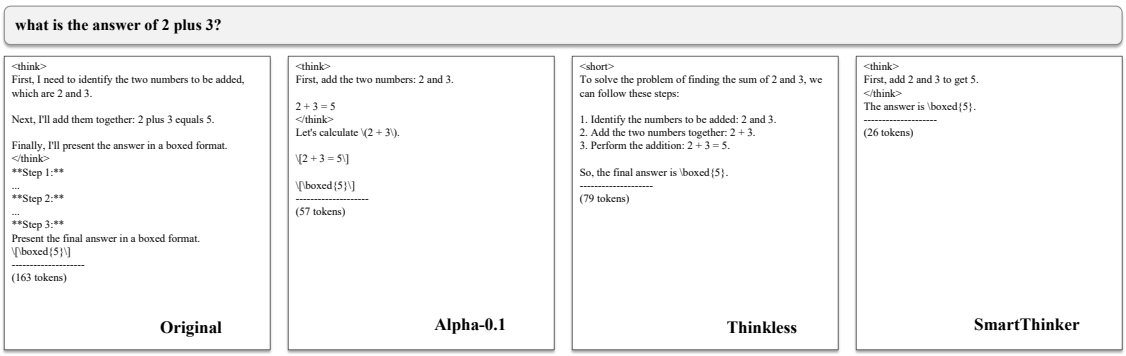


Figure 8: Examples of model responses to questions with straightforward solutions across different baselines.

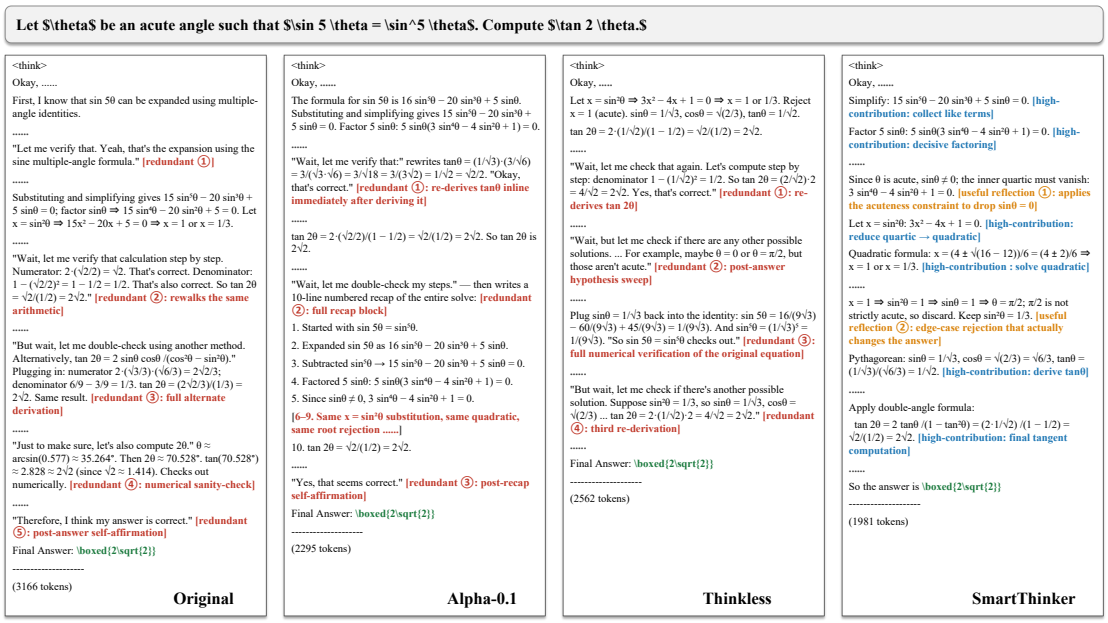


Figure 9: Examples of model responses to challenging questions across different baselines.

okay, alright, well, now, so now, at this point, so far.