The Green Reasoner: Cognition-Guided Policy Optimization for Efficient LLM Reasoning

Anonymous ACL submission

Abstract

Balancing reasoning accuracy and efficiency in large language models (LLMs) is a critical objective. Reinforcement learning (RL) has emerged as a leading paradigm to achieve this goal. However, current RL-based methods cannot effectively distinguish between redundant and essential reasoning paths. Moreover, these methods often lack flexibility in handling samples with different difficulty levels. To address these limitations, we present **Cognition-Guided Policy Optimization (CGPO)**, which consists of Cognitive Utility Reward (CUR) and Cognition-Adaptive Regulation (CAR). Specifically, CUR designs a multiplicative reward that scales correctness rewards with a non-linear length penalty to reduce redundancy. CAR adaptively adjusts Kullback-Leibler (KL) regularization based on real-time cognitive difficulty of each sample. Extensive experiments on nine datasets across three reasoning tasks demonstrate that CGPO achieves an effective balance between efficiency and reasoning accuracy. For instance, on mathematical reasoning benchmarks using the DeepScaleR-Preview-1.5B model, CGPO outperforms other methods by 0.2 to 3.2 points in average Pass@1 and reduces token usage by 0.7% to 38.4%.

1 Introduction

Recent advancements in large language models (LLMs) have catalyzed extensive research on their complex reasoning capabilities (Guo et al., 2025; Touvron et al., 2023; Yang et al., 2024). In particular, reinforcement learning (RL) has achieved significant progress across multiple domains (Chen et al., 2025; Ye et al., 2025; Wang et al., 2025). Nevertheless, this advanced reasoning is often constrained by high computational costs, since models often generate redundant reasoning paths (Chen et al., 2024; Yu et al., 2025). Such redundancy increases token consumption and the likelihood of erroneous intermediate steps, which ultimately degrades model performance (Turpin et al., 2024; Cuadron et al., 2025; Shen et al., 2025). As a result, improving the reasoning efficiency of LLMs without sacrificing accuracy has become an essential research goal.

To realize this goal, recent studies have explored RL-based approaches to promote concise reasoning. Existing approaches are broadly categorized into *fixed-budget reasoning methods* and *adaptive reasoning methods*. Specifically, *fixed-budget reasoning methods* aim to reduce token usage by enforcing explicit constraints. They typically use hard token limits or hierarchical token limits to prune the reasoning process. In contrast, *adaptive reasoning methods* empower models to self-regulate computational effort by applying explicit length-based rewards to promote concise reasoning. Despite their success, these methods still face two critical limitations that hinder their effectiveness:

First, current RL-based methods fail to effectively distinguish between redundant and necessary reasoning paths. While current RL-based methods are designed to promote conciseness, their reward shaping strategies often exhibit inherent limitations. To be specific, most methods apply simple length penalty mechanisms, subtracting points based on reasoning lengths. However, penalizing all tokens equally overlooks indispensable reasoning steps that increase reasoning lengths but are essential for producing correct answers (Xiang et al., 2025; Huang et al., 2025). By penalizing reasoning lengths indiscriminately, these methods encourage models to produce shorter but incorrect solutions. Therefore, such penalization effectively decouples efficiency from solution quality.

Second, current RL-based methods lack flexibility in handling samples of varying difficulty levels. To facilitate training stability, standard RL-based algorithms such as GRPO (Shao et al., 2024) typically apply a fixed Kullback-Leibler (KL) divergence coefficient to constrain policy updates.

Nevertheless, this strategy cannot adapt effectively to samples with varying difficulty levels. Specifically, simple tasks with low cognitive demands require less exploration and stricter constraints to ensure stability. On the contrary, complex tasks demanding extensive cognitive effort require more exploration to identify correct reasoning paths. Consequently, a fixed coefficient cannot satisfy both requirements simultaneously, thereby resulting in suboptimal policy optimization.

To address the above limitations, we present **Cognition-Guided Policy Optimization (CGPO)**, a novel RL-based algorithm developed to balance reasoning accuracy with efficiency. CGPO integrates two key innovations: (1) **Cognitive Utility Reward (CUR):** CUR formulates a non-linear multiplicative reward that treats token consumption as a cognitive cost. This ensures that reasoning efficiency is achieved without compromising accuracy. (2) **Cognition-Adaptive Regulation (CAR)**: CAR introduces a cognition-aware mechanism that adaptively adjusts the KL-divergence coefficient based on the model's cognitive effort for each sample. This design encourages exploration on cognitively demanding problems while preserving efficiency on simpler ones. Collectively, these designs empower CGPO to attain a superior balance between accuracy and efficiency. Extensive experiments on nine datasets across three reasoning tasks demonstrate that CGPO outperforms other baselines.

Our contributions are summarized as follows:

(1) We identify two key limitations in current RL-based methods for efficient reasoning. First, current methods cannot effectively distinguish between redundant and necessary reasoning paths. Second, these methods exhibit insufficient adaptation to samples with varying cognitive demands.

(2) We introduce CGPO, a novel RL-based algorithm that integrates a multiplicative reward function with a cognition-adaptive mechanism. This design enables models to reason more efficiently while maintaining reasoning accuracy across samples requiring varying cognitive demands.

(3) Experimental results on nine datasets covering three reasoning tasks showcase that CGPO outperforms other baselines in terms of reasoning accuracy and efficiency.

2 Related Work

Recent advancements in LLMs have spurred extensive research to improve reasoning efficiency by reducing excessive verbosity and redundant reasoning steps. Existing approaches are broadly classified into two main categories: *fixed budget reasoning methods* and *adaptive reasoning methods*.

Fixed Budget Reasoning Methods. These methods aim to reduce token usage by imposing explicit constraints on output length. Common strategies use hard token budgets (Aggarwal and Welleck, 2025; Xu et al., 2025) or leveraging hierarchical structures (Qi et al., 2025; Lyu et al., 2025; Hou et al., 2025) to constrain the reasoning process. Nonetheless, these methods often rely on manually specified budgets that fail to reflect problem difficulty. Consequently, they complicate the selection of appropriate token limits for samples with different difficulty levels.

Adaptive Reasoning Methods. In contrast, these approaches enable models to adapt their computational effort to the difficulty of each problem. One prominent strategy is reward shaping to discourage verbosity. A common implementation adopts an additive penalty, where a length-based cost is subtracted from the correctness reward (Yi et al., 2025; Xiang et al., 2025; Huang et al., 2025). However, this additive structure has limitations as it can distort the learning signal by assigning negative advantages to valid but long reasoning paths. Even alternative designs such as the multiplicative rewards using hard-thresholds (Liu et al., 2025) can suffer from similar issues that generate suppressive learning signals. Another approach is binary mode selection (Fang et al., 2025; Zhang et al., 2025; Tu et al., 2025), where models learn to switch between "thinking" and "non-thinking" modes. While effective for simple problems, this coarse-grained decision-making fails to capture the dynamics of reasoning difficulty during the training process.

3 Motivation for CGPO

In this section, we elaborate on the limitations of the current approaches mentioned in the Introduction section, illustrating the motivation behind our proposed CGPO.

3.1 Motivation for CUR

The results in Figure 1 reveal a critical limitation in existing reward shaping strategies. While these strategies can effectively reduce redundant reasoning, their reward structures often fail to distinguish between unnecessary redundancy and the extended reasoning required for complex problems.

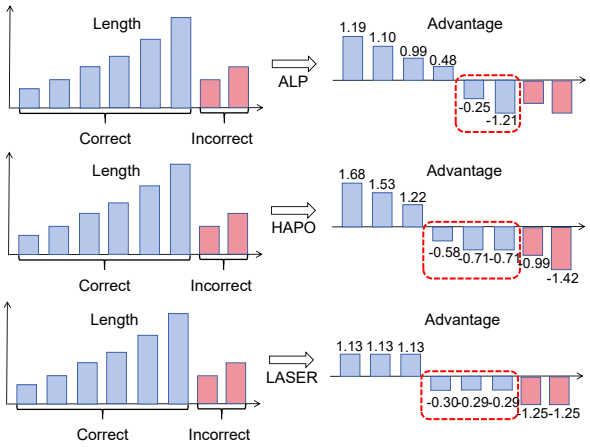

Figure 1: An analysis of learning signals (*i.e.*, advantages) from different reward designs. Advantages are computed by normalizing rewards within 8 rollouts (see Eq. (6)).

Specifically, additive penalties (*e.g.*, ALP (Xiang et al., 2025), HAPO (Huang et al., 2025)) tend to assign negative advantages to correct responses solely due to length. When the advantage becomes negative, it unintentionally drives the policy away from generating such answers even though they are correct. This issue arises because the reward functions fail to distinguish between redundant and necessary reasoning paths. This effect is harmful, as long-form reasoning is often essential to reaching the correct solution for complex questions. Suppressing these reasoning paths directly undermines the model's ability to reinforce the complex reasoning skills it has learned. Similarly, multiplicative formulations such as LASER (Liu et al., 2025) using hard thresholds exhibit the same limitation.

This indiscriminate restriction ignores that *efficiency and correctness should be jointly optimized.* Therefore, this underscores the need for an improved strategy that penalizes excessive reasoning while preserving positive reinforcement for correct reasoning. A more detailed analysis of the reward design is provided in Appendix D.1.

3.2 Motivation for CAR

Table 1 showcases the inherent rigidity of a static KL-divergence coefficient β, which governs the model's exploration capacity. Specifically, a loose constraint (*i.e.*, low β) may induce excessive exploration, often leading to unnecessarily prolonged reasoning paths on simple problems without improving accuracy. Conversely, a tight constraint (*i.e.*, high β) suppresses exploration, limiting the model's ability to develop the richer reasoning required for complex problems. This imbalance reveals a fundamental misalignment between the model's cognitive demands and its static regularization strength. As a result, a *cognition-adaptive regulator* is required to dynamically modulate β, enabling efficient reasoning by promoting concise exploration on easy samples while permitting deeper reasoning on cognitively challenging ones.

Level	β			
	0	0.001	0.002	0.005
Level 1	93.0	95.3	95.3	**97.6**
Level 2	92.2	**94.4**	93.3	92.2
Level 3	84.8	**86.7**	85.7	85.7
Level 4	73.4	**75.0**	74.2	72.7
Level 5	**72.4**	71.6	70.9	69.4

Table 1: Analysis of the KL-divergence coefficient β in GRPO on the MATH500 dataset, stratified by five predefined difficulty levels (Level 1 $\sim$ 5). The best result for each level is bolded.

4 Methodology

As depicted in Figure 2, CGPO integrates two innovations: Cognitive Utility Reward (CUR) and Cognition-Adaptive Regulation (CAR).

4.1 Cognitive Utility Reward (CUR)

In RL-based frameworks, each of the G sampled responses o_i receives a binary reward R_i. However, current methods often use suboptimal reward shaping strategies for efficient reasoning.

To address this limitation, we introduce **Cognitive Utility Reward (CUR)**, which redefines the reward function as a measure of *cognitive utility*. This utility quantifies the combined value of a reasoning outcome, jointly determined by its correctness and the associated cognitive cost. Formally, it is modeled as the product of response correctness and an efficiency factor. The core rationale is that cognitive efficiency decays as reasoning length increases. Each additional reasoning is assumed to incur an incremental cognitive cost, thereby reducing the overall efficiency of the reasoning process. To formalize this efficiency decay, a length-dependent marginal cost function $\gamma(L_i)$ is defined as a linear function of the response length L_i:

$$\gamma(L_i) = kL_i \tag{1}$$

where k is the coefficient of linear cost growth.

Intuition. Drawing inspiration from cognitive load theory (Sweller, 1988), each additional rea-

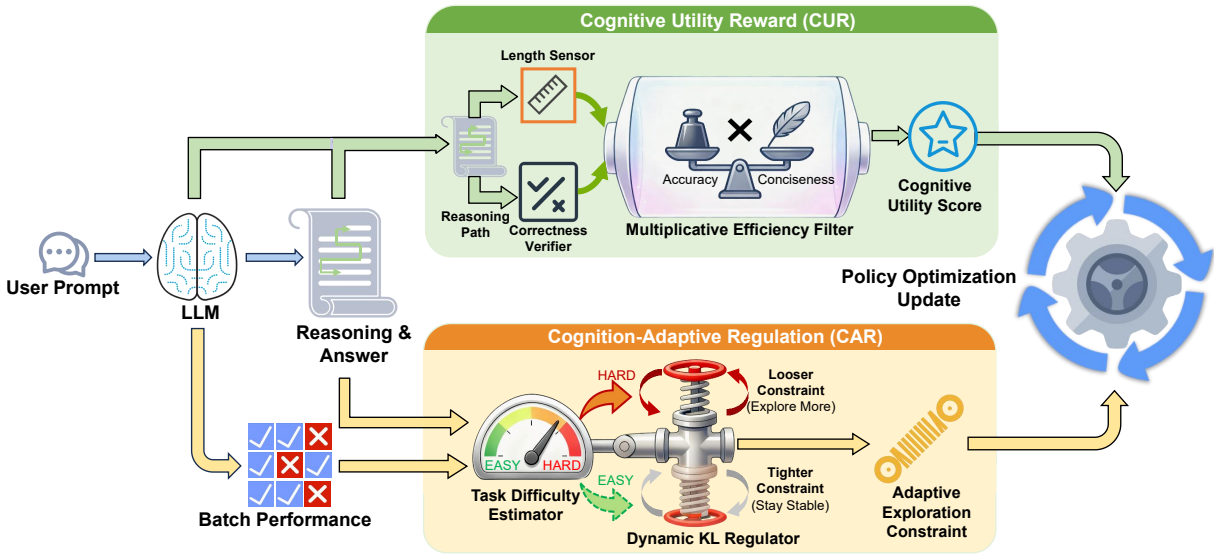

Figure 2: Illustration of CGPO, which is composed of Cognitive Utility Reward (CUR) and Cognition-Adaptive Regulation (CAR).

soning imposes greater cognitive burden and requires integration with an increasingly complex set of prior information. Furthermore, prolonged reasoning exhibits efficiency decay caused by noise accumulation (Shannon, 1948) and error propagation (Lee and Cummins, 2004; Chen et al., 2015). The proposed linear marginal cost function offers a *first-order approximation* of the escalating cognitive load, modeling each successive reasoning as increasingly computationally demanding.

The efficiency factor $E(L_i)$ is derived by setting its relative decay rate equal to $\gamma(L_i)$:

$$\frac{dE(L_i)}{dL_i} = -\gamma(L_i)E(L_i) \tag{2}$$

Subject to the boundary condition $E(0) = 1$ (*i.e.*, full efficiency at zero length), the solution to Eq. (2) is (see Appendix C for detailed derivation):

$$E(L_i) = \exp\left(-\frac{1}{2}kL_i^2\right) \tag{3}$$

The final utility U_i for o_i is computed as the product of the correctness reward R_i and the efficiency factor $E(L_i)$. For notational simplicity, the quadratic cost coefficient is defined as $\eta = k/2$:

$$U_i = R_i \cdot \exp\left(-\eta L_i^2\right) \tag{4}$$

In our implementation, the length is normalized by the maximum sequence length L_{max} within each batch to handle variation in sequence lengths. This yields the final CUR formulation:

$$U_i = R_i \cdot \exp\left(-\eta \cdot \frac{L_i^2}{L_{\text{max}}^2}\right) \tag{5}$$

To effectively guide policy updates, these utility scores U_i are used to compute the group-normalized advantage given as:

$$\hat{A}_i = \frac{U_i - \mu_U}{\sigma_U + \epsilon} \tag{6}$$

where μ_U and σ_U are the mean and standard deviation of $\{U_i\}_{i=1}^G$. ϵ is a small constant for numerical stability. By using this utility-based advantage, the model learns to prioritize responses that are both correct and efficient. Further analyses of CUR are shown in Appendices D.2, D.3, and D.4.

4.2 Cognition-Adaptive Regulation (CAR)

A static KL-divergence coefficient β cannot adapt to samples with varying difficulty levels and cognitive demands. This mismatch leads to overly loose exploration for simple problems and excessively restrictive exploration for cognitively complex ones. To mitigate this limitation, we propose **Cognition-Adaptive Regulation (CAR)** that adaptively modulates KL regularization strength based on real-time estimates of cognitive load for each sample. For each training batch, CAR operates as follows:

Step 1: To approximate the cognitive demand of each sample, the difficulty is quantified as the model's failure rate across G generated responses. Specifically, the difficulty for each sample $d(q_j)$ is computed as:

$$d(q_j) = 1 - \frac{1}{G}\sum_{i=1}^{G} R_i \tag{7}$$

where a higher failure rate indicates greater cognitive complexity, reflecting the increased cognitive effort required for reasoning correctness.

Step 2: Relative difficulty within a batch provides a more informative signal than absolute difficulty scores. For instance, a simple problem may induce greater cognitive effort compared with other samples. Accordingly, a standardized difficulty score z_j is derived for each sample:

$$z_j = \frac{d(q_j) - \mu_d}{\sigma_d + \epsilon} \tag{8}$$

where μ_d and σ_d are the batch mean and standard deviation of $\{d(q_j)\}$, respectively.

Step 3: z_j is mapped to an adaptive KL-divergence coefficient β_{ada} using a $\tanh$ function to establish a smooth, bounded, and inverse relationship (*i.e.*, reduced penalties for more cognitively demanding samples):

$$\beta_{\text{ada}} = \beta_{\text{base}} \cdot (1 - \tanh(z_j)) \tag{9}$$

where β_{base} serves as a scaling factor. This cognition-adaptive mechanism allows CAR to effectively allocate the reasoning capacity. To be specific, it relaxes constraints to encourage exploration on cognitively challenging samples while maintaining reasoning performance on simpler ones. Further analyses on Eq. (9) are shown in Appendices D.7 and D.8.

Step 4: Finally, β_{ada} is integrated into the standard GRPO objective (Shao et al., 2024) as follows:

$$\begin{aligned}\mathcal{L}(\theta) =& \mathbb{E}\Big[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big(r_{i,t}(\theta)\hat{A}_i, \\ & \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\Big) \\ & - \beta_{\text{ada}} D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\Big]\end{aligned} \tag{10}$$

where $r_{i,t}(\theta)$ is the importance sampling ratio. The policy optimization is then guided by the CUR-based advantage $\hat{A}_i$ and regularized by cognition-adaptive KL-divergence.

5 Experiments

Our experiments are designed to investigate four core research questions regarding the reasoning performance of CGPO: **RQ1:** How does CGPO perform against state-of-the-art RLVR baselines? **RQ2:** Does CGPO learn to adapt its reasoning depth based on task difficulty? **RQ3:** How do the individual components (CUR and CAR) contribute to the final results? **RQ4:** What is the impact of hyperparameters η and β_{base} on model sensitivity?

5.1 Settings

Models. In this study, DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025), DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025), and DeepScaleR-Preview-1.5B (Luo et al., 2025) were chosen for mathematical reasoning. For SQL generation and multi-modal reasoning tasks, we applied Qwen2.5-Coder-3B-Instruct (Hui et al., 2024) and Qwen2.5-VL-3B-Instruct (Bai et al., 2025), respectively.

Datasets. In this work, the MATH dataset (Hendrycks et al., 2021) was used for training in mathematical reasoning. Evaluation was carried out across widely-used benchmarks, encompassing MATH500 (Hendrycks et al., 2021), AIME24 (Li et al., 2024a), AIME25 (Codeforces), AMC23 (Ouyang et al., 2022), Minerva (Lewkowycz et al., 2022), and OlympiadBench (Huang et al., 2024). For SQL generation, we trained models using BIRD-Train (Li et al., 2024b) and tested on Spider-Dev (Yu et al., 2018) and BIRD-Dev (Li et al., 2024b). Multi-modal reasoning experiments relied on Geometry3K (Lu et al., 2021), which included dedicated training and test subsets. Additional details are provided in Appendix A.

Implementation Details. We use GRPO (Shao et al., 2024), AdaptThink (Zhang et al., 2025), HAPO (Huang et al., 2025), LASER (Liu et al., 2025), and ALP (Xiang et al., 2025) as our baseline methods. (See Appendix B for more details about the baselines). η, β_{base}, batch size, the number of rollouts, and the sampling temperature were set to 1.0, 0.001, 512, 8, and 1.0, respectively. AdamW (Zhou et al., 2024) is used with a learning rate of 1×10^{-6} and we trained each model for 400 steps. For evaluation, the sampling temperature was 0.6. We use Pass@1 score and average token length as evaluation metrics. Notably, the Pass@1 score denoted execution accuracy in SQL generation. Computing resources comprised eight NVIDIA GeForce A100 80GB GPUs.

5.2 Experimental Results

5.2.1 Main Results (RQ1)

Table 2 shows that CGPO establishes a superior accuracy-efficiency trade-off on mathematical reasoning. For instance, using DeepScaleR-Preview-1.5B, CGPO attains the highest average Pass@1

Methods	Math500		AIME24		AIME25		AMC23		Minerva		Olympiad		Avg.	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
DeepSeek-R1-Distill-Qwen-1.5B														
Base	77.6	4086	16.7	8024	16.7	7921	62.5	7628	27.6	4821	42.4	7638	40.6	6786
GRPO	81.8	3106	20.0	7626	20.0	7231	67.5	5254	30.5	3845	47.6	5832	44.6	5482
AdaptThink	79.6	**1728**	23.3	**6628**	23.3	7014	70.0	**3823**	29.8	3248	45.5	4723	45.3	**4527**
HAPO	81.4	2601	23.3	7469	23.3	7217	**72.5**	4371	30.2	3472	46.6	4812	46.2	4990
LASER	81.8	2426	**30.0**	7225	26.7	7523	**72.5**	4285	31.3	2873	48.2	**4626**	48.4	4826
ALP	79.2	2241	26.7	6671	**30.0**	6892	70.0	4176	30.9	2447	46.7	5094	47.3	4587
CGPO	**82.6**	2180	**30.0**	6662	**30.0**	**6826**	**72.5**	4049	**31.6**	**2835**	**48.7**	4646	**49.2**	4533
DeepSeek-R1-Distill-Qwen-7B														
Base	91.0	2836	50.0	9613	50.0	9425	77.5	5617	36.8	4673	63.1	6682	61.4	6474
GRPO	91.6	2654	50.0	8872	50.0	8545	80.0	5211	39.7	4153	64.2	6109	62.6	5924
AdaptThink	89.4	**1952**	53.3	**7104**	53.3	7315	80.0	**3908**	38.6	3105	64.5	4882	63.2	**4711**
HAPO	91.8	2413	53.3	7821	53.3	7534	82.5	4412	40.1	3519	65.9	5327	64.5	5171
LASER	92.0	2305	**56.7**	7618	56.7	7822	**85.0**	4306	40.8	3054	66.8	5113	66.3	5036
ALP	91.6	2108	53.3	7415	56.7	7429	82.5	4117	40.4	2956	66.5	4945	65.2	4828
CGPO	**92.2**	2083	**56.7**	7258	**56.7**	**7243**	**85.0**	4056	**41.2**	**2912**	**67.2**	**4867**	**66.5**	4737
DeepScaleR-Preview-1.5B														
Base	**86.4**	3063	26.7	8924	26.7	9127	67.5	4716	**34.6**	4261	51.8	5268	49.0	5893
GRPO	85.6	2736	26.7	7612	26.7	8162	67.5	4124	34.2	3562	52.7	4528	48.9	5121
AdaptThink	85.0	**1448**	26.7	6129	26.7	6368	65.0	3127	30.9	2529	48.2	3612	46.6	3869
HAPO	84.8	2434	26.7	6836	26.7	7023	67.5	3756	32.7	3268	50.6	4176	48.2	4582
LASER	85.8	2263	**30.0**	5926	26.7	6147	**70.0**	3548	32.0	3111	**53.3**	3925	49.6	4153
ALP	85.2	2052	26.7	5457	**30.0**	**5612**	**70.0**	2852	31.6	**2427**	47.9	**3551**	48.6	3659
CGPO	85.6	1632	26.7	**5398**	**30.0**	5624	**70.0**	**2811**	33.5	2674	53.1	3660	**49.8**	**3633**

Table 2: Experimental results of different baselines across various mathematical reasoning benchmarks in terms of Pass@1 score and reasoning length. The best results are highlighted in bold.

Methods	Geometry3K		Spider-Dev		BIRD-Dev		SQL Avg.	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
Base	25.7	552	70.2	378	43.6	536	56.9	457
GRPO	35.6	418	73.8	283	55.7	482	64.8	383
HAPO	33.9	386	72.8	297	54.7	446	63.8	372
LASER	35.6	351	73.5	242	55.4	458	64.5	350
ALP	34.6	338	73.2	**228**	54.2	**431**	63.7	**330**
CGPO	**36.1**	**322**	**74.3**	246	**56.0**	441	**65.2**	344

Table 3: Experimental results on multi-modal reasoning and SQL generation benchmarks. For each task, we report Pass@1 score and average response length. The best results in each dataset are highlighted in bold.

score of 49.8, surpassing GRPO by 1.3 points while reducing the average response length by 29.1%. This improvement in accuracy and efficiency highlights the superior optimization strategy of CGPO. Moreover, the superiority of CGPO is further showcased on SQL generation and multi-modal reasoning tasks (Table 3). Notably, it achieves the highest Pass@1 while maintaining superior token efficiency compared to most baselines. These gains across diverse domains affirms the broad applicability and robustness of CGPO.

5.2.2 Analysis of Adaptive Reasoning (RQ2)

Figure 3(a) shows that CGPO exhibits difficulty-aware resource allocation, where the model adaptively adjusts its reasoning depth based on sample difficulty. This adaptability is attributed to the synergy between CUR and CAR. CUR provides a global pressure for conciseness, while CAR grants the local flexibility to tackle complex problems. Meanwhile, the trend of decreasing length and increasing reward (Figures 3(b) and (c)) confirm that CGPO effectively balances accuracy and efficiency. To quantify this behavior, we introduce the *Cost per Accuracy (CpA)* metric:

$$\text{CpA} = \frac{\text{Average Length}}{\text{Pass@1}} \tag{11}$$

where a lower value denotes higher reasoning efficiency (*i.e.*, fewer tokens required per correct solu-

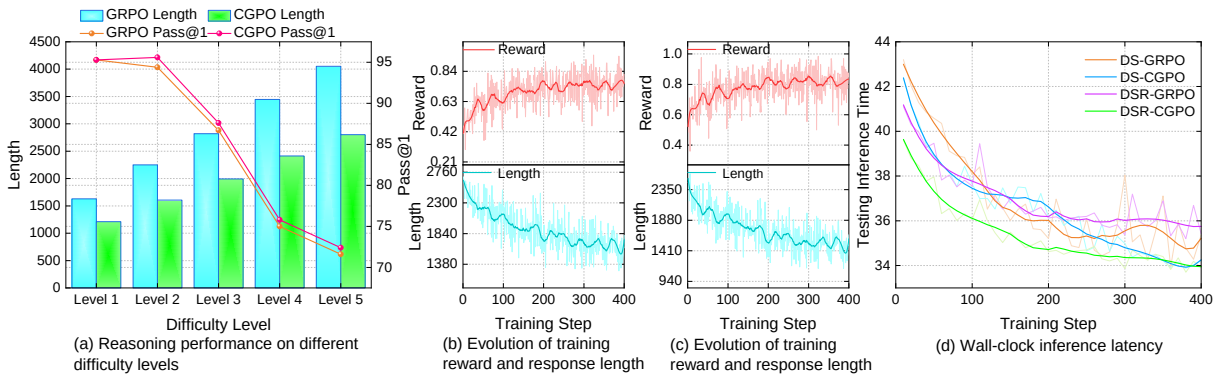

Figure 3: Evaluation of reasoning efficiency and training dynamics. (a) Average token length and Pass@1 on MATH500 stratified by sample difficulty. (b-c) Evolution of average reward and response length during training for DeepSeek-R1-Distill-Qwen-1.5B and DeepScaleR-Preview-1.5B. (d) Evolution of wall-clock inference latency on MATH500. DS and DSR represent DeepSeek-R1-Distill-Qwen-1.5B and DeepScaleR-Preview-1.5B, respectively.

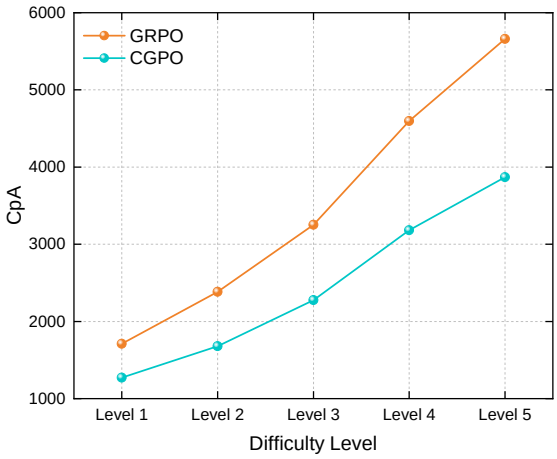

Figure 4: Comparison of CpA for CGPO and GRPO.

tion). Figure 4 reveals that CGPO achieves lower CpA than GRPO. Crucially, this efficiency gap diverges as task difficulty increases, indicating that CGPO invests its computational budget more effectively than GRPO.

Additionally, Figure 3(d) highlights benefits in inference efficiency, where the inference latency is lower for CGPO. This observation confirms that shortened reasoning paths translates directly into lower end-to-end latency. Regarding training overhead, the total wall-clock times for GRPO and CGPO using DeepSeek-R1-Distill-Qwen-1.5B are 649 and 654 minutes, respectively. Consequently, CGPO achieves substantial gains in both inference speed and reasoning performance with negligible additional training costs.

5.2.3 Ablation Study (RQ3)

Synergistic Effect of CUR and CAR. Figure 5(a) highlights the synergistic roles of CUR and CAR. While both modules surpass the GRPO baseline, CUR exhibits a greater standalone improvement. This is because CUR *structurally corrects the learning signal* and produces a better optimization landscape that benefits accuracy and efficiency. In contrast, CAR adaptively modulates the exploration policy within this landscape. The full CGPO integrates CUR's *foundational signal correction* with CAR's *fine-grained regularization control* to achieve the best accuracy-efficiency trade-off. (See Table 6 and Appendix D.9 for detailed results and further analysis).

Impact of Reward Formulation. To validate CUR's design, we compare it with two alternative reward formulations, including *Additive Penalty* and *Rational Quadratic Decay* (Figure 5(b)):

$$U_{i1} = R_i - \eta \frac{L_i^2}{L_{\max}^2} \tag{12}$$

$$U_{i2} = R_i \cdot \frac{1}{1 + \eta \frac{L_i^2}{L_{\max}^2}} \tag{13}$$

Notably, the additive penalty performs the worst, underscoring its flaw of decoupling correctness from efficiency. While the rational quadratic provides a strong baseline, CUR delivers a superior accuracy-efficiency balance, particularly in token reduction. This superiority stems from the mathematical properties of its quadratic exponential decay. To be specific, CUR's exponential decay penalizes extremely long responses more strongly than the rational quadratic. This behavior enables more precise discrimination between necessary complexity and excessive verbosity, which accounts for CUR's superior performance. (For more results

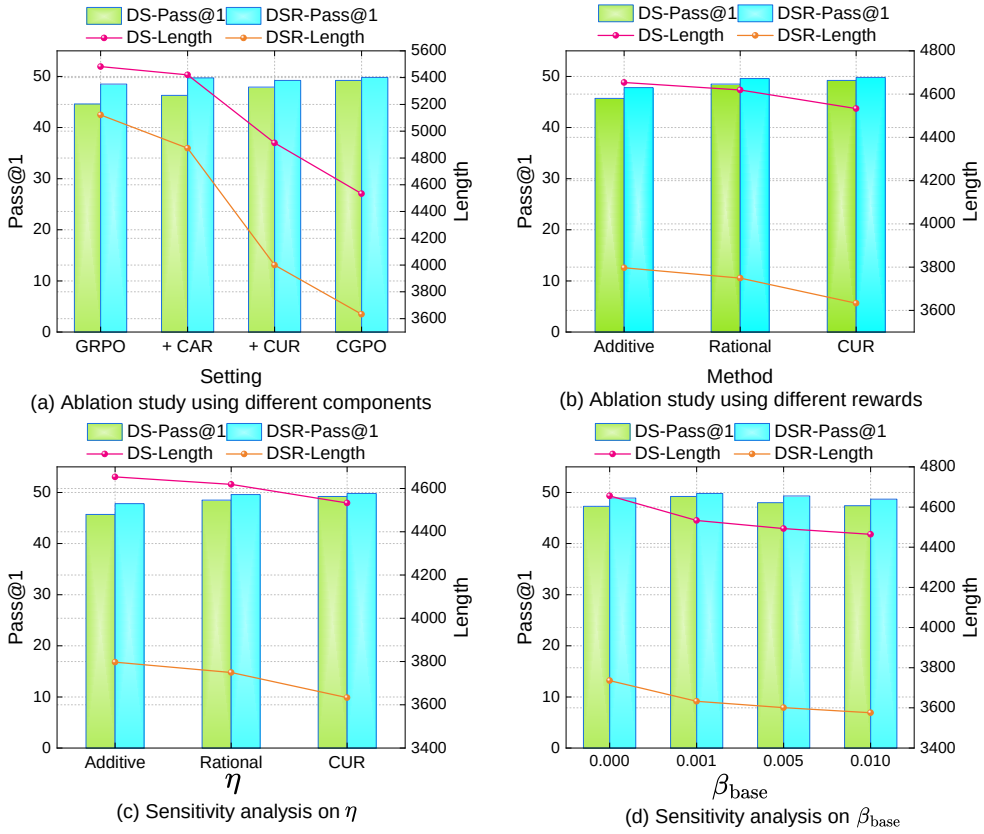

Figure 5: Ablation studies and sensitivity analysis. (a) Ablation study on the components in CGPO. (b) Ablation study on different reward formulations. (c) Sensitivity analysis on η. (d) Sensitivity analysis on β_{base}. Note that DS and DSR indicate DeepSeek-R1-Distill-Qwen-1.5B and DeepScaleR-Preview-1.5B, respectively.

and analyses of reward formulations and the normalization technique, see Table 9, Appendices D.5 and D.6).

5.2.4 Sensitivity Analysis (RQ4)

Impact of η. Figure 5(c) examines the effects of the efficiency coefficient η in CUR, revealing a clear accuracy-efficiency trade-off. Concretely, a low value ($\eta = 0.5$) representing a weaker penalty on length yields good accuracy but with the longest response lengths. Conversely, a high value ($\eta = 1.5$) aggressively shortens responses, which boosts efficiency but impairs performance on complex tasks. Our default $\eta = 1$ attains the highest accuracy while reducing response length compared to $\eta = 0.5$, validating it as the optimal choice. Detailed results and analysis are available in Table 7 and Appendix D.10, respectively.

Impact of β_{base}. Figure 5(d) shows the impact of the base KL coefficient β_{base}, which serves as the anchor point for our CAR mechanism. While CAR demonstrates robustness across a range of values, the choice of β_{base} is crucial for achieving optimal performance. Concretely, setting $\beta_{\text{base}} = 0$ yields suboptimal results, indicating the need for baseline regularization to promote model learning. In contrast, a high value (*e.g.*, $\beta_{\text{base}} = 0.01$) also degrades performance by excessively constraining exploration. Notably, $\beta_{\text{base}} = 0.001$ achieves the highest average Pass@1, confirming it as an effective anchor for adaptive regulation scheme. Additional results and analysis are shown in Table 8 and Appendix D.10, respectively. A case study showing the efficacy of CGPO is given in Appendix F.

6 Conclusion

We develop Cognition-Guided Policy Optimization (CGPO) to effectively balance reasoning accuracy and efficiency in LLMs. Specifically, CGPO combines Cognitive Utility Reward (CUR) and Cognition-Adaptive Regulation (CAR) within a unified RL-based framework. The experimental results conclusively demonstrate that CGPO generates reasoning paths that are both more accurate and more concise, offering a scalable path forward for efficient reasoning. This algorithm provides a general foundation for integrating cognitive principles into future reasoning optimization strategies.

Limitations

We acknowledge two primary constraints in our current study. (1) CGPO utilizes token length as the primary proxy for cost, which may inadvertently penalize necessary elaborations or self-corrections alongside redundancy. The method currently lacks the semantic granularity to distinguish between superfluous verbosity and valuable reasoning details, potentially sacrificing interpretability for brevity. (2) While effective in the difficulty estimation of CAR, this proxy may not fully capture a problem's intrinsic complexity. Future research could investigate more advanced difficulty estimators to improve the precision of cognitive regulation.

Ethical Considerations

We follow strict ethical standards throughout this work: (1) Regarding environmental impact, we acknowledge that training LLMs involves significant energy expenditure. However, our method's focus on token efficiency aims to mitigate the long-term inference cost. (2) To ensure transparency, all benchmarks are publicly accessible, ensuring that personal privacy is strictly protected. Moreover, all fine-tuned models are based on publicly released open-source frameworks, avoiding the utilization of proprietary models.

References

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Huayu Chen, Kaiwen Zheng, Qinsheng Zhang, Ganqu Cui, Yin Cui, Haotian Ye, Tsung-Yi Lin, Ming-Yu Liu, Jun Zhu, and Haoxiang Wang. 2025. Bridging supervised learning and reinforcement learning in math reasoning. *arXiv preprint arXiv:2505.18116*.

Ouhao Chen, Slava Kalyuga, and John Sweller. 2015. The worked example effect, the generation effect, and element interactivity. *Journal of Educational Psychology*, 107(3):689.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

MAA Codeforces. American invitational mathematics examination-aime 2024, 2024.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, et al. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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*.

Zhen Huang, Zengzhi Wang, Shijie Xia, Xuefeng Li, Haoyang Zou, Ruijie Xu, Run-Ze Fan, Lyumanshan Ye, Ethan Chern, Yixin Ye, et al. 2024. Olympicarena: Benchmarking multi-discipline cognitive reasoning for superintelligent ai. *Advances in Neural Information Processing Systems*, 37:19209–19253.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Michael D Lee and Tarrant DR Cummins. 2004. Evidence accumulation in decision making: Unifying the "take the best" and the "rational" models. *Psychonomic bulletin & review*, 11(2):343–352.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. 2024a. Numinamath: The largest public dataset in ai4maths

with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

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

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-chun Zhu. 2021. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6774–6786.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. 2025. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*.

Shangke Lyu, Linjuan Wu, Yuchen Yan, Xingyu Wu, Hao Li, Yongliang Shen, Peisheng Jiang, Weiming Lu, Jun Xiao, and Yueting Zhuang. 2025. Hierarchical budget policy optimization for adaptive reasoning. *arXiv preprint arXiv:2507.15844*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Optimizing anytime reasoning via budget relative policy optimization. *arXiv preprint arXiv:2505.13438*.

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

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

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*.

John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285.

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

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*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2024. Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36.

Yikun Wang, Yibin Wang, Dianyi Wang, Zimian Peng, Qipeng Guo, Dacheng Tao, and Jiaqi Wang. 2025. Geometryzero: Improving geometry solving for llm with group contrastive policy optimization. *arXiv preprint arXiv:2506.07160*.

Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. 2025. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. *arXiv preprint arXiv:2506.05256*.

Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Junnan Li, and Caiming Xiong. 2025. Scalable chain of thoughts via elastic reasoning. *arXiv preprint arXiv:2505.05315*.

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

Yufan Ye, Ting Zhang, Wenbin Jiang, and Hua Huang. 2025. Process-supervised reinforcement learning for code generation. *arXiv preprint arXiv:2502.01715*.

Jingyang Yi, Jiazheng Wang, and Sida Li. 2025. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. *arXiv preprint arXiv:2504.21370*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Zishun Yu, Tengyu Xu, Di Jin, Karthik Abinav Sankararaman, Yun He, Wenxuan Zhou, Zhouhao Zeng, Eryk Helenowski, Chen Zhu, Sinong Wang,

et al. 2025. Think smarter not harder: Adaptive reasoning with inference aware optimization. *arXiv preprint arXiv:2501.17974*.

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

Pan Zhou, Xingyu Xie, Zhouchen Lin, and Shuicheng Yan. 2024. Towards understanding convergence and generalization of adamw. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Appendices Content

A Appendix A: References to Models and Datasets **12**

B Appendix B: Baseline Implementation Details **12**

C Appendix C: Detailed Proof in CUR **13**

D Appendix D: Further Analysis **13**

D.1 Comparison between CUR and Suboptimal Reward Shaping . . . 13

D.2 Analysis of the Functional Form of CUR 14

D.3 Theoretical Foundations of CUR . 15

D.3.1 Derivation from Cumulative Error Dynamics . . . 15

D.3.2 Robustness Against Rank Reversal 16

D.4 Further Analysis of Optimization Dynamics 16

D.4.1 Correctness Consistency . 17

D.4.2 Pareto Optimality of the CUR Objective 17

D.5 Further Analysis on the Reward Formulation in CUR 18

D.6 Further Analysis on the Normalization Technique in CUR 18

D.7 Theoretical Justification for CAR . 19

D.8 Analysis of the Mapping Design for β_{ada} 19

D.9 Synergistic Effect of CUR and CAR 20

D.10 Further Analysis on the Impact of η and β_{base} 20

E Appendix E: Additional Experimental Results **21**

F Appendix F: Case Study **21**

A Appendix A: References to Models and Datasets

This section provides the links regarding the models and datasets used in this work. These datasets are publicly available under the CC BY-SA 4.0 license.

(1) **Models:**

- DeepSeek-R1-Distill-Qwen-1.5B: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B
- DeepSeek-R1-Distill-Qwen-7B: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
- DeepScaleR-Preview-1.5B: https://huggingface.co/agentica-org/DeepScaleR-1.5B-Preview
- Qwen2.5-Coder-3B-Instruct: https://huggingface.co/Qwen/Qwen2.5-Coder-3B-Instruct
- Qwen2.5-VL-3B-Instruct: https://huggingface.co/Qwen/Qwen2.5-VL-3B-Instruct

(2) **Datasets:**

- MATH: (12000 training data) https://huggingface.co/datasets/HuggingFaceH4/MATH
- MATH500: (500 testing data) https://huggingface.co/datasets/HuggingFaceH4/MATH-500
- AIME24: (30 testing data) https://huggingface.co/datasets/HuggingFaceH4/aime_2024
- AIME25: (30 testing data) https://huggingface.co/datasets/HuggingFaceH4/aime_2025
- AMC23: (40 testing data) https://huggingface.co/datasets/math-ai/amc23
- Minerva: (272 testing data) https://huggingface.co/datasets/svc-huggingface/minerva-math
- OlympiadBench: (674 testing data) https://huggingface.co/datasets/knoveleng/OlympiadBench
- Spider: (1034 testing data) https://yale-lily.github.io/spider
- BIRD: (9428 and 1534 data for training and testing https://bird-bench.github.io/
- Geometry3K: (2100 and 601 data for training and testing) https://huggingface.co/datasets/hiyouga/geometry3k

B Appendix B: Baseline Implementation Details

In this section, we provide a detailed description of the baseline implementations used in our experiments. For all baselines, we use the same set of general training hyperparameters as CGPO for a fair comparison. This includes the AdamW optimizer with a learning rate of 1×10^{-6}, a batch size of 512, and a sampling temperature of 1.0 during training. All models were trained for 400 steps.

- **Base:** This refers to the pre-trained model without any reinforcement learning fine-tuning. It serves as the foundational starting point to measure the improvements gained from RL-based methods.
- **GRPO** (Shao et al., 2024): As the foundational RL-based method, GRPO is implemented following its standard configuration. We use a clipping ratio of $\epsilon = 0.2$ and a static KL-divergence coefficient of $\beta = 0.001$.
- **AdaptThink** (Zhang et al., 2025): This method enables the model to adaptively switch between "thinking" and "non-thinking" modes. We implemented it following the official paper's guidelines, setting the number of sampled responses $K = 8$ and $\delta = 0.05$.
- **HAPO** (Huang et al., 2025): HAPO introduces a cosine-based penalty that considers historical response lengths. We adopt the key hyperparameters from the original paper, setting $w = 1.0$ and $c = -0.7$.
- **LASER** (Liu et al., 2025): This method uses a simple yet effective length-based reward shaping. Following its official implementation, we set $\alpha = 0.5$ and $L_T = 800$.

- **ALP** (Xiang et al., 2025): ALP proposes an additive penalty that adapts to the batch-level correctness rate. We set $\beta = 1 \times 10^{-7}$ following the original paper.

By adhering closely to the original configurations, we ensure that our comparisons are fair and accurately reflect the relative performance of each method against CGPO.

C Appendix C: Detailed Proof in CUR

Proposition 1 (Closed-Form Solution for the Efficiency Factor)**.** *Given the marginal cost function in Eq. (1) and the decay principle in Eq. (2), the unique closed-form solution for the efficiency factor $E(L_i)$ is:*

$$E(L_i) = \exp\left(-\frac{1}{2}kL_i^2\right) \tag{14}$$

Proof. We solve the differential equation from Eq. (2) using separation of variables.

1. **Separate Variables:** We rearrange Eq. (2) to get:

$$\frac{1}{E(L_i)}dE(L_i) = -\gamma(L_i)dL_i \tag{15}$$

2. **Integrate Both Sides:**

$$\int \frac{1}{E(L_i)}dE(L_i) = -k\int L_i dL_i \tag{16}$$

3. **Perform the Integration:**

$$\ln|E(L_i)| = -\frac{1}{2}kL_i^2 + C \tag{17}$$

where C is the constant of integration.

4. **Solve for $E(L_i)$:**

$$|E(L_i)| = e^C \cdot \exp\left(-\frac{1}{2}kL_i^2\right) \tag{18}$$

5. **Apply Boundary Condition:** Using $E(0) = 1$ at $L_i = 0$, we find:

$$|E(0)| = e^C \cdot e^0 \implies 1 = e^C \tag{19}$$

6. **State the Final Solution:** Substituting $e^C = 1$ and noting that $E(L_i) \geq 0$, we arrive at the unique solution shown in Eq. (14).

This completes the proof. □

D Appendix D: Further Analysis

D.1 Comparison between CUR and Suboptimal Reward Shaping

The primary advantage of CUR is its multiplicative structure, which realigns correctness and efficiency. While group normalization may still yield negative advantages for certain correct responses, CUR greatly reduces this issue, as evidenced in Tables 4 and 5.

Deficiency 1: Decoupling Error in Additive Formulations. Additive reward formulations (*i.e.*, $U_i = R_i -$ penalty) are structurally flawed as they decouple correctness rewards from length penalties. This separation allows length penalties to disproportionately reduce the utility of correct answers. While the base reward reflects correctness, excessive penalties can push long but valid responses below the group mean. Consequently, these valid responses receive negative advantages, which is a critical alignment failure. In particular, these methods discourage correct reasoning only due to length, which violates the principle that correctness should always be rewarded.

Deficiency 2: Polarization Error in Coarse Multiplicative Formulations. Notably, multiplicative methods such as LASER attempt to address the decoupling problem by setting zero utility for incorrect responses. However, they introduce an issue termed *reward polarization*. By employing a hard-threshold, LASER divides correct solutions into a high-reward group (*i.e.*, short answers) and a baseline-reward group (*i.e.*, long answers). This creates a significant disparity in the reward distribution. Under group normalization, inflated rewards for short answers raise the group mean and push longer correct responses below the mean. As a result, valid but long responses receive negative advantages.

CUR's Principled Multiplicative Solution. In contrast, CUR's multiplicative structure (*i.e.*, $U_i = R_i \cdot$ efficiency factor) functions as a *gating mechanism* to circumvent both deficiencies:

(1) **For the Decoupling Error**: By design, incorrect responses yield zero utility, and any correct response retains strictly positive utility. This separation preserves the ordering between correct and incorrect outcomes. As the advantages of correct answers are mostly not suppressed below zero due

Method	Reward Formulation
ALP (Xiang et al., 2025)	$R_i - \beta \cdot \lvert o_i \rvert \cdot \max(\text{mean}\{R_i\}, K^{-1})$
HAPO (Huang et al., 2025)	$R_i + w \cdot \max\left(\cos(\min(\frac{\pi}{2}\frac{\lvert o_i \rvert}{h(q)}, \pi)), c\right) R_i + w \cdot \min\left(\cos(\min(\frac{\pi}{2}\frac{\lvert o_i \rvert}{h(q)}, \pi)), 0\right)(1 - R_i)$
LASER (Liu et al., 2025)	$R_i + R_i \cdot \alpha \cdot \mathbb{I}(\lvert o_i \rvert \leq L_T)$

Table 4: Examples of reward designs for efficient reasoning, where $R_i = \mathbb{I}(o_i \text{ is correct}) \in \{0, 1\}$.

Method	Hyperparameter	Length	Correctness	Reward	Advantage
ALP	$\beta = 4 \times 10^{-4}$ $K = 8$	[400, 600, 1500, 900, 2800, 250, 1200, 4500]	[1, 1, 1, 0, 1, 1, 0, 1]	[0.88, 0.82, 0.55, -0.27, 0.16, 0.925, -0.36, -0.35]	[1.10, 0.99, 0.48, -1.06, -0.25, 1.19, -1.23, -1.21]
HAPO	$w = 1$ $c = -0.8$ $h(q) = 1000$	[400, 600, 1500, 900 2800, 250, 1200, 4500]	[1, 1, 1, 0, 1, 1, 0, 1]	[1.81, 1.59, 0.29, 0.00, 0.20, 1.92, -0.31, 0.20]	[1.53, 1.22, -0.58, -0.99, -0.71, 1.68, -1.42, -0.71]
LASER	$\alpha = 1.5$ $L_T = 800$	[400, 600, 1500, 900, 2800, 250, 1200, 4500]	[1, 1, 1, 0, 1, 1, 0, 1]	[2.50, 2.50, 1.00, 0.00, 1.00, 2.50, 0.00, 1.00]	[1.13, 1.13, -0.29, -1.25, -0.29, 1.13, -1.25, -0.30]
CGPO	$\eta = 1$	[400, 600, 1500, 900, 2800, 250, 1200, 4500]	[1, 1, 1, 0, 1, 1, 0, 1]	[0.992, 0.982, 0.895, 0, 0.679, 0.997, 0, 0.368]	[0.97, 0.94, 0.72, -1.57, 0.17, 0.98, -1.57, -0.63]

Table 5: An analysis of learning signals from different reward designs. Advantages are computed by normalizing rewards within a group of 8 samples (see Eq. (6)), and negative values for correct responses are highlighted in blue.

to length, CUR provides consistent signals that reinforce correctness.

(2) **For the Polarization Error**: Unlike LASER's discrete bonus, CUR employs a smooth and continuous decay in its efficiency factor. This produces a fine-grained, monotonic ranking of correct responses by length, substantially reducing unjustified penalties for moderately long but valid answers under group normalization.

Summary. Adopting a multiplicative formulation is a prerequisite for maintaining the proper hierarchical relationship between correctness and efficiency. However, this measure alone is insufficient. The advantage of CUR stems equally from the functional design of its efficiency component. By introducing a smooth decay rather than a coarse threshold, CUR establishes a more consistent foundation for policy optimization than both additive penalties and rigid multiplicative schemes.

D.2 Analysis of the Functional Form of CUR

The design of CUR is based on the assumption that the marginal cost of reasoning increases linearly, resulting in a quadratic exponential decay. To formalize these notion, we present the following lemma regarding the final CUR utility function for correct responses (*i.e.*, $R_i = 1$):

Lemma 1 (Properties of the CUR Function)**.** *For any given batch with a fixed $L_{\max}$, the CUR utility function $U(L) = \exp(-\eta \frac{L^2}{L_{\max}^2})$ is:*

1. ***Strictly Bounded:*** *The utility $U(L)$ is strictly bounded within the interval $(0, 1]$.*
2. ***Monotonically Decreasing:*** *For any length $L > 0$, $U(L)$ is a strictly decreasing function of L.*

Proof. The above two properties can be proven as follows:

1. **Boundedness:** Since $L^2/L_{\max}^2 \geq 0$ and $\eta > 0$, $-\eta \frac{L^2}{L_{\max}^2} \leq 0$ and $U(L) \leq e^0 = 1$. As the exponential function is always positive, we have $U(L) \in (0, 1]$. This guarantees that all correct solutions receive a positive utility, fundamentally distinguishing CUR from additive penalties.
2. **Monotonicity:** The first derivative of $U(L)$ with respect to L is:
$$\frac{dU(L)}{dL} = \exp\left(-\eta \frac{L^2}{L_{\max}^2}\right) \cdot \left(-\frac{2\eta L}{L_{\max}^2}\right) \tag{20}$$
For $L > 0$, the exponential term is strictly positive, while the term in the parentheses is strictly negative. Therefore, $\frac{dU(L)}{dL} < 0$, indicating that the function is monotonically decreasing. This ensures that shorter correct solutions are always preferred.

□

These proven properties directly support our designed CUR function:

Justification for the Linear Marginal Cost Formulation. The primary justification for a *linear marginal cost* (*i.e.*, $\gamma(L) = kL$) lies in its ability to represent *escalating cognitive load* in an analytically tractable manner. As a reasoning process extends, the cognitive burden of managing context and avoiding error propagation increases. A linear increase offers the simplest formal representation of this "harder-as-it-goes" phenomenon. In contrast, higher-order cost functions (*e.g.*, $\gamma(L) \propto L^2$) introduce additional complexity without compelling theoretical justification. Therefore, inspired by cognitive science principles (Sweller, 1988), the linear approximation provides a robust and interpretable foundation.

Justification for Batch-wise Normalization by $L_{\max}$. The use of *batch-wise normalization by* $L_{\max}$ is a practical strategy for addressing the intrinsic *non-stationarity* of reasoning tasks. Absolute length penalties are context-sensitive, as a response may be concise for one problem but verbose for another. Therefore, batch-wise normalization defines efficiency scores relative to the current distribution of solution lengths, introducing an adaptive baseline. This approach ensures that penalties remain meaningful and stable across batches with varying average response lengths. (See Appendix D.6 for further analysis).

D.3 Theoretical Foundations of CUR

In this work, we introduced CUR with a quadratic exponential decay form: $r_{cur} = R \cdot e^{-\eta L^2}$. While we provided an intuition based on cognitive load theory, this section derives CUR from two perspectives: (1) a probabilistic reliability model based on cumulative error dynamics, and (2) an optimization stability analysis based on gradient sign consistency.

D.3.1 Derivation from Cumulative Error Dynamics

We posit that the quadratic penalty term (L^2) naturally arises from the probability of a reasoning chain remaining valid as its length increases, particularly under the assumption of cumulative error accumulation.

Assumption 1 (Context-Dependent Error Drift)**.** *Consider a reasoning chain of length L. Within the effective reasoning where the model maintains coherence, the probability of making an error at token t (i.e., ϵ_t) is not constant. Due to attention dispersion and hallucination drift, the instantaneous error risk increases linearly:*

$$\epsilon_t \approx \lambda \cdot t \tag{21}$$

where $0 < \lambda \ll 1$ is a small drift coefficient representing the degradation of precision as context grows.

Remark 1 (Justification via Error Cascades)**.** *This linear drift assumption is supported by the phenomenon of error cascades in LLMs. Specifically, minor early hallucinations or logical deviations alter the context for subsequent tokens, increasing the conditional probability of future errors. Consequently, the risk of reasoning collapse grows cumulatively as the chain extends.*

Proposition 2 (Alignment with Linear Error Drift)**.** *Under Assumption 1 and within the regime of small local error probabilities ($\epsilon_t \ll 1$), the probability that a reasoning chain of length L remains entirely correct decays according to a quadratic exponential function. Therefore, the CUR objective aligns with maximizing the expected validity of the response.*

Proof. The probability that the entire chain is valid $P(\text{Valid}|L)$ is the product of the success probabilities $(1 - \epsilon_t)$ at each token t:

$$P(\text{Valid}|L) = \prod_{t=1}^{L}(1 - \epsilon_t) = \prod_{t=1}^{L}(1 - \lambda t) \tag{22}$$

To analyze the decay rate, we examine the log-probability:

$$\ln P(\text{Valid}|L) = \sum_{t=1}^{L} \ln(1 - \lambda t) \tag{23}$$

We focus on the high-fidelity regime where the model produces plausible reasoning (*i.e.*, $\epsilon_t \ll 1$). In this range, we can apply the Taylor expansion approximation $\ln(1 - x) \approx -x$:

$$\ln P(\text{Valid}|L) \approx \sum_{t=1}^{L}(-\lambda t) = -\lambda \sum_{t=1}^{L} t \tag{24}$$

Using the summation formula for an arithmetic series $\sum_{t=1}^{L} t = \frac{L(L+1)}{2} \approx \frac{L^2}{2}$ (for sufficiently large L):

$$\ln P(\text{Valid}|L) \approx -\frac{\lambda}{2} L^2 \tag{25}$$

Exponentiating back to the probability space yields:

$$P(\text{Valid}|L) \approx \exp\left(-\frac{\lambda}{2}L^2\right) \tag{26}$$

By setting the hyperparameter $\eta = \frac{\lambda}{2}$, we recover the exact form of the CUR scaling factor. This demonstrates that the L^2 term provides a theoretically motivated approximation for the cumulative probability of success under linear error drift. □

D.3.2 Robustness Against Rank Reversal

In group-based policy optimization (*e.g.*, GRPO), the optimization direction is determined by the advantage $A_i = \frac{r_i - \mu}{\sigma}$, where μ is the group mean. A critical requirement for stability is that the relative preference between a correct response and an incorrect response must be preserved, regardless of the baseline μ.

Definition 1 (Advantage Monotonicity)**.** *Let y_c be any correct response and y_i be any incorrect response. A reward function preserves Advantage Monotonicity if, for any shared baseline μ and scale σ:*

$$A(y_c) > A(y_i) \iff r(y_c) > r(y_i) \tag{27}$$

Theorem 1 (Rank Reversal in Additive Penalties)**.** *Additive penalty functions cause Rank Reversal, where the optimizer is explicitly encouraged to prefer incorrect responses over sufficiently long correct responses. CUR is immune to this pathology.*

Proof. Assume the standard setting where incorrect responses receive zero base reward ($r(y_i) = 0$) and correct responses receive a positive base reward of 1 scaled by efficiency.

Case 1: Additive Failure. Let $r_{add} = 1 - \beta L$. If a correct chain length $L > 1/\beta$, then $r_{add}(y_c) < 0$. Consequently, $r_{add}(y_c) < r(y_i)$. Applying the advantage formula:

$$A(y_c) = \frac{r_{add}(y_c) - \mu}{\sigma} < \frac{0 - \mu}{\sigma} = A(y_i) \tag{28}$$

Result: $A(y_c) < A(y_i)$. The policy gradient update will increase the probability of the incorrect response y_i relative to the correct response y_c. The model actively avoids correct long-form reasoning to prevent length penalty.

Case 2: CUR Stability. Let $r_{cur} = 1 \cdot e^{-\eta L^2}$. For any finite L, $r_{cur}(y_c) > 0$. Therefore, $r_{cur}(y_c) > r(y_i)$ is globally true.

$$A(y_c) = \frac{e^{-\eta L^2} - \mu}{\sigma} > \frac{0 - \mu}{\sigma} = A(y_i) \tag{29}$$

Result: $A(y_c) > A(y_i)$ always holds. □

Corollary 1 (Critical Breakdown Threshold)**.** *Specifically, for an additive penalty coefficient β, there exists a critical length difference beyond which the reward system fails. If a correct response has length L_c such that:*

$$L_c > \frac{1}{\beta} \tag{30}$$

then $R_{add}(y_c) < 0$. Assuming incorrect responses receive 0 reward, the optimizer will explicitly suppress this correct reasoning path. This prevents the model from solving complex problems that require lengths exceeding $1/\beta$.

Remark 2. *It is important to note that in CUR, $A(y_c)$ can indeed be negative (if $r_{cur}(y_c) < \mu$, i.e., the correct response is significantly less efficient than the average of the current batch). However, Theorem 1 guarantees that it is strictly less negative than the advantage of an incorrect response $A(y_i)$. Therefore, the optimizer correctly prioritizes:*

Efficient Correct $\succ$ Inefficient Correct $\succ$ Incorrect

In contrast, additive penalties distort this hierarchy to:

Efficient Correct $\succ$ Incorrect $\succ$ Inefficient Correct

which fundamentally misaligns the optimization objective.

Summary. The dual perspectives above provide a holistic justification for CUR. Concretely, the probabilistic derivation ensures the reward magnitude reflects the *intrinsic reliability* of long chains, while the rank analysis guarantees *optimization stability*. Collectively, they establish CUR as a theoretically sound objective for efficient reasoning. More analysis on CUR is shown in Appendix D.4.

D.4 Further Analysis of Optimization Dynamics

In this section, we provide further analysis to justify the superiority of CUR from the perspectives of *Correctness Consistency* and *Pareto Efficiency*.

D.4.1 Correctness Consistency

A fundamental risk in efficiency-oriented RL is *policy degeneration*, where the model unlearns valid reasoning paths since the length penalty outweighs the correctness reward. We define *Correctness Consistency* to analyze this phenomenon.

Definition 2 (Correctness Consistency)**.** *Let y_{pos} be any correct response and y_{neg} be any incorrect response. A reward shaping mechanism satisfies* ***Correctness Consistency*** *if the reward for correctness always strictly exceeds the reward for incorrectness, regardless of length:*

$$R(y_{pos}, L_{pos}) > R(y_{neg}, L_{neg}), \quad \forall L_{pos}, L_{neg} \tag{31}$$

Theorem 2 (Violation of Consistency in Additive Penalties)**.** *Additive penalty methods violate Correctness-Consistency, leading to Catastrophic Rank Reversal. Specifically, there exists a critical length difference ΔL_{crit} where a valid long response receives a lower reward than an invalid short response.*

Proof. Let $R_{add}(y) = \mathbb{I}(y) - \beta L(y)$. Consider y_{long} (correct) and y_{short} (incorrect). The condition for rank reversal is $R(y_{long}) < R(y_{short})$:

$$1-\beta L_{long} < 0-\beta L_{short} \implies L_{long}-L_{short} > \frac{1}{\beta} \tag{32}$$

When this occurs, the optimizer assigns a higher value to the incorrect response. In the context of advantage estimation, this can lead to $A(y_{short}) > A(y_{long})$, causing the policy to *reinforce the incorrect behavior* over the correct but verbose one. □

Theorem 3 (Consistency Guarantee of CUR)**.** *CUR satisfies Correctness Consistency, ensuring that gradient signals always prioritize correctness over length.*

Proof. The CUR formulation is $R_{cur}(y) = \mathbb{I}(y) \cdot \phi(L)$. For any y_{long} (correct) and y_{short} (incorrect):

$$R_{cur}(y_{long}) = \phi(L_{long}) > 0$$

$$R_{cur}(y_{short}) = 0$$

Since $\phi(L)$ is strictly positive, $R(y_{long}) > R(y_{short})$ holds universally. □

Remark 3. *While $A(y_{long})$ may become negative if $R(y_{long})$ falls below the group mean μ (i.e., it is less efficient than peers), it is structurally guaranteed to remain ranked higher than any incorrect response. This ensures the negative signal targets inefficiency, whereas additive methods risk targeting correctness itself due to the rank flip.*

D.4.2 Pareto Optimality of the CUR Objective

We further clarify that the design of CUR is a solution to a constrained optimization problem on the accuracy-efficiency Pareto frontier.

Proposition 3 (Equivalence to Constrained Optimization)**.** *Maximizing the expected CUR objective $J_{CUR}(\pi)$ is equivalent to solving the dual problem of maximizing accuracy subject to a soft constraint on the reasoning length variance.*

Proof. Consider the primary objective of maximizing the expected correctness R and a constraint to keep the reasoning steps within a computational budget. We formulate this as:

$$\max_{\pi} \mathbb{E}_{y\sim\pi}[R(y)] \quad \text{s.t.} \quad \mathbb{E}_{y\sim\pi}[L(y)^2] \leq C \tag{33}$$

where L^2 represents the quadratic cognitive cost (as derived in Appendix C). The Lagrangian of this problem is:

$$\mathcal{L}(\pi, \lambda) = \mathbb{E}_{y\sim\pi}[R(y)] - \lambda(\mathbb{E}_{y\sim\pi}[L(y)^2] - C) \tag{34}$$

Rearranging the terms inside the expectation:

$$\mathcal{L}(\pi, \lambda) = \mathbb{E}_{y\sim\pi}[R(y) - \lambda L(y)^2] + \lambda C \tag{35}$$

While the standard Lagrange form is additive ($R - \lambda L^2$), consider the optimization in the utility space. For correct responses ($R = 1$), the CUR objective implies:

$$\ln(U_{CUR}) = \ln(1) - \eta \frac{L^2}{L_{max}^2} = -\lambda L^2 \tag{36}$$

This mirrors the cost penalty term in the Lagrangian. Therefore, maximizing the expected CUR utility effectively imposes a soft constraint on the squared length variance to push the policy towards the Pareto frontier, where correctness is maximized for a given computational budget defined by η. □

Remark 4 (Physical Meaning of η)**.** *From the perspective of Lagrangian multipliers, the hyperparameter η in CUR represents the exchange rate between accuracy and computation on the Pareto frontier.*

$$\mathcal{L}(\pi, \eta) = \mathbb{E}[R] - \eta \cdot \mathbb{E}[L^2] \tag{37}$$

Tuning η allows us to traverse the Pareto frontier, where a larger η prioritizes efficiency and a smaller η favors accuracy. This insight offers a practical guideline for hyperparameter selection. In particular, η should be aligned with the latency constraints of the application. For instance, real-time systems requiring rapid responses warrant a larger η, whereas correctness-oriented reasoning tasks benefit from a smaller η.

D.5 Further Analysis on the Reward Formulation in CUR

The superiority of CUR's design over additive and rational quadratic formulations derives from the distinct curvature profiles of their decay functions.

The *additive penalty* (*i.e.*, $R_i - \eta(L/L_{\max})^2$) is fundamentally flawed since it decouples correctness from efficiency. Consequently, a correct but verbose solution may yield a lower raw reward than a short incorrect one, resulting in rank reversal and misleading advantage signals.

The comparison with the *rational quadratic formulation* (*i.e.*, $R_i \cdot 1/(1 + \eta(L/L_{\max})^2)$) is more nuanced. Both methods are multiplicative and preserve positive rewards for correct answers. However, their decay rates differ significantly. We formalize this difference in decay characteristics via the following theorem.

Theorem 4 (Asymptotic Dominance of Exponential Decay)**.** *Let the efficiency factor of CUR be $f_{cur}(L) = e^{-cL^2}$ and that of the rational quadratic be $f_{rq}(L) = \frac{1}{1+cL^2}$, where $c = \eta/L_{\max}^2 > 0$. As $L \to \infty$, the exponential decay factor approaches zero orders of magnitude faster than the rational quadratic factor:*

$$\lim_{L\to\infty} \frac{f_{cur}(L)}{f_{rq}(L)} = \lim_{L\to\infty} \frac{1 + cL^2}{e^{cL^2}} = 0 \tag{38}$$

Proof. $\lim_{L\to\infty} \frac{1+cL^2}{e^{cL^2}}$ is an indeterminate form of type $\frac{\infty}{\infty}$. Applying L'Hôpital's Rule by differentiating the numerator and denominator with respect to L yields:

$$\lim_{L\to\infty} \frac{\frac{d}{dL}(1 + cL^2)}{\frac{d}{dL}(e^{cL^2})} = \lim_{L\to\infty} \frac{2cL}{2cLe^{cL^2}} = \lim_{L\to\infty} \frac{1}{e^{cL^2}} \tag{39}$$

As $L \to \infty$, the term e^{cL^2} grows infinitely large, and the limit is 0. This proves that the exponential decay $f_{\text{cur}}(L)$ diminishes significantly faster than the rational decay $f_{\text{rq}}(L)$ for large L. □

Although training operates within a bounded length $L \in [0, L_{\max}]$, this asymptotic dominance governs the curvature of the penalty function within the effective range. As depicted in Figure 6, the exponential function imposes a substantially stronger penalty as $L \to L_{\max}$. This renders CUR particularly effective at curbing excessive overthinking. While its penalty is consistently stricter, the initial region remains sufficiently flat to safeguard essential reasoning steps. However, its subsequent steep decline delivers a decisive signal against emerging verbosity.

Essentially, the exponential decay provides a closer approximation to a *soft-threshold mechanism*. It permits complexity up to a certain point before becoming rapidly suppressive. In contrast, the rational quadratic decay is overly gradual, making it less adept at discriminating between necessary reasoning and unproductive verbosity. This structural advantage enables CUR to strike a superior balance, maintaining accuracy while achieving greater token reduction.

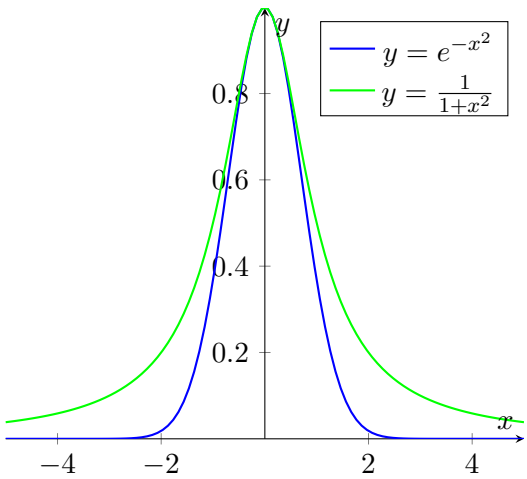

Figure 6: Comparison of the exponential decay function used in CUR (blue) and the rational quadratic decay function (green).

D.6 Further Analysis on the Normalization Technique in CUR

A critical design choice in CUR is the use of batch-wise normalization by maximum length ($L_{\max}$) in Eq. (5). Our analysis indicates that this approach provides greater practical robustness than a non-normalized, absolute length penalty.

Without normalization, CUR reduces to

$$U_i = R_i \cdot \exp(-\eta \cdot L_i^2) \tag{40}$$

This function is inherently brittle for two reasons. First, it makes model performance highly sensitive to the hyperparameter η. A fixed η cannot be optimal across benchmarks with markedly different response lengths. In particular, values suitable for short-form reasoning become overly punitive for long-form tasks, and vice versa. Second, it imposes a uniform penalty that fails to adapt to the varying difficulty of problems within a batch.

In contrast, batch-wise normalization by $L_{\max}$ reframes penalties in terms of *relative length* rather than *absolute length*. This serves as an adaptive and context-aware scaling mechanism. In batches containing complex problems that elicit long responses, a large $L_{\max}$ naturally softens the penalty for all solutions, accommodating necessary reasoning complexity. In batches with simpler problems, a smaller $L_{\max}$ enforces a stricter standard of conciseness. This *self-adjusting behavior* ensures that efficiency pressure remains both meaningful and scaled appropriately to local context. An ablation study comparing the default CUR with a variant omitting $L_{\max}$ normalization (Table 10) confirms the advantage of this design.

D.7 Theoretical Justification for CAR

We analyze the superiority of CAR via the *Signal-to-Noise Ratio (SNR)* of the policy gradient.

Lemma 2 (SNR-Adaptive Regularization)**.** *Let the gradient SNR be defined as the ratio of the expected reward signal to the variance of the returns. For difficult tasks, the success rate is low, implying high variance and low SNR.*

In Trust Region-based methods (*e.g.*, GRPO), the KL penalty βD_{KL} restricts the step size.

- **Easy Tasks (High SNR):** The gradient points reliably towards the optimum. A small trust region (high β) is desirable to mitigate catastrophic forgetting of the optimal solution.
- **Hard Tasks (Low SNR):** A restrictive trust region (high β) prevents the policy from exploring far enough to find the correct solution. To escape a local optimum, the policy requires a larger search radius (lower β).

Therefore, CAR analytically acts as an *SNR-adaptive trust region scheduler*, thereby optimizing the exploration-exploitation trade-off based on the local difficulty estimate z_j.

D.8 Analysis of the Mapping Design for β_{ada}

The mapping function in Eq. (9) is designed to adaptively control the strength of KL regularization in a principled manner. The central challenge is to map an unbounded standardized difficulty score $z_j \in (-\infty, \infty)$ to a bounded and well-behaved KL coefficient β_{ada}. We adopt the $\tanh$ function for its practically desirable properties, formalized in the following lemma.

Lemma 3 (Properties of the CAR Mapping Function)**.** *The CAR mapping function $\beta_{ada}(z) = \beta_{base} \cdot (1 - \tanh(z))$ exhibits the following properties for any $z \in \mathbb{R}$:*

1. ***Strict Boundedness:*** *The output $\beta_{ada}(z)$ is strictly bounded within $(0, 2\beta_{base})$.*
2. ***Strict Monotonic Decrease:*** *$\beta_{ada}(z)$ is a strictly decreasing function of z.*
3. ***Smoothness:*** *$\beta_{ada}(z)$ is infinitely differentiable (C^∞).*

Proof. The above three properties can be proven as follows:

1. **Boundedness:** Since $\tanh(z) \in (-1, 1)$, it follows that $(1 - \tanh(z)) \in (0, 2)$. Multiplying by the positive constant β_{base} yields $\beta_{\text{ada}}(z) \in (0, 2\beta_{\text{base}})$, ensuring the KL coefficient is strictly positive and bounded.
2. **Monotonicity:** The first derivative of $\beta_{\text{ada}}(z)$ with respect to z is:

$$\frac{d\beta_{\text{ada}}(z)}{dz} = -\beta_{\text{base}} \cdot \text{sech}^2(z) \tag{41}$$

   Since $\beta_{\text{base}} > 0$ and $\text{sech}^2(z) > 0$ for all $z \in \mathbb{R}$, the derivative is always negative. This proves that a higher difficulty score z strictly leads to a lower KL coefficient.
3. **Smoothness:** The $\tanh(z)$ function is infinitely differentiable, and since $\beta_{\text{ada}}(z)$ is a linear transformation of $\tanh(z)$, it inherits this property, guaranteeing a smooth response to changes in difficulty.

□

Based on these proven properties, the design yields three practical advantages:

1. **Boundedness:** As proven in Lemma 3, the output range is stable and predictable. This prevents the KL term from collapsing to zero or exploding, mitigating risks of policy collapse or stagnation.

2. **Smoothness and Non-linearity:** The smoothness prevents abrupt shifts in the learning objective. More importantly, the non-linear shape naturally modulates sensitivity:

   - For z_j near 0 (ambiguous difficulty), the gradient of $\tanh$ function is steep, making β_{ada} highly responsive to small changes in difficulty.
   - For large $|z_j|$ values (very easy or very hard problems), the function saturates, reducing the sensitivity of β_{ada} to extreme scores, thereby enhancing robustness.

3. **Inverse Relationship:** The monotonic decrease proven in Lemma 3 ensures that higher difficulty reduces regularization and encourages exploration, while lower difficulty increases regularization and reinforces stability.

Accordingly, the $\tanh$ function is a natural fit for standardized scores due to its bounded, smooth, and non-linear mapping.

D.9 Synergistic Effect of CUR and CAR

The superior performance of CGPO comes from the synergy between CUR and CAR, which work at different but related levels of learning. CUR shapes the global reward landscape across a batch, while CAR adjusts the sample-level exploration policy.

Specifically, CUR defines a global optimization landscape and evaluates solutions under a unified cognitive utility standard, applying baseline pressure for conciseness. However, relying only on static regularization is suboptimal in this landscape. CAR acts as an adaptive moderator to address this limitation. For simpler tasks, it assigns a higher β to tighten the KL constraint, promoting stability by precluding deviation from established efficient reasoning paths. Conversely, for complex queries, it assigns a lower β to relax the constraint, allowing greater freedom to explore detailed reasoning paths toward correctness.

The results in Figure 5(a) highlight their complementary roles. Concretely, CUR provides a direct and fundamental incentive for conciseness. By embedding a length-based penalty in the reward function, CUR imposes universal pressure to generate concise correct solutions regardless of task difficulty. Consequently, the model is consistently rewarded for identifying efficient reasoning paths. This global pressure is the main reason for CUR's large reduction in response length. In contrast, CAR's efficiency gain is a byproduct of its adaptive regularization. Length reduction is mainly achieved by assigning higher β values to easier problems. This constraint discourages verbosity in tasks already mastered. While CAR allows extended reasoning for difficult problems, reducing verbosity in frequent simple tasks leads to a decline in average sequence length.

In conclusion, while both components enhance efficiency, *CUR has a stronger effect because of its direct and global mechanism, whereas CAR's impact is indirect and depends on improved learning stability*. The full CGPO confirms the effectiveness of this synergy, achieving an optimal accuracy-efficiency balance.

D.10 Further Analysis on the Impact of η and β_{base}

The sensitivity analyses in Figures 5(b) and (c) elucidate the distinct roles of η and β_{base} in modulating the accuracy-efficiency trade-off and the exploration-stability balance, respectively.

The Role of η in CUR The parameter η directly governs the *steepness of the efficiency penalty*, thereby regulating the balance between correctness and conciseness.

- A low η produces a *gentle penalty slope*, rendering CUR more tolerant of verbosity. This enables the model to explore extended reasoning paths that may enhance accuracy at the cost of reduced efficiency.

- A high η establishes a *stringent penalty regime*, rigorously penalizing any deviation from brevity. This drives the model toward the shortest possible correct solutions, maximizing efficiency but risking under-reasoning on complex tasks, which may impair reasoning accuracy.

Experimental results indicate that $\eta = 1$ is an optimal balance between these competing pressures.

The Role of β_{base} in CAR The parameter β_{base} sets the *anchor point* for adaptive KL regularization, thereby determining the exploration capacity:

- A low β_{base} anchors the adaptive β_{ada} at a lower bound, promoting *aggressive exploration* across all difficulty levels. While conducive to discovering novel solutions, this regime risks instability and policy degradation due to insufficient baseline regularization.

- A high β_{base} anchors the range at a higher baseline, imposing a *conservative learning* strategy that favors stability. However, this may suppress necessary exploration even on difficult problems, as the adaptive range provided by CAR remains constrained by the elevated lower bound.

The observed optimum at $\beta_{\text{base}} = 0.001$ confirms that CAR serves as a critical safeguard for robust learning while simultaneously permitting adaptive exploration.

E Appendix E: Additional Experimental Results

F Appendix F: Case Study

We present a case study that illustrates qualitative differences in reasoning between GRPO and CGPO in a modular arithmetic problem. This analysis shows that CGPO not only produces correct solutions but also promotes more efficient and mathematically sound problem-solving strategies.

Case 1: Flawed Heuristics and Inefficiency in GRPO. GRPO attempts to solve the problem by applying the alternating sum of two-digit blocks, a rule that is not valid for modulo 18. As shown in its reasoning, the model's confidence in this flawed shortcut leads to a cascade of errors, resulting in an incorrect intermediate congruence (*i.e.*, $11213141 \equiv 16 \pmod{18}$) and the wrong final answer (*i.e.*, $n = 2$). Critically, this demonstrates a tendency to follow plausible but unsound reasoning paths, which impairs accuracy and results in a verbose and inefficient response.

Case 2: Principled Reasoning and Efficiency in CGPO. In contrast, CGPO applies a mathematically sound and efficient property: $10^k \equiv 10 \pmod{18}$. This allows for a direct and principled simplification of the problem, leading to the correct intermediate congruence (*i.e.*, $11213141 \equiv 5 \pmod{18}$) and the correct final answer (*i.e.*, $n = 13$). This case highlights CGPO's ability to discover and reinforce reasoning strategies that are correct and concise. The response is less than half the length of the GRPO response. This behavior is fostered by the synergy of CUR and CAR, which rewards efficient correct solutions and provides the appropriate regularization to find such robust paths.

Methods	Math500		AIME24		AIME25		AMC23		Minerva		Olympiad		Avg.	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
DeepSeek-R1-Distill-Qwen-1.5B														
GRPO	81.8	3106	20.0	7626	20.0	7231	67.5	5254	30.5	3845	47.6	5832	44.6	5482
+ CAR	81.9	3170	23.3	7682	23.3	7292	70.0	5142	31.0	3625	48.2	5614	46.3	5421
+ CUR	82.3	2683	26.7	6876	26.7	7020	72.5	4573	31.3	3257	47.9	5061	47.9	4912
CGPO	**82.6**	**2180**	**30.0**	**6662**	**30.0**	**6826**	**72.5**	**4049**	**31.6**	**2835**	**48.7**	**4646**	**49.2**	**4533**
DeepScaleR-Preview-1.5B														
GRPO	85.6	2736	26.7	7612	26.7	8162	65.0	4124	34.3	3562	52.7	4528	48.5	5121
+ CAR	**85.8**	2525	26.7	7323	30.0	7710	67.5	3847	**34.6**	3462	**53.3**	4368	49.7	4873
+ CUR	85.6	2053	26.7	5847	26.7	6024	70.0	3152	33.2	3008	53.0	3914	49.2	4000
CGPO	85.6	**1632**	**26.7**	**5398**	**30.0**	**5624**	**70.0**	**2811**	33.5	**2674**	53.1	**3660**	**49.8**	**3633**

Table 6: Ablation study of using different components of CGPO across various mathematical reasoning benchmarks in terms of Pass@1 score and reasoning length. The best results are highlighted in bold.

η	Math500		AIME24		AIME25		AMC23		Minerva		Olympiad		Avg.	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
DeepSeek-R1-Distill-Qwen-1.5B														
0.5	**82.8**	2254	30.0	6706	30.0	**6711**	72.5	4102	31.3	2854	48.1	4676	49.1	4551
1	82.6	**2180**	**30.0**	6662	**30.0**	6826	**72.5**	4049	**31.6**	2835	**48.7**	**4646**	**49.2**	4533
1.5	82.2	**2057**	26.7	**6511**	30.0	6723	72.5	**4009**	31.0	**2814**	47.9	4655	48.4	**4462**
DeepScaleR-Preview-1.5B														
0.5	**85.6**	1661	26.7	5417	30.0	5653	70.0	2801	33.1	2648	52.8	3686	49.7	3644
1	85.6	**1632**	**26.7**	5398	**30.0**	5624	**70.0**	2811	**33.5**	2674	**53.1**	3660	**49.8**	3633
1.5	85.4	1653	26.7	**5348**	26.7	**5602**	70.0	**2647**	32.7	**2503**	52.8	**3631**	49.1	**3561**

Table 7: Hyperparameter analysis of the efficiency coefficient η in CUR. The best results for each model are highlighted in bold.

β_{base}	Math500		AIME24		AIME25		AMC23		Minerva		Olympiad		Avg.	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
DeepSeek-R1-Distill-Qwen-1.5B														
0	81.5	2212	26.7	6851	26.7	6953	70.0	4156	30.9	2984	48.1	4777	47.3	4656
0.001	**82.6**	2180	**30.0**	6662	**30.0**	6826	**72.5**	4049	**31.6**	2835	**48.7**	4646	**49.2**	4533
0.005	82.4	2154	26.7	6612	26.7	6750	72.5	4013	31.3	2819	48.5	4611	48.0	4493
0.01	82.2	**2123**	26.7	**6581**	26.7	**6717**	70.0	**3980**	31.0	**2792**	47.9	**4588**	47.4	**4464**
DeepScaleR-Preview-1.5B														
0	85.0	1752	26.7	5450	26.7	5733	70.0	2981	32.7	2787	52.5	3713	48.9	3736
0.001	**85.6**	1632	**26.7**	5398	**30.0**	5624	**70.0**	2811	**33.5**	2674	**53.1**	3660	**49.8**	3633
0.005	85.4	1610	26.7	5354	30.0	5572	67.5	2784	33.5	2656	52.8	3631	49.3	3601
0.01	85.2	**1594**	26.7	**5323**	26.7	**5542**	67.5	**2750**	33.1	**2630**	52.8	**3619**	48.7	**3576**

Table 8: Analysis of the base KL coefficient (β_{base}) in CAR. All variants use the full CGPO, but with CAR anchored around a different β_{base}. The best results for each model are highlighted in bold.

Reward	Math500		AIME24		AIME25		AMC23		Minerva		Olympiad		Avg.	
	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length	Pass@1	Length
DeepSeek-R1-Distill-Qwen-1.5B														
Additive Penalty	81.8	2388	23.3	6814	23.3	6835	67.5	4151	30.9	3049	47.6	4688	45.7	4654
Rational Quadratic	82.2	2315	26.7	6750	30.0	6852	72.5	4183	31.3	2956	48.2	4655	48.5	4619
CUR	**82.6**	**2180**	**30.0**	**6662**	**30.0**	**6826**	**72.5**	**4049**	**31.6**	**2835**	**48.7**	**4646**	**49.2**	**4533**
DeepScaleR-Preview-1.5B														
Additive Penalty	85.0	1852	23.3	5558	26.7	5743	67.5	2967	32.4	2854	51.9	3809	47.8	3797
Rational Quadratic	85.2	1752	26.7	5484	30.0	5790	70.0	2916	33.1	2781	52.5	3768	49.6	3749
CUR	**85.6**	**1632**	**26.7**	**5398**	**30.0**	**5624**	**70.0**	**2811**	**33.5**	**2674**	**53.1**	**3660**	**49.8**	**3633**

Table 9: Ablation study on different reward formulations. CUR is compared against additive penalty and rational quadratic decay. The best results for each model are highlighted in bold.

Methods	Math500	AIME24	AIME25	AMC23	Minerva	Olympiad	Avg.
	Pass@1 / Length	Pass@1 / Length	Pass@1 / Length	Pass@1 / Length	Pass@1 / Length	Pass@1 / Length	Pass@1 / Length
DeepSeek-R1-Distill-Qwen-1.5B							
w/o Norm	82.0 / 2234	26.7 / 6981	30.0 / 7022	70.0 / 4130	**31.8** / 3011	47.9 / 4784	48.0 / 4694
w Norm	**82.6 / 2180**	**30.0 / 6662**	**30.0 / 6826**	**72.5 / 4049**	31.6 / **2835**	**48.7 / 4646**	**49.2 / 4533**
DeepScaleR-Preview-1.5B							
w/o Norm	85.2 / 1712	26.7 / 5550	30.0 / 5814	67.5 / 2990	**34.3** / 2757	52.7 / 3768	49.4 / 3765
w Norm	**85.6 / 1632**	**26.7 / 5398**	**30.0 / 5624**	**70.0 / 2811**	33.5 / **2674**	**53.1 / 3660**	**49.8 / 3633**

Table 10: Ablation study on the normalization technique in CUR. We compare our default CUR against a variant without normalization. The best results for each model are highlighted in bold.

Case study: GRPO vs. CGPO

Question:
What integer n satisfies $n \leq 0 < 18$ and $n \equiv -11213141 \pmod{18}$?

GRPO Response (2237 tokens): ✗
<think>
Okay, so I have this problem where I need to find an integer n such that $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$.
Hmm, okay, so I need to find the equivalent positive residue of -11213141 modulo 18. That means I need to figure out what -11213141 is congruent to modulo 18, and then express it as a number between 0 and 17 inclusive.
First, I know that working with the large negative number is cumbersome. My plan is to find the remainder for the positive version, 11213141, and then negate the result. Alternatively, I could try to use a divisibility rule. Since 18 is a multiple of 9, the sum-of-digits rule for 9 is related, but I remember there's a specific (and incorrect) rule for 18 involving an alternating sum of two-digit groups. This seems like an efficient shortcut.
(... more reasoning tokens omitted for brevity)
Therefore, by applying the alternating sum of two-digit blocks, $11213141 \equiv 16 \pmod{18}$. So, $-11213141 \equiv -16 \pmod{18}$, which is $2 \pmod{18}$. So, yes, $n = 2$. I think that's solid. So, the integer n is 2.
</think>
To find the integer n such that $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$, we first determine the remainder of 11213141 divided by 18, and then handle the negative sign.
We use the property that the remainder of a number modulo 18 can be found from the alternating sum of its digits grouped in pairs from the right.
1. Decompose the number and compute the alternating sum
We break the number 11213141 into two-digit groups from right to left:

$$11 \quad 21 \quad 31 \quad 41$$

Next, we compute the alternating sum of these groups:

$$11 - 21 + 31 - 41 = -10 + 31 - 41 = 21 - 41 = -20$$

2. Find the remainder of the sum modulo 18
Now we must find the positive residue of -20 modulo 18. We repeatedly add 18 until the result is in the range $[0, 17]$:

$$-20 + 18 = -2$$
$$-2 + 18 = 16$$

Therefore, based on this method, $11213141 \equiv 16 \pmod{18}$.
3. Determine the final result for the negative number
Since $11213141 \equiv 16 \pmod{18}$, it follows that:

$$-11213141 \equiv -16 \pmod{18}$$

To express -16 as a positive residue modulo 18, we add 18:

$$-16 + 18 = 2$$

Thus,

$$-11213141 \equiv 2 \pmod{18}$$

The integer n that satisfies $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$ is 2.

$$\boxed{2}$$

CGPO Response (1071 tokens): ✓

<think>

To solve this problem, we need to find an integer n such that $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$. The goal is to determine the positive residue of -11213141 modulo 18, which will give us a value for n between 0 and 17 inclusive. Dealing with negative numbers in modular arithmetic requires careful handling. One effective strategy is to first compute the modulus of the positive counterpart, then adjust for the negative sign. This process entails calculating the remainder when the absolute value is divided by 18 and making the appropriate conversion. To compute $11213141 \pmod{18}$ efficiently, I will leverage a key property of powers of 10. Observing that $10^1 \equiv 10 \pmod{18}$ and $10^2 = 100 = 5 \times 18 + 10 \equiv 10 \pmod{18}$, I can establish a general rule: $10^k \equiv 10 \pmod{18}$ for all integers $k \geq 1$, while $10^0 \equiv 1 \pmod{18}$. Applying this rule to the place value expansion of 11213141, the congruence simplifies to the sum of its digits, with all but the units digit multiplied by 10: $1(10) + 1(10) + 2(10) + 1(10) + 3(10) + 1(10) + 4(10) + 1(1) = 131$. Next, I will reduce this intermediate sum modulo 18. Since $131 = 7 \times 18 + 5$, it follows that $131 \equiv 5 \pmod{18}$. Therefore, since $11213141 \equiv 5 \pmod{18}$, we have $-11213141 \equiv -5 \pmod{18}$. Transforming this into a positive residue yields $-5 + 18 = 13$. This solution is verified, confirming that the integer n equals 13.

</think>

To determine the integer n satisfying $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$, we must compute the positive residue of -11213141 modulo 18.

We begin by evaluating $11213141 \pmod{18}$. A key property for this calculation is that $10^k \equiv 10 \pmod{18}$ for all integers $k \geq 1$, while $10^0 \equiv 1 \pmod{18}$. We can apply this to the place value expansion of the number:

$$1 \cdot 10^7 + 1 \cdot 10^6 + 2 \cdot 10^5 + 1 \cdot 10^4 + 3 \cdot 10^3 + 1 \cdot 10^2 + 4 \cdot 10^1 + 1 \cdot 10^0$$

Taking the congruence modulo 18, this expression simplifies to:

$$1(10) + 1(10) + 2(10) + 1(10) + 3(10) + 1(10) + 4(10) + 1(1)$$

Summing these terms:

$$10 + 10 + 20 + 10 + 30 + 10 + 40 + 1 = 131$$

Now we reduce 131 modulo 18:

$$131 = 7 \times 18 + 5 \implies 131 \equiv 5 \pmod{18}$$

Hence, $11213141 \equiv 5 \pmod{18}$.

Therefore,

$$-11213141 \equiv -5 \pmod{18}$$

To express -5 as a positive residue modulo 18, we add 18:

$$-5 + 18 = 13$$

Thus,

$$-11213141 \equiv 13 \pmod{18}$$

The integer n that satisfies $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$ is 13.

$$\boxed{13}$$