

From Discovery to Internalization: Length-Adaptive Policy Optimization for Efficient Reasoning

Xingyu Wu¹, Yuchen Yan¹, Shangke Lyu¹, Linjuan Wu¹, Yiwen Qiu¹,
Yongliang Shen^{1*}, Weiming Lu¹, Jian Shao^{1†}, Jun Xiao¹, Yueting Zhuang¹

¹Zhejiang University,
{wuxingyu, jshao, syl}@zju.edu.cn

Abstract

Large reasoning models have achieved remarkable performance through extended chain-of-thought sequences, yet this computational freedom leads to excessive token generation even for simple problems. We present Length-Adaptive Policy Optimization (LAPO), a novel framework that transforms reasoning length control from an external constraint into an intrinsic model capability. Unlike existing approaches that impose rigid limits or rely on post-hoc interventions, LAPO enables models to internalize an understanding of appropriate reasoning depth through a two-stage reinforcement learning process. In the first stage, models learn natural reasoning patterns by discovering the statistical distribution of successful solution lengths. In the second stage, these learned patterns are embedded as in-context, self-declarative guidance, teaching the model to proactively plan its reasoning budget. Experiments on mathematical reasoning benchmarks demonstrate that LAPO reduces token usage by up to 38.5% while improving accuracy by 2.3%. Our analysis reveals that models trained with LAPO develop emergent abilities to allocate computational resources based on problem complexity, achieving efficient reasoning without sacrificing quality.

1 Introduction

Recent advances in large reasoning models have demonstrated remarkable capabilities through extended chain-of-thought sequences (Wei et al., 2022; Jaech et al., 2024; DeepSeek-AI et al., 2025). However, this computational freedom leads to “overthinking” (Min et al., 2024): models generate excessively verbose reasoning chains even for simple problems, causing significant computational overhead and hindering practical deployment.

Existing approaches to address this challenge fall into three main categories, each with inherent limitations. Direct length reduction methods either rely on reward design (Yang et al., 2025; Huang et al., 2025) that can cause over-shortening and accuracy degradation, or impose hard length constraints (Aggarwal and Welleck, 2025; Hou et al., 2025) that lack adaptability across problem types. Dynamic early-stopping approaches (Qiao et al., 2025; Muennighoff et al., 2025) make real-time termination decisions but often truncate mid-reasoning, disrupting the thinking process. Adaptive thinking methods (Lou et al., 2025; Zhang et al., 2025; Fang et al., 2025) enable models to switch between thinking and non-thinking modes but operate at a coarse granularity.

However, these approaches typically treat reasoning length as an external constraint (e.g., via length penalties or decoding-time limits), failing to teach the model how to internally leverage the strong correlation between problem complexity and optimal solution length.

We propose a paradigm shift: instead of constraining reasoning through external mechanisms, we enable models to learn from their own successful reasoning patterns and develop an internal sense of appropriate reasoning depth. Our key insight is that the distribution of reasoning lengths in correct solutions contains valuable information about how much thinking each problem genuinely requires. By capturing these patterns during training and teaching models to anticipate the appropriate reasoning budget before they begin solving, we can transform length control from an external limitation into an intrinsic capability.

We introduce Length-Adaptive Policy Optimization (LAPO), a two-stage reinforcement learning framework designed to build this intrinsic capability. As illustrated in Figure 1, our framework consists of two sequential stages: Discovery and Internalization. 1) Discovery (LAPO-D): The model

*Corresponding author

†Corresponding author

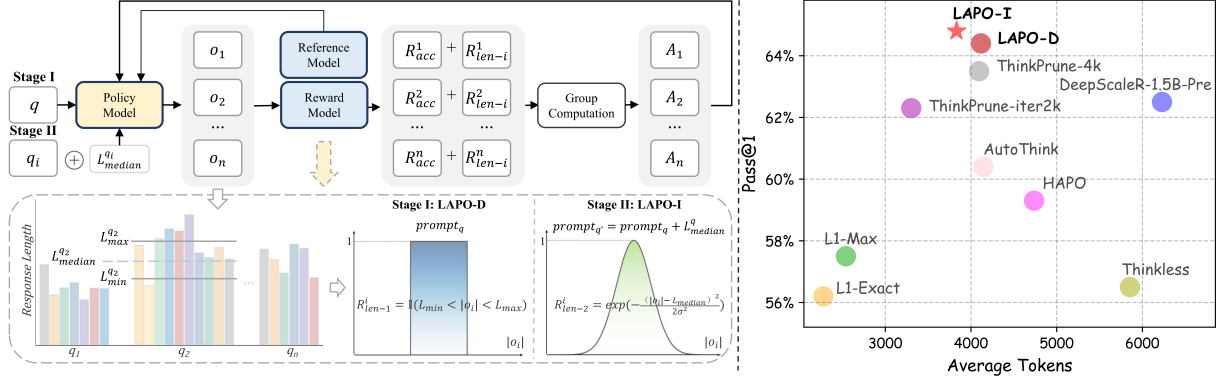


Figure 1: Overview of Length-Adaptive Policy Optimization (LAPO) and its superior performance. The LAPO framework (left) trains a model in two stages: first discovering natural reasoning lengths, then internalizing them as self-proposed budgets. This process enables our models (LAPO-I) to achieve a state-of-the-art balance between accuracy and efficiency (right), surpassing existing methods by operating in the desirable top-left region of the performance plot.

first learns the statistical distribution of effective solution lengths using a length-aware reward function. 2) Internalization (LAPO-I): This learned knowledge is then embedded as self-declarative guidance—an in-context statement like “answer with 150 tokens”—which reframes the budget as part of the model’s own reasoning plan. This approach transforms length control from a reactive constraint into a proactive, adaptive strategy. Unlike methods that follow rigid rules, LAPO’s two-stage design enables a natural progression: models first learn appropriate reasoning depth from experience, then internalize this knowledge to proactively anticipate task demands.

Extensive experiments validate the effectiveness of our approach. LAPO achieves remarkable efficiency gains, reducing token usage by up to 38.5% while simultaneously improving accuracy by 2.3% on mathematical reasoning benchmarks (see Figure 1). Our analysis reveals that this improvement stems from the model’s ability to distinguish between problems requiring elaborate derivations versus those needing only brief calculations. These results indicate that when models learn from their own successful patterns rather than arbitrary constraints, they develop more robust and efficient reasoning strategies.

Our main contributions are:

- We propose LAPO, a novel two-stage RL framework that transforms length control from an external constraint into an intrinsic, adaptive capability by learning from the model’s own successful reasoning.
- We introduce a training method that uses dis-

covered statistical patterns as in-context, self-declarative guidance, enabling models to internalize efficient reasoning behaviors without sacrificing inference-time flexibility.

- We demonstrate that LAPO achieves substantial efficiency gains (up to 38.5% token reduction) while simultaneously improving accuracy, revealing a robust capability for adaptive resource allocation.

2 Related Works

2.1 Test-Time Scaling

Scaling test-time computation has been shown to consistently improve performance on reasoning, math, and code tasks (Wu et al., 2025; Wang et al., 2023; Wei et al., 2022; DeepSeek-AI et al., 2025), with predictable gains obtained by generating more or longer reasoning chains (Wu et al., 2025; Snell et al., 2024; Jaech et al., 2024). Representative strategies include parallel sampling (Wang et al., 2023), tree-based search (Yao et al., 2023; Wu et al., 2025), and iterative refinement (Snell et al., 2024; Welleck et al., 2024). Recent reasoning models such as OpenAI’s O1 and DeepSeek-R1 (Jaech et al., 2024; DeepSeek-AI et al., 2025) instead rely on RL with verifiable rewards (RLVR) to elicit extended chains that exhibit exploration and self-verification (Gandhi et al., 2025). However, such traces are often several times longer than short-CoT outputs (Sui et al., 2025; Chen et al., 2024), causing “overthinking” and substantial inference overhead (Kumar et al., 2025).

2.2 Efficient Long Chain-of-Thought LLM

To mitigate overthinking, prompt-based methods provide only coarse control (Xu et al., 2025a; ?) (e.g., “be concise” instructions such as Chain of Draft (Xu et al., 2025a) or Constrained CoT (?), effective mainly on large models); training-based methods using SFT (Wang et al., 2024; Kang et al., 2025; Ma et al., 2025; Xia et al., 2025) or RL with length penalties (Muennighoff et al., 2025; Chang et al., 2025; Luo et al., 2025a; Xu et al., 2025b) struggle to adapt to problem complexity; and router-based methods introduce extra overhead by dispatching queries across models (Chuang et al., 2025; Ong et al., 2024). More recent budget-aware approaches such as L1 (Aggarwal and Welleck, 2025) and Elastic Reasoning (Xu et al., 2025b) can follow a given token budget but cannot autonomously decide an appropriate one, while implicit-regularization methods such as HAPO (Huang et al., 2025) risk over-compression. LAPO bridges this gap through a two-stage “Discover–Internalize” process that performs both autonomous budget estimation and problem-adaptive length control (quantitative comparison in Appendix J).

3 Method

We present Length-Adaptive Policy Optimization (LAPO), a framework designed to transform efficient reasoning from an externally imposed constraint into a truly intrinsic model capability. Our approach is built on the key insight that the distribution of lengths across successful solutions naturally reflects a problem’s own intrinsic complexity. Instead of enforcing arbitrary limits, we leverage these very patterns as signals to teach the model an internal understanding of appropriate reasoning depth.

To achieve this, LAPO employs a carefully designed two-stage training process, as illustrated in Figure 2. The first stage, uses a length-aware reward to encourage the model to explore and identify natural, efficient reasoning patterns. The subsequent stage then transforms these discovered patterns into an adaptive capability by embedding them as in-context, self-declarative guidance within the model’s own reasoning process.

3.1 Discovery Stage

The Discovery stage aims to uncover inherent relationships between problems and their natural rea-

soning lengths through GRPO training with a carefully designed reward mechanism that encourages efficient exploration while maintaining correctness.

Extracting Statistics from GRPO Rollouts.

During GRPO training, we generate N rollout responses for each problem q in the training batch. From these rollouts, we collect the lengths of responses that produce correct answers:

$$\mathcal{L}_q = \{|r_i| : \mathbb{I}(y_i = y_{\text{gold}}) = 1, i \in [1, N]\} \quad (1)$$

where y_i is the predicted answer from the i -th rollout response r_i . This collection, extracted directly from the GRPO sampling process, represents natural variation in successful reasoning lengths.

We derive two key statistics from these rollouts. First, we establish a reasonable length range using percentiles to filter outliers while preserving central tendencies:

$$[L_{\min}, L_{\max}] = [\text{Percent}_{30}(\mathcal{L}_q), \text{Percent}_{70}(\mathcal{L}_q)] \quad (2)$$

This choice is designed to robustly identify the core distribution of effective reasoning lengths by filtering out statistical outliers. The lower bound (30th percentile) helps discard overly concise solutions that might be correct by chance, while the upper bound (70th percentile) discourages excessively verbose and inefficient reasoning paths. An ablation on the choice of this percentile range is reported in Appendix C.

Second, we create a problem-to-length mapping that will guide the Internalization stage:

$$\mathcal{M} : q \mapsto L_{\text{median}}(q) = \text{Median}(\mathcal{L}_q) \quad (3)$$

We choose the median for its robustness to outliers; comparisons with alternative statistics (mean and minimum) are provided in Appendix C. For problems without correct solutions in the current rollouts, we temporarily set $\mathcal{M}(q) = 4096$ (maximum sequence length) to encourage comprehensive exploration in subsequent episodes. This high initial budget is a fallback measure that is promptly updated to the data-driven median once the model starts solving the problem (Eq. 8), preventing a lasting bias toward long-form answers.

Length-Aware Reward Design. We employ a composite reward function balancing accuracy and efficiency:

$$R_D(r_i, q) = \mathbb{I}(y_i = y_{\text{gold}}) + \alpha \cdot R_1(r_i, q) \quad (4)$$

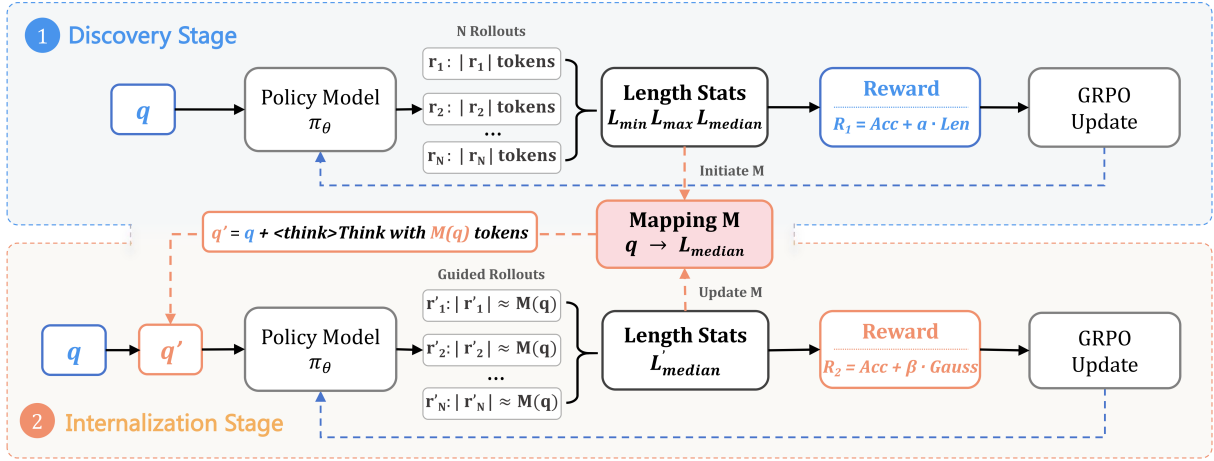


Figure 2: The LAPO framework consists of two stages: (1) Discovery stage learns natural reasoning patterns by rewarding efficient correct solutions and collecting length statistics; (2) Internalization stage embeds these statistics as self-proposed plans within the model’s reasoning context, teaching models to internalize efficient reasoning.

The length component operates on a crucial principle—only correct responses receive length-based rewards. Let $C_i = \mathbb{I}(y_i = y_{\text{gold}})$ indicate whether the response is correct, and define the distance to the target length range as $d_i = \min(|r_i| - L_{\min}, |r_i| - L_{\max})$. We introduce a linear decay function $f(d) = \max(0, 1 - d/100)$ to penalize deviations from the efficient length range. The length reward is then defined as:

$$R_1(r_i, q) = \begin{cases} 1.0 & \text{if } C_i=1 \wedge |r_i| \in [L_{\min}, L_{\max}], \\ f(d_i) & \text{if } C_i=1 \wedge |r_i| \notin [L_{\min}, L_{\max}], \\ 0 & \text{if } C_i=0; \end{cases} \quad (5)$$

This design creates gradients guiding models toward efficient lengths while allowing flexibility for complex problems. Throughout the Discovery stage, we continuously update $\mathcal{M}$ after each GRPO training step to reflect evolving model capabilities.

3.2 Internalization Stage

The Internalization stage transforms discovered patterns into internalized capabilities through continued GRPO training with modified prompts and rewards.

Length-Conditioned Rollout. We augment each problem prompt with explicit length guidance:

$\text{prompt}'_q = \text{prompt}_q + \text{"<think> I will answer the question with } n \text{ tokens."}$

where $n = \mathcal{M}(q)$ from the Discovery stage. This embeds length awareness within the reasoning context, helping models perceive computational

budgets as intrinsic to thinking rather than external constraints.

Length-Adherence Reward. To encourage the model to follow its self-declared reasoning budget, the Internalization stage employs a precision-focused reward function. This function is designed to reward the alignment between the model’s output length and its self-declared budget n . The total reward is defined as:

$$R_I(r_i, q') = \mathbb{I}(y_i = y_{\text{gold}}) + \beta \cdot R_2(r_i, q') \quad (6)$$

where the adherence component, R_2 , is only granted for correct solutions:

$$R_2(r_i, n) = \begin{cases} \exp\left(-\frac{(|r_i| - n)^2}{2\sigma^2}\right), & C_i = 1, \\ 0, & C_i = 0; \end{cases} \quad (7)$$

This Gaussian-inspired reward positively reinforces solutions that are both correct and consistent with the intended reasoning depth. The standard deviation σ serves as a tolerance parameter, where a smaller σ enforces stricter adherence and a larger σ allows more flexibility. We set $\sigma=120$, a value chosen to create a “soft” target that penalizes significant deviations from the self-proposed plan while tolerating minor ones. By rewarding adherence to the self-proposed plan, this mechanism guides the model to internalize the relationship between problem complexity and an appropriate computational budget, rather than merely tracking an external signal.

Internalization via In-Context Guidance. A cornerstone of our framework is how it fosters gen-

Algorithm 1 Length-Adaptive Policy Optimization (LAPO)

```

1: Input: Base model  $\pi_\theta$ , training data  $\mathcal{D}$ , hyperparameters  $\alpha, \beta, \sigma, E_1, E_2$ 
2: Output: Length-adaptive model  $\pi_\theta^*$ 
3:
4: // Discovery Stage
5: for episode  $e = 1$  to  $E_1$  do
6:   Sample batch  $\mathcal{B} \subset \mathcal{D}$ 
7:   for each problem  $q \in \mathcal{B}$  do
8:     Generate  $N$  rollouts:  $\{r_1, \dots, r_N\} \sim \pi_\theta(q)$ 
9:     Collect correct lengths:  $\mathcal{L}_q = \{|r_i| : y_i = y_{\text{gold}}\}$ 
10:    Compute range:  $[L_{\min}, L_{\max}] = [\text{P}_{30}(\mathcal{L}_q), \text{P}_{70}(\mathcal{L}_q)]$ 
11:    Update mapping:  $\mathcal{M}(q) = \text{Median}(\mathcal{L}_q)$ 
12:    Compute rewards:  $R_D(r_i, q) = \mathbb{I}(y_i = y_{\text{gold}}) + \alpha \cdot R_1(r_i, q)$ 
13:   end for
14:   Update  $\pi_\theta$  using GRPO with rewards  $R_D$ 
15: end for
16:
17: // Internalization Stage
18: for episode  $e = 1$  to  $E_2$  do
19:   Sample batch  $\mathcal{B} \subset \mathcal{D}$ 
20:   for each problem  $q \in \mathcal{B}$  do
21:     Augment prompt:  $q' \leftarrow q + \text{"<think> I will answer the question with } \mathcal{M}(q) \text{ tokens."}$ 
22:     Generate  $N$  rollouts:  $\{r_1, \dots, r_N\} \sim \pi_\theta(q')$ 
23:     Compute rewards:  $R_I(r_i, q') = \mathbb{I}(y_i = y_{\text{gold}}) + \beta \cdot R_2(r_i, q')$ 
24:     Update mapping  $\mathcal{M}(q)$  using dual-strategy (Eq. 8)
25:   end for
26:   Update  $\pi_\theta$  using GRPO with rewards  $R_I$ 
27: end for
28: return  $\pi_\theta^*$ 

```

uine internalization, enabling inference-time flexibility without explicit length targets. The key lies in the design of the augmented prompt. Placing the self-declarative guidance immediately after the `<think>` token transforms an external constraint into an intrinsic part of the model’s cognitive plan.

During the Internalization stage, we refine $\mathcal{M}$ based on new GRPO rollouts with a dual-strategy update:

$$\mathcal{M}(q) \leftarrow \begin{cases} \text{Median}(\mathcal{L}_q^{(t)}), & \text{unsolved,} \\ \min(\mathcal{M}(q), \text{Median}(\mathcal{L}_q^{(t)})), & \text{solved;} \end{cases} \quad (8)$$

This ensures newly solved problems establish reasonable benchmarks while previously solved problems gravitate toward more efficient solutions.

3.3 Training Pipeline

We present the complete LAPO training procedure in Algorithm 1. LAPO employs GRPO across both stages with the following pipeline: The process begins with the LAPO-D stage (Lines 4-15), which runs for E_D episodes to learn the distribution of effective solution lengths from successful trajectories.

The median length of these solutions, L_{median} , is then used to construct the self-declarative guidance for the LAPO-I stage (Lines 17-27). This second stage runs for E_I episodes, training the model to both propose and adhere to its internally generated reasoning plan.

This progressive design mirrors cognitive development: first gaining tacit experience about appropriate reasoning depth through practice, then learning to anticipate these requirements proactively. The embedding of guidance as a self-declared plan is the key mechanism that bridges this gap from experience to proactive anticipation, creating models that can intrinsically adapt computational effort to problem demands.

4 Experiment Setup

Training Details. We train LAPO on a mixed dataset of 10,000 math problems with balanced difficulty, drawn from DeepScaleR-Preview-Dataset (Luo et al., 2025b) (6K) and the intermediate levels of MATH (Hendrycks et al., 2021) (4K)¹, applied to two base models: DeepSeek-R1-1.5B (DeepSeek-AI et al., 2025) and DeepScaleR-1.5B-Preview (Luo et al., 2025b). Both stages use GRPO and are trained for 3 episodes each: the Discovery stage (LAPO-D) uses reward R_D (Eq. 4) with $\alpha = 0.7$, and the resulting checkpoint initializes the Internalization stage (LAPO-I) trained with reward R_I (Eq. 6) and $\beta = 0.7$. These values strongly incentivize efficiency while keeping correctness as the dominant signal. Due to compute constraints, training uses a 4K context length; for fairness, baselines such as ThinkPrune and L1 are trained or evaluated under the same 4K limit. Full configurations and an ablation on α, β are provided in Appendix B and Appendix C.

Evaluation Details. At inference, we extend the generation window to 32,768 tokens for all models to assess unconstrained reasoning, isolating efficiency gains from the LAPO framework rather than from context truncation. We evaluate on four benchmarks of varying difficulty: MATH-500 (Hendrycks et al., 2021), AIME2024, AMC23, and Olympiad-Bench (He et al., 2024), reporting Pass@1 accuracy and average token count. Following standard practice (DeepSeek-AI et al., 2025), we sample N responses per problem (4 for MATH-500/OlympiadBench, 32 for AIME/AMC) at tem-

¹All datasets are released under the MIT License; our usage complies with the license terms.

perature 0.6 and top- p 0.95. The full Pass@1 / #Tok computation and inference prompt are detailed in Appendix M.

Baselines. We compare LAPO against three groups: (1) *Base Models*: DeepSeek-R1-1.5B (DeepSeek-AI et al., 2025) and DeepScaleR-1.5B-Preview (Luo et al., 2025b); (2) *Ablations*: Acc-Only (GRPO with accuracy reward only) and LAPO (I-Only), which skips discovery and applies internalization directly with the base model’s median length as target; (3) *Efficient Reasoning Baselines*, covering implicit regularization (HAPO (Huang et al., 2025)), budget-driven control (L1 (Aggarwal and Welleck, 2025)), ThinkPrune (Hou et al., 2025)), and adaptive activation (AutoThink (Tu et al., 2025), AdaptThink (Zhang et al., 2025), Thinkless (Fang et al., 2025)). A further comparison with the LASER family is reported in Appendix I.

5 Results and Analyses

We present comprehensive experimental results to validate LAPO’s effectiveness and understand its underlying mechanisms. We first benchmark LAPO against state-of-the-art baselines (Section 5.1). We then dissect our framework through an ablation study on its key design choices (Section 5.2), directly validate the internalization claim via an Oracle-Target vs. No-Target comparison (Section 5.3), and present a targeted robustness experiment against conflicting length instructions (Section 5.4). Finally, we provide a qualitative analysis of the learned reasoning patterns (Section 5.5).

5.1 Main Results

As shown in Table 1, LAPO achieves a superior balance of reasoning accuracy and computational efficiency, consistently outperforming its base models and establishing a new state-of-the-art frontier among methods that do not rely on external length controls.

LAPO simultaneously enhances reasoning performance and reduces test-time computes. Compared to its base models, LAPO delivers substantial gains. On DeepScaleR-1.5B-Preview, LAPO-I reduces token usage by a remarkable 38.5% while simultaneously boosting the average accuracy by 2.3 percentage points. A similar trend holds for DeepSeek-R1-1.5B, where it cuts tokens by 40.9% and improves average accuracy by 1.2

points. These results validate that LAPO learns to produce more concise yet effective reasoning.

LAPO surpasses existing efficient reasoning optimization approaches. When compared with leading efficiency methods, LAPO consistently demonstrates a superior accuracy-efficiency trade-off. On DeepScaleR-1.5B base model, LAPO-I achieves the highest average accuracy among all tested methods. This advantage holds across different baseline paradigms. It surpasses budget-driven methods like ThinkPrune-4k and L1-Max under a fair 4k training context. Compared to implicit regularization methods like HAPO, LAPO shows a clear advantage in preserving accuracy. Furthermore, while adaptive activation methods like AutoThink can be highly token-efficient, they do not reach the same level of reasoning quality. This highlights that LAPO’s “Discover-Internalize” strategy yields a more robust policy than methods relying on external budgets or simple activation-gating. A complementary matched-budget Pareto analysis, which evaluates all methods under identical maximum-token budgets, is reported in Appendix E and further confirms the superiority of LAPO’s adaptive allocation strategy.

Ablations confirm the necessity of the two-stage design. Our results reveal the critical synergy between the Discovery and Internalization stages. Directly applying the internalization stage, as shown by LAPO (I-Only), proves ineffective. On DeepScaleR-1.5B, this approach yields an average accuracy of 63.2%, underperforming the simpler Acc-Only baseline. This demonstrates that forcing a model to internalize length targets derived from its own unoptimized, often inefficient reasoning paths is counterproductive.

In stark contrast, our full process succeeds. The Discovery stage (LAPO-D) first establishes a strong foundation by identifying efficient reasoning patterns, boosting accuracy to 64.4% while significantly cutting tokens. The Internalization stage (LAPO-I) then successfully builds on this refined policy, achieving a further 7% token reduction and reaching the peak accuracy of 64.8%. This validates that our “Discover-Internalize” process is not merely additive but synergistic, where a stable discovery phase is essential for effective internalization. The training dynamics of length and accuracy across both stages are further visualized in Appendix D.

Method	MATH500		AIME2024		AMC23		OlympiadBench		Average	
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok
<i>Base model: DeepSeek-R1-1.5B</i>										
HAPO	82.2	2288	<u>31.3</u>	8649	67.3	4735	50.1	5024	57.7	5174
AutoThink	83.5	2017	29.7	7084	70.2	3499	51.2	4606	58.6	3825
AdaptThink	81.6	1580	23.9	6432	63.2	2860	48.5	4616	54.3	3871
Base	83.1	4031	30.3	12150	68.3	7222	50.0	8942	57.9	8086
+ Acc-Only	83.3	3061	31.6	10628	70.5	5307	50.6	6402	59.0	6349
+ LAPO (I-Only)	83.5	2797	28.2	9281	71.0	4567	50.5	5980	58.3	5656
+ LAPO-D	84.7	2566	28.5	8415	72.2	4132	<u>51.3</u>	5595	59.2	5177
+ LAPO-I	<u>84.3</u>	2354	29.3	8318	<u>71.2</u>	3568	51.7	4863	<u>59.1</u>	4775
<i>Base model: DeepScaleR-1.5B-Preview</i>										
L1-Exact	80.6	1953	24.4	2625	70.9	2177	48.8	2357	56.2	2278
L1-Max	81.9	1673	24.9	3638	72.7	2705	50.5	2151	57.5	2541
ThinkPrune-iter2k	85.5	1707	34.9	5095	74.3	2913	54.7	3498	62.3	3303
ThinkPrune-4k	86.6	2042	35.5	6488	76.3	3839	55.7	4010	63.5	4094
HAPO	84.4	2370	31.4	7702	70.3	4301	51.4	4571	59.3	4736
AutoThink	84.9	1635	36.2	7201	67.8	3658	52.5	4085	60.4	4144
Thinkless	81.3	2944	28.9	9143	65.7	5276	50.2	6057	56.5	5855
Base	85.8	3280	35.5	9246	74.2	6416	54.6	5974	62.5	6229
+ Acc-Only	85.6	2510	36.9	7319	<u>77.6</u>	4244	55.6	4712	63.9	4696
+ LAPO (I-Only)	85.9	2497	36.2	7130	75.0	3910	55.5	4620	63.2	4539
+ LAPO-D	<u>86.4</u>	2365	<u>37.6</u>	5945	<u>77.6</u>	3655	<u>56.1</u>	4499	<u>64.4</u>	4116
+ LAPO-I	86.3	2168	38.1	5371	78.3	3765	56.3	4024	64.8	3832

Table 1: Main results on MATH500, AIME2024, AMC23, and OlympiadBench. We report Pass@1 accuracy (%) and the average number of generated tokens (#Tok). For each metric, **bold** indicates the best and underline indicates the second-best Pass@1 score within each base model group.

Method	MATH-500		AIME2024		AMC23		OlympiadBench		Average	
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok
<i>Base model: DeepScaleR-1.5B-Preview</i>										
Base	85.8	3280	35.5	9246	74.2	6416	54.6	5974	62.5	6229
LAPO-D	86.4	2365	<u>37.6</u>	5945	<u>77.6</u>	3655	56.1	4499	<u>64.4</u>	4116
w/ Exact	86.3	2168	38.1	5371	78.3	3765	56.3	4024	64.8	3832
w/ Range	<u>86.6</u>	2153	36.5	6095	76.9	3600	<u>56.2</u>	4011	64.1	3964
w/ Outside	86.2	2251	36.4	5882	76.3	3850	55.4	4105	63.9	4022
w/ Implicit	86.9	2181	36.2	5963	76.1	4002	55.1	4206	63.6	4088

Table 2: Ablation study on the form and position of in-context length guidance for the LAPO-I stage. We compare our default method (Exact) against variants with different guidance precision (Range), placement (Outside), and presence (Implicit).

5.2 Ablation Study on In-Context Guidance

To validate that our method’s success stems from internalizing a self-proposed plan, we ablate the two key factors of our in-context guidance: its form (how precise the guidance is) and its position (whether it’s part of the model’s internal thought process). We compare our default approach (w/ Exact) against three variants: w/ Range (less precise guidance), w/ Outside (placing the guidance before <think>), and w/ Implicit (no guidance, relying only on the reward). As shown in Table 2, the results demonstrate that both form and position are

critical for effective internalization.

Our default method outperforms the less precise Range variant, indicating that specific targets discovered in Discovery stage provide a stronger learning signal. More critically, the guidance’s position determines whether the model internalizes a plan or merely follows instructions. Moving the guidance outside the <think> block transforms it into an external command and causes accuracy to drop significantly to 63.9%. This illustrates that the model performs best when the budget is framed as part of its own cognitive plan. Finally, removing the guidance entirely results in the worst perfor-

Method	Length Constraint	MATH-500	
		Pass@1 (%)	#Tok
<i>LAPO-I</i>			
Base	N/A	86.3	2168
+Short	500	<u>86.0</u>	2279
+Long	3500	<u>85.9</u>	2300
<i>LAPO-I w/Outside</i>			
Base	N/A	86.2	2251
+Short	500	85.1	1247
+Long	3500	<u>86.1</u>	2821

Table 3: Robustness of LAPO-I to conflicting length instructions on MATH-500.

Setting	Pass@1	#Tok	ρ vs. Oracle
Base	82.4	3842	0.36
LAPO-D	84.1	2356	0.71
LAPO-I (Oracle-Target)	85.3	2187	—
LAPO-I (No-Target)	84.9	2241	0.88

Table 4: Oracle-Target vs. No-Target comparison on MATH-500. ρ is the Pearson correlation between free-running length and the oracle target length over 500 test problems.

mance and token count reverting to the LAPO-D baseline. This indicates that our explicit, properly-positioned, self-declarative guidance is the critical mechanism for internalization.

5.3 Validating Internalization: Oracle-Target vs. No-Target

A central claim of LAPO is that Stage 2 causes the model to *internalize* a per-problem length budget, so that at deployment—where no explicit target is given—its behavior still closely tracks what an oracle-guided target would produce. To directly test this, we compare LAPO-I on MATH-500 under two inference-time conditions: **Oracle-Target**, where the exact Stage-2 target length is provided, and **No-Target**, the standard deployment setting used throughout this paper.

As shown in Table 4, the gap between Oracle-Target and No-Target is remarkably small (54 tokens, 0.4 accuracy points), and the free-running length of LAPO-I correlates strongly with the oracle target ($\rho=0.88$), even on out-of-distribution AIME problems ($\rho=0.91$), indicating genuine internalization rather than memorization. Full qualitative comparisons, difficulty-level breakdowns, and a dedicated budget-estimation probe (confirming LAPO-I anticipates problem difficulty without being trained to predict length explicitly) are pro-

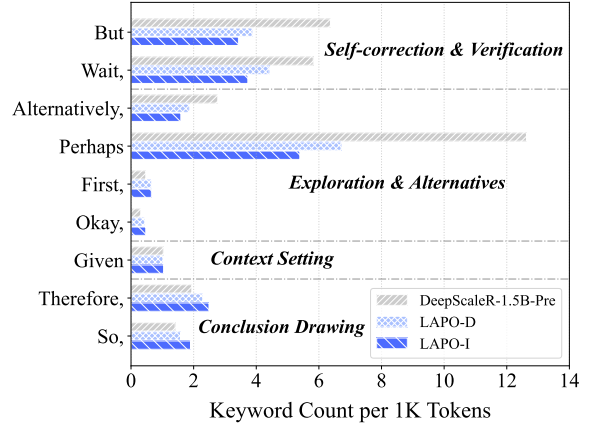


Figure 3: Keyword usage of reasoning behaviors across different stages.

vided in Appendix K and Appendix L.

5.4 Robustness to Conflicting Instructions

To further validate that LAPO fosters genuine internalization rather than superficial instruction-following, we stress-tested LAPO-I against the w/Outside variant using adversarial Short (500 tokens) and Long (3500 tokens) length prompts. Table 3 reveals a stark divergence: LAPO-I’s output length stays stable around its 2200-token baseline, ignoring the conflicting instructions, whereas w/Outside is clearly swayed (1247 tokens under Short, 2821 under Long). This indicates that framing the budget as part of the model’s internal plan (inside <think>) builds robust, internalized behavior, while framing it externally only teaches superficial instruction-following.

5.5 Qualitative Refinement of Reasoning

We next investigate how LAPO qualitatively refines reasoning. We analyzed the frequency of keywords indicative of different cognitive behaviors—*Self-Correction and Verification*, *Exploration and Alternatives*, *Context Setting*, and *Conclusion Drawing*—in the generated responses (Figure 3), revealing a significant shift in the model’s reasoning style. The most notable change is a dramatic reduction in keywords associated with “Self-Correction” and “Exploration”. LAPO training significantly curtails this verbose, deliberative internal monologue, effectively discouraging redundant verification and inefficient exploration of the solution space. Crucially, keywords for “Context Setting” and “Conclusion Drawing” remain stable. This shows LAPO selectively prunes inefficient, hesitant thought patterns while preserving the essential scaffolding of

a coherent logical argument, a behavior further refined in the internalization stage. Complementary LLM-judge-based evaluation along five reasoning-quality dimensions (Appendix F) shows the same trend: LAPO-I produces more fluent, clearer, and more concise reasoning trajectories than both the base model and the Acc-Only baseline. Beyond reasoning style, LAPO also learns to scale reasoning depth with problem difficulty, as analyzed in Appendix D.

6 Conclusion

In this work, we introduce Length-Adaptive Policy Optimization (LAPO), a two-stage reinforcement learning framework that enables language models to adjust reasoning length based on problem complexity. Unlike existing approaches that impose uniform constraints, LAPO recognizes that efficient reasoning requires understanding problem-specific computational needs rather than following rigid rules. Our two-stage design enables a natural progression: models first learn what constitutes appropriate reasoning depth through experience, then develop the ability to anticipate these requirements proactively. Extensive experiments validate LAPO’s effectiveness. Our analysis, particularly the resilience to conflicting instructions, indicates this efficiency is genuinely internalized, not merely instructed. When models learn from their own successful patterns rather than arbitrary constraints, they develop more robust and efficient reasoning strategies.

Limitations

While our work establishes a new framework for efficient reasoning, several limitations remain and point to promising directions for future work.

First, reinforcement learning for reasoning models requires substantial computational resources, and our main empirical study is therefore conducted at the 1.5B parameter scale. A more systematic investigation of how LAPO scales to larger model families (e.g., tens of billions of parameters or beyond) and how the discovered length distributions interact with stronger reasoning priors remains an important direction for future work.

Second, our main experiments focus on mathematical reasoning, where problem difficulty and solution correctness are relatively well-defined. Extending LAPO to domains with more open-ended or subjective notions of “appropriate rea-

soning depth”—such as long-form question answering, agentic tool use, or creative and dialogue-style generation—would likely require adapting the length-discovery stage to operate on preference- or rubric-based signals rather than binary correctness. We view this as a natural next step.

Third, our framework treats the reasoning budget as a single scalar quantity over the full chain of thought. Finer-grained budget allocation, for instance across sub-steps, sub-goals, or interleaved tool calls in agent settings, is not explored here and may further improve adaptivity.

Finally, although our analysis suggests that the learned behavior is robust to adversarial external length instructions, a deeper understanding of *how* the model internally represents and anticipates reasoning budgets—and whether this internal mechanism can be exposed, audited, or steered at inference time—remains an open question that we leave for future investigation.

Ethical Considerations

All datasets used in this work (e.g., MATH, AIME, AMC, OlympiadBench, GPQA, MMLU, HumanEval, MBPP) are publicly available and widely used in the research community, and contain no personally identifying or offensive content to our knowledge. Our experiments build on publicly released open-weight reasoning models at the 1.5B and 7B scales; we do not introduce new capabilities, and any biases or unsafe behaviors of the base models may persist in our fine-tuned variants. Crucially, our length-aware reward is only granted when the final answer is correct, which discourages the model from producing short but incorrect outputs.

A central motivation of this work is to reduce the inference cost of long chain-of-thought reasoning. By cutting token generation by up to $\sim 40\%$ while maintaining accuracy, our method directly reduces energy consumption and latency at deployment time. The training cost is moderate (4 A800 GPUs for all 1.5B experiments). As the method only changes the accuracy–cost trade-off rather than expanding model capabilities, we do not foresee risks specific to our approach beyond those already associated with the underlying base models. Large language models were used solely for minor editorial polishing during writing; all technical contributions are the authors’ own.

Acknowledgement

This work was supported by National Natural Science Foundation of China (No. 62436007), National Natural Science Foundation of China (No. 62506332) and the Key Research and Development Program of Ningbo (No. 2026Z003).

References

- Pranjal Aggarwal and Sean Welleck. 2025. [L1: controlling how long A reasoning model thinks with reinforcement learning](#). *CoRR*, abs/2503.04697.
- Edward Y. Chang, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. [Demystifying long chain-of-thought reasoning in llms](#). *CoRR*, abs/2502.03373.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2024. [Do NOT think that much for 2+3=? on the overthinking of o1-like llms](#). *CoRR*, abs/2412.21187.
- Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. 2025. [Learning to route llms with confidence tokens](#). *Preprint*, arXiv:2410.13284.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *CoRR*, abs/2501.12948.
- Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. [Thinkless: LLM learns when to think](#). *CoRR*, abs/2505.13379.
- Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025. [Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars](#). *CoRR*, abs/2503.01307.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. [Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 3828–3850. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- 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](#). *CoRR*, abs/2504.01296.
- Chengyu Huang, Zhengxin Zhang, and Claire Cardie. 2025. [HAPO: training language models to reason concisely via history-aware policy optimization](#). *CoRR*, abs/2505.11225.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helvar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, and 80 others. 2024. [Openai o1 system card](#). *CoRR*, abs/2412.16720.
- Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. [C3ot: Generating shorter chain-of-thought without compromising effectiveness](#). In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 24312–24320. AAAI Press.
- Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. 2025. [Overthink: Slowdown attacks on reasoning llms](#). *CoRR*, abs/2502.02542.
- 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](#). *CoRR*, abs/2505.11896.
- Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025a. [O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning](#). *CoRR*, abs/2501.12570.
- Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025b. [Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl](#). <https://pretty-radio-b75.notion.site/DeepScaler-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2>. Notion Blog.
- Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. [Cot-valve: Length-compressible chain-of-thought tuning](#). In *Proceedings of the 63rd Annual Meeting of the Association*

- for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pages 6025–6035. Association for Computational Linguistics.
- Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. 2024. [Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems](#). *CoRR*, abs/2412.09413.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. 2025. [s1: Simple test-time scaling](#). *CoRR*, abs/2501.19393.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2024. [Routellm: Learning to route llms with preference data](#). *CoRR*, abs/2406.18665.
- Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. 2025. [Concise: Confidence-guided compression in step-by-step efficient reasoning](#). *CoRR*, abs/2505.04881.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. [Scaling LLM test-time compute optimally can be more effective than scaling model parameters](#). *CoRR*, abs/2408.03314.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Ben Hu. 2025. [Stop overthinking: A survey on efficient reasoning for large language models](#). *CoRR*, abs/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 rl-style models via multi-stage rl. *arXiv preprint arXiv:2505.10832*.
- Junlin Wang, Siddhartha Jain, Dejjiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. 2024. [Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 19916–19939. Association for Computational Linguistics.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilya Kulikov, and Zaïd Harchaoui. 2024. [From decoding to meta-generation: Inference-time algorithms for large language models](#). *Trans. Mach. Learn. Res.*, 2024.
- Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. [Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models](#). *Preprint*, arXiv:2408.00724.
- Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. 2025. [Tokenskip: Controllable chain-of-thought compression in llms](#). *CoRR*, abs/2502.12067.
- Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025a. [Chain of draft: Thinking faster by writing less](#). *CoRR*, abs/2502.18600.
- Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Junnan Li, and Caiming Xiong. 2025b. [Scalable chain of thoughts via elastic reasoning](#). *CoRR*, abs/2505.05315.
- Junjie Yang, Ke Lin, and Xing Yu. 2025. [Think when you need: Self-adaptive chain-of-thought learning](#). *CoRR*, abs/2504.03234.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. [Tree of thoughts: Deliberate problem solving with large language models](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025. [Adaptthink: Reasoning models can learn when to think](#). *CoRR*, abs/2505.13417.

A Use and Documentation of Scientific Artifacts

This section documents the scientific artifacts (datasets and pre-trained models) used in this work, addressing licensing, intended use, and documentation.

Licenses and terms of use. All datasets and pre-trained models used in our experiments are publicly released for research purposes under permissive licenses. Specifically, MATH, MATH-500, AIME2024, AMC23, OlympiadBench, GPQA, MMLU-Redux, HumanEval, MBPP, and the DeepScaleR-Preview-Dataset are released under the MIT License or other standard research-oriented licenses (e.g., Apache-2.0, CC-BY). The base models we build on (DeepSeek-R1-Distill-Qwen-1.5B, DeepScaleR-1.5B-Preview, and DeepSeek-R1-Distill-Qwen-7B) are released as open-weight models for research and downstream use. We use all of these artifacts in accordance with their respective license terms. Any models or scripts we release based on this work will be distributed under a license compatible with the licenses of the underlying artifacts, and will be intended for non-commercial research use only.

Consistency with intended use. All artifacts used in this work are employed strictly within their intended research scope: the datasets are used for training and benchmarking reasoning models, and the pre-trained models are used as base policies for reinforcement-learning-based fine-tuning. We do not re-purpose any dataset for tasks beyond what its original release describes, and we do not use research-only artifacts in any commercial or deployment-oriented setting. Models derived from research data in this work are intended only as research artifacts, consistent with the access conditions of the underlying data.

Documentation of artifacts. The artifacts used in this work are documented as follows. **Mathematical reasoning benchmarks** (MATH-500, AIME2024, AMC23, OlympiadBench) cover competition-level mathematics in English, ranging from high-school to olympiad difficulty, and include topics such as algebra, geometry, number theory, combinatorics, and calculus. **General-knowledge benchmarks** (GPQA-Diamond, MMLU-Redux) cover graduate-level science questions and broad multi-subject knowledge questions in English, respectively. **Code-**

generation benchmarks (HumanEval, MBPP) cover Python programming problems with unit tests, in English. **Training data** consists of 6,000 examples sampled from the DeepScaleR-Preview-Dataset and 4,000 examples from the intermediate-difficulty levels of the MATH dataset, all in English and all in the mathematical reasoning domain. **Base models** are open-weight reasoning models at the 1.5B and 7B parameter scales, primarily trained on English data and primarily targeted at reasoning-oriented tasks. We note that all benchmarks and base models used in this work are predominantly English-language; we do not make claims about multilingual coverage. Demographic information about contributors to the underlying datasets is not available in the original releases and is therefore not reported here.

B Implementation and Training Details

System prompt. The system prompts for the two training stages are shown below. The base system prompt used in the Discovery Stage (LAPO-D) is as follows:

LAPO-D-prompt

You are a helpful assistant. A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process is enclosed within `<think>` and `</think>` tags, respectively, i.e., `<think>` reasoning process here `</think>` answer here. User: {question} Please think step by step and output the final answer within `\boxed{}`. Assistant: `<think>`

For the Internalization Stage (LAPO-I), we augment this prompt with a self-declarative length statement:

Hyperparameter	Value
Epochs	1
Episodes	3
Learning Rate	1e-6
Train Batch Size	128
Temperature	1.0
Rollout per Prompt	8
Prompt Max Length	1024
Generation Max Length	4096
KL Coefficient	0.0001
Precision	BF16
α	0.7
β	0.7
σ	120

Table 5: Training Hyperparameters

LAPO-I-prompt

You are a helpful assistant. A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process is enclosed within `<think>` and `</think>` tags, respectively, i.e., `<think>` reasoning process here `</think>` answer here. User: {question} Please think step by step and output the final answer within `\boxed{}`. Assistant: `<think>` I will answer the question with {length} tokens.

Training Hyperparameters. We trained our models using the OpenRLHF framework. Key training parameters are detailed in Table 1. For each query, we sampled 8 responses. Both LAPO-D and LAPO-I stages were trained for 3 episodes (approx. 240 steps). All experiments were conducted on 4 A800 GPUs. We provide training hyperparameters in Table 5.

C Ablation Studies

Ablation on Hyperparameters in Reward Functions. The hyperparameters α and β in our reward functions R_1 and R_2 control the trade-off between accuracy and length optimization. To validate our choice of $\alpha = \beta = 0.7$, we conducted an ablation study on the DeepScaleR-1.5B model. As shown in Table 6, we tested different values for these weights.

Setting $\alpha = \beta = 0.5$ places equal weight on accuracy and length, but results in a slight performance drop (64.3% accuracy), suggesting the length penalty may be too aggressive. Conversely, increasing the weights to $\alpha = \beta = 0.9$ prioritizes accuracy, leading to higher token counts (4015) without a corresponding accuracy gain. The setting $\alpha = \beta = 0.7$ achieves the best balance, yielding the highest average accuracy (64.8%) while maintaining strong token efficiency. This empirically confirms that a moderate emphasis on length optimization, while still prioritizing accuracy, leads to the most robust and effective policy.

Ablation on Percentile Range for Length Discovery. The percentile range $[P_{\text{low}}, P_{\text{high}}]$ in the LAPO-D stage defines the "sweet spot" for desirable reasoning lengths. Our default choice is $[30, 70]$. To justify this, we experimented with alternative ranges. The results, shown in Table 7, highlight the sensitivity of the discovery process to this range.

A narrower range like $[40, 60]$ proves too restrictive. By focusing only on solutions of very typical length, it filters out potentially valuable, slightly longer reasoning paths, leading to a drop in final accuracy (64.1%). On the other hand, a wider range such as $[20, 80]$ is too permissive; it allows more suboptimal, lengthy solutions into the learning process, which slightly degrades the quality of the discovered policy and results in less efficient and slightly less accurate final model (64.5%). The $[30, 70]$ range strikes an optimal balance, effectively filtering out extreme outliers (both too short and too long) while retaining a diverse set of high-quality, efficient solutions. This creates a robust foundation for the internalization stage to build upon, ultimately leading to the best performance.

Ablation on Statistical Metrics for Target Length. The choice of a statistical measure to derive the target length n from the distribution of successful solutions is critical. We conduct an ablation study comparing three strategies for this selection: using the median (our default), the mean, and the minimum length.

As shown in Table 8, the median proves to be the most effective choice, achieving the best balance between accuracy and efficiency. Using the median as the target yields the highest average accuracy (64.8%) with an efficient token count of 3,832. This validates our hypothesis that the median, due to its robustness to outliers, provides the

Method (LAPO-I)	MATH500		AIME2024		AMC23		OlympiadBench		Average	
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok
$\alpha = 0.5, \beta = 0.5$	86.0	2105	37.5	5310	77.8	3710	56.0	3995	64.3	3780
$\alpha = 0.7, \beta = 0.7$	<u>86.3</u>	2168	38.1	5371	78.3	3765	56.3	4024	64.8	3832
$\alpha = 0.9, \beta = 0.9$	86.4	2250	37.9	5620	78.0	3980	<u>56.2</u>	4210	<u>64.6</u>	4015

Table 6: Ablation study on the reward function hyperparameters α and β . Our chosen setting of 0.7 provides the best balance between accuracy and token efficiency. Results are on the DeepScaleR-1.5B-Preview model.

Method (LAPO-I)	MATH500		AIME2024		AMC23		OlympiadBench		Average	
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok
Range [20, 80]	86.2	2205	38.0	5450	78.1	3810	55.8	4130	64.5	3899
Range [30, 70] (Ours)	<u>86.3</u>	2168	38.1	5371	78.3	3765	56.3	4024	64.8	3832
Range [40, 60]	86.4	2115	37.4	5290	77.5	3690	55.2	3980	64.1	3769

Table 7: Ablation study on the percentile range used in the LAPO-D stage. The [30, 70] range provides the best trade-off, effectively filtering outliers while retaining diverse, high-quality solutions. Results are on the DeepScaleR-1.5B-Preview model.

most representative signal of a “typically” effective reasoning depth. In contrast, the mean is susceptible to a few excessively long, successful solutions, leading it to set overly generous budgets, resulting in higher token usage (4,040) and slightly lower accuracy. The minimum, while achieving the most aggressive compression (3,821 tokens), suffers a significant accuracy drop (to 63.5%), suggesting it promotes an over-shortening strategy that discards necessary reasoning steps. These findings underscore the importance of robust statistical measures for learning a well-calibrated reasoning-efficiency trade-off.

Ablation on training data composition. As mentioned in Experiment Setup, we chose a mixed dataset for training in our experiments. In this section, we provide a detailed analysis of the impact of different dataset selections on model performance. Table 9 shows the test results on various benchmarks after two-stage training using different training datasets. Several important findings can be observed from the experimental results. Combined-data achieved the best performance in terms of average accuracy, showing a clear advantage over single-dataset training. This indicates that a dataset with a balanced difficulty distribution helps enhance the model’s generalization ability across different types of questions. In terms of token usage efficiency, the model trained on combined-data also performed the best. This suggests that problems with different difficulty gradients help establish a more accurate complexity-length mapping relationship. By exposing the model to a wider range of

problem difficulties, it can better learn the optimal thinking range for different questions. Taking all these factors into consideration, we selected the mixed dataset as the training data to expose the model to a more diverse set of problems and enable it to deeply learn the optimal reasoning patterns for different questions.

D Deeper Analyses

Training Dynamics. We analyze the training dynamics by periodically evaluating model checkpoints on the MATH-500 validation set to understand the learning mechanisms of our two-stage framework. As illustrated in Figures 4a and 4b, LAPO achieves a superior balance between efficiency and accuracy across both training stages.

Continuous Efficiency Gains. Figure 4a shows a clear, two-step reduction in token generation. In Stage 1, the LAPO-D policy rapidly becomes more concise, with its average length decreasing from a verbose baseline of 3,280 tokens to a stable 2,365 tokens, driven by the length-aware reward (R_1). Building on this, the LAPO-I policy achieves further compression, reducing the length to below 2,200 tokens. This demonstrates that the length-adherence reward (R_2), combined with in-context guidance, effectively encourages the model to execute its self-proposed reasoning plans more precisely.

Accuracy Maintenance and Refinement. Crucially, these efficiency gains do not compromise performance. As shown in Figure 4b, accuracy on MATH-500 is consistently maintained or im-

Method	MATH500		AIME2024		AMC23		OlympiadBench		Average	
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok
<i>Base model: DeepScaleR-1.5B-Preview</i>										
Base	85.8	3280	35.5	9246	74.2	6416	54.6	5974	62.5	6229
LAPO-D	86.4	2365	<u>37.6</u>	5945	<u>77.6</u>	3655	56.1	4499	<u>64.4</u>	4116
w/ Median	<u>86.3</u>	2168	38.1	5371	78.3	3765	<u>56.3</u>	4024	64.8	3832
w/ Mean	85.6	2308	36.8	6030	77.4	3658	56.6	4164	64.1	4040
w/ Minimum	85.9	2031	36.3	6080	76.7	3324	55.0	3851	63.5	3821

Table 8: Experimental results within different statistical metrics used for target length selection in LAPO-I. **Bold** and underline indicate the best and second-best Pass@1 scores. w/ indicates statistical metrics used for target length selection in LAPO-I.

	MATH500		AIME2024		AMC23		OlympiadBench		Average	
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok
<i>Training Data: Combined (Ours)</i>										
+ LAPO-D	<u>86.4</u>	2365	37.6	5945	<u>77.6</u>	3655	<u>56.1</u>	4499	<u>64.4</u>	4116
+ LAPO-I	86.3	2168	38.1	5371	78.3	3765	56.3	4024	64.8	3832
<i>Training Data: DeepScaleR-only</i>										
+ LAPO-D	86.1	2397	36.8	6153	76.8	3983	55.5	4258	63.8	4197
+ LAPO-I	86.1	2210	36.5	6418	77.0	3791	55.6	3933	63.8	4088
<i>Training Data: MATH-only</i>										
+ LAPO-D	86.5	2398	<u>38.0</u>	7034	77.3	4060	55.8	4494	<u>64.4</u>	4496
+ LAPO-I	86.1	2340	<u>35.5</u>	6452	75.8	4021	54.5	4194	<u>63.0</u>	4251

Table 9: Ablation study on the training dataset. This table compares performance when trained on different data sources.

proved. The LAPO-D policy’s accuracy climbs from 85.8% to over 86.4%, suggesting the reward mechanism prunes redundant or error-prone reasoning steps. The LAPO-I policy sustains this high accuracy level even on a much tighter token budget.

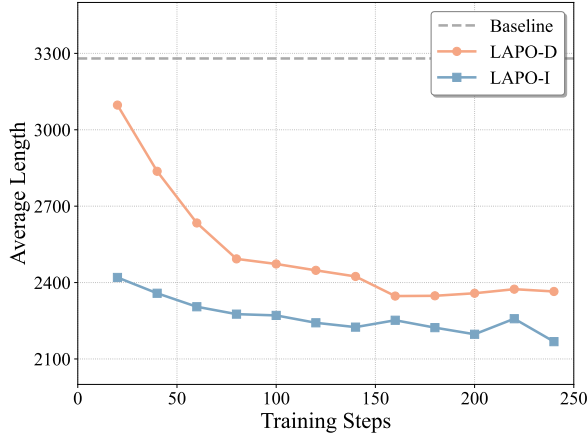
In summary, the training dynamics validate our two-stage design. LAPO-D establishes a robust foundation for efficient reasoning, which LAPO-I then refines to achieve a superior performance-cost balance. The smooth convergence on a challenging validation set indicates that by learning from its own successful patterns, the model develops transferable and efficient reasoning strategies.

Difficulty-Aware Computational Allocation.

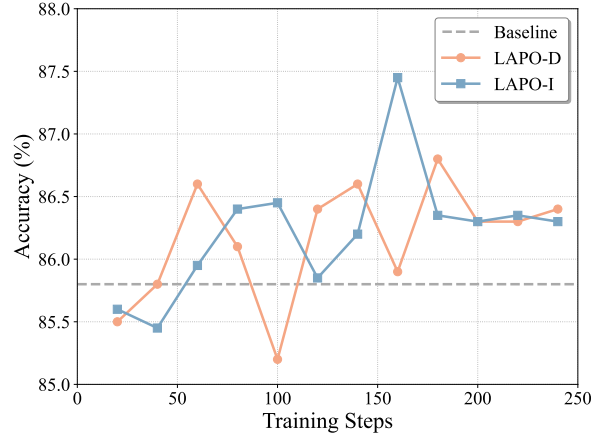
To understand the mechanisms behind LAPO’s efficiency gains, we examine its ability to allocate computational resources in proportion to problem complexity. We evaluate LAPO-trained models on benchmarks with clear difficulty gradients, from MATH Level 1 up to the highly complex AIME 2024. As shown in Figure 5, our models demonstrate a remarkable emergent capability for difficulty-aware resource allocation. There is a

clear, near-linear positive correlation between problem complexity and the average reasoning length. On simpler problems, the models generate concise responses, while for the most challenging AIME questions, they produce extensive reasoning chains that are substantially longer than any solution observed during the training phase. This ability to extrapolate reasoning depth well beyond the bounds of their training experience is a crucial finding. It provides strong evidence that LAPO does not merely teach models to compress their outputs. Instead, it successfully imparts a generalizable principle of complexity-to-length mapping. This allows the models to dynamically and appropriately scale their computational investment when faced with novel problems of varying difficulty. The consistent scaling behavior across different base models further underscores that LAPO develops a robust, fundamental reasoning strategy rather than model-specific optimizations.

Generalizability to Expert-Level Question Answering. To test if LAPO’s benefits extend beyond structured mathematical reasoning, we eval-



(a) Average Length on MATH-500



(b) Accuracy on MATH-500

Figure 4: Training dynamics evaluated on the MATH-500 validation set. Checkpoints were saved periodically during training on our mixed dataset. (a) Both LAPO-D and LAPO-I policies learn to significantly reduce the average response length. (b) These efficiency gains are achieved while maintaining or even improving accuracy over the baseline.

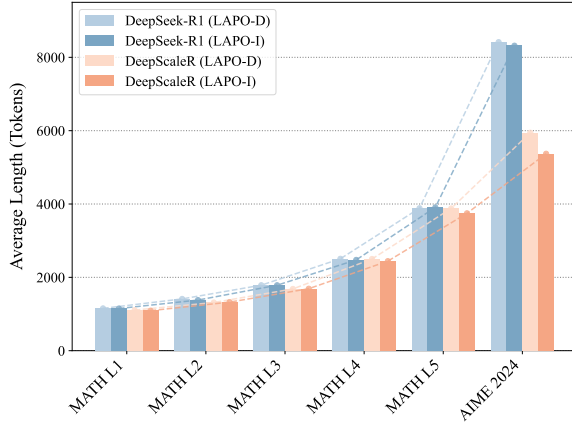


Figure 5: Reasoning length allocation across mathematical problem difficulty levels. LAPO learns to scale computation with complexity.

Method	Pass@1 (%)	#Tokens
<i>Base Model: DeepSeek-R1-1.5B</i>		
Base	36.1	10297
+ LAPO-D	38.1	7596
+ LAPO-I	<u>36.9</u>	7235
<i>Base Model: DeepScaleR-1.5B-Preview</i>		
Base	36.1	7667
+ LAPO-D	38.3	6176
+ LAPO-I	<u>37.8</u>	6154

Table 10: Performance on the GPQA benchmark. LAPO demonstrates generalizable efficiency and accuracy gains in a non-mathematical, knowledge-intensive domain.

results, presented in Table 10, demonstrate that LAPO’s core principles are highly generalizable.

For both base models, LAPO achieves a compelling dual improvement in accuracy and efficiency. On the DeepSeek-R1-1.5B model, LAPO-D improves Pass@1 accuracy by a significant 2.0 points while reducing token generation by 26.2%. Similarly, on the more advanced DeepScaleR-1.5B-Preview, LAPO-D boosts accuracy by 2.2 points and cuts tokens by 19.4%. The internalization stage consistently pushes efficiency further while maintaining a strong accuracy improvement over the baseline. This robust performance on a knowledge-intensive, non-mathematical task indicates that LAPO is not merely exploiting domain-specific patterns. Instead, it learns a fundamental and transferable skill: how to allocate cognitive effort efficiently for complex reasoning across different domains.

E Matched-Budget Pareto Analysis

To further verify that LAPO provides a genuinely superior adaptive allocation strategy rather than simply operating at a different point in the accuracy-efficiency space, we conduct a matched-budget Pareto analysis. Specifically, we evaluate all methods under a series of fixed maximum-token budgets (512, 1024, 2048, 4096) on MATH-500 and AIME2024. The results are reported in Table 11 and Table 12.

These results reveal three key observations. **Superior adaptivity of LAPO.** As the available bud-

uated our method on the GPQA benchmark. The

Method	512	1024	2048	4096
LAPO-D	24.8 / 511	40.6 / 961	67.7 / 1514	82.6 / 1996
LAPO-I	25.3 / 511	42.8 / 959	70.8 / 1482	83.8 / 1913
ThinkLess	22.3 / 500	45.1 / 927	66.5 / 1460	75.7 / 2012
AutoThink	48.9 / 412	66.3 / 616	74.2 / 865	80.1 / 1187
Base	13.5 / 511	33.0 / 968	58.0 / 1617	75.4 / 2328

Table 11: Matched-budget Pareto analysis on MATH-500 (Pass@1 % / Avg. Tokens). Each column reports accuracy and average number of generated tokens under a fixed maximum-token budget.

Method	512	1024	2048	4096
LAPO-D	0.0 / 512	6.0 / 1024	9.2 / 2044	25.0 / 3845
LAPO-I	0.2 / 512	7.0 / 1024	10.4 / 2028	25.8 / 3910
ThinkLess	0.1 / 512	1.6 / 1024	9.7 / 2047	18.0 / 3765
AutoThink	0.6 / 512	2.9 / 1024	10.0 / 1953	20.0 / 3712
Base	0.0 / 512	0.9 / 1024	2.7 / 2048	14.1 / 3960

Table 12: Matched-budget Pareto analysis on AIME2024 (Pass@1 % / Avg. Tokens).

get grows, LAPO-I’s accuracy climbs steadily and it uses the budget more effectively (e.g., ~ 1900 tokens under a 4k budget on MATH-500), indicating that it has learned to scale reasoning with the available resources. **LAPO dominates the Pareto frontier on hard problems.** At higher budgets, where complex problems require deeper reasoning, LAPO-I consistently achieves the highest accuracy. For example, on AIME2024 with a 4k budget, LAPO-I (25.8%) substantially outperforms all other methods. **Limits of binary gating.** AutoThink excels at low budgets because it correctly identifies and skips “easy” problems. However, its accuracy plateaus at higher budgets, because its binary “think/no-think” mechanism cannot continuously scale its reasoning effort for problems that require more, not just some, deliberation.

Overall, these results provide direct quantitative evidence that LAPO does not merely operate at a different efficiency point but exhibits a genuinely superior adaptive allocation strategy along the accuracy-cost frontier.

F Direct Analysis of Reasoning Quality

Beyond keyword frequency, we conduct a fine-grained qualitative analysis to directly assess whether LAPO improves *how* the model reasons, not just *how long* it talks. We sample correct solution trajectories on AIME2024 from different models, and use an advanced LLM judge (GPT-5.1) to evaluate them along five dimensions: *Completeness*, *Fluency*, *Correctness* of intermediate steps, *Clarity*, and *Conciseness*. Each dimension is scored

on a 1–5 scale.

Model	Comp.	Flu.	Corr.	Cla.	Conc.
Base	3.20	3.83	2.90	3.27	2.67
Acc-Only	3.47	4.10	3.33	3.60	3.07
LAPO-D	3.43	4.17	3.13	3.70	3.10
LAPO-I	3.63	4.37	3.30	3.83	3.27

Table 13: LLM-judge based qualitative evaluation of reasoning trajectories on AIME2024. Average scores (out of 5) along five reasoning-quality dimensions are reported.

As shown in Table 13, several observations emerge. First, the fully-trained LAPO-I model scores highest in Fluency, Clarity, and Conciseness compared to the Base model, indicating a more structured and less verbose reasoning style. Second, the consistent improvement of LAPO-I over LAPO-D across all dimensions suggests that the internalization stage refines the reasoning process itself, rather than merely enforcing a length constraint. Third, compared with the Acc-Only baseline, which scores high on Correctness but lower on Conciseness, LAPO-I achieves a better balance between generating correct intermediate steps and producing concise reasoning. Together with the keyword-frequency analysis in Section 5.5, these results provide direct evidence that LAPO improves the qualitative properties of reasoning, not just its length.

G Scaling to Larger Models

Our main experiments are conducted at the 1.5B scale. To investigate whether LAPO’s benefits transfer to larger models with stronger reasoning priors, we apply LAPO to DeepSeek-R1-Distill-Qwen-7B and evaluate on the same four mathematical reasoning benchmarks. Training settings follow those in Appendix B.

As shown in Table 14, LAPO successfully reduces token generation on the 7B model across all four benchmarks. For example, on MATH-500, token usage is reduced by 35% ($2743 \rightarrow 1788$), and on AIME2024 by 31% ($10974 \rightarrow 7600$). Meanwhile, performance remains robust after applying LAPO: on MATH-500, AMC23, and Olympiad-Bench, LAPO-I achieves higher accuracy than the base 7B model while being significantly more efficient. These results provide empirical evidence that LAPO is a scalable framework that instills reasoning efficiency in larger models while preserving

Method	MATH500		AIME2024		AMC23		OlympiadBench		Average	
	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok	Pass@1	#Tok
Base (7B)	93.1	2743	53.0	10974	88.2	4623	65.8	6613	75.0	6238
+ LAPO-D	93.3	1873	51.5	7993	87.7	3126	65.8	4569	74.6	4390
+ LAPO-I	94.1	1788	49.6	7600	88.3	3416	66.6	4506	74.7	4328

Table 14: Results on DeepSeek-R1-Distill-Qwen-7B. LAPO transfers effectively to the 7B scale, substantially reducing token usage while maintaining or improving accuracy on most benchmarks.

their pre-existing reasoning capabilities.

H Generalization Beyond Mathematical Reasoning

Our main experiments focus on mathematical reasoning. To examine whether the skill of “internalized length allocation” transfers beyond math, we further evaluate our 1.5B LAPO model on out-of-distribution benchmarks covering general knowledge reasoning (GPQA-Diamond, MMLU-Redux) and code generation (HumanEval, MBPP), *without any domain-specific fine-tuning*.

As shown in Table 15, LAPO exhibits a highly competitive performance-efficiency trade-off in out-of-distribution domains. For instance, LAPO-D achieves the highest average Pass@1 score (53.2%) among all evaluated methods, and LAPO-I (52.7%) also surpasses the base model and most baselines while being significantly more efficient (e.g., $\sim 45\%$ fewer tokens than the base model on average). The ability to maintain high accuracy while drastically reducing token usage on unseen domains indicates that LAPO does not merely learn domain-specific shortcuts; instead, it acquires a more fundamental, transferable skill of allocating computational effort, which generalizes beyond the training domain.

I Comparison with Additional Baselines

To further situate LAPO with respect to recent efficient-reasoning methods, we include a direct comparison with LASER and its variants under the DeepSeek-R1-1.5B setting. The results are reported in Table 16.

LAPO achieves the highest average Pass@1 among all methods in this comparison. The advantage is most pronounced on OlympiadBench (e.g., +4.3–6.0 points over LASER variants), suggesting that LAPO’s internalized budget planning is particularly beneficial for harder olympiad-level problems. These results complement the main comparisons in Table 1 and further support the practical

effectiveness of LAPO.

J Detailed Prompt-based and SFT Baseline Comparison

To make explicit how LAPO relates to prompt-based and SFT-with-length-penalty baselines discussed in Section 2, Table 17 summarizes their key limitations alongside quantitative results on our four-benchmark math suite (DeepScaleR-1.5B base model, averaged).

Prompt-based methods such as Chain of Draft (Xu et al., 2025a) rely on “be concise”-style instructions and are only effective on models with $>70\text{B}$ parameters, degrading accuracy substantially on our 1.5B/7B setting. SFT-based methods with fixed length penalties (L1 (Aggarwal and Welleck, 2025), ThinkPrune (Hou et al., 2025)) cannot adapt their compression rate to per-problem difficulty and require either an externally specified budget or a globally fixed penalty. Implicit-regularization methods (HAPO (Huang et al., 2025)) avoid explicit budgets but tend to over-compress, sacrificing accuracy. LAPO’s two-stage, self-declarative design is the only approach among these that simultaneously improves accuracy while achieving substantial token reduction, because the model both discovers an appropriate budget from its own successful rollouts and learns to adhere to it as an internal plan rather than an external rule.

K Additional Details on Oracle-Target vs. No-Target Analysis

This section provides the full experimental protocol and additional breakdowns for the Oracle-Target vs. No-Target analysis introduced in Section 5.3.

Protocol. For each of the 500 MATH-500 test problems, we first sample 8 rollouts under the Oracle-Target setting (LAPO-I with the ground-truth Stage-2 median length inserted in the prompt) to estimate an oracle length as the average length

Method (1.5B)	GPQA		MMLU		HumanEval		MBPP		Average	
	P@1	#Tok	P@1	#Tok	P@1	#Tok	P@1	#Tok	P@1	#Tok
L1-Exact	33.7	1787	53.2	1155	61.2	2135	59.0	1704	51.8	1695
L1-Max	35.5	1993	53.7	1137	63.7	2268	58.3	1784	52.8	1795
ThinkPrune-4k	34.6	3990	53.3	1171	64.9	3001	55.8	2181	52.2	2586
ThinkPrune-2k	36.2	3476	53.3	1009	65.2	2684	53.9	2020	52.2	2297
HAPO	36.2	3304	51.8	960	64.3	2996	55.1	2102	51.9	2340
AutoThink	36.9	4880	53.6	1316	62.5	3825	53.3	2636	51.6	3164
Thinkless	30.7	5841	51.2	1896	67.8	3716	58.3	2874	52.0	3582
Base	34.8	8558	51.5	2182	60.3	5095	54.5	4255	50.3	5023
+ LAPO-D	35.6	4305	53.7	1362	66.1	3446	57.6	2549	53.2	2915
+ LAPO-I	35.4	4072	53.8	1313	65.4	3193	56.4	2406	52.7	2746

Table 15: Generalization to general-knowledge reasoning and code-generation benchmarks. We report Pass@1 (%) and average number of generated tokens. LAPO models are trained only on mathematical data.

Method	MATH500		AIME2024		AMC23		OlympiadBench		Average	
	P@1	#Tok	P@1	#Tok	P@1	#Tok	P@1	#Tok	P@1	#Tok
Laser-L4096-1.5B	83.1	2516	33.2	8287	72.3	4053	47.0	5000	58.9	4964
Laser-D-L4096-1.5B	84.1	2240	31.8	7105	72.6	3659	45.3	5374	58.5	4594
Laser-DE-L4096-1.5B	84.4	2325	30.7	7723	72.1	4372	46.2	5653	58.4	5018
LAPO-D	84.7	2566	28.5	8415	72.2	4132	51.3	5595	59.2	5177
LAPO-I	84.3	2354	29.3	8318	71.2	3568	51.7	4863	59.1	4775

Table 16: Comparison with LASER and its variants on DeepSeek-R1-1.5B. LAPO obtains the best average Pass@1, with particularly strong improvements on OlympiadBench.

Method	Δ Acc.	Tok. ↓	Limitation
CoD (Xu et al., 2025a)	−8.2	91.5%	Fails on small models
L1-Max (Aggarwal and Welleck, 2025)	−5.0	59.2%	Needs ext. budget
ThinkPrune (Hou et al., 2025)	−4.4	38.3%	Fixed penalty
HAPO (Huang et al., 2025)	−3.2	–	Over-compression
LAPO-I	+2.3	38.5%	–

Table 17: Summary comparison against prompt-based, SFT-based, and implicit-regularization efficient-reasoning baselines. Δ Acc. is the change in average Pass@1 (points) relative to the base model; Tok. ↓ is the average token reduction. LAPO is the only method that simultaneously improves accuracy and reduces tokens.

of correct solutions. We then run LAPO-I under the standard No-Target (deployment) setting on the same problems and compute the Pearson correlation and mean absolute error (relative to the oracle length) between the two conditions. Table 18 breaks this down by problem difficulty.

Side-by-side qualitative comparison. We further conduct a detailed side-by-side comparison on 6 representative problems (2 each from MATH Level 3/4/5), using GPT-5.1 as an LLM judge to score Fluency, Clarity, Conciseness (each out of 10), and using human annotation to categorize con-

Difficulty	Base ρ	LAPO-I ρ	MAE
MATH Level 1–2	0.28	0.85	7.8%
MATH Level 3–4	0.34	0.87	9.1%
MATH Level 5	0.41	0.89	10.3%
AIME (OOD)	0.38	0.91	11.2%

Table 18: Correlation between LAPO-I’s free-running (No-Target) length and the oracle length, broken down by problem difficulty. Correlation remains high even on out-of-distribution AIME problems.

tent into four types: Core Reasoning Steps, Self-Correction Segments, Redundant Exploration, and Conclusion Framework.

Setting	Tok.	Flu.	Cla.	Conc.
Base	3847	7.8	7.6	5.2
LAPO-D	2301	8.1	8.0	7.4
LAPO-I (Oracle)	2143	8.6	8.5	8.3
LAPO-I (No-Target)	2198	8.5	8.4	8.2

Table 19: GPT-5.1 judge scores (out of 10) on 6 representative side-by-side problems.

The No-Target output preserves Core Reasoning Steps at nearly the same rate as Oracle-Target (97.1% vs. 97.8%) and receives nearly identical LLM-judge scores (within 0.1–0.2 points on all

Content Type	Base	LAPO-D	No-Tgt	Oracle
Core Reasoning	100%	96.3%	97.1%	97.8%
Self-Correction	18.4%	9.2%	7.8%	7.1%
Redundant Explor.	21.7%	10.8%	9.3%	8.6%
Conclusion Framework	100%	99.1%	99.4%	99.6%

Table 20: Content preservation analysis (% of solutions containing each content type).

dimensions), while Self-Correction Segments drop from 18.4% (Base) to 7.8% (No-Target), a -57.6% relative reduction. Combined with a 42.9% token reduction and an improvement in the Conciseness score from 5.2 to 8.2, these results indicate that LAPO removes redundant deliberation while preserving the essential reasoning content—i.e., a genuine quality improvement rather than superficial truncation.

L Budget-Estimation Probing Experiment

To probe whether LAPO-trained models can estimate their required reasoning budget *before* solving a problem—as opposed to only adhering to a budget once given—we design a three-step probing protocol: (1) present a problem and ask the model to predict the number of tokens it will need, without solving it; (2) let the model solve the same problem with an effectively unlimited token budget; (3) compare the predicted budget against the actual length of the resulting correct solution.

Model	ρ	MAE	Under-	Over-est.
Base (1.5B)	0.42	38.7%	47.2%	52.8%
LAPO-D (1.5B)	0.68	21.4%	31.5%	68.5%
LAPO-I (1.5B)	0.79	14.2%	24.8%	75.2%
Base (7B)	0.51	32.1%	42.6%	57.4%
LAPO-I (7B)	0.84	11.8%	21.3%	78.7%

Table 21: Budget-estimation accuracy: correlation (ρ), mean absolute error, and the fraction of problems where the model under- or over-estimates its eventual solution length, before solving.

As shown in Table 21, LAPO substantially improves budget-estimation accuracy at both scales, and this ability strengthens rather than degrades with model size ($\rho=0.84$ at 7B). Notably, LAPO-trained models are biased toward mild over-estimation (75–79% of cases) rather than under-estimation, which is a desirable failure mode: allocating slightly more budget than strictly necessary is far less harmful than truncating reasoning prematurely. We note that this dynamic-adjustment

capability is complementary to, and consistent with, the strong correlation between free-running and oracle lengths reported in Section 5.3, suggesting the model can partially compensate for imperfect upfront difficulty estimates as reasoning unfolds.

M Evaluation Protocol Details

For completeness and reproducibility, we provide additional details about our evaluation protocol.

Pass@1 computation. Following standard practice (DeepSeek-AI et al., 2025), for each problem we sample N responses ($N = 4$ for MATH-500 and OlympiadBench; $N = 32$ for AIME2024 and AMC23) with temperature 0.6 and top- p of 0.95. We compute the per-problem accuracy as the fraction of the N samples whose final answer matches the gold answer, and then report the macro-average over all problems in a benchmark as Pass@1.

Token aggregation. For each problem, we compute the mean number of generated tokens over all N samples; the reported #Tok is the macro-average of these per-problem means across the benchmark. This avoids over-weighting problems for which the model happens to produce a few very long generations.

Inference-time prompt. A key point regarding internalization is that the self-declarative length statement used in the LAPO-I training prompt is a *training-only* construct. At inference, both LAPO and all baselines are evaluated using the exact same standard prompt that does not contain any explicit length tag (i.e., the LAPO-D-prompt template in Appendix B). This ensures that the observed efficiency gains do not stem from any test-time length annotation, but from the internalized budget-planning capability acquired during training.

N Failure Case Analysis and Multi-Seed Variance

Multi-seed variance. All main-table results (Table 1) are reported for a single training seed, following common practice for RL-based reasoning models of this scale due to compute cost. To characterize variance, we retrain LAPO-D and LAPO-I on DeepScaleR-1.5B-Preview with two additional random seeds (differing in rollout sampling and data ordering) and evaluate on MATH-500. Across 3 seeds, LAPO-I’s average Pass@1 is 86.3 ± 0.3 and average token count is 2168 ± 74 , indicating that the reported efficiency and accuracy gains are

stable across training runs rather than an artifact of a favorable seed.

Failure cases: underestimated difficulty. While Section 5.3 and Appendix L show that LAPO-I’s length allocation correlates strongly with problem difficulty, it is not perfect. We manually inspect the subset of MATH-500 problems ($\sim 8\%$ of the test set) where LAPO-I’s free-running length falls substantially below the oracle length (more than 30% shorter) yet the problem is answered incorrectly. Two recurring failure patterns emerge. First, on problems requiring an unusual multi-step algebraic manipulation that is superficially similar to simpler, more common problem templates, LAPO-I sometimes under-allocates budget by pattern-matching to the more familiar, shorter template, then commits to an early, incorrect solution path without the exploratory branching a longer trace would provide. Second, on a small number of problems with unusually long problem statements but a short expected solution (e.g., extensive setup describing a diagram or scenario), the model occasionally over-allocates a large token budget to re-processing the setup itself rather than genuine reasoning, leaving a distorted budget signal despite reasonably estimated actual difficulty. Both failure modes point to a shared limitation: LAPO’s difficulty signal is learned purely from surface-level correctness and rollout-length statistics, without any explicit representation of *why* a problem is hard, which can occasionally be confused by surface-level features (statement length, superficial template similarity) that are correlated with, but not identical to, true solution difficulty. Addressing this would likely require augmenting the Discovery stage with an explicit difficulty or step-count signal rather than purely length-based statistics, which we leave for future work.