Self-Compression of Chain-of-Thought via Multi-Agent Reinforcement Learning

Anonymous ACL submission

Abstract

The inference overhead induced by redundant reasoning undermines the interactive experience and severely bottlenecks the deployment of Large Reasoning Models. Existing reinforcement learning (RL)-based solutions tackle this problem by coupling a length penalty with outcome-based rewards. This simplistic reward weighting struggles to reconcile brevity with accuracy, as enforcing brevity may compromise critical reasoning logic. In this work, we address this limitation by proposing a multi-agent RL framework that selectively penalizes redundant chunks, while preserving essential reasoning logic. Our framework, Self-Compression via MARL (SCMA), instantiates redundancy detection and evaluation through two specialized agents: **a Segmentation Agent** for decomposing the reasoning process into logical chunks, and **a Scoring Agent** for quantifying the significance of each chunk. The Segmentation and Scoring agents collaboratively define an importance-weighted length penalty during training, incentivizing **a Reasoning Agent** to prioritize essential logic without introducing inference overhead during deployment. Empirical evaluations across model scales demonstrate that SCMA reduces response length by 11.1% to 39.0% while boosting accuracy by 4.33% to 10.02%. Furthermore, ablation studies and qualitative analysis validate that the synergistic optimization within the MARL framework fosters emergent behaviors, yielding more powerful LRMs compared to vanilla RL paradigms.

1 Introduction

Recent advancements in Large Reasoning Models (LRMs) (Chen et al., 2025a; Guo et al., 2025) have demonstrated remarkable capabilities characterized as 'thinking'. Specifically, LRMs employ detailed Chain-of-Thought (CoT) sequences to facilitate complex problem-solving through self-reflection, backtracking, and verification. The deep thinking capacities are predominantly elicited through Reinforcement Learning (RL) (Shao et al., 2024; Ramesh et al., 2024), but this paradigm simultaneously introduces significant efficiency bottlenecks. This issue arises because existing RL optimization is primarily driven by sparse, outcome-based binary rewards. Lacking fine-grained guidance on the intermediate reasoning process, models often generate prolonged paths cluttered with non-essential steps or repetitive verifications to maximize reward certainty (Chen et al., 2024; Cuadron et al., 2025). This redundancy imposes substantial inference latency for deployment, necessitating methods that guide models to 'think less' by synthesizing concise, robust, and accurate reasoning chains.

Current methods, extensively discussed in Appendix A, primarily curb reasoning redundancy via global length penalties (Cheng et al., 2025; Tu et al., 2025; Dai et al., 2025; Zeng et al., 2025; Hou et al., 2025). However, this token-level penalty misaligns with the true nature of reasoning redundancy, which is inherently a *process-level* phenomenon. Redundancy manifests not merely as extraneous tokens, but as complex behavioral inefficiencies within the reasoning trajectory—such as repetitive verification, invalid backtracking, or low-value exploration (Zelikman et al., 2022). Consequently, undifferentiated length penalties pose a significant risk: models may inadvertently sacrifice critical reasoning steps just to satisfy length constraints, compromising final accuracy.

To precisely eliminate meaningless repetition while preserving core reasoning steps, a framework requires three core capabilities: *Generation* of candidate paths, *Decomposition* of continuous reasoning into logical units, and *Valuation* of each unit's contribution. Existing single-agent RL paradigms inherently struggle here. Guided only by sparse outcome rewards and global lengths, a single agent cannot discern which intermediate steps were essential versus mere repetition, thus lacking ex-

plicit mechanisms for fine-grained process regulation. This bottleneck necessitates a paradigm shift towards Multi-Agent Reinforcement Learning (MARL). Far from a superficial use of multiple prompts, MARL structurally decomposes *reasoning self-regulation* into a closed-loop system: a Reasoning Agent explores solution paths, a Segmentation Agent structures the reasoning flow, and a Scoring Agent translates these discrete units into granular value signals. Together, they collaborate to achieve simultaneous optimization of reasoning generation and fine-grained process control.

To bridge this gap, we propose Self-Compression via MARL (SCMA), an end-to-end multi-agent framework designed to achieve fine-grained *reasoning self-regulation* rather than mere sequence compression. SCMA innovatively reformulates this regulation task as a collaborative game, coordinating three specialized roles: the Reasoning Agent explores the solution space to generate reasoning paths, the Segmentation Agent structurally parses the reasoning paths into discrete logical chunks, and the Scoring Agent quantifies the substantive contribution of each chunk to the final derivation. In this setup, standard length penalties are replaced by an importance-weighted penalty derived collaboratively by the Segmentation and Scoring agents. This fine-grained feedback empowers the Reasoning Agent to actively self-regulate by selectively discarding redundancy while preserving essential logic. Crucially, since the Reasoning Agent's ability to balance correctness with conciseness hinges entirely on the fidelity of this dynamic feedback, the three roles are intrinsically coupled. Consequently, SCMA employs a shared reward to drive the co-evolution of all agents by multi-agent group relative policy optimization (GRPO), ensuring their objectives remain strictly aligned towards generating high-quality, efficient reasoning chains. In summary, our contributions can be concluded as follows:

- **Framework Proposal:** We propose SCMA, a framework that reformulates the Chain-of-Thought compression task by transitioning from the conventional single-agent RL to a MARL training paradigm, enabling fine-grained compression with no test-time overhead.
- **Mechanism Design:** We design a collaborative optimization mechanism governed by a multi-agent GRPO objective, incorporating an importance-weighted length penalty as a unified reward signal to drive the co-evolution of all agents.
- **Experimental Verification:** We empirically demonstrate that SCMA reduces reasoning length by **11.1%–39.0%** while simultaneously improving accuracy by **4.33%–10.02%**. Ablation studies further validate the superiority of MARL joint optimization, which significantly outperforms decoupled baselines relying on single-agent architectures.

2 Preliminary

2.1 General RL with Length Penalty

In post-training based on RL, LLM is modeled as a policy π_θ that autoregressively generates a response y given a prompt x. To compress the reasoning process, existing solutions typically incorporate a length penalty, formulating the reward function as $R(y|x) = R_{\text{acc}}(y|x) - \alpha f(|y|)$. R_{acc} denotes outcome-based rewards for correctness, α is the penalty strength coefficient, and $f(\cdot)$ applies group-wise normalization to the response length $|y|$, typically taking a linear or exponential form (Cheng et al., 2025; Zhang and Zuo, 2025). Consequently, the RL objective is to optimize θ for maximizing the expected reward:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)} \left[R_{\text{acc}}(y|x) - \alpha f(|y|)\right] \tag{1}$$

2.2 Group Relative Policy Optimization (GRPO)

GRPO (Shao et al., 2024) streamlines the alignment of large language models by eliminating the critic network inherent to PPO (Schulman et al., 2017), thereby reducing memory overhead and avoiding value approximation instability. Specifically, GRPO estimates the advantages by sampling a group of responses $\{y_i\}_{i=1}^G$ and normalizing the rewards within group. The advantage for each token t in the i-th response is computed as $\hat{A}_{i,t} = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$, where $\mathbf{r}$ is the reward vector for the group. The GRPO objective incorporates a clipping mechanism and a KL-divergence penalty to stabilize policy updates and prevent excessive

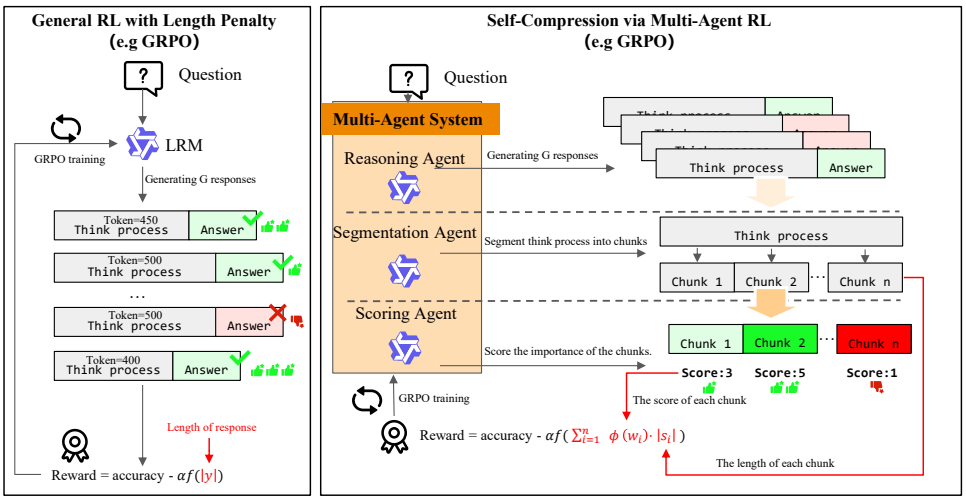


Figure 1: Overview of SCMA Compared to general RL with length penalty. **(Left)** The general RL calculates rewards by penalizing the length of the thinking process directly. **(Right)** The SCMA employs an importance-weighted length penalty within a multi-agent system.

deviation from the reference policy π_{ref}:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim P(x),\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \left\{ \min\left[r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\right] - \beta D_{\text{KL}}\left[\pi_\theta \parallel \pi_{\text{ref}}\right]\right\}, \tag{2}$$

where the importance sampling ratio is defined as:

$$r_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})}. \tag{3}$$

The optimization is constrained by the clipping hyperparameter ϵ and the KL penalty β.

3 Method

3.1 An Overview of SCMA

As illustrated in Fig 1, SCMA departs from the scalar penalties of general RL by instantiating three functionally distinct agents within a multi-agent system to achieve fine-grained compression. Crucially, all agents share parameters θ from a single base LLM π_{base} to achieve internal reasoning compression[1], a process we term **Self-Compression**. Under this paradigm, the multi-agent framework serves as a training-only auxiliary, enabling the exclusive deployment of the Reasoning Agent during inference without incurring additional computational overhead. Within the Self-Compression process, the agents are assigned the following specialized roles:

1. The **Reasoning Agent** ($\pi_\theta^{\text{reason}}$), which explores the solution space to generate an initial reasoning path y:
$$y \sim \pi_\theta^{\text{reason}}(\cdot \mid x) \tag{4}$$
2. The **Segmentation Agent** (π_θ^{seg}) structurally parses the reasoning path y into n discrete logical chunks:
$$\{s_1, \ldots, s_n\} \sim \pi_\theta^{\text{seg}}(y) \tag{5}$$
3. The **Scoring Agent** ($\pi_\theta^{\text{score}}$) then evaluates the inferential significance of each chunk, assigning an importance score w_i that quantifies its essentiality in deriving the final correct solution:
$$\{w_1, \ldots, w_n\} \sim \pi_\theta^{\text{score}}(\{s_1, \ldots, s_n\}) \tag{6}$$

Building upon the multi-agent system, the Segmentation and Scoring agents collaboratively define an **importance-weighted length penalty** to achieve fine-grained compression. This distinguishes SCMA from the general RL approach in Section 2.1 and mitigates the inherent risk of sacrificing critical reasoning steps, which is a common failure mode in monolithic sequence-level optimization. Specifically, the reformulated total reward is defined as:

$$R(y|x) = R_{\text{acc}}(y|x) - \alpha f\left(\sum_{i=1}^{n} \underbrace{\phi(w_i)\cdot|s_i|}_{\text{importance-weighted}}\right), \tag{7}$$

[1]For a detailed discussion on the rationale of this parameter-sharing mechanism, please refer to Appendix B.

where $|s_i|$ denotes the length of the i-th segment, and $\phi(w_i)$ is a weighting function that modulates the penalty scale based on the segment's importance score. Intuitively, $\phi(\cdot)$ is designed as a monotonically decreasing function to inversely map importance to penalty weight: a redundant segment ($w_i \to 0$) is assigned a high weight (e.g., $\phi(w_i) \approx 5$) to enforce compression, whereas a critical segment ($w_i \to 5$) receives a negligible weight (e.g., $\phi \approx 0$) to exempt it from the length penalty. This mechanism encourages the agent to preserve pivotal reasoning logic when compression.

We frame the optimization of SCMA as a Multi-Agent Reinforcement Learning (MARL) problem, where the reformulated total reward serves as a shared global payoff for all participating agents. This collective incentive ensures that the core objective of producing accurate and concise responses is inherently synchronized with the individual functional goals of each agent. In this process, the Reasoning Agent learns to generate accurate solutions while eliminating redundant reasoning steps. Simultaneously, the Segmentation Agent evolves to perform more rational and fine-grained partitioning, enabling the Scoring Agent to effectively assess the importance of the thinking process. We formalize the joint trajectory as a structured tuple that captures the sequential dependencies among the three agents:

$$\tau = \left\{ \begin{array}{l} \text{"response"} : y, \\ \text{"chunks"} : \mathcal{S} = \{s_1, s_2, \ldots, s_n\}, \\ \text{"scores"} : \mathcal{W} = \{w_1, w_2, \ldots, w_n\} \end{array} \right\} \quad (8)$$

The transition $(x, \tau, R(y|x))$ drives the MARL optimization, with each agent $\{\pi_\theta^{\text{reason}}, \pi_\theta^{\text{seg}}, \pi_\theta^{\text{score}}\}$ deriving its own advantage $\hat{A}_k(\tau, x)$ from the shared global reward.

3.2 Detailed Configuration for MARL

Grounded in the theory of Markov Games (Littman, 1994), the SCMA framework is formulated as a tuple:

$$\mathcal{M} = \langle \mathcal{G}, \mathcal{V}, \mathcal{O}, \mathcal{A}, \mathcal{R} \rangle, \quad (9)$$

where $\mathcal{G} = \{\pi_\theta^{\text{reason}}, \pi_\theta^{\text{seg}}, \pi_\theta^{\text{score}}\}$ represents the set of specialized agents responsible for reasoning, segmentation, and scoring, respectively. V represents a vocabulary of the underlying large language model, serving as the foundational set for both $\mathcal{O}$ and $\mathcal{A}$. $\mathcal{O} = \{\mathcal{O}_i\}_{i \in \mathcal{G}}$ represents the collection of each local observations, $\mathcal{A} = \prod_{i \in \mathcal{G}} \mathcal{A}_i$ is the joint action space, and $\mathcal{R}$ is a reward function shared for all agents. Generally, both the observation space and the action space of each agent are defined over the shared vocabulary $\mathcal{V}$ of the language model, treating the generation process as a sequential token-level decision problem (Ouyang et al., 2022). In practice, the effective observation space and action space of each agent are constrained by its role-specific prompt (Li et al., 2023). The concrete instantiations of the observation space, action space, and reward function for different agents are specified in the following.

The observation space of each agent: As formalized in Eq. 4, 5 and 6, the observation spaces exhibit a sequential dependency chain, where the output action of one agent constitutes the primary context for the next. Each agent's observation $o_i \in \mathcal{O}_i$ is constructed by concatenating its role-specific prompt P_i with the relevant task context. Specifically, the **Reasoning Agent** integrates the role-specific prompt P_{reason} with the question q as x to generate a reasoning path y. Subsequently, the **Segmentation Agent** treats reasoning path y and its role-specific prompt P_{seg} as its observation, parsing it into structured logical chunks. Finally, the **Scoring Agent** observes the sequence of discrete logical chunks $\{s_1, \ldots, s_n\}$ and role-specific prompt P_{score} to evaluate the importance of each chunk. $\mathcal{O}$ is given by:

$$\begin{aligned} \mathcal{O}^{\text{reason}} &= \{x, P_{\text{reason}}\} \\ \mathcal{O}^{\text{seg}} &= \{y, P_{\text{seg}} \mid y \sim \pi_\theta^{\text{reason}}(\cdot \mid x)\} \\ \mathcal{O}^{\text{score}} &= \{S, P_{\text{score}} \mid S \sim \pi_\theta^{\text{seg}}(y)\} \end{aligned} \quad (10)$$

The concrete implementation details and full templates for P_{reason}, P_{seg}, and P_{score} are provided in Appendix C.

The action space of each agent: Corresponding to the sequential observation process, the action spaces of the agents are defined by their specific generative objectives and formatting constraints. The **Reasoning Agent** operates within an unconstrained solution search space, generating a natural language sequence y from the Kleene closure (Kleene, 1956) of the model vocabulary $\mathcal{V}^*$. The **Segmentation Agent** performs a structural transformation, partitioning y into a sequence of steps delimited by <seg> tags, subject to the constraint that the composition of these segments reconstructs the original path. Finally, the **Scoring Agent** maps each segment to a discrete scalar value

$w_i \in \{1, \ldots, 5\}$, encapsulated by <score> tags.

$$\mathcal{A}^{\text{reason}} = \{y \mid y \in \mathcal{V}^*\}$$
$$\mathcal{A}^{\text{seg}} = \left\{ \{\texttt{<seg>}s_i\texttt{</seg>}\}_{i=1}^{n} \;\middle|\; \text{s.t.} \bigoplus_{i=1}^{n} s_i = y \right\}$$
$$\mathcal{A}^{\text{score}} = \{\{\texttt{<score>}w_i\texttt{</score>}\}_{i=1}^{n} \tag{11}$$

where $\mathcal{V}^*$ represents the set of all possible token sequences generated by the language model. The constraint in $\mathcal{A}^{\text{seg}}$ ensures lossless parsing, where the union of the segmented contents s_i must equate to the original reasoning path y.

The shared reward function: We introduce an importance-weighted length penalty to balance reasoning accuracy with conciseness, the design philosophy of which is formulated in Eq. 12. Specifically, the weighting function $\phi(w_i) = 5 - w_i$ maps the importance score $w_i \in \{1, \ldots, 5\}$ to a penalty coefficient, ensuring that critical segments with higher scores incur a reduced penalty. The function $f(\cdot)$ then applies group-wise normalization to the weighted length by computing $\sum \phi(w_i) \cdot |s_i|$ and normalizing it by the maximum total length among all correct reasoning paths in the candidate set $\mathcal{C}$. The total reward $R(y|x)$ is calculated as[2]:

$$\begin{cases} R_{\text{acc}}(y|x) + \alpha\left(1 - \dfrac{\sum_{i=1}^{n}(5-w_i)\cdot|s_i|}{\max_{j\in\mathcal{C}}\left(\sum_{m=1}^{n_j}|s_{j,m}|\right)}\right), & \text{if } y \in \mathcal{C} \\ 0, & \text{if } y \notin \mathcal{C}. \end{cases} \tag{12}$$

To mitigate bias from intrinsic problem complexity, the penalty is normalized against the maximum weighted length within the training batch $\mathcal{C}$. This relative scaling ensures equitable regularization across different tasks, while α balances the penalty's influence on the total reward. We show in Sec. D that this objective is equivalent to maximizing R_{acc} under a weighted length constraint.

In addition to the shared reward, we further define agent-specific format rewards (R_{fmt}) to ensure structural consistency and stabilize multi-agent training. A binary reward is assigned for strict adherence to the following XML-style tag protocols and the specific constraints of the action space: the Reasoning Agent must enclose thinking process within <think> tags; the Segmentation Agent must comprehensively demarcate segments using <seg> tags; and the Scoring Agent is required to assign scores to every segment within <score> tags.

[2]For a detailed discussion on the justification of this reward design, please refer to Appendix K.

3.3 Multi-Agent GRPO Optimization

As discussed in Section 2.2, the standard GRPO is inherently designed for single-agent settings, primarily focusing on calculating advantages for the reasoning agent. In this work, we extend GRPO to MARL context to collaboratively optimize the policies of three agents: $\mathcal{G} = \{\pi_\theta^{\text{reason}}, \pi_\theta^{\text{seg}}, \pi_\theta^{\text{score}}\}$. Adopting a configuration similar to MAPPO in *StarCraft II* (Yu et al., 2022; Feng et al., 2025), our Multi-Agent GRPO leverages a shared global reward and employs a parameter-sharing strategy to enhance sample efficiency and foster synergistic collaboration among agents. Consequently, all three agents are instantiated from a single base LLM π_{base} and are distinguished only by role-specific prompts P_{reason}, P_{seg}, P_{score}. The pseudocode for the multi-agent GRPO optimization is presented in Appendix H. The procedure initiates with the Multi-Agent Rollout phase, where the input data propagates sequentially through the Reasoning, Segmentation, and Scoring agents to generate cooperative reasoning chains. Following the trajectory collection, the Multi-Agent Optimization phase is executed. In this stage, group relative advantages are computed for each agent, and the shared parameters θ are updated by maximizing the GRPO objective for each agent respectively.

4 Experiment

We design experiments to answer the questions: **RQ1:** Can SCMA outperform existing baselines, particularly length-penalized RL methods, by achieving effective reasoning compression without compromising solution accuracy? **RQ2:** How does the hyperparameter α influence the training effectiveness and the stability of SCMA? **RQ3:** Is the multi-agent cooperative optimization essential for our framework? **RQ4:** How does the fine-grained compression capability emerge during training?

4.1 Experiment Setting

Dataset, Baselines and LLM Models: We evaluate our framework across a spectrum of difficulty levels, from fundamental arithmetic to competition mathematics, using GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), AMC23 (Li et al., 2024), and AIME24/25 (Li et al., 2024). We benchmark our approach against GRPO and two established baselines that employ length penalties: LC-R1_LP (Cheng et al., 2025) and RL+LP (Arora and Zanette, 2025). Detailed specifications for the

Table 1: Performance and reasoning length of SCMA and baselines across multiple large reasoning models.

Method	GSM8K		MATH500		AIME24		AIME25		AMC23		Overall			
	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc		Tokens	
DeepSeek-R1-Distill-Qwen-1.5B														
Vanilla	85.97	1496	78.40	3801	33.30	9237	20.00	9478	62.50	6261	56.03		6054	
GRPO	87.71	1063	79.00	3773	40.00	8224	30.00	9273	70.00	6069	**61.34**	+5.31%	**5680**	-6.2%
LC-R1_LP	79.68	451	77.40	3092	36.60	8656	26.60	8504	67.5	5414	**57.55**	+1.52%	**5223**	-13.7%
RL+LP	81.50	556	76.20	2718	30.00	7683	26.60	7851	62.5	4671	**55.36**	-0.67%	**4695**	-22.4%
SCMA (Ours)	86.20	661	79.00	3551	36.66	8672	30.00	8518	70.00	5504	**60.36**	+4.33%	**5381**	-11.1%
DeepSeek-R1-Distill-Qwen-7B														
Vanilla	92.40	1833	86.40	3080	50.00	8726	36.66	8195	82.50	5229	69.59		5412	
GRPO	93.17	941	86.80	3094	46.66	7987	40.00	8626	85.00	4981	**70.32**	+0.73%	**5125**	-5.2%
LC-R1_LP	82.10	121	86.40	2696	50.00	7951	40.00	7970	90.00	4631	**69.70**	+0.11%	**4673**	-13.6%
RL+LP	91.28	376	82.80	2416	46.66	7781	33.33	7586	90.00	4561	**68.81**	-0.77%	**4544**	-16.0%
SCMA (Ours)	93.02	588	86.20	2654	60.00	7473	43.33	7657	90.00	4521	**74.51**	+4.92%	**4578**	-15.4%
Qwen3-4B														
Vanilla	94.49	1320	86.60	4300	43.33	9805	30.00	9958	82.50	6943	67.38		6465	
GRPO	94.92	1001	87.20	3911	60.00	8680	40.00	9356	90.00	6441	**74.42**	+7.04%	**5877**	-9.0%
LC-R1_LP	93.70	378	88.2	2892	56.66	8096	36.66	7819	87.5	4032	**72.54**	+5.16%	**4643**	-28.1%
RL+LP	94.01	381	87.2	2249	53.33	7295	43.33	8040	87.5	4217	**73.07**	+5.69%	**4436**	-31.3%
SCMA (Ours)	94.16	351	88.00	1629	60.00	7100	43.33	7402	95.00	3242	**76.09**	+8.70%	**3944**	-39.0%
Qwen3-8B														
Vanilla	95.40	1888	85.80	4512	40.00	8944	33.33	9501	72.50	6928	65.40		6354	
GRPO	95.98	1342	88.40	4182	56.67	8789	46.66	9344	90.00	6844	**75.54**	+10.14%	**6100**	-3.99%
LC-R1_LP	95.98	447	88.80	2841	56.6	8304	40.00	7636	82.50	4937	**72.77**	+7.37%	**4833**	-23.9%
RL+LP	95.75	432	88.4	2539	53.33	7973	46.67	8425	90.00	3878	**74.83**	+9.43%	**4649**	-26.8%
SCMA (Ours)	94.99	369	89.20	1999	60.00	6475	43.33	7004	89.60	3599	**75.42**	+10.02%	**3889**	-38.8%

datasets and baselines are elaborated in Appendix F. Further supplementary experiments with additional baselines are available in Appendix G. We select DeepSeek-R1-Distill-Qwen (1.5B/7B) (DeepSeek-AI, 2025) and Qwen3 (4B/8B) (Team, 2025) series as base models. Our analysis focuses on accuracy (Acc), and response length (Tokens).

Implementation Details: We implement SCMA based on the open-source `verl` framework (Sheng et al., 2024), utilizing GRPO as the underlying optimization backbone for SCMA. To ensure experimental consistency, all models are fine-tuned using the GSM8K dataset, which contains 7,473 training samples. For further validation, we additionally employ the DeepScaleR-Preview dataset (Luo et al., 2025c) as a training set, which provided in Appendix G. Regarding hyperparameter configurations, we set the training context size to 11K, the batch size to 256, the α to 0.1, and the learning rate to 1×10^{-6}. For GRPO-specific settings, we configure the number of rollouts to 5 and the KL penalty coefficient to 0.001. During validation, we adopt a sampling decoding strategy with a temperature of 0.6 and a top-p value of 0.95. Answer verification are enabled by default in verl. Please refer to Appendix E for details. Note that final results are obtained by averaged across several steps.

4.2 Main results

Answer of Question 1: We evaluate SCMA against multiple baselines across five benchmark datasets: GSM8K, MATH500, AIME24, AIME25, and AMC23. As illustrated in Table 1, SCMA consistently achieves superior overall accuracy across models of varying parameter scales. Compared to the vanilla reasoning model, SCMA demonstrates advantages in both efficiency and performance, achieving an average reduction in thinking process ranging from 11.1% to 39.0% while simultaneously boosting accuracy by 4.33% to 10.02%.

Advantages on Length and Accuracy: Experimental results demonstrate that SCMA's efficacy amplifies with model scale, expanding from an 11.1% length reduction on DeepSeek-1.5B to nearly 39% on Qwen3-8B, where the MATH500 task notably exhibits a reduction exceeding 50% alongside an accuracy boost to 89.20%. This scalability, which allows larger models to achieve high compression rates while maintaining high accuracy, confirms SCMA's capacity to excise redundant reasoning chains and unleash the latent potential of high-performance models.

Cost-Performance Trade-Off: Generally, length penalties-based RL methods often struggle to balance efficiency and accuracy. For instance, while RL+LP reduces length by 16.0% on DeepSeek-7B, it causes accuracy to drop below the baseline (68.81% $<$ 69.59%). In contrast, SCMA achieves a superior trade-off: on Qwen3-4B, SCMA not only reduces length by 39.0% (significantly outperforming LC-R1_LP) but also attains the highest accuracy of 76.09%. This demonstrates that SCMA optimizes reasoning paths rather than merely truncating them, significantly enhancing logical density while eliminating redundancy.

Out-of-Distribution Generalization Although trained exclusively on GSM8K, SCMA exhibits exceptional generalization capabilities across the

remaining benchmarks. It not only significantly improves accuracy on unseen challenging problems (e.g., using DeepSeek-7B achieves 60.00% on AIME24, surpassing GRPO's 46.66%) but also maintains high efficiency across tasks. For example, in the MATH500 evaluation using Qwen3-4B, the response length plummets from 4,300 to 1,629 tokens (a reduction of over 62%). This confirms that SCMA does not overfit the training data; rather, it has genuinely mastered a generalizable and compact reasoning paradigm.

Inference Efficiency: SCMA functions strictly as a training-time auxiliary via shared parameters θ. Auxiliary agents are deactivated during deployment, leaving the Reasoning Agent to operate autonomously. This guarantees zero additional computational overhead, while the induced brevity further reduces total decoding latency. Please refer to Appendix I for a detailed analysis.

4.3 Hyperparameter Study and Stability

Answer of Question 2: Hyperparameter study please refer to Appendix L.

Analysis of Stability Please refer to Appendix J.

4.4 Ablation Studies

Answer of Question 3: To verify the necessity of the multi-agent RL optimization within the SCMA framework, we conducted ablation studies by freezing the Segmentation and Scoring agents. Specifically, we replaced the optimized internal modules with closed-source GPT-4o and Qwen models (1.7B, 4B, and 8B) invoked via API, while keeping the Reasoning agent (Qwen-4B) optimized.

Superiority of Cooperative Optimization. As shown in Table 2, SCMA achieves the highest overall accuracy of **73.38%**, significantly outperforming the unoptimized variants. When the Segmentation and Scoring agents are replaced with a frozen model of the same size (*w/o_optimization-4b*), the overall accuracy drops by **2.37%** to 71.01%, and token consumption increases by **4.62%**. This demonstrates that joint optimization aligns the agents' objectives more effectively than simply pipeline-chaining pre-trained models.

Beating Larger Models with Efficiency. Notably, SCMA even surpasses the variant utilizing a larger 8B model and GPT-4o model as the auxiliary agent (*w/o_optimization-8b*, *w/o_optimization-gpt4o*). For example, despite the 8B model's stronger intrinsic capabilities, the lack of cooperative tuning results in a lower overall accuracy (**73.21%**) and a significantly higher computational cost (**+6.09%** tokens). This indicates that the domain-specific collaboration established through our multi-agent optimization is more critical than the raw scale of the auxiliary models.

Impact of Segmentation and Scoring Quality. The performance gap widens further when smaller models are used. The *w/o_optimization-1.7b* variant exhibits the lowest performance (70.44% accuracy) and the highest inefficiency (**+11.94%** token usage). This decline underscores that the quality of segmentation and scoring directly bounds the reasoning capabilities. Without rational chunk segmentation and accurate importance scoring provided by the segmentation agent and the scoring agent, the Reasoning agent struggles to maintain high-quality reasoning chains.

4.5 Evolutionary Dynamics of SCMA

Answer of Question 4: By monitoring the evolution of key metrics and analyzing qualitative samples, as illustrated in Fig. 2, we observe a significant emergence of fine-grained compression capability as training steps increase:

Semantic Concentration & Redundancy Pruning As depicted in Fig. 2 (Top Left & Top Right), a distinct evolutionary pattern emerges between Step 0 and Step 100: the Average Chunk Num (blue line) declines sharply, while the Average Score (orange line) ascends synchronously towards convergence. This negative correlation implies a synergistic co-optimization between the Reasoning Agent ($\pi_\theta^{\text{reason}}$) and the Scoring Agent ($\pi_\theta^{\text{score}}$). Qualitatively, the model transitions away from the low-scoring, hesitant, and repetitive fragments typical of early training (e.g., the redundant *"...Wait, but is there something else..."* observed at Step 10), evolving instead to produce highly condensed reasoning kernels. We interpret this phenomenon through an information-theoretic lens as the model learning to maximize information entropy per unit fragment. By employing **Semantic Compression**, the model prunes redundancy to converge the reasoning process into critical steps with elevated **Semantic Density** and substantial logical contribution, simultaneously optimizing reasoning efficiency and quality.

Content-Adaptive Fine-grained Segmentation As illustrated in Fig. 2 (Bottom Left & Bottom Right), the trajectory of the Chunk Length Std (red line) offers critical insights into the evolution of the Segmentation Agent (π_θ^{seg}). Initially maintaining a low level, the standard deviation under-

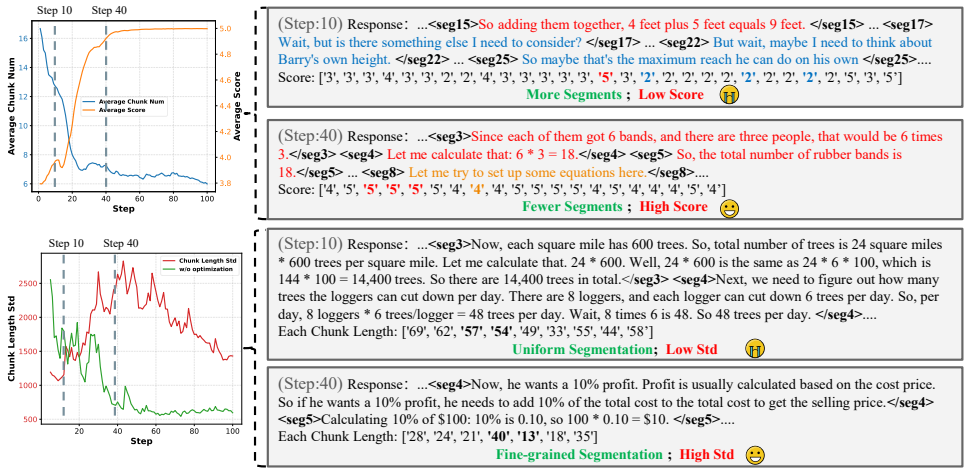


Figure 2: Evolution of the Scoring and Segmentation Agents. **(Left)** Curves showing the Average Score (orange), Average Chunk Num (blue), Chunk Length Std (red), the Chunk Length Std if Segmentation Agents are not optimized, for shot w/o optimization (green) over training steps. **(Right)** Case of agents at Step 10 versus Step 40, including segmentation points and score annotations.

Table 2: Ablation study on component optimization. Here, only the reasoning agent (Qwen-4B) is optimized, while the Segmentation and Scoring agents remain frozen and are implemented using off-the-shelf Qwen models via API.

Method	GSM8K		MATH500		AIME24		AIME25		AMC23		Overall	
	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens	Acc	Tokens
SCMA (Ours)	94.54	588	88.20	3360	50.00	8495	46.66	9016	87.50	5704	73.38	5432
Ablation Studies												
*w/o_optimization-**gpt4o***	93.40	702	87.20	3541	60.00	8418	43.33	9568	80.00	5830	**72.78**-0.59	**5611**+3.31%
*w/o_optimization-**8b***	92.83	798	86.6	3636	53.33	8989	43.33	9216	90.00	6176	**73.21**-0.17	**5763**+6.09%
*w/o_optimization-**4b***	94.76	855	87.80	3675	53.33	8802	36.66	9335	82.50	5752	**71.01**-2.37	**5683**+4.62%
*w/o_optimization-**1.7b***	94.76	984	85.80	4044	50.00	8873	36.66	10007	85.00	6497	**70.44**-2.94	**6081**+11.94%

goes a sharp ascent after Step 20. This statistical trend signifies a behavioral shift: *a transition from "uniform segmentation" to "fine-grained segmentation"*. Specifically, the low standard deviation observed in the early phase (e.g., Step 10) implies that the model tends to segment text into equal-length chunks, remaining agnostic to the underlying semantic importance. Conversely, the high variance at Step 40 indicates that the model has acquired the capability to dynamically allocate chunk length based on the cognitive load of the reasoning content. As corroborated by qualitative samples, the agent learns to allocate extended text blocks to encapsulate complete logical flows (such as complex numerical calculations or derivations), while assigning concise segments to simple transitions or definitions. This behavior demonstrates the model's capacity for **Content-Adaptive Fine-grained Segmentation**, regulating information flow based on semantic density rather than uniform chunking.

Qualitative Evolution Summary A comparative analysis of the textual outputs between Step 10 and Step 40 (Fig. 2, Right) delineates the clear evolutionary trajectory: **(Step 10)** The reasoning path is verbose and dominated by low-confidence (Low Score) exploratory phrases. The segmentation granularity appears rigid and indiscriminate, lacking semantic focus. **(Step 40)** The total number of chunks (n) decreases, while the importance score (w_i) assigned to each segment (s_i) increases significantly. This shift indicates that the model has learned to precisely identify logical units, achieving efficient compression while rigorously preserving the structural integrity of the reasoning chain.

5 Conclusion

This work addresses the issue of redundant reasoning chains in LRMs. Distinct from existing single-agent Reinforcement Learning methods that rely on coarse-grained length penalties, we construct a Multi-Agent Reinforcement Learning framework utilizing fine-grained reward shaping. Empirical evaluations confirm that our framework significantly shortens thinking process while maintaining reasoning performance. Furthermore, qualitative analysis delineates the clear evolutionary trajectory of the model's compression capabilities, validating the synergy inherent in the multi-agent system. Future work will scale this paradigm towards large-scale, heterogeneous systems, thereby empowering the resolution of complex settings.

Limitations

While SCMA successfully demonstrates that multi-agent collaborative optimization can significantly reduce the redundant reasoning of LRMs while boosting accuracy, we acknowledge a primary limitation regarding training cost. The multi-agent training paradigm (i.e., sequentially triggering the Reasoning, Segmentation, and Scoring agents) inevitably introduces longer training times compared to monolithic RL approaches. However, it is worth noting that this overhead remains strictly within a controllable range, and the framework demonstrates highly competitive sample efficiency. Crucially, the multi-agent system serves merely as a training-only auxiliary mechanism. Once deployed, the Segmentation and Scoring agents are safely discarded, ensuring that SCMA introduces zero extra computational overhead or latency during the inference phase.

Ethics Statement

This paper proposes a novel multi-agent reinforcement learning framework to enhance the inference efficiency of Large Reasoning Models, which simultaneously mitigates redundant reasoning and improves accuracy. The experiments employ widely adopted mathematical and reasoning benchmark datasets in the research community that, to our knowledge, involve no privacy concerns or ethical controversies. The experiments are conducted on open-source large language models (such as the Qwen and DeepSeek series). It should be emphasized that this work represents fundamental research focused exclusively on advancing reinforcement learning training paradigms and deployment optimization for LLMs, rather than developing specific end-user applications.

References

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

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

Ching Chang, Yidan Shi, Defu Cao, Wei Yang, Jeehyun Hwang, Haixin Wang, Jiacheng Pang, Wei Wang, Yan Liu, Wen-Chih Peng, and 1 others. 2025. A survey of reasoning and agentic systems in time series with large language models. *arXiv preprint arXiv:2509.11575*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Yiqun Chen, Erhan Zhang, Tianyi Hu, Shijie Wang, Zixuan Yang, Meizhi Zhong, Xiaochi Wei, Yan Gao, Yi Wu, Yao Hu, and 1 others. 2026. Jade: Bridging the strategic-operational gap in dynamic agentic rag. *arXiv preprint arXiv:2601.21916*.

Yiqun Chen, Erhan Zhang, Lingyong Yan, Shuaiqiang Wang, Jizhou Huang, Dawei Yin, and Jiaxin Mao. 2025b. Mao-arag: Multi-agent orchestration for adaptive retrieval-augmented generation. *arXiv preprint arXiv:2508.01005*.

Zhengxiang Cheng, Dongping Chen, Mingyang Fu, and Tianyi Zhou. 2025. Optimizing length compression in large reasoning models. *arXiv preprint arXiv:2506.14755*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, and 1 others. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*.

Muzhi Dai, Chenxu Yang, and Qingyi Si. 2025. S-grpo: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jinyuan Feng, Min Chen, Zhiqiang Pu, Yifan Xu, and Yanyan Liang. 2025. Ma2rl: Masked autoencoders for generalizable multi-agent reinforcement learning. *arXiv preprint arXiv:2502.17046*.

Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. 2026. Dr. mas: Stable reinforcement learning for multi-agent llm systems. *arXiv preprint arXiv:2602.08847*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

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*.

Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang, Tengchao Lv, Lei Cui, and Furu Wei. 2025. Think only when you need with large hybrid-reasoning models. *arXiv preprint arXiv:2505.14631*.

Stephen Cole Kleene. 1956. *Representation of events in nerve nets and finite automata*, volume 34. Princeton University Press Princeton.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and 1 others. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9.

Baohao Liao, Hanze Dong, Yuhui Xu, Doyen Sahoo, Christof Monz, Junnan Li, and Caiming Xiong. 2025. Fractured chain-of-thought reasoning. *arXiv preprint arXiv:2505.12992*.

Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, and 1 others. 2025a. Dler: Doing length penalty right-incentivizing more intelligence per token via reinforcement learning. *arXiv preprint arXiv:2510.15110*.

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

Xin Liu and Lu Wang. 2025. Answer convergence as a signal for early stopping in reasoning. *arXiv preprint arXiv:2506.02536*.

Haotian Luo, Haiying He, Yibo Wang, Jinluan Yang, Rui Liu, Naiqiang Tan, Xiaochun Cao, Dacheng Tao, and Li Shen. 2025a. Adar1: From long-cot to hybrid-cot via bi-level adaptive reasoning optimization. *arXiv e-prints*, pages arXiv–2504.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025b. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. 2025c. 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-S Notion Blog.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie

Zhou, Ju Ren, and Yaoxue Zhang. 2025. Concise: Confidence-guided compression in step-by-step efficient reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8021–8040.

Shyam Sundhar Ramesh, Yifan Hu, Iason Chaimalas, Viraj Mehta, Pier Giuseppe Sessa, Haitham Bou Ammar, and Ilija Bogunovic. 2024. Group robust preference optimization in reward-free rlhf. *Advances in Neural Information Processing Systems*, 37:37100–37137.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 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, Zhaoxiang Liu, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 2322–2331.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Qwen Team and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3).

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*.

Haoming Wen, Yushi Bai, Juanzi Li, and Jie Tang. 2025. Siri: Scaling iterative reinforcement learning with interleaved compression. *arXiv preprint arXiv:2509.25176*.

Wei Wu, Liyi Chen, Congxi Xiao, Tianfu Wang, Qimeng Wang, Chengqiang Lu, Yan Gao, Yi Wu, Yao Hu, and Hui Xiong. 2026. Anti-length shift: Dynamic outlier truncation for training efficient reasoning models. *arXiv preprint arXiv:2601.03969*.

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

Wei Yang, Jiacheng Pang, Shixuan Li, Paul Bogdan, Stephen Tu, and Jesse Thomason. 2025a. Maestro: Learning to collaborate via conditional listwise policy optimization for multi-agent llms. *arXiv preprint arXiv:2511.06134*.

Wei Yang and Jesse Thomason. 2025. Learning to deliberate: Meta-policy collaboration for agentic llms with multi-agent reinforcement learning. *arXiv preprint arXiv:2509.03817*.

Wei Yang, Muyan Weng, Jiacheng Pang, Defu Cao, Heng Ping, Peiyu Zhang, Shixuan Li, Yue Zhao, Qiang Yang, Mengdi Wang, and 1 others. 2025b. Toward evolutionary intelligence: Llm-based agentic systems with multi-agent reinforcement learning. *Available at SSRN 5819182*.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624.

Linan Yue, Yichao Du, Yizhi Wang, Weibo Gao, Fangzhou Yao, Li Wang, Ye Liu, Ziyu Xu, Qi Liu, Shimin Di, and 1 others. 2025. Don't overthink it: A survey of efficient r1-style large reasoning models. *arXiv preprint arXiv:2508.02120*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Zihao Zeng, Xuyao Huang, Boxiu Li, Hao Zhang, and Zhijie Deng. 2025. Done is better than perfect: Unlocking efficient reasoning by structured multi-turn decomposition. *arXiv preprint arXiv:2505.19788*.

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

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

Shengjia Zhang, Junjie Wu, Jiawei Chen, Changwang Zhang, Xingyu Lou, Wangchunshu Zhou, Sheng Zhou, Can Wang, and Jun Wang. 2025b. Othink-r1: Intrinsic fast/slow thinking mode switching for over-reasoning mitigation. *arXiv preprint arXiv:2506.02397*.

Yujie Zhao, Lanxiang Hu, Yang Wang, Minmin Hou, Hao Zhang, Ke Ding, and Jishen Zhao. 2025. Stronger-mas: Multi-agent reinforcement learning for collaborative llms. *arXiv preprint arXiv:2510.11062*.

Jason Zhu and Hongyu Li. 2025. Towards concise and adaptive thinking in large reasoning models: A survey. *arXiv preprint arXiv:2507.09662*.

A Related Work

Efficient Reasoning in LRMs Large Reasoning Models (LRMs) often depend on lengthy chains of thought (CoT) to achieve strong performance. However, such exhaustive reasoning can be unnecessary for low- to medium-complexity tasks, leading to overthinking and substantial computational overhead (Chen et al., 2024). To address this issue, recent studies propose early-exit mechanisms, adaptive inference, and dynamic prompting.

Early-exit methods aim to terminate generation once sufficient confidence is reached, leveraging signals such as internal state transitions or uncertainty thresholds (Liao et al., 2025; Team et al., 2024; Qiao et al., 2025), or by assessing the consistency between intermediate reasoning steps and candidate answers (Liu and Wang, 2025). Other works adopt implicit stopping strategies without explicit termination criteria (Dai et al., 2025). While effective and often training-free, these approaches rely heavily on heuristic rules and exhibit limited generalization. In contrast, adaptive inference methods dynamically adjust reasoning depth or patterns to better match task complexity. Representative strategies include reward-guided control of reasoning length (Jiang et al., 2025; Luo et al., 2025a) and dynamic switching between reasoning modes (Zhang et al., 2025b,a; Tu et al., 2025). Similarly, dynamic prompting reduces redundant reasoning through inference-time prompt engineering (Han et al., 2025; Muennighoff et al., 2025; Xu et al., 2025; Ma et al., 2025). Nevertheless, these approaches typically depend on predefined priors over task difficulty or output length, making it challenging to learn a unified and intrinsic mechanism for regulating inference behavior.

RL-based Post-Training for Efficient Reasoning With Reinforcement Learning with Verifiable Rewards (RLVR) emerging as a dominant paradigm for post-training LRMs (Yang et al., 2025a; Chang et al., 2025; Yang and Thomason, 2025; Yang et al., 2025b), it has been observed that optimizing solely for final answer correctness often induces excessively long chains of thought, exacerbating overthinking and increasing inference cost (Yue et al., 2025; Zhu and Li, 2025). Consequently, recent studies have explored explicitly incorporating efficiency considerations into the RL reward function to mitigate redundant reasoning while preserving accuracy.

One line of work introduces length-based penalties or constraints to directly account for reasoning cost. For instance, (Cheng et al., 2025; Arora and Zanette, 2025; Aggarwal and Welleck, 2025; Cui et al., 2025) penalize the output length during RL training, encouraging the generation of more concise reasoning traces and balancing accuracy with computational efficiency. O1-Pruner (Luo et al., 2025b), ThinkPrune (Hou et al., 2025) and Kimi (Team et al., 2025) further propose pruning-oriented RL strategies, applying stronger negative rewards to long or inefficient trajectories to guide the model toward early termination or compression of reasoning steps when appropriate. Although these methods have demonstrated empirical success, they typically treat efficiency coarsely as total sequence length and operate mainly over parallel chains, offering limited control over stepwise reasoning within a single chain. This can result in reduced accuracy when compressing reasoning chains or suppressing redundant steps, highlighting the need for RL approaches that more finely penalize unnecessary reasoning while preserving correctness.

B On the Rationale and Stability of Parameter Sharing

As introduced in Section 3.1, our SCMA framework instantiates three functionally distinct agents that share the identical parameters θ of a single base LLM π_{base}, differentiated solely by their specialized prompts. This design choice is strictly motivated by theoretical foundations, recent empirical consensus in LLM-based multi-agent systems, and significant efficiency advantages.

1. Historical Paradigm in MARL and Prior Validations. Parameter sharing is deeply rooted in classic Multi-Agent Reinforcement Learning (MARL). Historically, prominent algorithms such

as QMIX (Rashid et al., 2020) and MAPPO (Yu et al., 2022) have demonstrated that sharing parameters among agents is a fundamental and stable paradigm, highly effective in facilitating cooperative behaviors and accelerating convergence. Extending this philosophy to LLM-based agents, maintaining shared weights while distinguishing agent roles exclusively through prompt variations has been heavily utilized and empirically validated in our prior works, including MMOA-RAG (Chen et al., 2025b) and JADE (Chen et al., 2026), which proves the robust architectural soundness of this approach.

2. Theoretical Consistency in the Era of LLMs. Recent advances in multi-agent LLM systems, notably *Stronger-MAS* (Zhao et al., 2025) and *Dr. MAS* (Feng et al., 2026), have explicitly investigated the impact of shared versus unshared parameters. Both studies reached a consistent conclusion: parameter sharing yields performance remarkably on par with unshared (independent parameter) settings. The underlying rationale is that LLMs inherently possess strong zero-shot and multi-task capabilities. The model contains sufficient capacity to activate distinct reasoning pathways and functional behaviors when conditioned on role-specific prompts, rendering the maintenance of independent model copies functionally redundant.

3. Substantial Reductions in Computational Cost. Beyond theoretical alignment, parameter sharing offers overwhelming practical benefits. By maintaining only a single copy of the model parameters θ in GPU memory, we drastically reduce the training overhead (e.g., optimizer states, gradients, and intermediate activations). Furthermore, it minimizes real-world deployment costs, aligning perfectly with our objective of exclusive inference deployment for the Reasoning Agent without incurring extra multi-agent computational burden.

C Prompt Designs for SCMA

In this section, we provide the detailed prompt templates used for the three agents defined in our framework: the Reasoning Agent (P_{reason}), the Segmentation Agent (P_{seg}), and the Scoring Agent (P_{score}).

C.1 Reasoning Agent Prompt (P_{reason})

The Reasoning Agent generates the initial Chain-of-Thought solution path y given the input problem x. We utilize a standard zero-shot Chain-of-Thought prompt to encourage step-by-step reasoning.

Reasoning Prompt (P_{reason})

```
Let's think step by step and output the
final answer within \boxed{}.
```

C.2 Segmentation Agent Prompt (P_{seg})

The Segmentation Agent takes the generated reasoning path y and segments it into minimal reasoning units. The prompt focuses on preserving the semantic completeness of each unit without omitting any original text.

Segmentation Prompt (P_{seg})

```
You will be given a passage of text. Please
do the following:
1. Segment the text into "minimal reasoning
units.": A "minimal reasoning unit" is
the smallest chunk that is logically
or semantically complete enough to be
understood or evaluated on its own (it can
be a sentence, a clause, or a short phrase,
but do not break the text so much that you
destroy natural semantics).
Your segmentation must cover the entire
original text, without omissions and
without changing the order.
```

C.3 Scoring Agent Prompt (P_{score})

The Scoring Agent evaluates the importance of each segmented unit derived from the previous step. It assigns a score on a scale of 1 to 5 and formats the final output with specific tags.

Scoring Prompt (P_{score})

```
For each segmented unit, assign an
importance score based on how important
that unit is for the overall reasoning
process, using a 1-5 scale: 1: Almost not
important; has negligible impact on the
overall reasoning 2: Contributes something,
but below average importance 3: Average
level of contribution 4: Above-average
level of contribution 5: Very important;
plays a key or central role in the overall
reasoning
Output format requirements:
You must fully reproduce the original
text in your output, but divided into
the segmented units, each wrapped
with the following tags: For the
1st unit: <seg1>content of the first
segment</seg1><score1>score</score1> For
the 2nd unit: <seg2>content of the second
segment</seg2><score2>score</score2>
And so on, until the nth
unit: <segn>content of the n-th
segment</segn><scoren>score</scoren>
The concatenation of all <seg*> contents
(in order) must reconstruct the original
text exactly in terms of wording and order.
```

```
Minor differences in whitespace or line
breaks are acceptable, but you must not
add, delete, or paraphrase any words.
Output only the tagged structure described
above.  Do not include any explanations,
comments, or extra text.
Here is the text to process: [Input Text]
```

D Theoretical Analysis: Formulation as Constrained Optimization

Instead of viewing the reward function merely as a heuristic length penalty, we formulate our task as a **Constrained Markov Decision Process (CMDP)**. Our objective is to maximize the accuracy R_{acc} subject to a constraint on the "weighted length cost," where the cost is inversely related to the importance score w_i.

D.1 Primal Problem Definition

Let $C(y)$ denote the weighted length cost of a generated sequence y. We define this cost based on the premise that tokens with higher importance (w_i) should incur a lower penalty. The optimization problem is defined as:

$$\begin{aligned} \max_{\theta} \quad & \mathcal{J}(\theta) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \left[R_{\text{acc}}(y|x) \right] \\ \text{s.t.} \quad & \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \left[C(y) \right] \leq \beta \end{aligned} \tag{13}$$

where β is a pre-defined budget for redundancy. Consistent with our reward design, the weighted cost $C(y)$ is explicitly defined as:

$$C(y) = \frac{\sum_{i=1}^{n} (5 - w_i) \cdot |s_i|}{L_{\text{norm}}} \tag{14}$$

Here, $L_{\text{norm}} = \max_{j \in \mathcal{C}} \left(\sum_{m=1}^{n_j} |s_{j,m}| \right)$ acts as a normalization factor to scale the cost.

D.2 Lagrangian Relaxation

To solve this constrained problem, we employ the method of Lagrange multipliers. We introduce a Lagrange multiplier $\lambda \geq 0$ to incorporate the constraint into the objective function, converting the primal problem into the following unconstrained dual problem:

$$\min_{\lambda \geq 0} \max_{\theta} \mathcal{L}(\theta, \lambda) = \mathbb{E}_{y \sim \pi_\theta} \left[R_{\text{acc}}(y|x) - \lambda \cdot (C(y) - \beta) \right] \tag{15}$$

By fixing λ as a hyperparameter and rearranging the terms, the equivalent maximization objective for the policy π_θ becomes:

$$\max_{\theta} \mathbb{E}_{y \sim \pi_\theta} \left[R_{\text{acc}}(y|x) - \lambda \cdot C(y) + \lambda\beta \right] \tag{16}$$

D.3 Connection to Proposed Reward

We now demonstrate that our proposed reward function in Eq. 12 is a specific instantiation of the Lagrangian objective derived above. Recall our reward function design (for $y \in \mathcal{C}$):

$$R(y|x) = R_{\text{acc}}(y|x) + \alpha \left(1 - \underbrace{\frac{\sum_{i=1}^{n} (5 - w_i) \cdot |s_i|}{L_{\text{norm}}}}_{C(y)} \right) \tag{17}$$

Expanding this equation yields:

$$R(y|x) = \underbrace{R_{\text{acc}}(y|x)}_{\text{Accuracy}} - \underbrace{\alpha}_{\lambda} \cdot \underbrace{C(y)}_{\text{Weighted Cost}} + \underbrace{\alpha}_{\text{Constant Bias } (\lambda\beta)} \tag{18}$$

By comparing Eq. 18 with the Lagrangian objective in Eq. 16, we establish the following theoretical equivalences:

- The hyperparameter α functions as the **Lagrange multiplier** λ, controlling the trade-off between accuracy and the information density constraint.
- The term $(5 - w_i)$ in $C(y)$ implements the weighted constraint, where preserving high-importance information ($w_i \uparrow$) incurs a lower cost ($C(y) \downarrow$).
- The constant term $+\alpha$ corresponds to $\lambda\beta$, serving as a reward baseline that reduces variance during training without altering the optimal policy direction.

E Detailed Training Configurations

We conduct our training experiments on a single compute node equipped with 8×NVIDIA H800 GPUs. Specifically, when fine-tuning the 7B base model (e.g., DeepSeek-R1-Distill-Qwen-7B) on the DeepScaleR-Preview dataset for 2 epochs, the typical wall-clock training time is approximately 6 days.

Our implementation is built upon the open-source `verl` framework. To optimize memory footprint and throughput during the multi-agent generation phase, we utilize vLLM for rollout generation with a tensor parallel (TP) size of 2 and a GPU memory utilization limit of 0.6. Furthermore, we enable gradient checkpointing and FSDP parameter offloading for the reference model to accommodate ultra-long context training.

For the data configuration, we set the maximum prompt length to 4,000 tokens and the maximum response length to 11,000 tokens, which accommodates the extensive reasoning chains required for complex mathematical tasks. Any prompts exceeding this length are strictly filtered out. For the GRPO optimization, the group size (number of rollouts, G) is set to 8. We utilize a low-variance KL penalty with a coefficient of 0.001 to prevent the model from deviating excessively from the reference policy. The core hyperparameters used for our MARL training are summarized in Table 3.

Table 3: Core hyperparameters for SCMA training.

Hyperparameter	**Value**
Algorithm & Optimization	
Algorithm	GRPO
Actor Learning Rate	1×10^{-6}
Total Epochs	2
Global Batch Size	128
PPO Mini-Batch Size	16
Micro-Batch Size per GPU	2
Rollout & GRPO Settings	
Number of Rollouts (G)	8
KL Penalty Coefficient (β)	0.001
KL Loss Type	Low Variance
Tensor Parallel Size (vLLM)	2
Context & Generation	
Max Prompt Length	4,000
Max Response Length	11,000
Validation Temperature	0.6
Validation Top-p	0.95

F Evaluation datasets and baselines

Our implementation relies on the open-source `verl` training framework, which is distributed under the Apache-2.0 license. The base models utilized in our research, including the Qwen and DeepSeek series, strictly adhere to their respective open-source licenses, ensuring full compliance with reproducible and ethical research standards.

To evaluate the mathematical reasoning capabilities of our framework across a wide spectrum of difficulty levels that range from foundational arithmetic to elite competition mathematics, we utilize five representative benchmarks:

- **GSM8K** consists of several thousand grade-school math word problems that require multi-step reasoning and the application of basic arithmetic operations.
- **MATH500** serves as a curated subset of the MATH dataset, encompassing high-school level challenges across diverse domains such as algebra, geometry, and probability.
- **AMC23** features problems from the 2023 American Mathematics Competitions, which demand both solid subject knowledge and a high degree of logical flexibility.
- **AIME24** is derived from the 2024 American Invitational Mathematics Examination and represents a substantial leap in difficulty to test the model's precision in handling long-chain mathematical deductions.
- **AIME25** includes the most recent 2025 AIME problems, serving as a frontier benchmark to assess the performance limits on highly sophisticated and cognitively demanding mathematical tasks.

To evaluate the effectiveness of our approach, we compare it against several RL-based baselines that focus on reasoning efficiency and reward optimization:

- **GRPO** (Shao et al., 2024): Group Relative Policy Optimization is a reinforcement learning framework that eliminates the necessity of a critic model by estimating advantages through reward normalization across a set of responses generated from the same prompt.
- **LC-R1_LP**: This baseline stems from the LC-R1 (Cheng et al., 2025) framework, which originally integrates a Length Reward for conciseness and a Compress Reward for removing invalid thinking processes. In our experimental setup, we adapt this framework by excluding the LC-Extractor and exclusively employing its length penalty function.
- **RL+LP** (Arora and Zanette, 2025): This configuration implements a length penalty mechanism that assigns reward values based on the deviation of each correct response's length from the group mean. This approach specifically penalizes excessive verbosity by reducing the rewards for correct but unnecessarily long responses, thereby encouraging the model to generate more efficient reasoning chains.

- **DeepScaleR-Preview** (Luo et al., 2025c): A robust, unconstrained baseline that boosts reasoning performance by progressively expanding the context window (scaling from 8K up to 24K tokens) throughout the reinforcement learning phase.

- **OverThink** (Chen et al., 2021): A data-driven methodology designed to mitigate redundant reasoning. By utilizing efficiency metrics, it filters out unnecessary cognitive steps—such as excessive post-answer verification—to curate highly concise datasets for supervised fine-tuning.

- **DAST** (Shen et al., 2025): An adaptive framework that modulates rewards based on problem difficulty through a Token Length Budget (TLB). During RL, it heavily penalizes overly verbose answers for straightforward questions, while permitting extended reasoning trajectories for complex tasks.

- **O1-Pruner** (Luo et al., 2025b): An off-policy technique utilizing a length-harmonizing reward mechanism. It reduces inference overhead by matching the student model's output length distribution with that of a more concise reference model, while simultaneously penalizing any drops in accuracy.

- **Laser-DE** (Liu et al., 2025b): A dynamic length-based reward shaping approach. Its Dynamic & Exploration (DE) configuration fosters exploration by loosening length penalties on incorrect answers, whereas it enforces strict conciseness on correct solutions.

- **AdaptThink** (Zhang et al., 2025a): A dynamic mode-switching strategy that enables the model to assess query complexity. Based on this assessment, it explicitly chooses whether to engage in an extended "Thinking" process (Chain-of-Thought) or to provide a "NoThinking" direct response.

- **DLER-R1** (Liu et al., 2025a): An optimization strategy that improves upon basic truncation penalties. To prevent reward collapse during RL, it incorporates dynamic sampling techniques alongside batch-wise normalization of rewards.

- **SIRI** (Wen et al., 2025): A phased training schedule that systematically alternates between response compression and expansion stages.

G Additional Results on DeepScaleR-Preview dataset

In this section, we present the main empirical results of our proposed SMCA. Following the configurations detailed in Section 4.1, we utilize the DeepScaleR-Preview dataset (Luo et al., 2025c) for further training and validation. Specifically, the model is trained for a total of 2 epochs based on the *DeepSeek-R1-Distill-Qwen-7B* architecture. To comprehensively evaluate the effectiveness of our approach, we benchmark SMCA against a variety of additional state-of-the-art baselines.

The overall performance on four standard mathematical reasoning benchmarks (MATH-500, AMC-23, AIME-24, and AIME-25) is summarized in Table 4. As demonstrated in the results, SMCA achieves highly competitive accuracy while maintaining significantly shorter generation lengths. For instance, on the AMC-23 benchmark, our method achieves the highest accuracy (87.5%) with the shortest response length (2073 tokens). Similarly, on AIME-24 and AIME-25, SMCA delivers strong accuracy scores (56.66% and 40.00%, respectively) while reducing the generation length by more than half compared to the original model and the DAPO-DeepScaleR baseline. These results indicate that SMCA successfully mitigates the verbosity issue of reasoning models without compromising their problem-solving capabilities.

H The pseudocode of SCMA

The pseudocode for SCMA is provided in Algorithm 1.

I Discussion on Computational Overhead

A common concern regarding Multi-Agent Reinforcement Learning (MARL) for Large Language Models is the potential explosion of computational overhead, particularly during the rollout generation phase. In this section, we explicitly address the training and inference costs of our proposed SCMA framework, demonstrating that the training overhead is highly optimized and manageable, while the inference overhead is strictly zero.

Table 4: Performance comparison on mathematical reasoning benchmarks based on the *DeepSeek-R1-Distill-Qwen-7B* base model. Our proposed method, SMCA, is trained on the DeepScaleR-Preview dataset (Luo et al., 2025c) for 2 epochs. Some baseline results are cited from (Wu et al., 2026) and (Zhang et al., 2025a). For the accuracy (Acc) metric, the highest and second-highest values are denoted in **bold** and underlined, respectively. For the response length (Length), the shortest and second-shortest values are denoted in **bold** and underlined, respectively.

Method	MATH-500		AMC-23		AIME-24		AIME-25	
	Acc	Length	Acc	Length	Acc	Length	Acc	Length
DeepSeek-R1-Distill-Qwen-7B								
DAPO-DeepScaleR	92.5	3658	84.5	6508	**57.6**	9983	40.8	10705
OverThink (Chen et al., 2021)	89.4	2435	—	—	53.1	8744	—	—
DAST (Shen et al., 2025)	89.6	2162	—	—	45.6	7578	—	—
O1-Pruner (Luo et al., 2025b)	86.6	2534	—	—	49.2	9719	—	—
LC-R1 (Cheng et al., 2025)	90.1	1536	79.2	3765	48.5	7580	35.6	7984
Laser-DE-L4096 (Liu et al., 2025b)	92.6	1883	83.0	3381	53.5	5890	37.4	6324
AdaptThink (Zhang et al., 2025a)	91.0	2008	81.5	5177	55.2	10393	38.3	11723
DLER-R1 (Liu et al., 2025a)	92.4	**1438**	83.5	2262	50.6	**3241**	33.6	**3357**
SIRI (Wen et al., 2025)	**93.5**	2452	85.8	4015	56.1	6122	**41.5**	6386
SMCA (Ours)	87.4	1536	**87.5**	**2073**	56.66	4695	40.00	5662

Controllable Training Overhead via Implementation Optimization. In a naive MARL implementation, simulating interactions among multiple agents would require numerous sequential inference passes, severely bottlenecking the training throughput. However, from a system implementation perspective, we highly optimize this process.

Specifically, while our framework involves multiple agents, we mathematically and structurally concatenate the prompts and action spaces of the two auxiliary agents. This system-level optimization allows the two auxiliary agents to be merged into a single generation pass. Consequently, compared to standard single-agent reasoning training (e.g., standard GRPO), our SCMA setup only introduces **one additional inference step** during the rollout phase.

To quantify this, we compare the computational burden of SCMA against a standard single-agent GRPO baseline in Table 5. Both experiments are conducted on a single compute node equipped with 8×NVIDIA H800 GPUs. As shown in the table, standard GRPO takes approximately 2 days and 15 hours to complete. Thanks to our agent-merging optimization, the multi-agent generation does not cause a combinatorial explosion in latency. Instead, the total wall-clock time is strictly bounded to an increase of roughly 2.16× (totaling 5 days and 17 hours). Given the substantial reasoning capability improvements achieved, this bounded training overhead on a single 8-GPU node is highly acceptable and practical.

Zero-Overhead at Inference Time. Crucially, the multi-agent formulation in our framework serves *exclusively* as a training-time exploration and regularization mechanism. Once the training phase is complete, the auxiliary agents are entirely discarded.

During deployment and test-time inference, the primary reasoning model operates in a standard, standalone autoregressive manner, structurally identical to any conventional reasoning model (e.g., DeepSeek-R1). The model does not need to simulate multi-agent interactions or generate auxiliary responses to solve test-time problems. Therefore, our proposed method brings **absolutely zero additional overhead** during inference.

Summary. In conclusion, the proposed SCMA framework addresses the common efficiency concerns in MARL. **(1) The training overhead is controllable:** by merging agent outputs, the rollout only requires one extra inference pass. This restricts the total training time penalty to merely ∼2.16×, which is efficiently completed within 6 days on a single compute node. **(2) The inference overhead is strictly zero:** the resulting model is deployed exactly like a standard reasoning LLM, retaining all performance gains without any deployment penalties.

J Analysis of SCMA Stability and Dynamic Penalty Efficacy

To investigate the training dynamics and stability of SCMA compared to length penalties-based RL

Algorithm 1 SCMA: Multi-Agent Training with GRPO

Input: Initial shared parameters θ, dataset $\mathcal{D}$, group size G, role prompts $P_{\text{reason}}, P_{\text{seg}}, P_{\text{score}}$
Output: Optimized shared parameters θ (instantiating $\mathcal{G} = \{\pi_\theta^{\text{reason}}, \pi_\theta^{\text{seg}}, \pi_\theta^{\text{score}}\}$)

for step = $1, 2, \ldots, M$ **do**
  Sample a batch of questions $\mathcal{D}_b$ from $\mathcal{D}$
  % Phase 1: Multi-Agent Rollout
  for each question $x \in \mathcal{D}_b$ **do**
    for $k = 1, \ldots, G$ **do**
      1. Reasoning: $y_k \sim \pi_\theta^{\text{reason}}(\cdot \mid x, P_{\text{reason}})$ (Eq. 4)
      2. Segmentation: $S_k \sim \pi_\theta^{\text{seg}}(\cdot \mid y_k, P_{\text{seg}})$, where $S_k = \{s_{k,1}, \ldots, s_{k,n}\}$ (Eq. 5)
      3. Scoring: $W_k \sim \pi_\theta^{\text{score}}(\cdot \mid S_k, P_{\text{score}})$, where $W_k = \{w_{k,1}, \ldots, w_{k,n}\}$ (Eq. 6)
      Construct joint trajectory $\tau_k = (y_k, S_k, W_k)$ (Eq. 8)
      Compute global reward $R_k = R_{\text{acc}}(y_k|x) + R_{\text{penalty}}(S_k, W_k) + R_{\text{fmt}}$ (Eq. 12)
    end for
    % Phase 2: Group Advantage Estimation
    for $k = 1, \ldots, G$ **do**
      Compute shared group advantage: $\hat{A}_k = \frac{R_k - \text{mean}(\{R_1,\ldots,R_G\})}{\text{std}(\{R_1,\ldots,R_G\})}$
      Distribute $\hat{A}_k$ to the reasoning, segmentation, and scoring actions within τ_k
    end for
  end for
  % Phase 3: Multi-Agent Optimization
  Compute the joint GRPO objective $J(\theta) = J_{\text{GRPO}}^{\text{reason}} + J_{\text{GRPO}}^{\text{seg}} + J_{\text{GRPO}}^{\text{score}}$ using distributed $\hat{A}_k$
  Update shared parameters $\theta \leftarrow \theta + \eta \nabla_\theta J(\theta)$ by maximizing the joint objective
end for
return $\mathcal{G} = \{\pi_\theta^{\text{reason}}, \pi_\theta^{\text{seg}}, \pi_\theta^{\text{score}}\}$

Table 5: Comparison of training overhead between standard single-agent GRPO and our proposed SCMA framework. The wall-clock time is measured for 2 training epochs on a single 8×NVIDIA H800 node.

Metric	Standard GRPO	SCMA (Ours)
Inference Passes per Rollout	1	2 (Optimized via Agent Merging)
Relative Wall-Clock Time	1.00× (2d 15h 39m)	∼ 2.16× (5d 17h 42m)
Extra Inference Overhead (Latency)	**-**	**0 (Auxiliary Agents Discarded)**

methods, we visualize the training curves of both methods in Fig. 3. As observed, RL+LP is prone to training collapse during the mid-to-late stages, where the agent aggressively compresses response length to maximize rewards once accuracy converges, inadvertently driving the model into a "No Think" pattern. In contrast, our SCMA achieves a superior trade-off between accuracy and conciseness by employing a scoring agent that selectively penalizes only redundant reasoning while strictly preserving core logic.

As illustrated in Fig. 4, the importance-weighted length penalty demonstrates an autonomous dynamic adjustment process that significantly influences model convergence. The training process can be analyzed in two distinct phases to elucidate the underlying mechanism of this self-adaptive regulation:

- **The initial phase**, spanning from step 0 to step 30, is characterized by a high-penalty regime that drives rapid length compression. During this period, the model frequently generates responses containing redundant or low-quality semantic chunks. As indicated by the grey dashed line, the length penalty remains at an elevated level above 0.20 and reaches a local maximum around step 15. This behavior occurs because the importance-weighted length penalty identifies that these

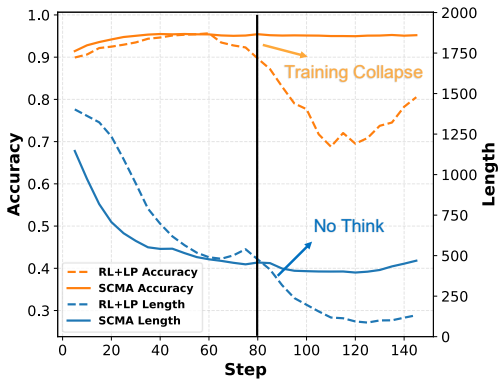


Figure 3: Training curves of SCMA and RL+LP. During training, the RL+LP model suffers from training collapse, with response length dropping significantly, indicating a "No Think" pattern.

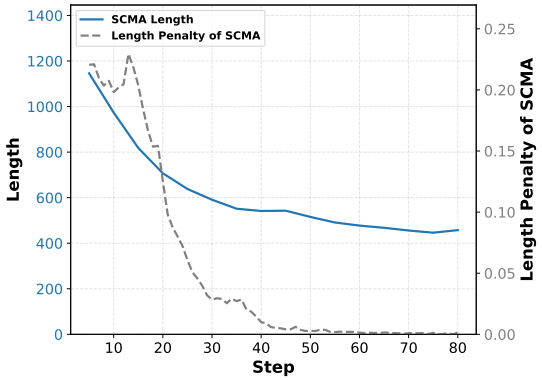


Figure 4: Dynamic evolution of response length and length penalty during SCMA training. The solid blue line (left axis) denotes the mean response length, while the dashed grey line (right axis) represents the importance-weighted length penalty. The figure illustrates how the dynamic penalty modulates response length over training steps to ensure convergence and stability.

low-scoring chunks contribute minimally to the overall logic of the response, thereby necessitating a substantial penalty. Consequently, under this intense negative feedback, the model effectively eliminates invalid content, resulting in a sharp decrease in the SCMA length from approximately 1150 to 600 as shown by the solid blue line.

- **The subsequent phase**, occurring between step 30 and step 80, transitions into a state where core logical structures are retained under a diminished penalty. As redundant information is successfully filtered, the model begins to focus on the production of high-quality outputs. The grey dashed line exhibits a precipitous decline toward zero during this interval. This reduction is attributed to the fact that the remaining segments are primarily high-scoring chunks essential to the core reasoning of the response. The importance-weighted length penalty identifies these components as necessary and ceases to apply significant pressure for further compression. As the penalty approaches zero, the SCMA length stabilizes into a plateau between 450 and 500, marking the end of the rapid reduction phase.

The results confirm that the importance-weighted length penalty functions as a sophisticated adaptive regulator rather than a static hyperparameter. By imposing rigorous constraints in the early stages to refine the output and subsequently relaxing those constraints to protect critical information, the penalty ensures a stable convergence. This process allows the SCMA length to reach an equilibrium that minimizes redundancy while preserving essential logical integrity, thereby preventing the model from experiencing either excessive compression or numerical oscillation.

K A Discussion on Reward Hacking

In the proposed Multi-Agent (MA) framework, a natural question arises regarding the behavior of the Score Agent: *Will the Score Agent fall into the trap of "reward hacking" by simply assigning the maximum score (e.g., 5) to all reasoning chunks?*

In short, the answer is no. The design of the normalization mechanism and the dynamic co-adaptation between agents within our MA system inherently prevent this trivial solution. We analyze this phenomenon by dividing the training process into two distinct stages: the early exploration phase and the late convergence phase.

Early Training Stage: The Necessity of Relative Importance. At the beginning of the training process, the generation agent's reasoning paths are far from optimal and inevitably contain a significant amount of redundant, irrelevant, or erroneous reasoning chunks. In this stage, if the Score Agent were to attempt reward hacking by assigning a maximum score (e.g., 5) to all chunks, the subsequent normalization process (e.g., softmax or linear normalization over the scores of a trajectory) would flatten the weight distribution.

Consequently, the relative importance between crucial reasoning steps and redundant chunks would be entirely lost. Because the redundant

chunks are weighted equally to the critical ones, the overall representation of the reasoning path degrades, leading to a suboptimal final prediction and drastically lowering the global reward (e.g., accuracy reward). To maximize the expected global reward, the MA system is forced to explore and update the Score Agent towards assigning differentiated scores. Thus, the Score Agent learns to penalize redundancy and highlight key reasoning steps, actively avoiding the "all 5s" trap.

Late Training Stage: Convergence vs. Reward Hacking. As the training progresses to the later stages, the generation agent adapts to the MA system's feedback and generates much higher-quality reasoning trajectories. The occurrence of redundant reasoning chunks is significantly minimized.

In this scenario, it is entirely possible that the Score Agent begins to output high scores (e.g., 5) for almost all generated chunks. However, **this is not reward hacking, but rather an indication of system convergence.** Since the reasoning chunks are now genuinely concise and crucial, they deserve uniform high scores. Because the scores are uniformly high and subsequently normalized, the length penalty mechanism (which aims to control the total length of the reasoning) exerts a relatively uniform and stable pressure across all remaining chunks. This prevents the generation agent from maliciously increasing the output length to gain more absolute scores. At this equilibrium, the generation agent outputs only necessary reasoning, and the Score Agent accurately reflects their high utility, achieving the optimal state of the MA system.

In conclusion, the Score Agent's scoring distribution dynamically evolves with the quality of the reasoning chunks. The system relies on normalization to enforce relative importance during the noisy early stages, and gracefully transitions to uniform high-scoring in the late stages as a natural consequence of algorithmic convergence rather than reward hacking.

L Hyperparameter Study

Fig. 5 presents the sensitivity analysis of SCMA regarding the length penalty hyperparameter α. The results indicate that as α increases from 0.05 to 0.40, the inference length decreases significantly across both datasets, exemplified by a reduction from 630 to 434 tokens on GSM8K, which underscores the high efficiency of SCMA in compressing the thinking process. Crucially, the model accuracy remains highly stable with only minor fluctuations despite the substantial shortening of the reasoning chains. This demonstrates that SCMA successfully retains core reasoning information while eliminating redundancy, thereby validating the robust performance of the framework under varying compression intensities.

M Comparative Case Study of GRPO and SCMA

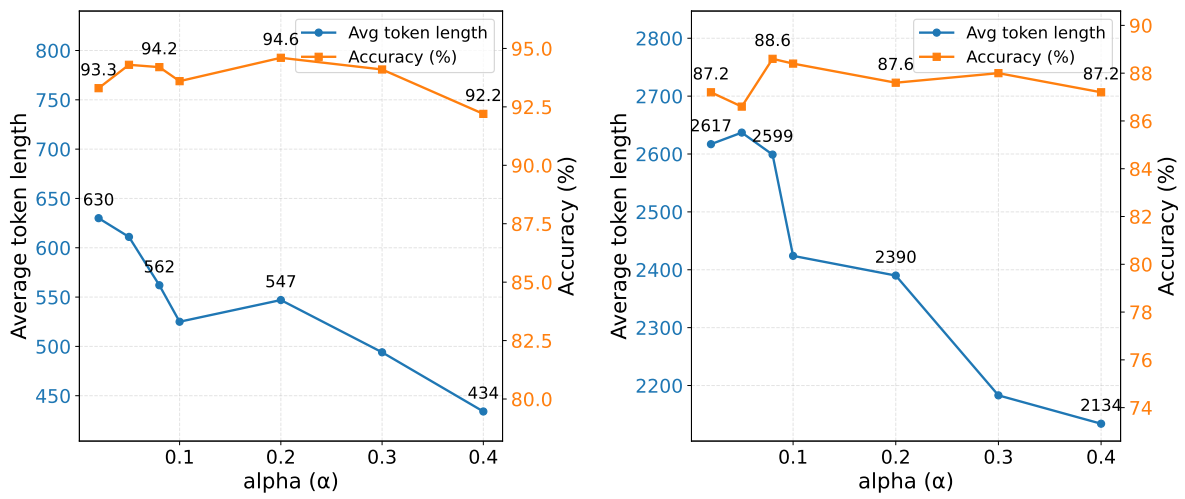
As illustrated in the comparison between SCMA and GRPO presented in Case Study 1 and 2, our method demonstrates substantial advantages in reasoning efficiency and logical precision. By analyzing the red-highlighted redundant sections in the GRPO output, we can categorize the superiority of SCMA into the following dimensions:

- **Enhanced Logical Density and Convergence**: SCMA exhibits a high degree of logical density by maintaining a convergent reasoning path from the initial prompt to the final answer. While GRPO adopts a divergent thinking style that leads to unnecessary computational overhead, SCMA focuses on the essential deductive steps and avoids the generation of low-information tokens.

- **Elimination of Redundant Meta-talk**: The response from GRPO is heavily populated with "meta-talk" such as filler phrases and transitional self-dialogue. Phrases like "Alright, let's break this down step by step" or "Wait, let me check that again" serve no functional purpose in solving the arithmetic problem. SCMA effectively prunes these linguistic redundancies and moves directly into the core mathematical logic.

- **Suppression of Unproductive Circular Verification**: A critical weakness observed in the GRPO baseline is its tendency to engage in circular verification. As shown in the red-highlighted text, GRPO repeats the same calculation of "(16 - (3+4)) * 2" three separate times without producing any new insights. In contrast, SCMA performs a single, high-efficiency verification at the end of the process, which ensures accuracy while minimizing token consumption.

- **Optimized Cognitive Processing of Simple Constraints**: GRPO demonstrates a form of



(a) GSM8K (b) MATH500

Figure 5: Average accuracy and response length of Qwen3-4B trained with different α

"over-thinking" by questioning simple provided facts, such as the number of eggs used for muffins. This excessive skepticism leads to unproductive reasoning loops that increase latency. SCMA recognizes the straightforward nature of the constraints and processes them with appropriate cognitive intensity, thereby avoiding the analytical paralysis seen in the baseline.

Summary of Improvements: The primary distinction between the two models lies in the efficiency of their internal reasoning chains. In this specific case, approximately 40% to 50% of the tokens generated by GRPO are redundant or repetitive. SCMA succeeds in teaching the model that logical rigor does not require linguistic verbosity, which results in a reasoning process that is both faster and more cost-effective for real-world deployment.

Case Study 1

Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market? Let's think step by step and output the final answer after "####"

Response of GRPO

<think>
Okay, let's see. So Janet's ducks lay 16 eggs each day. She uses some of them for breakfast and for baking muffins, and then sells the rest. The question is asking how much she makes every day at the farmers' market. Alright, let's break this down step by step.

First, the total number of eggs laid per day is 16. Then, she eats three eggs for breakfast every morning. So that's 3 eggs taken out. Then, she bakes muffins for her friends every day with four eggs. Wait, does that mean she uses four eggs for the muffins? So, total eggs used each day would be 3 for breakfast plus 4 for muffins. Let me check that again. The problem says she eats three for breakfast every morning and bakes muffins for her friends every day with four. So yeah, I think that's 3 + 4 = 7 eggs used each day.

So, total eggs laid: 16. Subtract the ones she uses: 16 - 7 = 9 eggs left to sell. Then, she sells the remainder at $2 per egg. So, 9 eggs times $2 per egg. Let me calculate that: 9 * 2 = $18. So, she makes $18 each day at the farmers' market.

Wait, let me double-check. Maybe I missed something. Let me go through each part again. Ducks lay 16 eggs per day. She eats 3 for breakfast. Then, she uses 4 eggs for muffins. So total eggs used are 3 + 4 = 7. 16 - 7 = 9. Then, 9 eggs sold at $2 each. 9 * 2 is indeed 18. So, $18 per day. That seems right.

Is there any chance that the muffins require more eggs? Like, maybe four muffins per day? But the problem says she bakes muffins for her friends every day with four. So, I think that's four eggs per day for muffins. So, yeah, 3 + 4 = 7. So, 16 - 7 = 9. 9 * 2 = 18.

I think that's correct. So the answer should be $18.
</think>

Response of SCMA

<think>
Okay, let's see. Janet's ducks lay 16 eggs each day. She uses some of them for breakfast and for muffins, and then sells the rest. I need to figure out how much she makes from selling the eggs.

First, how many eggs does she use each day? She eats three for breakfast every morning. Then she bakes muffins with four eggs. So that's 3 plus 4, which is 7 eggs total that she uses.

So if she starts with 16 eggs a day, subtract the 7 she uses, that leaves 16 minus 7. Let me calculate that: 16 - 7 equals 9 eggs. So she has 9 eggs left to sell each day.

Now, each egg is sold for $2. So 9 eggs times $2 per egg. Let me do that multiplication: 9 * 2 = $18.

Therefore, she makes $18 every day at the farmers' market. Let me just check my steps again to make sure I didn't make a mistake. Ducks lay 16 eggs. She uses 3 + 4 = 7. 16 - 7 = 9. 9 * 2 = 18. Yep, that seems right. I think that's the answer.
</think>

Case Study 2

A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? Let's think step by step and output the final answer after "####"

Response of GRPO

<think>
Okay, let's see. The problem says that a robe takes 2 bolts of blue fiber and half that much white fiber. I need to find out how many bolts in total it takes. Hmm, let me break this down step by step.

First, the blue fiber part is straightforward. It takes 2 bolts. Now, the white fiber is half of that much. Wait, half of what? The problem says "half that much," so "that" probably refers to the 2 bolts of blue fiber. So half of 2 bolts would be 1 bolt. So the white fiber requires 1 bolt.

Let me double-check that. If "half that much" refers to the blue fiber amount, which is 2 bolts, then yes, half of 2 is 1. So white fiber is 1 bolt. Then the total bolts needed would be blue plus white, which is 2 + 1 = 3 bolts.

Wait, is there another way to interpret "half that much"? Maybe someone might think "that much" refers to something else, but in the sentence structure, "that" is pointing back to the previous quantity mentioned, which is the 2 bolts of blue fiber. So I think my initial interpretation is correct.

So adding them together: 2 bolts (blue) + 1 bolt (white) equals 3 bolts total. That seems right. Let me just make sure there's no trick or anything. The problem is pretty straightforward. No mention of any other fibers or additional requirements. So yeah, total bolts needed are 3.

I think that's it. The answer should be 3.
</think>

Response of SCMA

<think>
Okay, let's see. The problem says a robe takes 2 bolts of blue fiber and half that much white fiber. I need to find the total number of bolts required. First, the blue

fiber is 2 bolts. Then the white fiber is half of that amount. Wait, half of what? Half of the blue fiber amount, right? So if the blue is 2 bolts, then half of that would be 1 bolt for the white fiber. So adding them together, 2 bolts for blue and 1 bolt for white makes a total of 3 bolts. Let me check that again. Half of 2 is 1, so yes, 2 + 1 equals 3. That seems right. I think the total is 3 bolts.
</think>