Adaptive Ability Decomposing for Unlocking Large Reasoning Model Effective Reinforcement Learning

Anonymous ACL submission

Abstract

Reinforcement learning with verifiable rewards (RLVR) has shown great potential to enhance the reasoning ability of large language models (LLMs). However, due to the limited amount of information provided during the RLVR process, the model can only engage in blind exploration, which often fails in challenging problems. To provide additional information for RLVR without relying on a teacher model, we propose **A^{2}D**, an **A**daptive **A**bility **D**ecomposing method for improving RLVR effectiveness. Concretely, we first train a decomposer via RLVR without distillation, enabling it to decompose complex questions into a set of simpler sub-questions. Next, we use this decomposer to annotate sub-questions for each question in the training dataset, and then train the reasoner through in-context distillation loss with sub-question guidance. To better understand A^{2}D, we first compare its performance with competitive baselines on eight mathematical tasks, demonstrating that A^{2}D consistently improves the effectiveness of RLVR across multiple training algorithms.

1 Introduction

Recently, reinforcement learning with verifiable rewards (RLVR) has emerged as a dominant paradigm for improving the reasoning capabilities of large language models (LLMs) (OpenAI, 2025; Guo et al., 2025). Through large-scale exploration and outcome-based supervision, RLVR enables LLMs to gradually acquire complex reasoning behaviors and evolve into large reasoning models (LRMs) (Chen et al., 2025c; Hu et al., 2025b; Chen et al., 2025b; Team et al., 2025).

Despite its empirical success, recent studies (Yue et al., 2025; Yao et al., 2025; Wu et al., 2025) show that RLVR often struggles to continuously improve reasoning performance as training progresses. A key reason is that RLVR operates in a *low-information setting* (Liu et al., 2025a; Castanyer et al., 2025), where the model performs exploration with only sparse final-answer rewards as supervision. Without informative intermediate guidance, the policy often fails to discover reasoning trajectories with meaningful learning value (Song et al., 2025; Wang et al., 2025b), leading to inefficient exploration and limited capability growth. Thus, improving the quality and structure of exploration has become one of the central challenges in RLVR.

Existing studies (Lu and Lab, 2025) have pointed out that RLVR provides substantially less information than distillation-based training. Following this, an intuitive direction is to introduce additional information into exploration. To achieve this, recent methods (Yan et al., 2025; Zhang et al., 2025c)) rely on stronger teacher models (*e.g.,* DeepSeek-R1) to generate plans, knowledge points, or reasoning trajectories to guide exploration. Although effective, they remain fundamentally within the paradigm of imitation learning, which often constrains the student model to the capability boundary of the teacher model (Shing et al., 2025; Trinh et al., 2025). This naturally raises an important question: *can a model autonomously generate useful guidance for itself during RLVR, without relying on stronger external teachers?*

To enable LLMs effectively explore based on guidance, we first need to identify an appropriate form of guidance, which can guide LLM exploration without directly revealing the answer. Inspired by that planning can significantly improve reasoning (Parmar et al., 2025; Erdogan et al., 2025), we investigate whether decomposing a complex problem into simpler sub-questions can provide the effective guidance. To verify this hypothesis, we conduct an empirical study in Fig. 1, observing that LLMs consistently achieve higher Pass@k scores with sub-question guidance. It suggests that the limitation of RLVR may not primarily stem from insufficient knowledge, but rather from the model's inability to effectively organize and com-

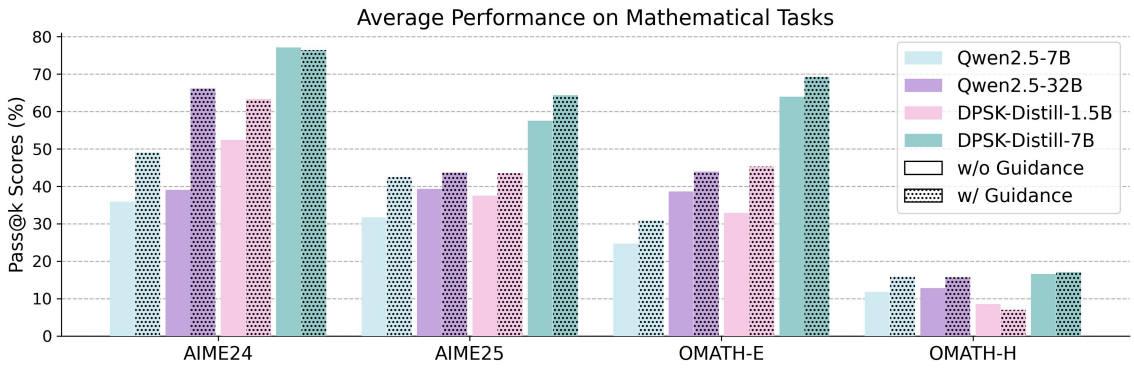


Figure 1: Pass@k scores of various models on mathematical tasks. By Integrate sub-question guidance from seed-1.6 (no-thinking) into the prompts, LLMs can perform effective exploration. Details are in Appendix D.

pose reasoning steps during exploration. By decomposing complex problems into simpler intermediate objectives, sub-question guidance reduces the exploration difficulty and helps LLMs discover higher-quality reasoning trajectories.

To enable LLMs to better generate guidance and perform reasoning, we argue that different abilities in RLVR should be learned separately. Using a single policy to simultaneously discover decomposition strategies and perform complex reasoning can therefore lead to inefficient exploration. Building on this, we propose **A**daptive **A**bility **D**ecomposing (**A^{2}D**), a novel framework for enhancing RLVR through structured exploration guidance. We conduct two complementary roles: a *decomposer* to generate sub-question guidance, and a *reasoner* to solve the original problem. Specifically, we first train a decomposer through RLVR to generate sub-questions that serve as intermediate guidance for reasoning. Then, instead of directly inserting generated guidance into the training data as in prior work (Yan et al., 2025; Zhang et al., 2025c), we introduce an *in-context distillation loss*, which encourages the reasoner to learn from the informative trajectories induced by sub-question-guided exploration. This design allows A^{2}D to serve as a plug-and-play component that can be integrated into various RLVR algorithms. The major contributions can be summarized as follows:

- We identify that one fundamental bottleneck of RLVR lies in the lack of structured intermediate guidance during exploration, which limits the discovery of useful reasoning trajectories.
- We propose A^{2}D, a self-evolving RLVR framework that leveraging a single LLM to build a decomposer and a reasoner. The decomposer can adaptively generate sub-question guidance, and the reasoner can learn through in-context distillation loss under the guidance
- We conduct extensive experiments on eight mathematical reasoning benchmarks and demonstrate that A^{2}D consistently improves the effectiveness of RLVR across multiple training algorithms.

2 Related Work

In this section, we briefly introduce the related work, and the details can be found in Appendix A.

Improving LLMs via Value-based Guidance. RLVR utilizes reward scores to supervise LRMs (Jaech et al., 2024; DeepSeek-AI, 2025; Team et al., 2025). During RLVR, the reward design (Arnal et al., 2025; Su et al., 2025), advantage shaping (Deng et al., 2025; Chen et al., 2025d), and sampling mechanism (Wang et al., 2025a; Zhu et al., 2025b) are leveraged to balance the exploration and exploitation of LRMs. Given that LLMs struggle with hard tasks (Zhang et al., 2025c; Xue et al., 2024; Peng et al., 2024), in this work, we focus on how to guide the model to transfer its ability from solving sub-questions toward solving more complex problems, thereby enhancing its abilities.

Test-time Scaling via Language-based Guidance. Supervised finetuning (SFT) is a kind of language-based guidance to elicit inner knowledge of LRMs (Hugging Face, 2025; Guha et al., 2025). Moreover, prior studies have shown that responses from the teacher model (Yan et al., 2025; Zhang et al., 2025c) and self-reflection (Sareen et al., 2025; Jiang et al., 2025a) can be used to enhance RLVR effectiveness. Besides, parallel reasoning boosts the performance of LRMs by enabling them to perform multi-turn reasoning (Qi et al., 2025; Snell et al., 2024; Luo et al., 2025). In this work,

we investigate the usage of sub-questions during RLVR to aid the model in accurately solving these problems and to activate its internal knowledge.

Compositional Generalization of LLMs. Compositional generalization refers to combine simple knowledge to solve complex problems (Liu et al., 2020; An et al., 2023). It is a vital ability for LLM self-evolution (Sakai et al., 2025; Yang et al., 2024), but is limited (Andreas et al., 2016; Sakai et al., 2025; Yang et al., 2024). Prior studies adopt several simple problems to synthesize multi-hop problems and utilize these problems to train the models (Yuan et al., 2025; Sun et al., 2025b), or guide the model to perform multi-turn reasoning (Jiang et al., 2025b; Zhu et al., 2025a). Inspired by this, in this work, we focus on teaching LLMs to combine the knowledge it has learned to answer complex questions in RLVR.

3 Approach

In this section, we first review the RLVR (Sec. 3.1). Next, we introduce **A^{2}D**, including decomposer training (Sec. 3.2) and RLVR with guidance (Sec. 3.3). Finally, we conduct the discussion in Sec. 3.4 and present the overview of A^{2}D in Fig. 2.

3.1 Preliminary of RLVR

RLVR can further enhance the reasoning abilities of LLMs (Jaech et al., 2024; DeepSeek-AI, 2025), *e.g.,* GRPO (Shao et al., 2024). In its training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, the i-th training instance contains a question x_i and an answer y_i. Given the question x, a policy with the parameter θ (*a.k.a.,* π_θ) is required to generate a solution with a final answer $\hat{y}$. Once the predicted answer $\hat{y}$ is obtained, a verifier provide the reward r based on the answer y. When $\hat{y}$ is correct, the verifier provides a positive reward R_{pos}, while it provides a negative reward R_{neg} for other situations. Based on x, $\hat{y}$, and r, the objective function of the RLVR (take GRPO as an example) can be formulated as follows,

$$\mathcal{J}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D},\{\hat{y}\}_{i=1}^{G}\sim\pi_\theta(\cdot|x)}[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\hat{y}_i|}\sum_{t=1}^{|\hat{y}_i|} \min\left(r_{i,t}\hat{A}_{i,t}, \text{clip}\left(r_{i,t}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t} - \beta D_{\text{KL}}\right)], \quad (1)$$

where r_t and $\hat{A}_t$ refer to the importance sampling coefficient and estimated advantages, and G denotes the rollout number. In GRPO, the average value and variance of the rewards in the same group are utilized to estimate the advantages as follows,

$$\hat{A}_{i,1} = \ldots, \hat{A}_{i,|y_i|} = \hat{A}_i = \frac{R_i - \text{mean}(R_1, \ldots, R_G)}{\text{std}(R_1, \ldots, R_G)}, \quad (2)$$

where mean() and std() refer to average and standard deviation of the rewards in the group. Each token in $\hat{y}_i$ will receive the same advantage $\hat{A}_i$.

3.2 Training Decomposer via RLVR

We use RLVR to guide LLM to decompose complex questions. We first introduce the RLVR dataset construction, and then present the RLVR details.

Training Dataset Construction. We collect the questions and their corresponding answer to construct the training dataset. Formally, we construct a training dataset $\mathcal{D} = \{\langle x_i, y_i\rangle\}_{i=1}^{n}$, where x_i, y_i, and n denote the question, the corresponding ground truth answer, and the number of instances. As discussed in prior work (Yao et al., 2023), decomposing the original question is essentially a planning process, during which the model possesses a greater capacity to explore potential solutions. Under this assumption, we think that the trained decomposer can provide useful guidance for the subsequent reasoner RLVR, despite no teacher guidance is introduced into the training.

RLVR for Decomposer. We design a multi-dimensional reward function, including a quality reward R_{Q} and a format reward R_{F}. Concretely, to assess the quality of the generated sub-questions, we employ a proxy reasoner to reason for n_{proxy} times based on the original question and the decomposed sub-questions. The quality reward is set to 1 if and only if at least one response from the proxy reasoner is correct, *i.e.,* $R_{\text{Q}} = 1$. Otherwise, the R_{Q} is set to 0, *i.e.,* $R_{\text{Q}} = 0$. For format reward, we examine whether the responses from the decomposer are in the required format (Details is in Appendix G). The format reward will be set to 1 when and only when the response passes the format check (*i.e.,* $R_{\text{F}} = 1$), or it will be set to 0 (*i.e.,* $R_{\text{F}} = 0$). Based on R_{Q} and R_{F}, we compute the response reward by multiplying these two rewards, *i.e.,* $R = R_{\text{Q}} \times R_{\text{F}}$. In RLVR, we encourage $\pi_{\theta_{\text{D}}}$ to produce responses in the required format and to assist the reasoner in generating correct answers.

3.3 Improving with Sub-question Guidance

We present RLVR details of the reasoner $\pi_{\theta_{\text{R}}}$. After obtaining the trained decomposer $\pi_{\theta_{\text{D}}}$, we introduce an in-context distillation loss (IDL) to guide

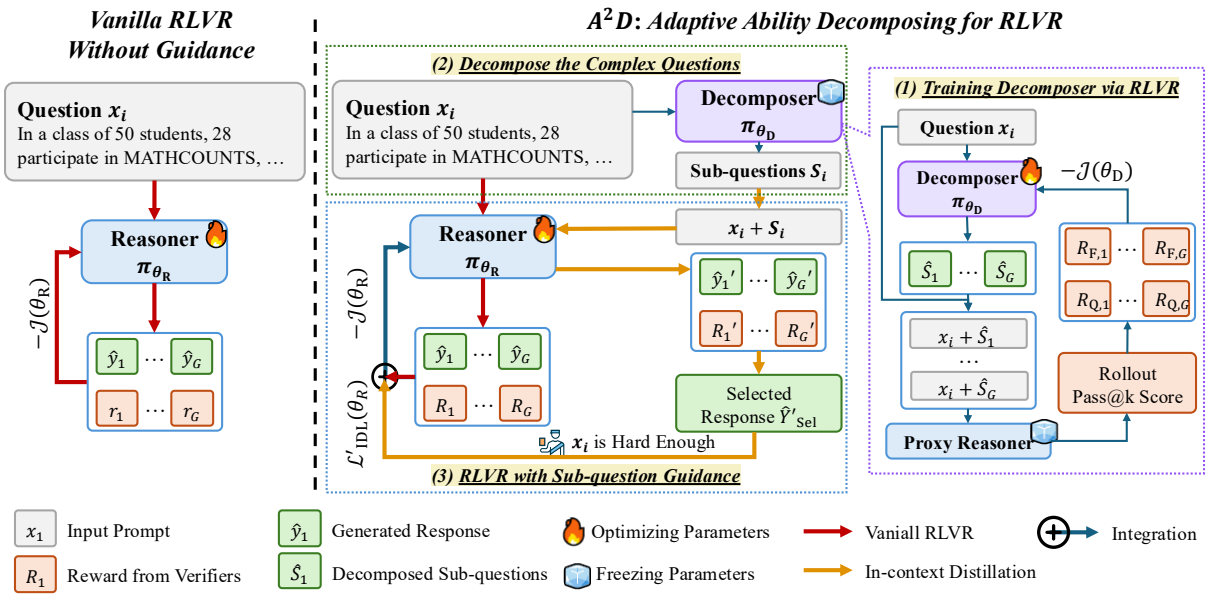


Figure 2: The comparison between vanilla RLVR and our A²D. In A²D, we first train a decomposer π_{θ_D} through RLVR, guided by the format reward R_F and quality reward R_Q. Next, we use π_{θ_D} to annotate sub-questions for each training question, and design the in-context distillation loss (IDL), *i.e.,* Eq. 5, to integrate the sub-question guidance into the RLVR for reasoner π_{θ_R}. To better present A²D, we present the pseudo code in Appendix H.

LLM exploration with sub-question hints. Then, we integrate IDL into RLVR.

Annotating Sub-question Guidance. We introduce the approach to train a decomposer only with the dataset containing questions and corresponding answers $\mathcal{D}$ in Sec. 3.2. This dataset will also be utilized in the training process for the reasoner. To provide additional guidance, we use the decomposer to decompose each question in $\mathcal{D}$ and annotate it with a list of its potential sub-questions. Formally, given the question x_i, the decomposer will decompose this question into several sub-questions $\mathcal{S}_i = \{s_{i,1}, \ldots, s_{i,p}\}$, and construct them into triples $\langle x_i, y_i, \mathcal{S}_i \rangle$. After annotating all questions in $\mathcal{D}$, we construct the training dataset with sub-question guidance, *i.e.,* $\mathcal{D}_{SQ} = \{\langle x_i, y_i, \mathcal{S}_i \rangle\}_{i=1}^{n}$. These sub-questions can be viewed as an idea of solving the original question. We concatenate x_i and S_i using a specific template (Appendix F), which serves as a sub-question-guided prompt x'_i for guiding LLLM reasoning.

In-context Distillation Loss. We design *in-context distillation loss* (IDL) to enhance the ability of LLMs with the sub-question guidance. IDL is a plug-and-play component that is compatible with various RLVR algorithms. Specifically, given the sub-question-guided prompt x'_i, π_{θ_R} generates multiple solutions $\hat{\mathcal{Y}}' = \{\hat{y}'_{i,1}, \ldots, \hat{y}'_{i,n_{\text{rollout}}}\}$. With the guidance, the generated solutions $\hat{\mathcal{Y}}$ are more likely to contain the correct answer. After generation, we verify the correctness of each response in $\hat{\mathcal{Y}}'$, obtaining the rewards $\{R'_{i,1}, \ldots, R'_{i,n_{\text{rollout}}}\}$. Then, we can select the positive solutions to optimize the reasoner π_{θ_R}. Since no sub-question guidance is available in evaluation, we *remove guidance* from prompts and train the model only based on the question, aiming to internalize the correct problem-solving ideas into the reasoner's inner knowledge. The IDL can be formulated as follows,

$$\mathcal{L}_{\text{IDL}}(\theta_R) = -\frac{1}{N_{\text{pos}}} \sum_{j=1}^{n_{\text{rollout}}} \pi_{\theta_R}(\hat{y}'_{i,j}|x_i) \cdot \mathbb{I}[R'_{i,j} = 1], \quad (3)$$

where $N_{\text{pos}} = \sum_{j=1}^{n_{\text{rollout}}} \mathbb{I}[R'_{i,j} = 1]$ refers to the number of correct responses and $\mathbb{I}[R'_{i,j} = 1]$ returns 1 when and only when $R'_{i,j} = 1$ while returns 0 for others. By optimizing π_{θ_R} to minimize $\mathcal{L}_{\text{IDL}}$, it might be able to surpass its prior ability boundary.

Integrating IDL into RLVR. Following the vanilla RLVR, for a given problem, we first have the reasoner generate multiple solutions $\hat{\mathcal{Y}} = \{\hat{y}_{i,1}, \ldots, \hat{y}_{i,n_{\text{rollout}}}\}$, and then compute the average reward of each generated responses $\bar{R}$. For integrating the IDL function into RLVR, the purpose is to provide auxiliary guidance for questions for which π_{θ_R} is unable to obtain correct answers through its exploration. To achieve this, we preset a threshold k_1 $(0 < k_1 < 1)$ such that when the average reward

Table 1: Comparison between A^{2}D and prior work. "Solu", "KP", "Abs", "Ref", and "SQ" refer to solution, knowledge point, abstraction, reflection, and sub-question. "TT" and "TI" denote whether utilize teacher LLM in training and inference process. "CS" refers whether performing cold start on backbone LLM.

Methods	CS	Guide	RLVR	TT	TI
LUFFT (Yan et al., 2025)	✗	Solu	✔	✔	✗
S-G (Zhang et al., 2025c)	✗	KP	✔	✔	✗
RLAD (Qu et al., 2025)	✔	Abs	✗	✗	✔
STP (Dong and Ma, 2025)	✔	Ref	✗	✗	✔
UFT (Liu et al., 2025b)	✔	Solu	✗	✔	✗
P-RFT (Huang et al., 2025)	✗	Solu	✔	✔	✗
A^{2}D (Ours)	✗	**SQ**	✔	✗	✗

falls below this threshold (*i.e.,* $k_1 > \bar{R}$), the IDL is applied. Otherwise, the IDL is not activated. Since π_{θ_R} is likely to follow the guidance to solve the question, the generated responses might be highly similar to each other. Thus, to prevent overfitting, we retain at most $k_2 \times n_{\text{rollout}}$ $(0 < k_2 < 1)$ positive responses generated under the sub-question guidance $\hat{\mathcal{Y}}'$ as the instances for IDL. Formally, the *selected positive responses* can be denoted as

$$\hat{\mathcal{Y}}'_{\text{Sel}} = \{\hat{y}_{i,1,\text{Sel}}, \ldots, \hat{y}_{i,q,\text{Sel}}\}, \quad q = \min(N_{\text{pos}}, k_2 \times n_{\text{rollout}}). \tag{4}$$

Furthermore, we observe that several generated solutions in $\hat{\mathcal{Y}}'_{\text{Sel}}$ are not rigorous, as they might fail to consider the scenarios not covered in the hints, thereby performing imprecise reasoning. In this situation, directly fine-tuning π_{θ_R} on these imprecise data will hurt its performance. Thus, we utilize *different templates* for the original problem and use these prompts to segment the model's abilities, like prior work (He et al., 2025; Shin et al., 2025). The following formula presents the adjusted IDL,

$$\mathcal{L}'_{\text{IDL}}(\theta_R) = -\frac{1}{|\hat{\mathcal{Y}}'_{\text{Sel}}|} \sum_{j=1}^{|\hat{\mathcal{Y}}'_{\text{Sel}}|} \pi_{\theta_R}(\hat{y}'_{i,j,\text{Sel}}|x_i, p). \tag{5}$$

Based on the above discussion, we can integrate IDL into the RLVR training process as follows,

$$\mathcal{L}_{\text{A}^2\text{D}}(\theta_R) = -\mathcal{J}(\theta_R) + \alpha \mathcal{L}'_{\text{IDL}}(\theta_R) \cdot \mathbb{I}[k_1 > \bar{R}], \tag{6}$$

where $\mathcal{J}(\theta_R)$ is the objective function of RLVR (Eq. 1), and α is a hyperparameter to balance the effect of the above two objective functions.

3.4 Discussion

To better demonstrate our A^{2}D, we provide the pseudocode in Appendix H. Further, to facilitate a clearer comparison with prior work and highlight our contribution, we present a table summarizing the key differences between our method and existing approaches in Table 1. From the comparison, we observe that prior work generally relies on powerful teacher LLMs to guide the training process or to distill data for model initialization. As mentioned above, incorporating external teacher models during training can lead the policy model to overfit to imitation, making it difficult to surpass its reasoning limits. Our work focuses on how to further improve training performance in the absence of assistance from external strong models. Based on this goal, our A2D does not require a teacher model for guidance during training, nor does it rely on external models during inference, making it an efficient and effective approach.

4 Experiments

In this section, we assess the effectiveness of A^{2}D and analyze its features. First, we report the experimental settings (Sec. 4.1). Next, we present the evaluation results (Sec. 4.2). Finally, we present the detailed analysis of A^{2}D (Sec. 4.3).

4.1 Experimental Settings

In this part, we introduce the datasets that are utilized in experiments and the baselines. We report the implementation details in Appendix E.

Datasets. We utilize the dataset proposed in prior work (Chen et al., 2025c) as the training dataset for the decomposer and reasoner. In evaluation, we adopt eight mathematical benchmarks as the downstream tasks, including AIME24 (AIME2024, 2024), AIME25 (AIME2025, 2025), AMC23 (AMC2023, 2023), BeyondAIME (ByteDance-Seed, 2025a), MATH500 (Lightman et al., 2024), Minerva (Minerva), OlymMATH-Easy (Sun et al., 2025a), and OlymMATH-Hard (Sun et al., 2025a).

Baselines. For evaluation, we conduct various SFT-based and RLVR-based methods as the baselines. For SFT-based methods, we utilize seed-1.6 (no thinking) (ByteDance-Seed, 2025b) to generate the solution and use the generated instances for SFT (*i.e.,* SFT w/ CoT). Besides, given the sub-questions from the decomposer, we combine these generated sub-questions and the solutions to conduct the SFT process (*i.e.,* SFT w/ CoT & SQ). For RLVR-based methods, we conduct LUFFY (Yan et al., 2025) and Scaf-GRPO (Zhang et al., 2025c)

Table 2: Accuracy of different approaches on mathematical tasks. BeAIME, OMATH-E, and OMATH-H denote BeyondAIME, OlymMATH-Easy, and OlymMATH-Hard, respectively. "Avg." refers to average accuracy.

	AIME24	**AIME25**	**AMC23**	**BeAIME**	**MATH500**	**Minerva**	**OMATH-E**	**OMATH-H**	**Avg.**
				Qwen2.5-7B-Instruct					
Baseline	9.6	6.3	47.5	3.4	70.8	22.9	4.8	2.1	20.9
+ GRPO	15.0	8.3	52.8	5.3	73.3	24.8	4.6	2.9	23.4
+ GRPO w/ $n_{\text{rollout}} = 64$	15.0	6.7	51.9	5.4	73.9	25.3	4.6	**3.0**	23.2
+ SFT w/ CoT	7.1	2.9	41.3	2.8	63.6	24.4	2.9	1.4	18.3
+ SFT w/ CoT & SQ	5.8	1.3	40.0	2.5	62.7	23.3	4.3	1.4	17.7
+ LUFFY	10.4	8.8	47.2	3.9	69.2	22.4	4.4	2.8	21.1
+ Scaf-GRPO	14.6	9.6	**54.7**	5.5	74.5	25.1	5.6	2.8	24.1
+ A^{2}D (Ours)	**20.0**	**17.1**	53.4	**6.6**	**75.6**	**28.9**	**8.4**	1.8	**26.5**
				LLaMA3.2-3B-Instruct					
Baseline	1.7	0.0	10.9	0.1	26.8	6.6	1.0	0.4	5.9
+ GRPO	11.3	0.8	29.4	0.6	52.5	13.4	2.5	1.3	14.0
+ GRPO w/ $n_{\text{rollout}} = 64$	10.8	1.7	30.0	1.1	52.5	14.6	2.8	1.3	14.4
+ SFT w/ CoT	2.1	1.7	22.5	0.5	41.5	9.6	2.1	0.6	10.1
+ SFT w/ CoT & SQ	3.8	0.4	21.9	0.8	41.6	9.9	1.5	1.0	10.1
+ LUFFY	1.7	0.0	9.7	0.6	26.4	6.6	1.6	1.1	6.0
+ Scaf-GRPO	11.3	0.8	30.6	0.8	51.2	**15.3**	2.6	0.9	14.2
+ A^{2}D (Ours)	**13.8**	**2.1**	**30.9**	**2.0**	**52.6**	**15.3**	**3.1**	**1.6**	**15.2**
				Qwen2.5-Math-7B-Instruct					
Baseline	10.8	6.7	56.3	6.4	78.5	23.4	5.9	2.0	23.8
+ GRPO	10.4	8.8	58.1	8.3	79.7	24.6	**8.7**	2.0	25.1
+ A^{2}D (Ours)	**11.7**	**11.3**	**58.8**	**9.0**	**80.3**	**25.6**	8.4	**3.0**	**26.0**
				LLaMA3.1-8B-Instruct					
Baseline	2.5	0.0	13.8	0.1	33.3	10.9	0.4	0.6	7.7
+ GRPO	7.1	0.0	28.8	0.8	50.3	19.6	**1.6**	1.0	13.7
+ A^{2}D (Ours)	**8.3**	**1.3**	**30.6**	**1.4**	**50.6**	**19.9**	**1.6**	**1.6**	**14.4**

as baselines. Since these methods require a teacher model for guidance, we utilize our well-trained decomposer mentioned in Sec. 3.2 to provide the guidance for a fair comparison.

4.2 Main Results

We present the evaluation results in Table 2, and provide a significant test in Appendix I.2.

First, from the results, we observe that A^{2}D consistently yields substantial improvements across different backbone models and achieves superior performance on mathematical tasks. Compared with both the SFT-based (*i.e.,* SFT w/ CoT and SFT w/ CoT & SQ) and RLVR-based (*i.e.,* GRPO, LUFFY, and Scaf-GRPO) baselines, the models trained with our method also surpass these methods. The experimental results demonstrate that our method is effective across models of different families, scales, and capability levels.

Second, A^{2}D shows the robustness across various models. During the RLVR of the decomposer, we use Qwen2.5-7B-Ins as the proxy reasoner. The decomposed sub-questions from the trained decomposer can assist not only the RLVR of Qwen2.5-7B-Ins but also that of Qwen2.5-Math-7B-Ins and LLaMA family, yielding performance improvements. This indicates that $\pi_{\theta_{\text{D}}}$ can guide the RLVR of various models without requiring a specific proxy reasoner, showing that our method A^{2}D is both effective and robust.

Third, comparing the two SFT-based methods, we find that adding decomposer-generated sub-questions to the training data does not improve model performance. Instead, it even leads to performance degradation in certain scenarios, *e.g.,* AIME25 and AMC23. The underlying reasons are twofold. First, learning to decompose the original question may not directly aid in solving it, as these are two dissimilar abilities. Second, although the sub-questions can stimulate exploration that occasionally leads to the correct answer, they do not necessarily represent a complete reasoning process and may overlook certain cases. For these reasons, directly learning the decomposer-generated sub-questions does not contribute to improving the reasoning ability of LLMs. Our method effectively avoids this issue by leveraging the decomposed sub-questions through IDL, thereby providing richer guidance for RLVR.

Finally, directly training LLM using teacher-

Table 3: Results of the ablation study. We first present the ablation study about the training process of the decomposer, and then provide the analysis of the effectiveness of each module in our approach. The best is in bold.

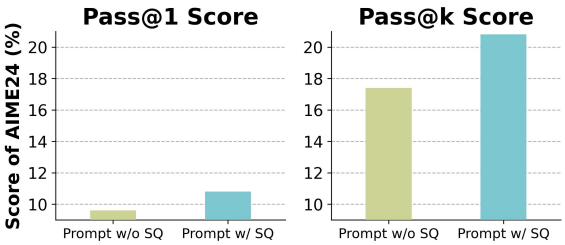
	AIME24	AIME25	MATH500	Minerva	Avg.
Pass@k Performance of the Decomposer					
A^2D (Ours)	**20.8**	**18.3**	**82.9**	**30.7**	**38.2**
w/o Format Reward w/ Pass@k Reward	15.8	16.7	83.2	30.4	36.5
w/o Format Reward w/ Pass@1 Reward	16.3	14.9	81.5	28.5	35.3
w/o RLVR	17.1	13.0	78.6	28.1	34.2
Pass@1 Performance of the Reasoner					
A^2D (Ours)	**20.0**	**17.1**	**75.6**	**28.9**	**35.4**
w/o Removing Guidance	15.0	9.6	74.5	25.0	31.0
w/o Selection	16.3	11.7	73.7	26.6	32.1
w/o Selection w/o Different Prompt	13.3	10.0	69.6	22.6	28.9
w/o Sub-question Guidance	15.0	8.3	73.3	24.8	30.4

generated solutions without the thinking process (*i.e.,* SFT w/ CoT and LUFFY) is unlikely to yield improvements and may even impair the ability. A possible explanation is that these backbones have already learned a substantial amount of instructions during post-training. Further training on such data may lead to overfitting, thereby degrading their performance. Compared to SFT w/ CoT, LUFFY introduces data distilled from the teacher model only when the model struggles to answer a question, reinforcing knowledge that the model has not yet mastered and thereby partially alleviating the overfitting issues. However, this approach requires the introduction of an external model and relies on the teacher model possessing a strong ability to generate correct and complete solutions.

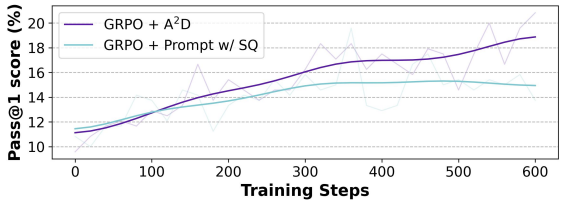
4.3 Detailed Analysis

To further understand A^2D, we present the detailed analysis, including the ablation study, the effects of integrating A^2D into various RLVR, and the effects of the sub-question guidance utilization. We also present the statistical analysis and the case study of the generated sub-questions in Appendix I.

Ablation Study. To evaluate the capability of the decomposer, we concatenate the generated sub-questions with the original question as a combined prompt, and then ask Qwen2.5-7B-Ins to produce an answer. We report the Pass@k score in the top part of Table 3. Based on the results, the format reward and Pass@k reward both enhance the decomposer's ability. Format reward teaches the decomposer to generate responses formally, and Pass@k reward helps it to present the sub-problems that are more helpful to the reasoner. By training with the above two rewards, the decomposer can better assist the subsequent RLVR process of the reasoner,



(a) Pass@1 and Pass@k performance without training.



(b) Pass@1 score of AIME24 during GRPO.

Figure 3: Performance comparison of Qwen2.5-7B-Ins of whether sub-questions are included in the prompt.

thereby improving the training effect. In the bottom part of the Table 3, we present the results of removing the Selection and Different Prompt mechanism of A^2D. We can observe the decrease in the model's performance without these two modules. For problems that the model is already capable of solving through its exploration, it is unnecessary to introduce additional external guidance, so as to avoid potential conflicts between such guidance and the model's inherent problem-solving trajectory. For problems that require external guidance, training with diverse prompts can mitigate the influence of external knowledge on the model, thereby improving its overall performance.

Adaptation to Other RLVR Algorithms. Our method introduces the IDL function to guide the model to leverage guidance in RLVR, which is orthogonal to RLVR methods and can be combined

Table 4: Effect of integrating A^2D into various RLVR on mathematical tasks. BeAIME, OMATH-E, and OMATH-H denote BeyondAIME, OlymMATH-Easy, and OlymMATH-Hard, respectively. "Avg." refers to average accuracy.

	AIME24	**AIME25**	**AMC23**	**BeAIME**	**MATH500**	**Minerva**	**OMATH-E**	**OMATH-H**	**Avg.**
Baseline	9.6	6.3	47.5	3.4	70.8	22.9	4.8	2.1	20.9
+ GRPO	15.0	8.3	52.8	5.3	73.3	24.8	4.6	2.9	23.4
+ A^2D (Ours)	20.0	17.1	53.4	6.6	75.6	28.9	8.4	1.8	**26.5**
+ RLOO	12.9	7.1	52.5	4.6	74.9	25.2	5.9	2.8	23.2
+ A^2D (Ours)	17.9	12.5	55.0	5.8	75.2	26.8	6.4	2.3	**25.2**
+ REINFORCE++	14.2	8.3	50.6	3.6	73.6	26.1	3.6	2.1	22.8
+ A^2D (Ours)	14.6	10.4	52.8	5.6	72.7	26.7	4.9	2.5	**23.8**

with other methods to improve the effectiveness. To validate it, we collect commonly used RLVR algorithms (*i.e.,* GRPO (Shao et al., 2024), RLOO (Ahmadian et al., 2024), and REINFORCE++ (Hu et al., 2025a)) and combined A^2D with these methods. In Table 4, we observe that traditional RLVR can enhance the model's reasoning ability. Building on these algorithms, A^2D can further improve the model's ability, achieving better performance. For more challenging tasks (*e.g.,* AIME25 and BeyondAIME), vanilla RLVR struggles to yield improvements, as the model's limited ability makes it hard to discover correct solutions through self-exploration. In contrast, sub-question guidance can provide the reasoner with novel ideas, extending its ability boundaries and enabling it to tackle these complex tasks during RLVR. In all, A^2D can be adapted to various RLVR algorithms, verifying that it is a plug-and-play component.

Utilization of Sub-question Guidance. In fact, the trained decomposer can not only be used to decompose questions in the training set but also for test-time queries, thereby assisting the model in solving questions. We experiment with the effectiveness of different methods. In Fig. 3a, we observe that adding decomposed sub-questions into the prompt (*i.e.,* Prompt w/ SQ) leads to improvements in both the Pass@1 score and the Pass@k score, indicating that decomposing the original problem can guide LLMs to explore and reason more effectively, resulting in improved performance. This also serves as one of the key motivations behind our method. Moreover, we added sub-question guidance to both the training and test sets, and trained the reasoner using GRPO on this augmented data, *i.e.,* GRPO + Prompt w/ SQ. In Fig. 3b, we compare its performance with our approach (*i.e.,* GRPO + A^2D). We observe that in the first 100 steps, GRPO + Prompt w/ SQ performs better, because the sub-question guidance can directly provide the reasoner with solution strategies, leading to improved performance. However, as training progresses, the model becomes overly reliant on the sub-question guidance from the decomposer, which diminishes its exploratory ability and prevents further improvement in reasoning ability. In contrast, during A^2D, the model uses guidance only when the question is hard enough, while relying on self-exploration in all other cases. As a result, A^2D does not lead to a rapid decline in the model's exploratory ability, which enables the model to possess better reasoning ability.

5 Conclusion

In this work, we proposed A^2D, adaptive ability decomposing for enhancing the effectiveness of the RLVR. Concretely, we first trained a decomposer to decompose the complex questions into several sub-questions, and then utilized these sub-questions to assist the reasoner RLVR. With the additional guidance, the reasoner can explore more effectively and further enhance its reasoning ability during RLVR. To better understand the features of the decomposer, we conducted a detailed analysis of the training approach and its behaviours. Experiments demonstrated strong adaptability and robustness, and can serve as a plug-and-play component compatible with different RLVR algorithms. Moreover, by analyzing the role and usage of sub-question guidance, we find that only the proper use of guidance can effectively help the lead LLM break through its capability boundary.

In future work, whether our algorithm can be extended to online settings and scenarios is also an important question. When AI assistants help humans solve problems, they must possess the ability to evolve and improve based on user feedback, thereby becoming more helpful and useful.

Limitations

In this section, we discuss the limitations of our paper. First, limited by the paper length, only the evaluation results of mathematical tasks are presented in the main text. To enhance the credibility of experimental findings, evaluations are conducted across diverse tasks, which validate the effectiveness of the proposed algorithm. Furthermore, the results of tasks from other domains are illustrated in Appendix C to further verify the conclusion. Additionally, potential ethical issues are not discussed in this work. Nevertheless, the model is trained solely on mathematical and scientific datasets, leading to an extremely low risk of relevant ethical concerns.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 12248–12267. Association for Computational Linguistics.

AIME2024. 2024. Aime2024.

AIME2025. 2025. Aime2025.

AMC2023. 2023. Amc2023.

Shengnan An, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Jian-Guang Lou, and Dongmei Zhang. 2023. How do in-context examples affect compositional generalization? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 11027–11052. Association for Computational Linguistics.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 39–48. IEEE Computer Society.

Charles Arnal, Gaëtan Narozniak, Vivien Cabannes, Yunhao Tang, Julia Kempe, and Rémi Munos. 2025. Asymmetric REINFORCE for off-policy reinforcement learning: Balancing positive and negative rewards. *CoRR*, abs/2506.20520.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press.

ByteDance-Seed. 2025a. Beyondaime: Advancing math reasoning evaluation beyond high school olympiads.

ByteDance-Seed. 2025b. Seed1.6 tech introduction.

Roger Creus Castanyer, Johan S. Obando-Ceron, Lu Li, Pierre-Luc Bacon, Glen Berseth, Aaron C. Courville, and Pablo Samuel Castro. 2025. Stable gradients for stable learning at scale in deep reinforcement learning. *CoRR*, abs/2506.15544.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: Process supervision without process. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Jiangjie Chen, Qianyu He, Siyu Yuan, Aili Chen, Zhicheng Cai, Weinan Dai, Hongli Yu, Qiying Yu, Xuefeng Li, Jiaze Chen, Hao Zhou, and Mingxuan Wang. 2025a. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles. *CoRR*, abs/2505.19914.

Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, Yufeng Yuan, Yu Yue, Lin Yan, Qiying Yu, Xiaochen Zuo, Chi Zhang, Ruofei Zhu, Zhecheng An, Zhihao Bai, Yu Bao, and 80 others. 2025b. Seed1.5-thinking: Advancing superb reasoning models with reinforcement learning. *CoRR*, abs/2504.13914.

Zhipeng Chen, Yingqian Min, Beichen Zhang, Jie Chen, Jinhao Jiang, Daixuan Cheng, Wayne Xin Zhao, Zheng Liu, Xu Miao, Yang Lu, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen. 2025c. An empirical study on eliciting and improving r1-like reasoning models. *CoRR*, abs/2503.04548.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. 2025d. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *CoRR*, abs/2508.10751.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *CoRR*, abs/2506.14758.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Jia Deng, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2025. Decomposing the entropy-performance exchange: The missing keys to unlocking effective reinforcement learning. *CoRR*, abs/2508.02260.

Kefan Dong and Tengyu Ma. 2025. STP: self-play LLM theorem provers with iterative conjecturing and proving. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

Etash Kumar Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. Openthoughts: Data recipes for reasoning models. *CoRR*, abs/2506.04178.

Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, Haoqi Fan, Jian Wang, Jianyu Jiang, Jiawei Wang, Jingji Chen, Jingjia Huang, Kang Lei, Liping Yuan, Lishu Luo, Pengfei Liu, Qinghao Ye, Rui Qian, Shen Yan, and 81 others. 2025. Seed1.5-vl technical report. *CoRR*, abs/2505.07062.

Qianyu He, Siyu Yuan, Xuefeng Li, Mingxuan Wang, and Jiangjie Chen. 2025. Thinkdial: An open recipe for controlling reasoning effort in large language models. *CoRR*, abs/2508.18773.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025a. Reinforce++: Stabilizing critic-free policy optimization with global normalization. *CoRR*, abs/2501.03262.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025b. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *CoRR*, abs/2503.24290.

Zeyu Huang, Tianhao Cheng, Zihan Qiu, Zili Wang, Yinghui Xu, Edoardo M. Ponti, and Ivan Titov. 2025. Blending supervised and reinforcement fine-tuning with prefix sampling. *CoRR*, abs/2507.01679.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, and 80 others. 2024. Openai o1 system card. *CoRR*, abs/2412.16720.

Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, Zheng Liu, Dong Yan, Jian Xie, Zhongyuan Wang, and Ji-Rong Wen. 2024. Technical report: Enhancing LLM reasoning with reward-guided tree search. *CoRR*, abs/2411.11694.

Yuhua Jiang, Yuwen Xiong, Yufeng Yuan, Chao Xin, Wenyuan Xu, Yu Yue, Qianchuan Zhao, and Lin Yan. 2025a. PAG: multi-turn reinforced LLM self-correction with policy as generative verifier. *CoRR*, abs/2506.10406.

Zhouyu Jiang, Mengshu Sun, Lei Liang, and Zhiqiang Zhang. 2025b. Retrieve, summarize, plan: Advancing multi-hop question answering with an iterative approach. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pages 1677–1686. ACM.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jiarui Liu, Kaustubh Dhole, Yingheng Wang, Haoyang Wen, Sarah Zhang, Haitao Mao, Gaotang Li, Neeraj Varshney, Jingguo Liu, and Xiaoman Pan. 2025a. Stabilizing reinforcement learning for honesty alignment in language models on deductive reasoning. *arXiv preprint arXiv:2511.09222*.

Mingyang Liu, Gabriele Farina, and Asuman E. Ozdaglar. 2025b. UFT: unifying supervised and reinforcement fine-tuning. *CoRR*, abs/2505.16984.

Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. 2020. Compositional generalization by learning analytical expressions. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kevin Lu and Thinking Machines Lab. 2025. On-policy distillation. *Thinking Machines Lab: Connectionism*.

Tongxu Luo, Wenyu Du, Jiaxi Bi, Stephen Chung, Zhengyang Tang, Hao Yang, Min Zhang, and Benyou Wang. 2025. Learning from peers in reasoning models. *CoRR*, abs/2505.07787.

Minerva. Minerva.

OpenAI. 2025. Introducing openai o3 and o4-mini.

Mihir Parmar, Palash Goyal, Xin Liu, Yiwen Song, Mingyang Ling, Chitta Baral, Hamid Palangi, and Tomas Pfister. 2025. PLAN-TUNING: post-training language models to learn step-by-step planning for complex problem solving. *CoRR*, abs/2507.07495.

Yixing Peng, Quan Wang, Licheng Zhang, Yi Liu, and Zhendong Mao. 2024. Chain-of-question: A progressive question decomposition approach for complex knowledge base question answering. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 4763–4776. Association for Computational Linguistics.

Jianing Qi, Xi Ye, Hao Tang, Zhigang Zhu, and Eunsol Choi. 2025. Learning to reason across parallel samples for LLM reasoning. *CoRR*, abs/2506.09014.

Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Setlur, Ruslan Salakhutdinov, Chelsea Finn, and Aviral Kumar. 2025. RLAD: training llms to discover abstractions for solving reasoning problems. *CoRR*, abs/2510.02263.

Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2025. Revisiting compositional generalization capability of large language models considering instruction following ability. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 31219–31238. Association for Computational Linguistics.

Kusha Sareen, Morgane M. Moss, Alessandro Sordoni, Rishabh Agarwal, and Arian Hosseini. 2025. Putting the value back in RL: better test-time scaling by unifying LLM reasoners with verifiers. *CoRR*, abs/2505.04842.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Jiho Shin, Clark Tang, Tahmineh Mohati, Maleknaz Nayebi, Song Wang, and Hadi Hemmati. 2025. Prompt engineering or fine-tuning: An empirical assessment of llms for code. In *22nd IEEE/ACM International Conference on Mining Software Repositories, MSR@ICSE 2025, Ottawa, ON, Canada, April 28-29, 2025*, pages 490–502. IEEE.

Makoto Shing, Kou Misaki, Han Bao, Sho Yokoi, and Takuya Akiba. 2025. TAID: temporally adaptive interpolated distillation for efficient knowledge transfer in language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314.

Yuda Song, Julia Kempe, and Rémi Munos. 2025. Outcome-based exploration for LLM reasoning. *CoRR*, abs/2509.06941.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. 2025. Crossing the reward bridge: Expanding RL with verifiable rewards across diverse domains. *CoRR*, abs/2503.23829.

Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, Lei Fang, and Ji-Rong Wen. 2025a. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. *CoRR*, abs/2503.21380.

Yiyou Sun, Shawn Hu, Georgia Zhou, Ken Zheng, Hannaneh Hajishirzi, Nouha Dziri, and Dawn Song. 2025b. OMEGA: can llms reason outside the box in math? evaluating exploratory, compositional, and transformative generalization. *CoRR*, abs/2506.18880.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 75 others. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599.

Khoa Trinh, Gaurav Menghani, and Erik Vee. 2025. GUIDE: guided initialization and distillation of embeddings. *CoRR*, abs/2510.06502.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025a. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. *CoRR*, abs/2506.01939.

Zhen Wang, Zhifeng Gao, and Guolin Ke. 2025b. Masked-and-reordered self-supervision for reinforcement learning from verifiable rewards. *arXiv preprint arXiv:2511.17473*.

Fang Wu, Weihao Xuan, Ximing Lu, Zaïd Harchaoui, and Yejin Choi. 2025. The invisible leash: Why RLVR may not escape its origin. *CoRR*, abs/2507.14843.

Shangzi Xue, Zhenya Huang, Jiayu Liu, Xin Lin, Yuting Ning, Binbin Jin, Xin Li, and Qi Liu. 2024. Decompose, analyze and rethink: Solving intricate problems with human-like reasoning cycle. In *Advances in Neural Information Processing Systems 38: Annual*

Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. *CoRR*, abs/2504.14945.

Haoran Yang, Hongyuan Lu, Wai Lam, and Deng Cai. 2024. Exploring compositional generalization of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop, NAACL 2024, Mexico City, Mexico, June 18, 2024*, pages 16–24. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Xinhao Yao, Lu Yu, Xiaolin Hu, Fengwei Teng, Qing Cui, Jun Zhou, and Yong Liu. 2025. The debate on RLVR reasoning capability boundary: Shrinkage, expansion, or both? A two-stage dynamic view. *CoRR*, abs/2510.04028.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476.

Lifan Yuan, Weize Chen, Yuchen Zhang, Ganqu Cui, Hanbin Wang, Ziming You, Ning Ding, Zhiyuan Liu, Maosong Sun, and Hao Peng. 2025. From f(x) and g(x) to f(g(x)): Llms learn new skills in RL by composing old ones. *CoRR*, abs/2509.25123.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *CoRR*, abs/2504.13837.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, Wanli Ouyang, and Dongzhan Zhou. 2025a. Llama-berry: Pairwise optimization for olympiad-level mathematical reasoning via o1-like monte carlo tree search. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 7315–7337. Association for Computational Linguistics.

Kaiyi Zhang, Ang Lv, Jinpeng Li, Yongbo Wang, Feng Wang, Haoyuan Hu, and Rui Yan. 2025b. Stephint: Multi-level stepwise hints enhance reinforcement learning to reason. *CoRR*, abs/2507.02841.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. 2025c. Scaf-grpo: Scaffolded group relative policy optimization for enhancing LLM reasoning. *CoRR*, abs/2510.19807.

Rongzhi Zhu, Xiangyu Liu, Zequn Sun, Yiwei Wang, and Wei Hu. 2025a. Mitigating lost-in-retrieval problems in retrieval augmented multi-hop question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 22362–22375. Association for Computational Linguistics.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. 2025b. The surprising effectiveness of negative reinforcement in LLM reasoning. *CoRR*, abs/2506.01347.

A Detailed Related Work

Improving LLMs via Value-based Guidance. RLVR utilizes reward scores to supervise LRMs (Jaech et al., 2024; DeepSeek-AI, 2025; Team et al., 2025). During the RLVR process, the reward design (Arnal et al., 2025; Su et al., 2025), advantage shaping (Deng et al., 2025; Chen et al., 2025d; Cheng et al., 2025), and sampling mechanism (Wang et al., 2025a; Zhu et al., 2025b) are leveraged to balance the exploration and exploitation of LRMs, and guide them to find the global optimal. In addition to one-pass generation, previous works (Yao et al., 2023; Besta et al., 2024; Chen et al., 2024; Jiang et al., 2024) have employed reward models to provide supervision signals for each reasoning step. Besides, existing work (Zhang et al., 2025a) also adopts outcome-level reward models to guide LRMs to find the better solution. Given that models can easily solve simple problems but struggle with multi-hop tasks (Zhang et al., 2025c; Xue et al., 2024; Peng et al., 2024), this work focuses on how to guide the model to transfer its ability from solving sub-questions toward solving more complex problems, thereby enhancing the effectiveness of RLVR.

Test-time Scaling via Natural Language Guidance. Supervised finetuning (SFT) is a kind of natural language guidance to elicit inner knowledge of LRMs and improve their exploration ability (Hugging Face, 2025; Guha et al., 2025). Moreover, previous studies have shown that responses from the teacher model (Yan et al., 2025; Zhang et al., 2025c) and self-reflection (Sareen et al., 2025; Jiang et al., 2025a) can be used to elicit the reasoning ability of LRMs. Besides, parallel reasoning enhances the performance of LRMs by enabling the model to explore multiple reasoning paths (Qi et al., 2025; Snell et al., 2024; Luo et al., 2025). , which is an alternative way of leveraging natural language guidance. In this work, we investigate the usage of sub-questions during RLVR to aid the model in accurately solving these problems and to activate its internal knowledge.

Compositional Generalization Ability of LLMs. Compositional generalization ability refers to the ability to combine known simple knowledge to solve more complex problems (Liu et al., 2020; An et al., 2023). However, the compositional generalization ability of LLMs is limited (Andreas et al., 2016; Sakai et al., 2025; Yang et al., 2024).

Table 5: Performance of Qwen2.5-7B-Ins on various domain tasks.

	MMUL-Pro	Enigmata	Avg.
Baseline	56.3	4.7	30.5
+ GRPO	62.3	12.9	37.6
+ A^2D (Ours)	**65.1**	**20.8**	**43.0**

Specifically, the model has acquired some basic knowledge, but it still cannot combine these related pieces of knowledge to solve more complex tasks, *e.g.,* puzzle tasks (Chen et al., 2025d,a) and question answering tasks (Jin et al., 2025; Yang et al., 2018). To improve such an ability, prior studies adopt several simple problems to synthesize multi-hop problems and utilize these problems to train the models (Yuan et al., 2025; Sun et al., 2025b), or guide the model to perform multi-turn reasoning on these problems (Jiang et al., 2025b; Zhu et al., 2025a). The compositional generalization ability is vital for LLM self-evolution (Sakai et al., 2025; Yang et al., 2024), where LLMs should explore the potential solutions to the given question and learn from these experiences (Zhang et al., 2025c,b). Inspired by this, in this work, we focus on teaching LLMs to combine the knowledge it has learned to answer complex questions in RLVR.

B Discussion about Computational Cost

During experiments, we observe that the decomposer requires only a small number of training steps to converge. To ensure a fair comparison with other baselines, we reduce the number of training steps in A D and conduct comparisons under similar computational budgets.

Concretely, based on Qwen2.5-7B-Instruct, GRPO cost 735.1 GPU hours (about 800 Steps), while A D cost 741.2 GPU hours (training decomposer cost 120.9 GPU hours and training reasoner costs 620.3 GPU hours). Under comparable training time, A^2 D (26.5%) outperforms the baseline methods (23.4% for GRPO), demonstrating its effectiveness.

These results and analyses further strengthen our contributions. We will incorporate the above discussion on computational cost into our paper.

C Evaluation on Various Domain Tasks

We focus on mathematical tasks and conduct evaluations on multiple mathematical tasks. These tasks cover a wide range of difficulty levels, which is

sufficient to show the effectiveness of A^2D.

Moreover, we conduct MMLU-Pro (**?**) (scientific tasks) and Enigmata (Chen et al., 2025a) (logical reasoning tasks) for evaluation. The evaluation results are presented in Table 5. After GRPO, Qwen2.5-7B-Ins achieves 62.3% and 12.9% on MMLU-Pro and Enigmata, respectively. After A^2D, Qwen2.5-7B-Ins achieves 65.1% and 20.8% on MMLU-Pro and Enigmata, respectively. The results further show that A^2D is effective across a wide range of domains.

D Implementation of Empirical Study

We utilize Seed-1.6 (ByteDance-Seed, 2025b) to decompose the original questions into several simple sub-questions. To guide the Seed-1.6 to decompose the questions, we employ the following template to construct the prompt that will be fed into the model.

Prompt for Seed-1.6
{QUESTION}
Please reason step-by-step and decompose this question into several sub-questions. You only provide the sub-questions and do not provide the any details reasoning steps and solution. Use the format "<subquestion>" and "<\subquestion>" to identify each sub-question.

To prevent the model from first solving the problem and then generating subproblems based on the solution, we disable its reasoning function and limit its generation length to 512 tokens, forcing the model to generate subproblems intuitively without deliberate reasoning. Moreover, we set the hyperparameters of the decoding process `temperature` and `top_p` as 1.0 and 1.0, respectively.

E Implementation Details of Experiment in Sec. 4

We adopt the Qwen2.5-7B-Instruct as the backbone model of the decomposer and the proxy reasoner, and employ GRPO (Shao et al., 2024) as the backbone RLVR algorithm for the whole training process. For the training process of the decomposer, we guide the proxy reasoner to generate 4 solutions to compute the quality reward. For the training process of the reasoner, we adopt $k_1 = 0.1$ and $k_2 = 0.2$ for in-context distillation loss.

To enhance the effectiveness, we follow the previous work (Yu et al., 2025) to set the clip ratio $\varepsilon_{\text{low}} = 0.2$ and $\varepsilon_{\text{high}} = 0.28$. Moreover, we set the `max_prompt_length` and `max_response_length` as 2048 and 6144, and set the `train_batch_size` and `mini_ppo_batch_size` as 128 and 32, respectively. For each question, we perform 32 times rollout process to conduct the experience for RLVR training, and utilize 1.0 and 1.0 as the `temperature` and `top_p`. For the evaluation procedure, we adopt the `temperature` as 1.0 and `top_p` as 1.0 for the decoding process. Besides, to reduce the influence of random values, we repeat the experiment 8 times and report the average accuracy.

F Prompt Templates

In this part, we introduce the details of the prompt templates that are utilized in out experiments.

Below is the prompt used to train the decomposer model, where the placeholder "{QUESTION}" is replaced with the corresponding question x_i.

Prompt for Decomposer
{QUESTION}
Please reason step-by-step and decompose this question into several sub-questions. You only provide the sub-questions and do not provide the any details reasoning steps and solution. Use the format "<subquestion>" and "<\subquestion>" to identify each sub-question.

Below is the prompt used to guide the reasoner to perform reasoning, which is employed in the objective function $\mathcal{J}(\theta_R)$ in Eq. 1. The placeholder "`{QUESTION}`" is replaced with the corresponding question x_i.

Vanilla Prompt for Reasoner
{QUESTION}
Please reason step by step, and put your final answer within `\\boxed{}`.

Below is the prompt used by the reasoner for inference under sub-question guidance. The placeholder "`{QUESTION}`" is replaced with the corresponding original question x_i, and "`{LIST OF SUB-QUESTIONS}`" is replaced with the decomposed sub-questions $\mathcal{S}_i$.

Prompt with Sub-question Guidance for Reasoner

{QUESTION}

Now, I give you some tips and you can refer to the provided tips to solve the problem. Tips:
{LIST OF SUB-QUESTION}

Please reason step by step, and put your final answer within \\boxed{}.

Below is the prompt used during IDN Loss training, and the placeholder "QUESTION is replaced with the corresponding question.

IDN Loss Prompt for Reasoner

{QUESTION}

You can try to reason step by step through different approaches, and try to explore the correct the solution. You should put your final answer within \\boxed{}.

G Details of the Format Reward for Decomposer Training

We enforce a series of constraints to verify whether the model responses follow the required format.

- The model's generated response is required to begin with the "<subquestion>" tag.
- Each sub-question is not allowed to contain more than one "<subquestion>" tag.
- The model's generated response must contain more than 10 characters.

H Pseudo Code of A^{2}D

In Algorithm 1, we present the pseudocode the our A^{2}D to better demonstrate its training pipeline.

I Further Analysis

I.1 Case Study of the Generated Sub-questions from Different Models

In Fig. 4, we present the case study of the generated sub-questions of different models, *i.e.,* our method (*i.e.,* A^{2}D) and the approach that removes the Format Reward while using the Pass@1 Reward to replace the Pass@k Reward (*i.e.,* A^{2}D w/o Format Reward w/ Pss@1 Reward).

The example problem involves sets and requires knowledge of intersections and unions. The model trained with our method decomposes this problem into two sub-questions: first, it identifies that the problem involves sets, guiding the reasoner to select relevant knowledge for solving the problem; then, it guides the reasoner to apply this knowledge to the specific question. The Decomposer performs a relatively coarse-grained decomposition of the problem, which can inspire the reasoner to explore while preventing over-reliance on the hints. This phenomenon reflects the effect of using the Pass@k Reward during training.

In contrast, the model trained by removing the Format Reward and using the Pass@1 Reward instead of the Pass@k Reward produces more concrete content, with the sub-questions degenerating into step-by-step reasoning solutions. This phenomenon indicates that high-level, coarse-grained hints can guide the model's exploration, thereby improving its Pass@k performance. In contrast, fine-grained hints better enhance the model's exploitative capability, leading to higher Pass@1 performance. The analysis of the model's generated content validates the rationale of our training method.

I.2 Significance Test

To enhance the persuasiveness of our experiments, we adopt the commonly used settings from previous works (Chen et al., 2025c; Wang et al., 2025a; Hu et al., 2025b), *i.e.,* conducting the experiments multiple times and reporting the average result of these repeated trials as the final score of the model, which has been report in Appendix E. Multiple replicate experiments are conducted to eliminate performance fluctuations caused by randomness, thereby validating the effectiveness of the algorithm.

To further demonstrate the effectiveness of our algorithm, we conduct a significance test and present the results in Table 6. Through the validity tests, we find that the performance improvements of our algorithm over previous works are all greater than one standard deviation. This phenomenon further demonstrates that the improvement in model performance stems from the enhancements to our algorithm, rather than random errors.

I.3 Analysis of the Generated Sub-questions

To further understand the feature of the decomposer, we collect the generated sub-questions from decomposer for four dataset, *i.e.,* AIME24,

Algorithm 1: The Pseudo Code for A^{2}D.

Input : A backbone model π_θ and the training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$.
Output : A powerful reasoner π_{θ_R}.

```
# (1) Training Decomposer π_θD via RLVR.
```
Initialize the decomposer π_{θ_D} and proxy reasoner using the backbone model π_θ.
for *each training step* **do**
  Prompt the decomposer π_{θ_D} to decompose the questions in the training dataset.
  Compute the quality reward of the decomposed question R_Q by the proxy reasoner.
  Compute the format reward of the decomposed question R_F by rules in Appendix G.
  Utilize Eq. 1 to optimize the decomposer π_{θ_D} based on the reward $R = R_\text{Q} \times R_\text{F}$.
Obtain a decomposer π_{θ_D}, which can decompose the complex question into several simpler sub-questions.

```
# (2) Decompose the Complex Questions.
```
for *each instance* $\langle x_i, y_i \rangle$ *in training dataset* $\mathcal{D}$ **do**
  Use decomposer π_{θ_D} to decompose the question x_i, obtaining a list of sub-questions $\mathcal{S}_i$.
  Add the triple $\langle x_i, y_i, \mathcal{S}_i \rangle$ into the training dataset of the reasoner $\mathcal{D}_\text{SQ}$.
Obtain the training dataset of the reasoner $\mathcal{D}_\text{SQ} = \{\langle x_i, y_i, \mathcal{S}_i \rangle\}_{i=1}^n$.

```
# (3) Training Reasoner π_θR through RLVR with Sub-question Guidance.
```
Initialize the reasoner π_{θ_R} using the backbone model π_θ.
for *each instance* $\langle x_i, y_i, \mathcal{S} \rangle$ *in training dataset* $\mathcal{S}_{SQ}$ **do**
  Prompt the reasoner π_{θ_R} to generate the solutions $\hat{\mathcal{Y}}$, based on the question x_i.
  Compute the loss function of RLVR $\mathcal{J}(\theta_\text{R})$ using Eq. 1, based on $\hat{\mathcal{Y}}$.
  Prompt the reasoner π_{θ_R} to generate the solutions $\hat{\mathcal{Y}}'$, based on the question x_i and the sub-questions $\mathcal{S}_i$.
  Process the generated solutions $\hat{\mathcal{Y}}'$ through Eq. 4.
  Compute the in-context distillation loss $\mathcal{L}'_\text{IDL}(\theta_\text{R})$ using Eq. 5, based on $\hat{\mathcal{Y}}'$.
  Utilize Eq. 6 to optimize the reasoner π_{θ_R} based on $\mathcal{J}(\theta_\text{R})$ and $\mathcal{L}'_\text{IDL}(\theta_\text{R})$.
Obtain the powerful reasoner π_{θ_R}.

Table 6: The mean and standard deviation of multiple runs for different algorithms on AIME24 and AIME25.

Methods	AIME24		AIME25		MATH500		Minerva	
	mean@8	std@8	mean@8	std@8	mean@8	std@8	mean@8	std@8
Qwen2.5-7B-Instruct								
Baseline	9.6	-	6.3	-	70.8	-	22.9	-
+ GRPO	15.0	8.6	8.3	8.3	73.3	0.9	24.8	3.3
+ LUFFY	10.4	7.6	8.8	6.6	69.2	1.6	22.4	5.4
+ Scaf-GRPO	14.6	4.9	9.6	7.9	74.5	0.8	25.1	6.2
+ A^{2}D (Ours)	20.0	4.5	17.1	6.1	75.6	1.0	28.9	4.7

AIME25, MATH500, and Minerva, and present the statistical information about the generated sub-questions in Table 7.

First, by examining the number of sub-questions produced by the decomposer and the number of generated tokens, we observe that both remain at relatively low levels. Specifically, on average, a question can be decomposed into two sub-questions, and describing these sub-questions requires only about 60 tokens. This phenomenon indicates that the sub-questions produced by the decomposer operate at a relatively high level of abstraction and can be regarded as a coarse-grained reasoning process rather than a step-by-step solution.

Moreover, we find that the responses generated by the decomposer do not contain keywords related to the final answer, indicating that the decomposer provides problem-solving guidance without revealing the solution itself, thereby not constraining the reasoner's reasoning and exploratory process. Such a decomposer can provide the reasoner with guidance and inspiration while retaining a certain degree of exploratory capability, which is precisely what we aim for. The decomposer does not engage in overly detailed reasoning that would excessively reinforce exploitative behavior and consequently diminish its exploratory capacity.

Question

In a class of 50 students, 28 participate in MATHCOUNTS, 21 participate in science club, and 6 students participate in neither. How many students participate in both MATHCOUNTS and science club?

A^2D

<subquestion>How can the given information be manipulated to apply principles of set theory for a solution?
\subquestion>How can the given conditions be used to set up equations or relationships between the given sets?

A^2D w/o Format Reward w/ Pass@1 Reward

<subquestion>First, we need to determine how many students participate in at least one of the activities (MATHCOUNTS or the science club). Since there are 50 students in total and 6 do not participate in either, we can calculate the number of students involved in at least one activity as follows:

$$50 - 6 = 44$$

So, 44 students are either involved in MATHCOUNTS, science club, or both.

Let's denote:

- (|M|) as the number of students who participate in MATHCOUNTS, which is 28.
- (|S|) as the number of students who participate in the science club, which is 21.
- (|M ∩ S|) as the number of students who participate in both MATHCOUNTS and the science club.
- (|M ∪ S|) as the number of students who participate in either MATHCOUNTS or the science club or both, which we determined to be (44).

According to the principle of inclusion and exclusion, we know:

$$|M \cup S| = |M| + |S| - |M \cap S|$$

Substituting what we know:

$$44 = 28 + 21 - |M \cap S|$$

Now we solve for (|M ∩ S|):

$$44 = 49 - |M \cap S|$$
$$|M \cap S| = 49 - 44$$
$$|M \cap S| = 5$$

Thus, the number of students who participate in both MATHCOUNTS and the science club is $\boxed{5}$.

Figure 4: An example of the generated sub-question from models trained through different methods.

Table 7: The statistical information of the generated sub-questions from A^2D.

	AIME24	AIME25	MATH500	Minerva
Number of Sub-questions				
Average value	2.07	2.1	2.02	1.99
Standard deviation	0.25	0.4	0.15	0.28
Maximum value	3.00	4.00	3.00	5.00
Minimum value	2.00	2.00	1.00	1.00
Number of Tokens of Generated Sub-questions				
Average value	60.37	58.33	50.08	53.01
Standard deviation	23.93	22.19	15.99	17.33
Maximum value	115.00	154.00	130.00	167.00
Minimum value	31.00	36.00	26.00	16.00
Contained Content				
Whether contain "`\boxed{}`"	✗	✗	✗	✗
Whether contain "`Answer:`"	✗	✗	✗	✗
Whether contain "`Answer is`"	✗	✗	✗	✗