Process Rewarding via Solution-Space Alignment

Anonymous ACL submission

Abstract

Reinforcement learning with verifiable or outcome-level rewards has recently become an effective post-training paradigm for improving the reasoning abilities of large language models, with GRPO-style optimization achieving strong results on mathematical and other verifiable tasks. However, for multi-step reasoning, outcome-level supervision remains intrinsically sparse: a trajectory that contains partially correct intermediate derivations but makes a late mistake is still treated as a complete failure. As a result, useful reasoning prefixes are penalized together with erroneous continuations, and informative failed trajectories are underutilized. PRMs provide denser step-level feedback, but they often rely on costly process annotation or auxiliary process verifiers, and optimizing local process rewards alone may fail to preserve the global objective of final correctness, leaving room for reward misspecification and reward hacking during reinforcement fine-tuning. We propose PRSA(Process Rewarding via Solution-Space Alignment), a simple framework that bridges outcome supervision and process supervision without requiring human step-level labels. PRSA automatically extracts intermediate-value steps from model outputs and reference solutions, aligns them within the solution space under equivalence and order constraints to estimate reasoning progress, and uses this signal for both reward shaping and token-level prefix-suffix credit assignment. In this way, PRSA preserves the global anchor of final-answer correctness while providing denser and better localized learning signals for partially correct trajectories.

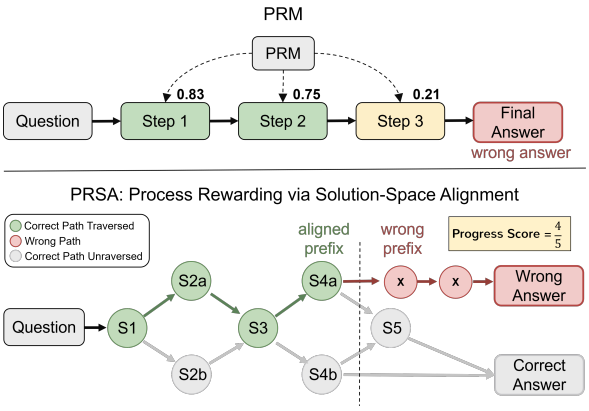

Figure 1: Illustration of PRSA. Top: PRM-style stepwise supervision can assign positive local scores to intermediate steps even when the overall trajectory leads to a wrong final answer, revealing a mismatch between local plausibility and global correctness. Bottom: PRSA measures progress by matching a sampled reasoning trajectory to valid intermediate states in the solution space. A partially correct trajectory therefore receives credit for its aligned prefix, while the diverged suffix is identified for suppression.

1 Introduction

Large language models have shown increasingly strong multi-step reasoning abilities through chain-of-thought prompting and rationale bootstrapping (Wei et al., 2023; Zelikman et al., 2022). In parallel, reinforcement-learning-based post-training, rooted in preference learning and PPO-style policy optimization (Christiano et al., 2023; Schulman et al., 2017; Ouyang et al., 2022), has become an important route for improving reasoning under verifiable rewards. In particular, GRPO, introduced in DeepSeekMath and later adopted in DeepSeek-R1, demonstrates that substantial gains can be obtained from simple outcome-level rewards on mathematics and related reasoning tasks (Shao et al., 2024; Guo et al., 2025a). More broadly, recent work on reinforcement learning with verifiable rewards has renewed interest in whether outcome-level RL truly improves reasoning rather than merely reweighting existing outputs (Wen et al., 2025).

Despite these advances, standard outcome-level

optimization remains a blunt instrument for multi-step reasoning. In GRPO-style training, supervision is largely determined by a scalar reward assigned to the final response. Consequently, a trajectory may follow the correct derivation for most of its length and deviate only near the end, yet once the final answer is wrong, the entire response receives an unfavorable update. This creates two related problems: *reward sparsity*, because failed trajectories provide little information beyond the fact that the final answer is incorrect, and *misaligned credit assignment*, because correct reasoning prefixes are penalized together with later erroneous suffixes. Recent work on process supervision and fine-grained evaluation has repeatedly highlighted the mismatch between final-answer correctness and process correctness in mathematical reasoning (Lightman et al., 2023; Yu et al., 2024; Luo et al., 2024; Guo et al., 2025b).

A natural remedy is process supervision. Step-level supervision has been shown to outperform pure outcome supervision for mathematical reasoning (Lightman et al., 2023), motivating scalable alternatives to manual annotation, including automatically constructed process rewards (Wang et al., 2024), search-derived process labels (Zhang et al., 2024), automated process supervision pipelines (Luo et al., 2024), and process verifiers based on stepwise progress (Setlur et al., 2024). However, PRMs do not fully solve the problem: they often still rely on expensive labels or complex auxiliary machinery (Lightman et al., 2023; Wang et al., 2024; Zhang et al., 2024; Luo et al., 2024; Setlur et al., 2024), and optimizing local process rewards may not preserve global task success, leaving room for reward hacking during RL fine-tuning (Cheng et al., 2025). Moreover, recent studies show that answer-only supervision is insufficient for evaluating reasoning, since correct final answers can still conceal flawed intermediate steps (Wen et al., 2025; Guo et al., 2025b). These limitations motivate methods that retain final-answer correctness as the global objective while still exploiting partially correct intermediate reasoning.

In this paper, we propose **PRSA** (**P**rocess **R**ewarding via **S**olution-**S**pace **A**lignment), a simple framework that combines the strengths of outcome supervision and process-aware learning. Instead of learning a dedicated PRM from dense step labels, PRSA automatically extracts intermediate-value steps from sampled solutions and reference solutions, aligns them within a shared solution space under equivalence and order constraints, and treats the resulting alignment depth as a proxy for reasoning progress. This progress signal is then used in two complementary ways: it provides shaped rewards for trajectories that are partially correct but ultimately fail, and it defines a prefix-suffix boundary for token-level credit reassignment so that aligned prefixes are protected while diverged suffixes are selectively suppressed. In this way, PRSA occupies a middle ground between outcome-level GRPO and learned PRMs: it retains the global anchor of final-answer correctness, while injecting process sensitivity precisely where outcome-level RL is weakest.

Our contributions are summarized as follows:

- We identify a key limitation of GRPO-style outcome-level reinforcement learning for multi-step reasoning: the lack of intermediate-state awareness can lead to coarse or mislocalized credit assignment.
- We propose PRSA, a process-aware framework that derives dense progress signals from reference solutions without requiring step-level human annotation.
- Experiments on mathematical reasoning benchmarks show that PRSA consistently outperforms GRPO and strong baselines, achieving state-of-the-art results in our setting.

2 Related Work

2.1 Reasoning Elicitation and Reinforcement Learning for Reasoning

Recent progress in complex reasoning has been driven first by advances in reasoning elicitation. Chain-of-thought prompting shows that explicitly exposing intermediate reasoning steps can substantially improve performance on multi-step tasks, while rationale bootstrapping further demonstrates that such reasoning traces can be iteratively expanded and reused for training (Wei et al., 2023; Zelikman et al., 2022). Together, these works suggest that reasoning ability is not only a property of the model itself, but can also be revealed or amplified by appropriately structuring intermediate computation.

Beyond prompting and supervised rationale construction, reinforcement-learning-based post-training has become a central paradigm for improv-

ing reasoning ability in verifiable domains (Christiano et al., 2023; Schulman et al., 2017; Ouyang et al., 2022). In particular, GRPO-style optimization has shown strong effectiveness on mathematical reasoning by using simple outcome-level rewards derived from final-answer correctness (Shao et al., 2024; Guo et al., 2025a). However, such outcome-level RL provides only sparse trajectory-level supervision, making it difficult to exploit partially correct reasoning paths and often resulting in coarse credit assignment across long reasoning trajectories.

2.2 Process Supervision and Fine-Grained Credit Assignment

To address the limitations of outcome-only supervision, a growing line of work studies process supervision and process reward modeling. *Let's Verify Step by Step* shows that step-level supervision can outperform outcome supervision for mathematical reasoning (Lightman et al., 2023). Subsequent works aim to scale this idea while reducing reliance on expensive human annotation, including automatically constructed process supervision in Math-Shepherd (Wang et al., 2024), search-based process labeling in ReST-MCTS* (Zhang et al., 2024), automated pipelines such as OmegaPRM (Luo et al., 2024), and progress-based process verifiers that reward intermediate advancement rather than only final success (Setlur et al., 2024).

Related work also studies alternative ways to assign fine-grained credit, including outcome-supervised value estimation for partial reasoning states (Yu et al., 2024) and recent tree-sampling-based RL methods such as TreeRPO, which seek to provide denser step-level learning signals without relying on an explicitly trained process reward model (Yang et al., 2025). At the same time, prior studies highlight important failure modes of process reward optimization, such as reward hacking and mis-specified local objectives (Cheng et al., 2025), and recent analyses further argue that final-answer correctness alone is insufficient for evaluating reasoning quality because correct answers may still conceal flawed intermediate steps (Wen et al., 2025; Guo et al., 2025b). Different from prior PRM-based approaches, our method does not train a separate process reward model. Instead, it extracts intermediate-value steps from sampled and reference solutions, uses step matching to derive a lightweight progress signal, and incorporates this signal into both reward shaping and token-level credit reassignment.

3 PRSA: Methodology

PRSA extends outcome-level GRPO with automatically derived process supervision. It builds a solution-space graph from valid derivations, estimates partial reasoning progress by aligning sampled intermediate steps to this graph, and uses the resulting signal for both trajectory-level reward shaping and token-level credit reallocation. The overall procedure is summarized in Algorithm 1.

3.1 Problem Setup

Let $\mathcal{D} = \{(x_u, u)\}$ denote a set of reasoning problems, where x_u is an input question and u is its unique identifier. For each problem, the policy π_θ samples a group of trajectories

$$\mathcal{Y}_u = \{y^{(1)}, \ldots, y^{(G)}\}, \qquad y^{(n)} \sim \pi_\theta(\cdot \mid x_u),$$

where each trajectory contains intermediate reasoning and a final answer. A verifier assigns a final reward

$$R^{(n)} = R(x_u, y^{(n)}),$$

which typically depends only on final-answer correctness.

Standard GRPO applies this sequence-level signal uniformly across all tokens, which is often too coarse for multi-step reasoning. PRSA augments it with a process-aware signal, constructed without human step annotations, by aligning intermediate-value steps within a feasible solution space to estimate partial progress and using that signal for both reward shaping and token-level credit assignment.

3.2 Solution-Space Graph

For each problem, PRSA collects a set of *valid derivations* from final-correct solutions. These solutions may come from successful rollouts observed during training, or from an external teacher when no correct sample is available in the current batch. From each valid derivation, we extract an ordered sequence of intermediate-value steps, such as equality-bearing expressions or explicit intermediate results, using deterministic rules detailed in Appendix A.2. To improve robustness, these intermediate values are normalized so that mathematically equivalent expressions written in different surface forms can be treated as the same state.

The normalized derivations are then organized into a structured *solution-space graph*, which represents the feasible reasoning space for the problem.

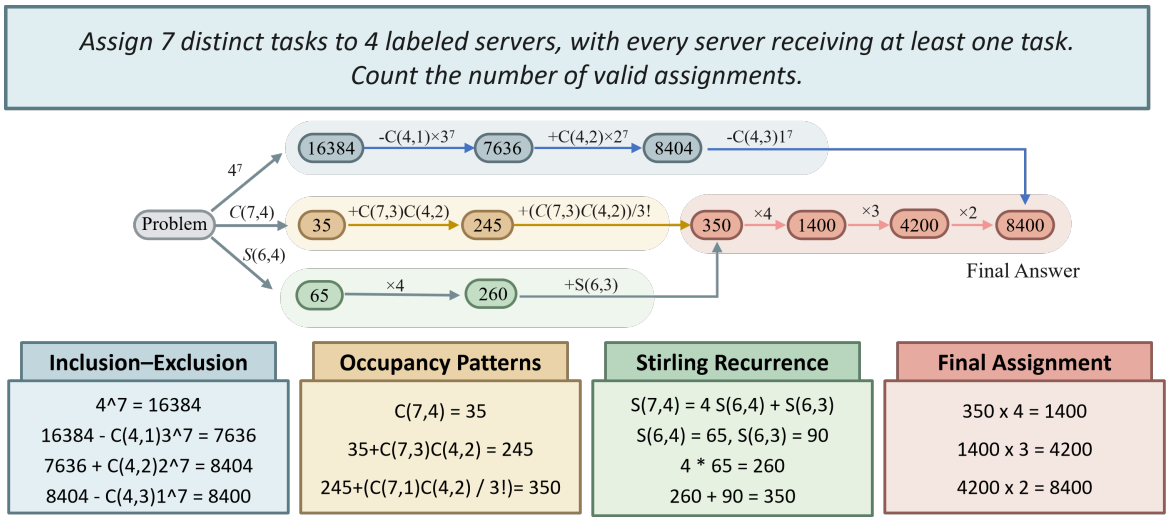

Figure 2: An example of a solution-space graph. Multiple valid reasoning paths can be organized into a structured solution space that leads to the same final answer.

Each valid derivation corresponds to a path in the graph, while nodes represent normalized intermediate states and edges connect states that appear consecutively in a correct solution. This construction allows multiple correct reasoning paths to coexist in a single representation, rather than forcing the model to follow one canonical chain of thought. As a result, PRSA measures progress against a structured space of valid intermediate states instead of a single reference rationale. An illustrative example is shown in Fig. 2.

3.3 Progress Scoring

Given a sampled trajectory, we extract its intermediate-value sequence and compare it with the solution-space graph through *window-constrained equivalence alignment*. Concretely, a predicted intermediate value at position i is considered aligned to a valid state at position j only if they are equivalent after normalization and their positions are locally consistent:

$$\phi(p_i) = \phi(r_j), \qquad |i - j| \leq w,$$

where w is a matching window. This local constraint suppresses accidental matches caused by repeated values or disordered reasoning.

Based on the alignment result, we measure how far the sampled trajectory has progressed along the feasible solution space. For each valid reference trajectory induced by the graph, we record the farthest matched step position and compute a normalized progress ratio as the matched prefix length divided by the total number of reference steps. The final progress score is obtained by averaging this ratio over all matched references:

$$\text{Progress}(y, \mathcal{G}_u) = \frac{1}{|\mathcal{R}'|} \sum_{r \in \mathcal{R}'} \frac{m_r}{L_r},$$

where $\mathcal{R}'$ denotes the set of reference trajectories with at least one matched step, m_r is the farthest matched position in reference trajectory r, and L_r is the total number of extracted steps in r. Intuitively, this score measures partial reasoning progress rather than simple step overlap.

We then convert progress into a trajectory-level training score. If the final answer is correct, we keep the original verifier reward. If the final answer is wrong but the trajectory still aligns with part of the feasible solution space, we assign a compressed shaped reward:

$$S = \begin{cases} R, & \text{if answer is correct,} \\ \alpha \cdot \text{Progress}(y, \mathcal{G}_u), & \text{otherwise,} \end{cases}$$

where $\alpha \in (0, 1)$ controls the strength of progress-based shaping and prevents partially correct but ultimately wrong solutions from dominating the learning signal. If no valid graph is available, or no reference trajectory can be matched, PRSA simply falls back to the original outcome reward.

3.4 Token Credit Reallocation

Trajectory-level shaping alone is insufficient for fine-grained credit assignment, because it still leaves open *which part* of a response should be reinforced or suppressed. To address this, PRSA uses

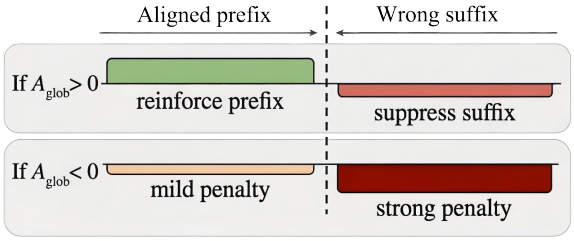

Figure 3: Illustration of token credit reallocation in PRSA.

the last aligned intermediate step to divide each trajectory into two contiguous regions: an aligned *prefix*, consisting of tokens generated before the final aligned step boundary, and a diverged *suffix*, consisting of tokens generated after the trajectory has departed from the feasible solution space. This partition allows the training signal to distinguish between reasoning that is still consistent with a valid derivation and reasoning that has already gone off track. As illustrated in Fig. 3, PRSA applies asymmetric token-level credit assignment: aligned prefix tokens are treated more conservatively, while diverged suffix tokens receive stronger suppression.

For each sample group, we first normalize the trajectory-level score S into a standard GRPO-style global advantage

$$A_{\text{glob}} = \frac{S - \mu_g}{\sigma_g + \epsilon},$$

where μ_g and σ_g are the mean and standard deviation of the group scores. We then decompose the global advantage into its positive and negative parts,

$$A^+ = \max(A_{\text{glob}}, 0), \qquad A^- = \max(-A_{\text{glob}}, 0),$$

so that positive and negative updates can be treated asymmetrically. Intuitively, when a trajectory receives a positive group-level signal, the aligned prefix should be reinforced while the diverged suffix should still be discouraged; when the trajectory receives a negative signal, the whole response should not be penalized uniformly, because the aligned prefix may still reflect valid reasoning progress and should therefore receive only a reduced negative update.

Accordingly, PRSA defines the token-level advantage by assigning different values to the prefix and suffix:

$$A_t = \begin{cases} A_{\text{pre}} = A^+ - \eta_{\text{pre}} A^-, & t \in \mathcal{T}_{\text{pre}}, \\ A_{\text{suf}} = -\lambda_{\text{suf}} |A_{\text{glob}}|, & t \in \mathcal{T}_{\text{suf}}, \end{cases}$$

Algorithm 1 PRSA for one training group

Require: Problem x_u, sampled trajectories $\mathcal{Y}_u = \{y^{(n)}\}_{n=1}^G$, verifier rewards $\{R^{(n)}\}_{n=1}^G$

Build a solution-space graph $\mathcal{G}_u$ from valid derivations of x_u
for each sampled trajectory $y^{(n)}$ **do**
    Extract intermediate-value steps from $y^{(n)}$
    Align them to $\mathcal{G}_u$ under equivalence and window constraints
    Compute progress score $\text{Prog}^{(n)}$
    Convert it into shaped score $S^{(n)}$
    Identify the last aligned step and split tokens into prefix/suffix
end for
Normalize $\{S^{(n)}\}$ within the group to obtain global advantages
Reallocate token-level advantages based on prefix/suffix boundaries
Update the policy with the standard GRPO objective

where $\eta_{\text{pre}} \in (0, 1)$ reduces negative updates on the aligned prefix, and $\lambda_{\text{suf}} > 0$ controls the penalty applied to the diverged suffix. The resulting token-level credit assignment is intentionally asymmetric: positively scored trajectories reinforce the aligned prefix while continuing to suppress the suffix, whereas negatively scored trajectories do not penalize all tokens equally. Instead, tokens in the aligned prefix receive only a milder negative update, while the main penalty is concentrated on the suffix, which is more likely to contain the actual reasoning error. This allows PRSA to preserve useful partial progress even in failed trajectories. In practice, we choose $\eta_{\text{pre}} < \lambda_{\text{suf}}$ so that suffix tokens are penalized more strongly than prefix tokens.

4 Experiment

We evaluate PRSA on diverse math reasoning benchmarks spanning high-school contests to olympiad-level problems. Across settings, PRSA delivers consistent pass@1 gains over GRPO and other strong baselines under sparse verifier rewards, and the improvements generalize across model scales and benchmark difficulty. We further conduct analyses on cross-backbone transfer, teacher-assisted solution-space recovery in all-zero rollout groups, and the contributions of progress-based re-

ward shaping and token-level prefix-suffix credit reallocation.

4.1 Experimental Setup

Benchmarks. We evaluate on six math reasoning benchmarks: AIME24, AIME25, AMC23, MATH500 (Hendrycks et al., 2021), Minerva (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024), covering competition-style problems from high-school to olympiad level. In a text-only setting, we use the same temperature and a maximum generation length that avoids truncating valid solutions, and report the average performance across multiple independent runs. Concretely, we use Avg@32 for AIME24, AIME25, and AMC23, and Avg@4 for the remaining benchmarks.

Compared Methods. We compare PRSA with RLVR-based baselines implemented in a shared framework. *BaseModel* is the backbone without RLVR fine-tuning, and *GRPO* is standard Group Relative Policy Optimization. We further include Dr.GRPO (Liu et al., 2025), GPG (Chu et al., 2025), DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025), GRPO-PRM, which adds the average process reward to the outcome reward, TreeRPO(Yang et al., 2025), GRPO-lead (Zhang and Zuo, 2025) and DGPO(Dai et al., 2026). All methods use the same data, verifier, and decoding settings.

Implementation Details. We fine-tune Qwen2.5-Math-7B (Yang et al., 2024) on the MATH dataset (Hendrycks et al., 2021) with our PRSA algorithm under a standard reasoning-oriented RLVR setup (Liu et al., 2025). To test model-agnosticism, we also train Qwen2.5-Math-1.5B and Qwen2.5-3B with the same training split, preprocessing, and prompt templates as prior RLVR/GRPO-style work (Liu et al., 2025). All experiments run on 8 NVIDIA A100 GPUs using a modified `verl-0.4.0` GRPO implementation. For PRM-based scoring, we use Qwen2.5-Math-PRM-7B as the process reward model following the official guidance, with its outputs normalized to the $[0, 1]$ range.

During RL training, we set the sampling temperature to 0.6 and roll out 8 responses for each problem. The learning rate is set to 1×10^{-6}. The coefficients for the KL divergence loss and entropy loss are both set to 0.001. The GRPO training batch size is 128 and the mini-batch size is 64. Unless otherwise specified, the maximum prompt length is 4096 and the maximum response length is 8192. For PRSA, we set $\lambda_{\text{suf}} = 0.5$ and $\alpha = 0.5$. For each backbone and method, we share the same training schedule. PRSA adds no learnable parameters and only introduces lightweight solution-space alignment during training, so its computational cost is comparable to GRPO under the same sampling budget. Full hyperparameters, prompt formats, and verifier details are provided in Appendix.

4.2 Main Results

Table 1 shows that PRSA delivers the strongest overall performance among all GRPO-style objectives. On Qwen2.5-Math-7B, PRSA achieves an average pass@1 of 40.69, improving over vanilla GRPO by +3.08 points and surpassing the strongest competing baseline by a clear margin. More importantly, this gain is not concentrated on only one benchmark, but appears consistently across six math reasoning datasets. Compared with GRPO, PRSA improves AIME24, AIME25, AMC23, MATH500, Minerva, and Olympiad by +1.50, +2.59, +1.43, +4.50, +5.97, and +2.51 points, respectively. This supports our central claim that pure outcome-level optimization is insufficient for long-horizon mathematical reasoning, because it collapses useful intermediate progress into a single binary signal and therefore cannot distinguish between an almost-correct derivation and a completely uninformative failure.

A closer look at the results also helps explain why PRSA is stronger than PRM-based variants such as GRPO-PRM. Although GRPO-PRM introduces denser supervision, its average improvement over GRPO is only marginal, whereas PRSA yields a much larger gain. We argue that this difference comes from the quality and faithfulness of the supervision signal rather than from reward density alone. PRM-style methods rely on a separately learned or externally constructed process reward, so the optimization target can drift toward locally plausible but globally unfaithful reasoning patterns; in contrast, PRSA derives progress directly from alignment with valid intermediate states in the solution space and still keeps final-answer correctness as the global anchor, which makes the reward shaping more reliable and the token-level credit assignment better localized. This mechanism is especially valuable when a trajectory follows the correct path for most of the reasoning process but makes a mistake near the end: instead of penalizing the whole response uniformly, PRSA preserves the aligned prefix and concentrates suppression on the

Methods	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./ΔGRPO
BaseModel	12.19	4.79	35.23	48.60	15.07	16.33	22.04
GRPO	20.94	8.44	58.98	72.20	27.76	37.33	37.61
Dr.GRPO	21.04	8.23	58.59	72.05	28.58	35.89	37.40 (-0.21)
GPG	21.98	9.06	59.61	72.05	27.21	37.67	37.93 (+0.32)
DAPO	21.25	8.75	58.20	72.70	29.50	37.22	37.94 (+0.33)
GSPO	19.38	8.33	60.16	73.00	28.12	37.26	37.71 (+0.10)
GRPO-PRM	20.92	9.01	58.63	72.85	28.68	36.73	37.81 (+0.20)
TreeRPO	21.08	8.92	59.75	73.40	28.68	37.98	38.30 (+0.69)
GRPO-lead	21.56	9.48	59.06	73.25	29.14	37.07	38.26 (+0.65)
DGPO	**23.85**	10.21	**61.02**	74.25	31.07	38.33	39.79 (+2.18)
PRSA (ours)	22.44	**11.03**	60.41	**76.70**	**33.73**	**39.84**	**40.69** (+3.08)

Table 1: Overall performance of different GRPO-style objectives on math reasoning benchmarks using Qwen2.5-Math-7B as the base model.

Model / Methods	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./ΔGRPO
Qwen2.5-Math-1.5B							
BaseModel	6.87	3.65	30.94	34.95	8.55	21.93	17.82
GRPO	11.35	3.96	46.48	64.85	20.13	29.59	29.39
PRSA (ours)	12.17	4.33	47.94	66.70	21.97	30.71	30.64 (+1.25)
Qwen2.5-3B							
BaseModel	2.81	0.73	22.66	48.65	13.69	19.37	17.99
GRPO	5.31	1.56	33.28	63.35	22.89	26.41	25.47
PRSA (ours)	6.83	2.17	35.94	66.85	25.46	28.71	27.66 (+2.19)

Table 2: Performance of PRSA on different Qwen2.5 backbones. The consistent improvements over GRPO across both model sizes indicate that the benefits of PRSA generalize beyond a specific backbone scale.

diverged suffix.

4.3 Cross-backbone Results

The cross-backbone results are shown in Table 2. PRSA consistently improves over GRPO on both the 1.5B and 3B backbones, indicating that its effectiveness does not depend on a particular model scale. On Qwen2.5-Math-1.5B, the average score increases from 29.39 to 30.64, while on Qwen2.5-3B it rises from 25.47 to 27.66. Although the gains on the smaller backbone are relatively moderate, the improvements remain stable across all six benchmarks, suggesting that process-aware supervision can still provide useful training signals even when the underlying reasoning capacity is limited. Additional Qwen3 and Llama-family results are in Appendix B.1.

The advantage becomes more pronounced on the 3B model, where PRSA yields larger and more visible improvements across the evaluation sets. This trend suggests that as the backbone acquires stronger basic reasoning ability, it can make better use of the progress information provided by PRSA, leading to more effective credit assignment during training. Overall, these results show that solution-space-aware progress estimation is not a training trick tailored to a single model, but a generally effective supervision mechanism that transfers across model sizes and remains beneficial under different reasoning capacities.

4.4 Different Teachers for Solution-Space Assistance under All-Zero Groups

Table 3 shows that recovering a valid solution-space signal in all-zero groups is important for stable learning. Even the *No Teacher* variant already improves over vanilla GRPO by +1.32 points on average, suggesting that the PRSA objective itself can partially alleviate sparse-reward collapse. However, introducing teacher-based assistance yields a much larger gain: using *Qwen2.5-Math-7B-Instruct* as a self-teacher improves the average score to 40.02, and replacing it with *GPT-4o* further raises performance to 40.38. The best overall result is obtained by *Claude-Sonnet-4-6*, which reaches 40.69 average pass@1, confirming that stronger auxiliary derivations provide more reliable guidance when the on-policy rollout group contains no positive sample. Teacher trajectories only construct the solution space and are not imitation targets.

The gains are also clearly non-uniform across benchmarks. Teacher assistance is most helpful

Teacher	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./ΔGRPO
GRPO	20.94	8.44	58.98	72.20	27.76	37.33	37.61
No Teacher	22.00	9.92	59.44	73.00	30.61	38.61	38.93 (+1.32)
Qwen2.5-Math-7B-Instruct	22.33	10.75	59.88	74.55	32.90	39.69	40.02 (+2.41)
GPT-4o	22.42	11.00	60.25	75.25	33.55	39.80	40.38 (+2.77)
Claude-Sonnet-4-6	**22.44**	**11.03**	**60.41**	**76.70**	**33.73**	**39.84**	**40.69** (+3.08)

Table 3: Effect of different teacher models used for solution-space assistance when an all-zero rollout group is encountered. Claude-Sonnet-4-6 is the default teacher used in the full PRSA system.

Methods	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./ΔGRPO
GRPO	20.94	8.44	58.98	72.20	27.76	37.33	37.61
PRM+Outcome	20.92	9.00	58.63	72.85	28.68	36.76	37.81 (+0.20)
PRSA w/o Reward Shaping	21.67	9.67	59.44	73.90	29.96	38.54	38.86 (+1.25)
PRSA w/o Token Credit Reallocation	22.08	10.33	59.88	74.95	31.71	39.32	39.71 (+2.10)
PRSA w/ Zero Suffix Credit	22.33	10.67	60.19	75.90	32.81	39.54	40.24 (+2.63)
PRSA (ours)	**22.44**	**11.03**	**60.41**	**76.70**	**33.73**	**39.84**	**40.69** (+3.08)

Table 4: Ablation study of PRSA on Qwen2.5-Math-7B. The results show that both reward shaping and token-level credit reallocation are important for achieving the full performance gain.

on harder settings such as AIME25, MATH500, Minerva, and Olympiad, where all-zero groups are more likely to occur and sparse rewards are less informative. In contrast, the differences on AMC23 are relatively small, indicating that easier datasets rely less on external recovery of correct reasoning trajectories. We also note that self-teaching already captures a substantial portion of the benefit, which shows that PRSA does not strictly depend on an external frontier model; nevertheless, stronger teachers still provide consistent additional improvements in the most challenging sparse-reward regime.

4.5 Ablation Study

We further ablate the two key ingredients of PRSA, namely progress-based reward shaping and token-level prefix-suffix credit reallocation, and compare them with a direct PRM+OUTCOME baseline. Specifically, PRSA W/O REWARD SHAPING removes the progress-based shaped trajectory score and keeps only boundary-aware token credit reassignment under the original outcome reward. PRSA W/ ZERO SUFFIX CREDIT keeps the prefix-suffix partition but sets the credit of the diverged suffix to zero instead of applying an explicit negative penalty. PRSA W/O TOKEN CREDIT REALLOCATION applies the shaped trajectory score uniformly to all tokens. PRM+OUTCOME augments GRPO with direct process reward modeling while retaining the outcome-level reward.

As shown in Table 4, both reward shaping and token-level credit reallocation contribute non-trivially to the final performance. Removing reward shaping leads to the largest drop among the core ablations, indicating that dense progress-aware supervision is the primary source of improvement under sparse verifier rewards. Applying shaped rewards without token-level reallocation remains effective, but still underperforms the variants that explicitly localize credit to the aligned prefix and diverged suffix. Moreover, replacing the explicit negative suffix penalty with zero suffix credit yields a further drop, suggesting that actively suppressing already-diverged suffixes is more effective than merely withholding positive updates. Finally, direct PRM+OUTCOME only brings a limited gain over vanilla GRPO and remains clearly behind all PRSA variants, which supports our claim that preserving final-answer correctness as the global optimization anchor is more reliable than directly optimizing local process rewards.

5 Conclusion

We presented PRSA, a framework for enhancing outcome-level reinforcement learning in multi-step mathematical reasoning. PRSA derives process-aware supervision by aligning sampled intermediate states with a solution space built from valid derivations, and uses this signal for reward shaping and prefix-suffix credit reallocation. Experiments and ablations show that PRSA consistently outperforms GRPO and strong baselines, suggesting that solution-space alignment is a practical middle ground between sparse outcome supervision and learned process reward modeling.

Limitations

Although PRSA improves outcome-level reinforcement learning for mathematical reasoning, it still has several limitations. First, our method relies on the ability to extract, normalize, and align intermediate-value states from both sampled trajectories and valid solutions. This assumption is relatively natural in symbolic mathematics, where intermediate expressions often admit explicit equivalence checking, but it may be substantially harder to satisfy in more open-ended reasoning domains such as commonsense, legal, or scientific writing, where intermediate progress is less discretizable and correctness is harder to verify.

Second, the quality of PRSA depends on the coverage and faithfulness of the constructed solution space. When the available valid derivations are sparse, narrow, or biased toward a small set of reasoning styles, the resulting graph may underrepresent legitimate alternative solution paths. In such cases, partially correct trajectories that follow unconventional but valid reasoning may receive underestimated progress scores. Relatedly, our all-zero-group recovery mechanism can benefit from stronger teacher models, which means that performance in difficult sparse-reward settings may still depend on the availability and quality of auxiliary solution providers.

References

Jie Cheng, Gang Xiong, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Yisheng Lv, and Fei-Yue Wang. 2025. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. *arXiv preprint arXiv:2504.15275*.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences. *Preprint*, arXiv:1706.03741.

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. 2025. Gpg: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*.

Yanqi Dai, Yuxiang Ji, Xiao Zhang, Yong Wang, Xiangxiang Chu, and Zhiwu Lu. 2026. Harder is better: Boosting mathematical reasoning via difficulty-aware grpo and multi-aspect question reformulation. *Preprint*, arXiv:2601.20614.

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

Jiaxing Guo, Wenjie Yang, Shengzhong Zhang, Tongshan Xu, Lun Du, Da Zheng, and Zengfeng Huang. 2025b. Right is not enough: The pitfalls of outcome supervision in training llms for math reasoning. *Preprint*, arXiv:2506.06877.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of ACL 2024*, pages 3828–3850.

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. In *Proceedings of NeurIPS 2021*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

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

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

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024.

Rewarding progress: Scaling automated process verifiers for llm reasoning. *Preprint*, arXiv:2410.08146.

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

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Preprint*, arXiv:2312.08935.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *Preprint*, arXiv:2506.14245.

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

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. 2025. Treerpo: Tree relative policy optimization. *Preprint*, arXiv:2506.05183.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2024. Ovm, outcome-supervised value models for planning in mathematical reasoning. *Preprint*, arXiv:2311.09724.

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

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Preprint*, arXiv:2203.14465.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. *Preprint*, arXiv:2406.03816.

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*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, and et al. Yang, An. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Appendix

A Experiment Details

A.1 Detailed Prompt of Ours

In this section, we present the exact prompt template used in PRSA for inference and policy rollouts. Compared with a standard zero-shot chain-of-thought prompt, our template further encourages the model to expose intermediate results in a structured equation-only format. This design makes intermediate-value steps easier to extract, normalize, and align with valid solution paths during training.

PRSA Inference Prompt. For standard policy sampling and GRPO-style training rollouts, we append the following instruction to each question:

PRSA Inference Instruction

```
Let's think step by step and output the final answer
↪  within \boxed{}.
In the steps, write intermediate results as
↪  standalone equation lines (e.g., x = ...),
and keep those equation lines free of English
↪  narration.
```

This prompt serves two purposes. First, it preserves the standard step-by-step reasoning style needed for mathematical problem solving. Second, by encouraging intermediate results to appear as clean standalone equation lines, it improves the reliability of extracting intermediate-value states from sampled trajectories, which is important for the solution-space alignment and progress scoring used in PRSA.

A.2 Extraction, Matching, and Cost Clarifications

PRSA uses a deterministic rule-based extraction pipeline rather than an LLM extractor. We parse sampled and reference solutions line by line, keep standalone equation-style lines, equality-bearing expressions, and explicit intermediate-value statements, discard malformed or non-mathematical fragments, and normalize the remaining expressions before alignment. The same extraction rule is applied to sampled trajectories and reference derivations.

The extra cost mainly comes from graph construction and window-constrained matching over extracted states. If a rollout group contains C correct trajectories with K extracted states on average, graph construction is $O(CK)$. If each of the G sampled rollouts contains T extracted states, alignment is $O(GTw)$ for window size w, so the overhead is roughly linear in the number of extracted intermediate states and requires no auxiliary model calls. Accidental matches are reduced by equivalence normalization, local order constraints, and bounded partial credit for final-wrong trajectories. If no valid graph exists or no state can be matched, PRSA falls back to outcome supervision.

A.2.1 Tasks and Data Details

We assess our method on a diverse collection of competition-level mathematical reasoning benchmarks, covering problems from advanced high-school contests to olympiad-style settings. Table 5 lists the datasets used in our experiments together with their evaluation sizes and, when applicable, the training split adopted in our setup.

Table 5: Benchmarks used in our experiments and their corresponding data splits.

Dataset	Test Size	Train Size
AIME24	30	–
AIME25	30	–
AMC23	40	–
MATH500	500	–
Minerva	272	–
OlympiadBench	674	–
MATH	5,000	7,500

AIME24 AIME24 contains 30 integer-answer problems collected from the 2024 editions of AIME I and AIME II. These problems typically involve multi-step symbolic reasoning, algebraic transformations, and nontrivial case distinctions. Since the target outputs are integers, evaluation can be performed using exact-match accuracy.

AIME25 AIME25 follows the same setting as AIME24, but uses the 2025 AIME I and AIME II problems. This benchmark is mainly used to examine whether the model can transfer to a more recent exam set under an identical answer format and similar reasoning demands.

AMC23 AMC23 is constructed from problems in the 2023 AMC 12 exams (Forms A and B) and contains 40 questions. Compared with AIME, the questions are often shorter in wording, but still require deliberate reasoning over topics such as

algebra, geometry, combinatorics, and number theory. We formulate this benchmark as free-form mathematical reasoning with exact final-answer evaluation.

MATH500 MATH500 is a commonly used 500-problem evaluation subset sampled from the MATH test set. It covers seven mathematical domains and multiple difficulty levels, and is widely adopted as a reliable benchmark for assessing both reasoning quality and answer accuracy.

Minerva Minerva-Math is a relatively compact benchmark consisting of 272 competition-style mathematics problems. It spans a range of quantitative reasoning scenarios, including algebra, calculus, and applied mathematics, and is frequently used to test the robustness of math reasoning models beyond the distribution of MATH-style training data.

OlympiadBench We use the English, text-only mathematics portion of OlympiadBench, which contains 674 challenging olympiad-level problems. The benchmark emphasizes long-horizon reasoning and often requires more elaborate derivations in areas such as combinatorics, geometry, and number theory.

MATH Dataset The MATH dataset is a large-scale benchmark of 12,500 competition mathematics problems paired with step-by-step reference solutions. In our experiments, we use 7,500 training examples for optimization and the standard 5,000-problem test split for evaluation where appropriate. The dataset includes problems from seven subject areas and a broad range of difficulty levels.

A.2.2 Compared Methods

We compare our method with a set of representative reinforcement-learning-based post-training algorithms for verifiable reasoning tasks. While these approaches all operate under the general paradigm of sampling multiple candidate solutions, assigning reward signals based on verifiable outcomes, and updating the policy accordingly, they differ substantially in how they estimate advantages, define clipping strategies, and address issues such as response length bias and training instability.

GRPO Group Relative Policy Optimization (GRPO) is a critic-free alternative to PPO-style training for reasoning models. Rather than learning a separate value function, GRPO computes relative advantages from the reward statistics of a sampled response group for each prompt. This design reduces memory overhead and simplifies optimization while retaining strong empirical performance on reasoning tasks.

Dr. GRPO Dr. GRPO, short for "GRPO Done Right," is an improved variant of GRPO designed to mitigate optimization artifacts in the original formulation. In particular, it aims to remove biases that can unintentionally favor overly long generations, especially among incorrect samples. By revising the gradient estimation procedure, Dr. GRPO improves token efficiency and yields more controlled training behavior.

GPG Group Policy Gradient (GPG) revisits the standard policy-gradient formulation and adapts it to the group-based reasoning setting. In contrast to GRPO-style surrogate objectives, GPG more directly optimizes the original RL objective and adopts a streamlined design that avoids auxiliary critics, reference policies, and explicit KL regularization, while still targeting strong reasoning performance.

DAPO Decoupled Clip and Dynamic sAmpling Policy Optimization (DAPO) is a GRPO-style method developed for long-chain reasoning optimization. It modifies the clipping mechanism and introduces dynamic sampling strategies to improve both stability and efficiency. In addition, DAPO includes several training heuristics at the system and objective levels to reduce reward noise and prevent training collapse in long-response regimes.

GSPO Group Sequence Policy Optimization (GSPO) is a sequence-level alternative to token-level group RL optimization. Instead of relying on token-wise importance ratios, GSPO computes importance weights at the sequence level and performs clipping and optimization accordingly. This formulation is intended to suppress token-level variance and deliver more stable optimization in large-scale reasoning training.

GRPO-LEAD GRPO-LEAD extends GRPO with mechanisms tailored to mathematical reasoning difficulty. It incorporates reward shaping that favors concise correct solutions, imposes explicit penalties on incorrect responses, and reweights learning signals according to problem difficulty. Together, these components aim to provide sharper

supervision and stronger optimization pressure on more challenging examples.

GRPO-PRM GRPO-PRM is a process-reward-augmented variant of GRPO that combines final-answer reward with additional supervision from a process reward model (PRM). In practice, it assigns step-level PRM scores to a reasoning trajectory, aggregates them into a single process score, and blends this score with the outcome reward for standard GRPO optimization. Compared with vanilla GRPO, this method provides denser supervision over intermediate reasoning steps.

DGPO Difficulty-Aware Group Policy Optimization (DGPO) is a GRPO-based method proposed in *Harder Is Better*. It is motivated by the observation that vanilla GRPO tends to produce weaker effective updates on harder questions. To address this issue, DGPO introduces difficulty-balanced group advantage estimation and difficulty-aware question weighting, so that more challenging problems contribute stronger learning signals during training.

B Implementation Details

We implement our training pipeline using the VEneralizable RL (`verl`) framework, with `vLLM` as the rollout backend for efficient generation.

Training Configuration For the reinforcement learning stage, we train the model for 1 epoch with a train batch size of 128. The actor is optimized with a learning rate of 1×10^{-6}. We set the PPO mini-batch size to 64 and the PPO micro-batch size per GPU to 64. To improve training stability, gradient clipping is applied with a threshold of 0.6. We further use an entropy coefficient of 0.0 and a KL coefficient of 0.001 during optimization. For PRSA-specific reward shaping, the progress shaping coefficient α is set to 0.5, the suffix penalty λ_{suf} is set to 0.5, and the prefix penalty scaler η_{pre} is set to 0.4. All experiments are conducted on a single node with 8 GPUs.

Generation and Rollout During rollout, we use `vLLM` for efficient response sampling. For each query, we sample $N = 8$ candidate responses. To support long-form mathematical reasoning, we set the maximum prompt length to 4096 tokens and the maximum response length to 8192 tokens. The sampling temperature is fixed at 0.6 to maintain diversity in generated reasoning trajectories. In addition, the log-probability micro-batch size per GPU is set to 160, and the GPU memory utilization ratio for rollout is configured as 0.85. Table 6 summarizes the detailed hyperparameter settings used in our experiments.

Table 6: Hyperparameters used for PRSA training.

Hyperparameter	**Value**
Base Framework	`verl`
Rollout Engine	`vLLM`
Number of GPUs	8
Training	
Total Epochs	1
Train Batch Size	128
PPO Mini-Batch Size	64
PPO Micro-Batch Size / GPU	64
Learning Rate	$1\text{e}{-6}$
Gradient Clipping	0.6
Entropy Coefficient	0.0
KL Coefficient	0.001
PRSA Specifics	
Progress Shaping Coefficient α	0.5
Suffix Penalty λ_{suf}	0.5
Prefix Penalty Scaler η_{pre}	0.4
Match window w	2
Rollout	
Number of Rollouts (N)	8
Temperature	0.6
Max Prompt Length	4096
Max Response Length	8192
Log-Prob Micro-Batch Size / GPU	160
GPU Memory Utilization	0.85

Solution-Space Matching Details To provide denser supervision for incorrect rollouts, PRSA performs step-level matching in the intermediate solution space rather than relying solely on the final outcome reward. Given a group of sampled responses for the same problem, we first extract intermediate equation-style results from each solution trajectory. Specifically, we retain standalone equation lines and treat the right-hand-side values as candidate intermediate states. To improve robustness, we discard malformed or non-informative entries, such as pure English strings, pure symbolic fragments, or expressions with unbalanced brackets.

After extraction, each intermediate state is normalized into a canonical form before compari-

Model / Method	AIME24	AIME25	AMC23/AMC	MATH500	Avg.
Qwen3-4B GRPO	46.46	40.94	87.81	89.17	66.10
Qwen3-4B PRSA	50.21	42.92	88.98	89.59	67.93
Qwen3-8B GRPO	54.06	39.37	88.05	89.91	67.85
Qwen3-8B PRSA	55.73	41.56	89.53	90.22	69.26
Llama-3.2-3B Before RL	6.88	0.42	17.47	44.40	17.29
Llama-3.2-3B GRPO	10.62	0.21	22.89	47.50	20.31
Llama-3.2-3B PRSA	10.83	0.42	23.44	48.10	20.70

Table 7: Additional cross-backbone results. Avg. is computed over the displayed benchmarks.

Setting	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg.
$\alpha = 0.3, \lambda_{\text{suf}} = 0.5$	22.60	9.90	60.55	75.80	32.90	39.09	40.14
$\alpha = 0.5, \lambda_{\text{suf}} = 0.5$	22.44	11.03	60.41	76.70	33.73	39.84	40.69
$\alpha = 0.7, \lambda_{\text{suf}} = 0.5$	21.88	10.73	59.92	76.25	34.01	40.13	40.49
$\alpha = 0.5, \lambda_{\text{suf}} = 0.3$	22.08	10.31	60.62	75.95	32.72	39.43	40.19
$\alpha = 0.5, \lambda_{\text{suf}} = 0.7$	22.71	10.83	60.70	76.45	33.18	39.58	40.58

Table 8: Hyperparameter sensitivity for progress shaping and suffix suppression on Qwen2.5-Math-7B.

son. The normalization removes superficial formatting differences and standardizes mathematically equivalent expressions, such as redundant spaces, outer parentheses, fraction formats, and common LaTeX variants. This allows equivalent intermediate results written in different surface forms to be matched reliably.

For each problem, we then construct a reference solution space from valid correct trajectories. When correct rollouts are available within the sampled group, their extracted intermediate states are used as references; otherwise, we fall back to previously collected correct solutions when available. The resulting reference bank contains both the set of normalized intermediate states and their relative positions in each reference trajectory.

Given an incorrect rollout, we align its extracted intermediate states to the reference bank using relative-position matching. Let the predicted trajectory contain intermediate states $\{s_i\}_{i=1}^{T}$, and let a reference trajectory contain states $\{r_j\}_{j=1}^{L}$. A predicted state s_i is considered matched if there exists a reference state r_j such that the two states are mathematically equivalent after normalization and their relative positions satisfy $|i - j| \leq w$, where w is a fixed matching window. This windowed criterion makes the alignment tolerant to small variations in reasoning granularity while still preserving step-order consistency.

Based on the alignment result, we measure how far the incorrect rollout has progressed along the valid solution space. For each reference trajectory, we record the farthest matched step position and compute a normalized progress ratio as the matched prefix length divided by the total number of reference steps. The final progress score is obtained by averaging this ratio over all matched references:

$$\text{Progress} = \frac{1}{|\mathcal{R}'|} \sum_{r \in \mathcal{R}'} \frac{m_r}{L_r},$$

where $\mathcal{R}'$ denotes the set of references with at least one matched step, m_r is the farthest matched position in reference r, and L_r is the total number of extracted steps in that reference.

This progress signal is then used to shape rewards for incorrect samples. Correct trajectories keep their original final reward, while incorrect trajectories receive a softened score proportional to their progress in the aligned solution space:

$$\tilde{r} = \begin{cases} r_{\text{final}}, & \text{if the final answer is correct,} \\ \alpha \cdot \text{Progress}, & \text{otherwise,} \end{cases}$$

where α controls the strength of progress-based shaping.

In addition to scalar reward shaping, PRSA also performs prefix-aware token-level credit assignment. Once the last matched intermediate state is identified, tokens before this matched boundary are treated as constructive reasoning tokens, while tokens after it are down-weighted with a suffix penalty. This design encourages the model to

preserve useful partial reasoning while discouraging unproductive continuations beyond the valid progress frontier. If no valid reference exists or no intermediate state can be matched, PRSA falls back to the standard group-based optimization objective without applying stepwise shaping.

B.1 Additional Results and Sensitivity

To further check cross-backbone behavior, we additionally evaluate PRSA on Qwen3-series models and Llama-3.2-3B-Instruct. As shown in Table 7, PRSA improves over GRPO on Qwen3-4B, Qwen3-8B, and Llama-3.2-3B-Instruct. Teacher trajectories, when used, are only used to construct the solution-space graph and are not imitation targets.

We also conduct an error-localization diagnostic on Qwen2.5-Math-7B rollouts over MATH-7500. Among incorrect samples, 2,366 trajectories initially align with a valid derivation and later diverge; the first erroneous step appears on average at step 3.05, corresponding to 55.52% of the extracted reasoning process. This supports the claim that outcome-only supervision can give coarse or mislocalized credit for long reasoning trajectories.

Table 8 shows that the default $\alpha = 0.5, \lambda_{\text{suf}} = 0.5$ performs best on average, while nearby settings remain competitive. Smaller α weakens the use of partially correct failed trajectories, whereas larger α can make progress shaping too strong relative to the final-answer anchor.