SRM-GRPO: Learning Strategy-Level Rewards for Mathematical Reasoning

Anonymous ACL submission

Abstract

Reinforcement learning with verifiable outcome rewards has greatly improved mathematical reasoning in large language models. However, its credit assignment remains coarse: a scalar terminal reward provides little feedback on whether early high-level decisions, such as theorem selection and subgoal ordering, are strategically sound. To address this issue, we use the coarse solution outline before detailed computation as a natural unit of intermediate credit. This paper presents **SRM-GRPO**, a three-stage framework: (1) the policy is trained via supervised fine-tuning to emit explicit strategy segments; (2) strategy quality is estimated through counterfactual rollouts, and a Strategy Reward Model (SRM) is trained on confidence-aware pairwise preferences; and (3) the frozen SRM is integrated into GRPO to provide a segment-level reward. In particular, strategy quality is estimated once in an offline stage, avoiding the need to repeatedly recompute strategy rewards during reinforcement learning. Experiments on three policy backbones show that SRM-GRPO improves average pass@1 over outcome-only GRPO by 1.87%, 2.64%, and 2.93% respectively. It also achieves competitive or better accuracy than stronger reward baselines while requiring substantially lower GRPO-stage compute.

1 Introduction

Reinforcement learning (RL) under verifiable outcome rewards has become a scalable paradigm for improving mathematical reasoning in large language models (LLMs) (Guo et al., 2025). Group relative policy optimization (GRPO) (Shao et al., 2024) estimates advantages from grouped rollouts, making large-scale RL on benchmarks such as GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) tractable. However, the dominant training signal remains the correctness of the final answer, with little differentiation among the intermediate decisions that produced it.

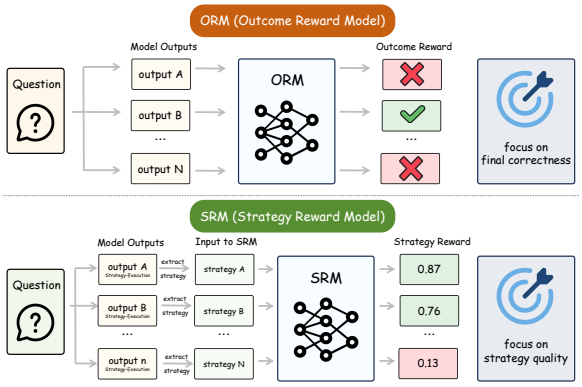

Figure 1: **Outcome vs. strategy rewards.** An outcome reward (top) scores the final answer uniformly across the trajectory. Our SRM (bottom) scores the strategy segment directly, distinguishing productive routes from unproductive ones.

Such coarse outcome supervision creates a familiar credit-assignment problem. A scalar terminal reward provides little information about whether the early, high-level decisions in a trajectory (which theorem to invoke, which quantities to introduce first, or how to order subgoals) were strategically sound. Outcome supervision cannot distinguish a clean strategy from a fragile one that happens to succeed, leaving the policy with no signal about which routes are worth pursuing. Sparse outcome signals thus underspecify the contribution of the *strategy*: the coarse solution outline that precedes detailed computation. Figure 1 contrasts our setting with conventional outcome supervision: we score the strategy segment of each rollout directly, rather than scoring only the terminal answer.

Prior work establishes explicit intermediate structure as a reliable lever for reasoning, through chain-of-thought prompting, decomposition, plan-and-solve templates, and inference-time search (Wei et al., 2022; Wang et al., 2023; Yao et al.,

2023), but treats planning as a prompting convention or inference-time procedure. We ask whether the downstream utility of a strategy can instead be learned as a *reusable, training-time* reward signal that improves RL credit assignment.

We study mathematical reasoning under a two-level view of generation. A *strategy* segment fixes a coarse solution outline (which tools or lemmas to use, which intermediate targets to pursue, and in what order), while an *execution* segment carries out the concrete derivation and produces the final answer. We operationalize strategy quality not by surface plausibility of the plan text, but by the success rate of independent execution rollouts conditioned on a fixed strategy under controlled sampling. This definition situates strategy-level reward between two granularities: outcome supervision, which scores entire solutions (Cobbe et al., 2021; Ouyang et al., 2022), and process supervision, which scores individual reasoning steps (Lightman et al., 2024; Wang et al., 2024). We target the *strategy block* as an intermediate unit of credit, which complements rather than replaces process reward models: the two operate at different granularities and can be combined (§4.3).

Our approach proceeds in three stages. First, we perform supervised fine-tuning (SFT) that teaches a base policy to emit an explicit strategy segment before solving, establishing a format compatible with downstream RL. Second, we sample *counterfactual* strategies for each question and estimate their quality through multiple independent execution rollouts. Pairwise preferences induced by empirical success rates, with confidence-aware filtering to mitigate label noise from finite Monte Carlo estimation, supervise a *strategy reward model* (SRM) that scores question–strategy pairs (q, s). Third, we integrate the SRM into GRPO as a *segment-level* reward applied to the strategy prefix, jointly with correctness and format constraints. Group-wise normalization and the suppression of strategy rewards when the final answer is incorrect are used to sharpen credit assignment while reducing incentives to game the learned signal. Conceptually, this *amortizes* expensive online estimation of plan usefulness into an offline-learned SRM that is reused throughout training.

We organize the empirical investigation around three questions:

RQ1 Does the learned strategy reward transfer beyond the policy update? When used as a standalone selector, do its top-ranked strategies improve the accuracy of a frozen executor?

RQ2 Does augmenting GRPO with strategy-level SRM rewards improve performance beyond outcome and format rewards alone?

RQ3 Given a reusable pretrained SRM, what is the GRPO-stage compute cost compared to baselines that re-estimate plan quality online during RL (Dou et al., 2025)?

Contributions. We summarize our contributions as:

- We formulate *strategy-level* credit for mathematical reasoning and learn an SRM from counterfactual rollouts, with *confidence-aware* preference construction to mitigate noise from finite Monte Carlo estimation.
- We present **SRM-GRPO**, integrating the frozen SRM into GRPO as a segment-level reward with correctness gating and group-wise normalization.
- We report experiments across three policy backbones with GRPO-stage cost comparisons, a token-matched CoT-SFT control, and ablations. We will release all SRM weights, preference data, and policy checkpoints.

2 Related Work

2.1 Explicit Plans and Decompositions

Chain-of-thought prompting and self-consistency demonstrate that step-by-step rationalization and multi-path aggregation improve robustness on arithmetic and symbolic tasks (Wei et al., 2022; Wang et al., 2022). Decomposition-oriented methods (Zhou et al., 2022; Khot et al., 2022) and plan-and-solve templates (Wang et al., 2023) make planning explicit, while tree- and graph-based search expand the solution space at inference time (Yao et al., 2023; Besta et al., 2024; Hao et al., 2023). These approaches treat planning as a prompting convention or inference-time procedure. We ask instead whether strategy utility can be *learned* as a reusable training-time reward signal.

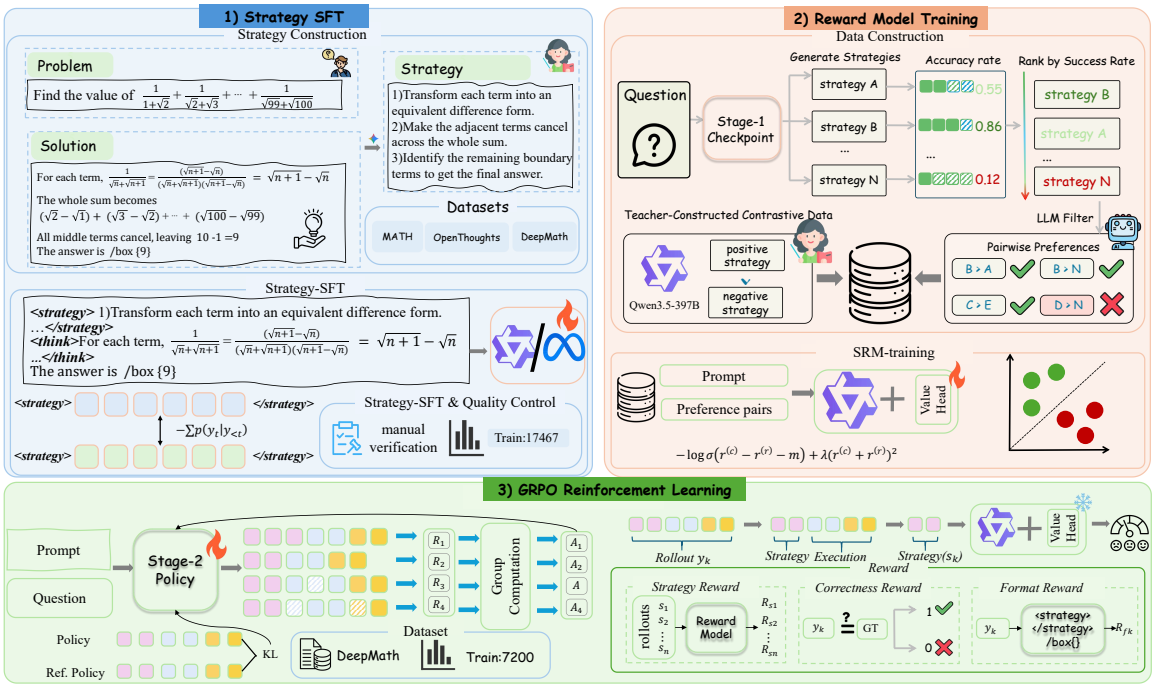

Figure 2: Overview of SRM-GRPO. **(1) SFT**: teacher strategies fine-tune the student on (question, strategy, execution) triples. **(2) SRM Training**: candidate strategies condition rollouts, and confidence-aware preference pairs train the SRM. **(3) GRPO**: composite reward fuses correctness, format, and frozen SRM score with group normalization and correctness gating.

2.2 Reward Models for Math Reasoning

Outcome reward models (ORMs) score complete solutions for reranking or terminal RL supervision (Cobbe et al., 2021; Ouyang et al., 2022), inheriting sparse credit-assignment limitations. Process reward models (PRMs) provide denser supervision by scoring individual reasoning steps (Lightman et al., 2024; Uesato et al., 2022; Rezaei et al., 2026; Yue et al., 2026; Xu et al., 2025; Yuan et al., 2026; Chen et al., 2026). Automated Monte Carlo labels reduce annotation cost (Wang et al., 2024, 2025) but introduce noise and generalization challenges (Zhang et al., 2025b). PRIME documents reward hacking risks when process-level signals misalign with terminal correctness (Cui et al., 2025). We target an intermediate granularity, the *strategy segment*, complementing PRMs: step-level models localize algebraic errors, while an SRM assesses whether the model chose a productive high-level approach.

2.3 RL with Planning-Aware Rewards

GRPO and related group-normalized policy gradient methods avoid a separate critic by estimating advantages from grouped rollouts, and have become standard for mathematical RL (Shao et al., 2024; Guo et al., 2025; Wen et al., 2025; He et al., 2025a; Zhang et al., 2025a). The default reward remains answer-level, underspecifying credit for early strategic choices, and learned rewards can harm RL when policies exploit spurious correlations (Gao et al., 2024). Plan-guided RL methods address this by jointly optimizing answer correctness and high-level guidance quality (Dou et al., 2025), but estimate plan quality *online* during RL, requiring repeated resampling as the policy evolves. We amortize this into an offline SRM trained once from counterfactual rollouts and integrated into GRPO as a segment-level reward.

3 Methodology

We present **SRM-GRPO**, a three-stage framework that learns a strategy-level reward model from counterfactual rollouts and integrates it into group-relative policy optimization for mathematical reasoning. Figure 2 provides a schematic overview. We formalize the strategy–execution decomposition (§3.1), then describe strategy-guided SFT (§3.2), SRM data construction and training (§3.3–§3.4), and GRPO integration (§3.5).

3.1 Strategy–Execution Decomposition

For a mathematical problem x, we factor the model response into a **strategy** s and an **execution** y. The strategy is a high-level solution outline specifying tool choice, intermediate targets, and

subgoal ordering, while the execution is the concrete derivation producing the final answer. The strategy itself excludes detailed computation and the final answer.

Strategy quality is defined by downstream execution success, not surface plausibility:

$$q(x, s) = P(\text{correct} \mid x, s). \quad (1)$$

This anchors strategy evaluation in a measurable quantity. The decomposition is enforced by structured output tags `<strategy>` `</strategy>` and `<think>` `</think>`, taught during SFT and rewarded during RL. Figure 1 (top vs. bottom) contrasts this construction with an outcome reward model: where an ORM marks an entire generation right or wrong, the SRM separates the route choice from its execution and learns a ranking over alternative routes for the same problem.

3.2 Strategy-Guided SFT

The first stage establishes the strategy–execution format through supervised fine-tuning. We draw problems from the MATH training split, OpenThoughts, and DeepMath, prompt Qwen3.5-397B-A17B with each problem and its reference solution to generate a strategy capturing the high-level route without intermediate computations (prompt in Appendix B), and manually verify the resulting (question, strategy, execution) triples. After verification, the SFT corpus contains 17,467 examples. Each instance concatenates the teacher strategy with the reference execution under the structured tags, and the policy is trained with standard next-token prediction loss over the full sequence. SFT targets format establishment rather than reasoning improvement, yielding a policy that reliably emits structured outputs for the subsequent RL stages.

3.3 SRM Training Data Construction

The SRM is trained on pairwise preferences (x, s^+, s^-) where s^+ yields higher empirical execution success than s^-. We construct preferences from two complementary sources, starting from 32,000 problems sampled from DeepMath and OpenThoughts.

3.3.1 Rollout-Induced Preference Data

For each training problem, we sample $K = 8$ candidate strategies from π_{SFT} with temperature $T_s > 0$, then condition on each strategy and sample $M = 24$ independent execution rollouts with the strategy prefix held fixed, yielding 192 rollouts per problem. The empirical success rate of strategy $s^{(k)}$ is computed as:

$$\hat{q}(x, s^{(k)}) = \frac{1}{M} \sum_{j=1}^{M} \mathbf{1}[\text{rollout } j \text{ is correct}]. \quad (2)$$

We construct pairwise preferences with confidence-aware filtering. Modeling each execution outcome as Bernoulli with probability $q(x, s)$, we retain a pair $(s^{(i)} \succ s^{(j)})$ when the posterior probability that $s^{(i)}$ outperforms $s^{(j)}$ exceeds 0.9:

$$\mathbb{P}\big[q(x, s^{(i)}) > q(x, s^{(j)})\big] \geq 0.9, \quad (3)$$

computed via a Beta-Binomial model with $\text{Beta}(1, 1)$ prior. This suppresses pairs where the empirical gap is plausibly noise, addressing reliability concerns in Monte Carlo-based supervision (Wang et al., 2024; Zhang et al., 2025b). After this filter, 53,464 candidate pairs survive. We further filter by Qwen3.5-397B-A17B preference verification (Zheng et al., 2023): given the problem and the proposed labeled preference (s^+, s^-), the judge decides whether $s^+ \succ s^-$ is reasonable. Pairs marked unreasonable are discarded, retaining 48,413 pairs as $\mathcal{D}_{\text{rollout}}$ (prompt template in Appendix C.3).

3.3.2 Teacher-Constructed Contrastive Data

Rollout-induced preferences are limited to strategies the SFT policy can produce. For broader coverage, we use Qwen3.5-397B-A17B to directly construct positive strategies s^+ (correct theorems, feasible subgoal ordering) and negative strategies s^- (flawed, incomplete, or misleading approaches) for each question (prompt template in Appendix C.4). These 17,197 pairs form $\mathcal{D}_{\text{teacher}}$, providing cleaner contrastive signal with recognizable structural errors underrepresented in rollout-induced data.

3.3.3 Merging and Filtering

We merge $\mathcal{D}_{\text{rollout}}$ and $\mathcal{D}_{\text{teacher}}$ into $\mathcal{D}_{\text{SRM}} = \{(x_i, s_i^+, s_i^-)\}_{i=1}^{N_{\text{SRM}}}$, yielding 65,610 preference pairs in total (see Appendix C for full statistics). We balance the two sources via sampling to prevent overfitting to rollout noise or teacher stylistic biases.

3.4 SRM Training

The SRM $r_\phi(x, s)$ is initialized from Qwen2.5-7B-Instruct (Yang et al., 2024a) and adapted with

LoRA (rank 64) (Hu et al., 2021). A scalar value head over the final strategy token's hidden state replaces the LM head. The SRM never sees the execution segment or final answer.

We train with a margin-augmented Bradley–Terry pairwise ranking loss (Bradley and Terry, 1952) with a scale-anchoring regularizer (Christiano et al., 2017):

$$\mathcal{L}_{\text{SRM}}(\phi) = -\log \sigma\big(r^{(c)} - r^{(r)} - m\big) + \lambda\big(r^{(c)} + r^{(r)}\big)^2, \tag{4}$$

where $r^{(c)} = r_\phi(x, s^+)$ and $r^{(r)} = r_\phi(x, s^-)$ are the chosen and rejected scores, $m \geq 0$ enforces a minimum gap between preferred and dispreferred strategies, and the squared sum keeps absolute reward magnitudes bounded so the learned scale does not drift over training. Rollout-induced pairs are weighted by the confidence score from Equation 3, while teacher pairs use uniform weights. The trained SRM is then frozen and deployed as a reward component in GRPO. We validate it behaviorally via a frozen-executor selection study (§4.4).

3.5 SRM-GRPO

The third stage integrates the frozen SRM into GRPO (Shao et al., 2024). Each response receives a composite reward:

$$R = \alpha\, R_{\text{correct}} + \beta\, R_{\text{format}} + \gamma\, R_{\text{SRM}}, \tag{5}$$

where $R_{\text{correct}} \in \{0, 1\}$ indicates answer correctness, $R_{\text{format}} \in \{0, 1\}$ enforces the structured output format, and $R_{\text{SRM}} = r_\phi(x, s)$ scores the extracted strategy segment.

The SRM reward is applied to the strategy prefix tokens only. Execution tokens receive zero SRM reward. This segment-level assignment means the SRM score is *computed* from strategy text alone (it never observes execution tokens or the final answer), providing credit that is informationally independent of how any particular execution unfolded. Segment-level reward also separates SRM from process reward models (Lightman et al., 2024; Wang et al., 2024): PRMs score step-level correctness, while SRM evaluates whether the chosen high-level approach was productive.

We apply two stabilizers motivated by reward design analyses (Gao et al., 2024; Cui et al., 2025): **group-wise normalization** of SRM scores to zero mean and unit variance within each GRPO group, preventing scale drift; and **correctness gating**:

$$R_{\text{SRM}}^{\text{gated}} = R_{\text{correct}} \cdot R_{\text{SRM}}, \tag{6}$$

which zeroes the SRM reward on failed rollouts. This is an anti-hacking measure: since the SRM is an approximation of true strategy quality, granting strategy reward on incorrect rollouts would let the policy exploit the gap between SRM score and actual execution success. With gating, the SRM acts as a *sharpening* signal within the set of correct rollouts, distinguishing those that succeeded via a sound strategy from those that succeeded via a fragile or inefficient route, rather than as a reward that substitutes for correctness.

GRPO training proceeds with the composite reward and gating, using the standard clipped surrogate objective (Schulman et al., 2017) with group-based advantage normalization and KL regularization toward π_{SFT} (Shao et al., 2024). RL is run on 7,200 problems sampled from DeepMath, with the SRM frozen throughout. Unlike methods that estimate plan quality online during RL (Dou et al., 2025), SRM-GRPO amortizes strategy-quality estimation into a one-time offline cost. In §4.6 we isolate the GRPO-stage cost, treating the SRM as a reusable pretrained reward model.

4 Experiments

We evaluate SRM-GRPO on competition mathematics benchmarks across three policy backbones, contrasting it with outcome-only GRPO, process-reward (PRM) integration, and online plan-reward estimation. The experiments are organized around the three research questions of §1: whether strategy quality can be learned from rollout outcomes (RQ1), whether the learned reward improves RL beyond outcome-only training (RQ2), and how a reusable pretrained SRM compares to online plan-reward methods in GRPO-stage compute (RQ3).

4.1 Experimental Setup

Models. We instantiate the policy on three backbones spanning two families and a 2.5× parameter range: Qwen3-4B, Qwen3-8B (Yang et al., 2025), and Llama-3.2-3B-Instruct. The strategy reward model is initialized from Qwen2.5-7B-Instruct (Yang et al., 2024a) and adapted with LoRA (rank 64) (Hu et al., 2021). All teacher-side strategy generation and judge filtering use Qwen3.5-397B-A17B.

Training data. Strategy-guided SFT uses 17,467 manually verified (question, teacher-strategy, reference-execution) triples drawn from MATH (Hendrycks et al., 2021),

Table 1: Main results: pass@1 (%) on four mathematical reasoning benchmarks. **Bold** marks the best result within each backbone block. SRM-GRPO improves the average over outcome-only GRPO, online plan-reward (PTA-GRPO), and a token-matched CoT-SFT control across all three backbones.

Backbone	Method	AIME24 ↑	AIME25 ↑	AMC23 ↑	MATH500 ↑	Avg ↑
Qwen3-4B	Base	23.00	22.67	64.00	76.76	46.61
	CoT-SFT (token-matched)	36.67	24.33	76.50	77.88	53.85
	SFT (strategy-guided)	42.00	29.67	80.00	81.82	58.37
	GRPO (outcome+format)	40.67	30.00	80.50	82.32	58.37
	PTA-GRPO (Dou et al., 2025)	40.33	31.00	82.25	81.54	58.78
	SRM-GRPO (ours)	**42.67**	**31.00**	**84.25**	**83.02**	**60.24**
Qwen3-8B	Base	26.00	21.00	62.75	78.20	46.99
	CoT-SFT (token-matched)	37.33	23.67	77.25	78.24	54.12
	SFT (strategy-guided)	41.33	31.33	82.75	82.02	59.36
	GRPO (outcome+format)	39.33	31.00	79.25	82.10	57.92
	PTA-GRPO (Dou et al., 2025)	39.67	31.33	82.50	81.94	58.86
	SRM-GRPO (ours)	**42.33**	**32.67**	**84.50**	**82.72**	**60.56**
Llama-3.2-3B-Instruct	Base	3.33	3.33	19.50	39.40	16.39
	CoT-SFT (token-matched)	2.67	1.33	21.00	43.20	17.05
	SFT (strategy-guided)	2.33	1.00	21.25	44.36	17.24
	GRPO (outcome+format)	3.00	2.00	25.00	46.60	19.15
	PTA-GRPO (Dou et al., 2025)	3.33	2.33	25.25	46.66	19.39
	SRM-GRPO (ours)	**6.67**	**3.33**	**30.00**	**48.32**	**22.08**

OpenThoughts (Guha et al., 2025), and DeepMath (He et al., 2025b). The SRM is trained on the merged preference set $\mathcal{D}_{\text{SRM}}$ of §3.3. GRPO is run on a 7,200-problem subset of DeepMath with group size $G = 8$, KL-regularized toward the SFT policy following Shao et al. (2024).

Benchmarks. We report pass@1 on four mathematical benchmarks of varying difficulty: AIME 2024, AIME 2025, AMC 2023, and MATH500. The *Avg* column is the arithmetic mean over the four benchmarks. For each problem we draw 10 samples at temperature 0.6 and report the mean pass@1. Evaluation variance is captured through the 10 inference samples per problem. Our SFT data draws from the MATH training split, OpenThoughts, and DeepMath. MATH500 is the MATH test split and is disjoint from our training data. The PRM is the publicly released Qwen2.5-Math-PRM-7B from Zhang et al. (2025b); Yang et al. (2024b), kept identical across PRM-only and PRM+SRM runs.

Compared methods. We organize comparisons along three axes: (i) the SFT/RL backbone (Base, strategy-guided SFT); (ii) the RL reward composition (GRPO with outcome+format only, with SRM, with PRM, with PRM+SRM); (iii) two reference baselines: a token-matched CoT-SFT control that distills the same teacher into a non-segmented format under a matched output-token budget, and PTA-GRPO (Dou et al., 2025), which estimates plan quality online during RL.

4.2 Main Results (RQ2)

Table 1 reports pass@1 across the four benchmarks and three backbones. The first comparison addresses RQ2: does adding a learned strategy-level reward to GRPO improve over outcome-only GRPO?

SRM-GRPO improves over outcome-only GRPO across backbones. On the average column, SRM-GRPO exceeds GRPO by +1.87 on Qwen3-4B (58.37 → 60.24), +2.64 on Qwen3-8B (57.92 → 60.56), and +2.93 on Llama-3.2-3B-Instruct (19.15 → 22.08). The gain is largest on the weakest backbone, consistent with the hypothesis that strategy-level credit is most useful when the outcome reward is otherwise sparse.

The strategy structure, not output length, drives the SFT gain. Under the same teacher and a matched output-token budget, the strategy-guided SFT outperforms a vanilla CoT-SFT control on all three backbones: +4.52 on Qwen3-4B (53.85 → 58.37), +5.24 on Qwen3-8B (54.12 → 59.36), and +0.19 on Llama-3.2-3B-Instruct (17.05 → 17.24). The Qwen-side gap rules out the simplest alternative explanation (that strategy-guided SFT helps merely because it produces longer outputs or distills a stronger teacher signal) and isolates the contribution of the explicit strategy/execution decomposition before any RL is applied. The much smaller Llama gap indicates that on this weaker backbone the structural advantage is largely latent at the SFT stage and only emerges in full once GRPO begins, consistent with the largest absolute SRM-GRPO gain we observed

on Llama in the previous paragraph.

Online plan rewards alone do not match the offline SRM. PTA-GRPO (Dou et al., 2025), which estimates plan quality online by resampling executions during RL, lands within ± 1 Avg of vanilla GRPO on all three backbones (Qwen3-4B 58.78 vs. 58.37; Qwen3-8B 58.86 vs. 57.92; Llama 19.39 vs. 19.15) and trails SRM-GRPO by 1.5–2.7 Avg points (Qwen3-4B 58.78 vs. 60.24, Qwen3-8B 58.86 vs. 60.56, Llama 19.39 vs. 22.08). On AMC23, both plan-aware methods improve over outcome-only GRPO, but the offline SRM scales further (Qwen3-4B GRPO 80.50, PTA 82.25, SRM-GRPO 84.25; Qwen3-8B 79.25 / 82.50 / 84.50). This is consistent with the amortization claim of §3.5: offline construction can afford substantially more rollouts per (question, strategy) pair than each online estimate can. Section 4.6 examines this trade-off under matched GRPO-stage compute, given a pretrained SRM.

Outcome-only GRPO can degrade SFT on the larger backbone. On Qwen3-8B, GRPO with only correctness and format rewards underperforms the SFT checkpoint on the average column (57.92 vs. 59.36) and drops on AIME24 and AMC23. SRM-GRPO recovers and exceeds the SFT baseline on every benchmark. This pattern, taken with the consistent gain across the other two backbones, indicates that the strategy-level signal is acting as a stabilizer rather than as a redundant proxy for correctness.

Hardest subsets benefit most. The Olympiad-level subsets concentrate the gain. On Qwen3-4B, AMC23 moves from 80.50 (GRPO) to 84.25 (SRM-GRPO), and AIME24 moves from 40.67 to 42.67. On Llama-3.2-3B-Instruct, AIME24 more than doubles ($3.00 \rightarrow 6.67$) and AMC23 rises by 5 points. Easier benchmarks (MATH500) move less, in agreement with the view that strategy choice matters more on problems requiring non-local planning.

4.3 Compatibility with Process Rewards

A natural concern is whether SRM redundantly captures signal already provided by step-level process rewards. We probe this by comparing SRM-GRPO against (i) GRPO with the publicly released Qwen2.5-Math-PRM-7B from Zhang et al. (2025b) and (ii) GRPO with both rewards combined (Table 2).

Table 2: PRM+SRM combination on Qwen3 backbones (Qwen2.5-Math-PRM-7B, Zhang et al., 2025b). The two rewards operate at different granularities and accumulate: GRPO+PRM+SRM achieves the best average and AIME24 across all configurations.

Backbone	Method	AIME24	AIME25	AMC23	MATH500	Avg
Qwen3-4B	SRM-GRPO (ours)	42.67	31.00	84.25	**83.02**	60.24
	GRPO + PRM	43.00	**34.33**	82.00	82.34	60.42
	GRPO + PRM + SRM (ours)	**45.67**	33.33	**86.00**	82.70	**61.93**
Qwen3-8B	SRM-GRPO (ours)	42.33	32.67	**84.50**	82.72	60.56
	GRPO + PRM	44.33	30.33	82.50	82.24	59.85
	GRPO + PRM + SRM (ours)	**45.33**	**33.33**	83.50	**83.08**	**61.31**

PRM and SRM are complementary. PRM-only and SRM-only achieve comparable averages, with PRM slightly stronger on the smaller backbone and SRM stronger on the larger one (Qwen3-4B: 60.42 vs. 60.24; Qwen3-8B: 59.85 vs. 60.56); their per-benchmark profiles also differ. PRM-only is stronger on AIME25 (Qwen3-4B 34.33), where local-step verification helps catch arithmetic mistakes. SRM-only is stronger on AMC23 (Qwen3-4B 84.25, Qwen3-8B 84.50), where the gain comes from selecting a viable solution route. Combining the two yields the highest average on both Qwen3 backbones (Qwen3-4B 61.93, Qwen3-8B 61.31) and the best AIME24 result overall (45.67, 45.33).

4.4 Does the SRM Select Useful Strategies?

The previous results entangle the SRM-induced training signal with the policy parameters it produces. To probe the SRM signal in isolation, we run a frozen-executor study. For 2,000 held-out problems we sample candidate strategies from a strategy generator, score each with the SRM, and select three strategies at high, medium, and low SRM scores to test whether downstream success tracks the SRM ranking. For each selected strategy we run a frozen Qwen3-4B executor with 10 different seeds and average pass@1 across seeds and across the 2,000 problems. The control runs the executor on the question alone with the same 10 seeds.

As shown in Figure 4 (Appendix F), executor pass@1 monotonically tracks SRM score: high-scored strategies outperform medium-scored ones, which in turn outperform low-scored ones, and all three exceed the no-strategy control. The signal is therefore not a side-effect of the policy update but is recoverable as a standalone selector at inference time, consistent with the SRM having learned a transferable notion of strategy quality (RQ1).

Table 3: Ablation on Qwen3-4B. Removing correctness gating decreases Avg by 0.99 (with the largest AIME24 drop of 2.00), and removing group normalization decreases Avg by 0.98.

Variant	AIME24	AIME25	AMC23	MATH500	Avg
SRM-GRPO (full)	**42.67**	**31.00**	84.25	**83.02**	**60.24**
− correctness gating	40.67	30.33	83.75	82.24	59.25
− group normalization	39.33	30.67	**84.50**	82.54	59.26

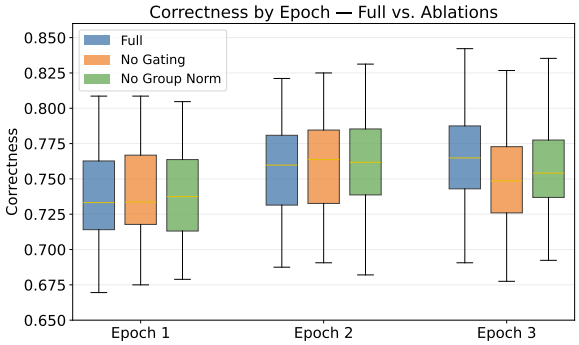

Figure 3: Per-epoch correctness on Qwen3-4B. Full SRM-GRPO improves through epoch 3, while removing either stabilizer causes plateau or regression in the final epoch.

4.5 Ablations and Stability Across Training

We isolate the contribution of the two stabilizers in §3.5 by removing them one at a time on Qwen3-4B. Table 3 reports final pass@1 after the full RL run. Figure 3 tracks the same three configurations across the three RL epochs to expose how the ablations evolve over training.

Correctness gating contributes most on the hardest subset. Removing the gate drops the final average by 0.99 points, with the AIME24 drop (−2.00) being the largest single-cell decrease. This is consistent with reward-hacking analyses (Gao et al., 2024): without the gate, the policy can drift toward strategies the SRM scores favorably but that do not translate into correct answers, an effect most visible on the hardest problems. Figure 3 shows the temporal signature of this drift: the No-Gating variant peaks at epoch 2 and *regresses* by epoch 3, while the full configuration is still improving.

Group-wise normalization protects against late-training drift. Removing normalization decreases Avg by 0.98, comparable to the gating ablation, but Figure 3 reveals a more specific pattern: the No-Group-Norm variant tracks the full configuration through epoch 2 and only diverges in epoch 3, with the median falling below the full curve. Score-scale drift across rollouts within a group accumulates as RL proceeds, eventually pushing the policy toward strategies whose SRM scores are inflated relative to peers. Group normalization is a cheap protection against this late-training failure mode.

Table 4: GRPO-stage cost on Qwen3-4B. GPU-hours include method-specific overhead: frozen SRM inference, online resampling, or PRM inference. SRM-GRPO matches GRPO+PRM accuracy with 45% less compute (174.05 vs. 318.30 GPU-h). The marginal overhead over outcome-only GRPO is ∼6 GPU-h.

Method	GRPO-stage GPU-h	AMC23	MATH500
GRPO (outcome+format)	168.06	80.50	82.32
PTA-GRPO	235.29	82.25	81.54
GRPO + PRM	318.30	82.00	82.34
SRM-GRPO	174.05	84.25	83.02

4.6 GRPO-Stage Cost Given a Pre-Trained SRM (RQ3)

PTA-style methods estimate plan quality online during RL by repeatedly resampling executions from the current policy (Dou et al., 2025). SRM amortizes that estimate into a pre-trained reward model that is frozen before RL begins. We measure the GRPO-stage GPU cost of each method: for SRM-GRPO this covers GRPO training plus frozen SRM inference during rollout scoring; for PTA-GRPO it covers GRPO training plus per-step online resampling overhead; for GRPO+PRM it covers GRPO training plus PRM inference.

SRM-GRPO outperforms PTA-GRPO by +2.00 on AMC23 and +1.48 on MATH500 while running 61 GPU-hours faster at the GRPO stage. Against GRPO+PRM, SRM-GRPO achieves comparable accuracy (+2.25 AMC23, +0.68 MATH500) with 45% less GRPO-stage compute (174.05 vs. 318.30 GPU-h).

5 Conclusion

We presented SRM-GRPO, a framework that learns strategy-level credit by training a Strategy Reward Model on confidence-aware pairwise preferences from counterfactual rollouts. Integrated into GRPO with correctness gating and group-wise normalization, SRM-GRPO consistently improves over outcome-only GRPO across three policy backbones, complements rather than replaces step-level PRMs, and achieves competitive accuracy at substantially lower GRPO-stage compute than online plan-reward baselines. The SRM also transfers as a standalone strategy selector, suggesting it captures a reusable notion of strategy quality.

Limitations

Our experiments are conducted on competition mathematics benchmarks; generalization to other reasoning domains such as code synthesis, scientific question answering, or multi-step planning is not assessed. The policy backbones span 3B–8B parameters on two model families; whether the strategy-level signal remains beneficial at larger scales is an open question. The SRM is initialized from Qwen2.5-7B-Instruct and trained on strategies generated by Qwen3.5-397B-A17B; preference quality depends on the capability of the teacher model, and adapting the pipeline to weaker or domain-specialized teachers may require retuning the confidence threshold and judge filter. Finally, the strategy–execution decomposition relies on structured output tags established during SFT; applying SRM-GRPO to base models that have not been fine-tuned on this format would require an additional SFT stage.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Luoxin Chen, Yichi Zhou, and Huishuai Zhang. 2026. Learning to generate formally verifiable step-by-step logic reasoning via structured formal intermediaries. *arXiv preprint arXiv:2603.29500*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

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

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Zhihao Dou, Qinjian Zhao, Zhongwei Wan, Dinggen Zhang, Weida Wang, Towsif Raiyan, Benteng Chen, Qingtao Pan, Yang Ouyang, Zhiqiang Gao, and 1 others. 2025. Plan then action: High-level planning guidance reinforcement learning for llm reasoning. *arXiv preprint arXiv:2510.01833*.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. 2024. On designing effective rl reward at training time for llm reasoning. *arXiv preprint arXiv:2410.15115*.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, and 1 others. 2025. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*.

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

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Haoran He, Yuxiao Ye, Qingpeng Cai, Chen Hu, Binxing Jiao, Daxin Jiang, and Ling Pan. 2025a. Random policy valuation is enough for llm reasoning with verifiable rewards. *arXiv preprint arXiv:2509.24981*.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, and 1 others. 2025b. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

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 *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

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

Mohammad Rezaei, Jens Lehmann, and Sahar Vahdati. 2026. Llm reasoning with process rewards for outcome-guided steps. *arXiv preprint arXiv:2604.02341*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics*, pages 2609–2634.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 9426–9439.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, and 1 others. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. 2025. Training llms with logicreward for faithful and rigorous reasoning. *arXiv e-prints*, pages arXiv–2512.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, and 1 others. 2024a. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024b. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

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. *Advances in neural information processing systems*, 36:11809–11822.

Huining Yuan, Zelai Xu, Huaijie Wang, Xiangmin Yi, Jiaxuan Gao, Xiao-Ping Zhang, Yu Wang, Chao Yu, and Yi Wu. 2026. Verifiable process rewards for agentic reasoning. *arXiv preprint arXiv:2605.10325*.

Chuhuai Yue, Chengqi Dong, Yinan Gao, Hang He, Jiajun Chai, Wei Lin, and Guojun Yin. 2026. Promoting efficient reasoning with verifiable stepwise reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34530–34538.

Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. 2025a. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1859–1869.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren

Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516, Vienna, Austria.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Comparison with Online Plan-Reward

SRM-GRPO differs structurally from methods that estimate plan quality online during RL (Dou et al., 2025). We detail three architectural distinctions:

Offline vs. online estimation. These methods estimate plan quality from rollouts of the *current* policy throughout training, requiring repeated sampling and quality re-estimation as the policy evolves. The SRM is trained once from rollouts of a fixed SFT policy and reused without modification throughout GRPO. This amortizes strategy-quality estimation: the computational investment in the two-level rollout procedure (§3.3) is a one-time offline cost, rather than a recurring tax on each RL update.

Unit of credit. Such methods typically evaluate plan–action trajectories holistically, conflating plan quality with execution success. The SRM isolates the strategy prefix and scores it based on strategy text alone, never observing execution tokens or the final answer. Within the set of correct rollouts, this produces finer-grained credit: the policy is rewarded more strongly for succeeding via a sound strategy than via a fragile or inefficient one, a distinction that is lost when plan and execution are scored jointly.

GRPO-stage cost. We measure the cost of the GRPO stage itself, assuming a strategy evaluator is already available. For SRM-GRPO, the only overhead over outcome-only GRPO is a frozen inference call per rollout (∼6 GPU-h). For PTA-GRPO, the overhead is per-step online resampling that recurs throughout training. This framing (cost given a pre-trained SRM) is explicit in the section title and table caption of §4.6. SRM training is a one-time offline investment that is reused across runs, backbones, and tasks.

The SRM occupies a specific design point (offline, strategy-level, outcome-grounded) that is complementary to both process reward models and online plan estimation.

B SFT Strategy Generation Prompt

For Stage 1 SFT, we prompt Qwen3.5-397B-A17B with each problem and its reference chain-of-thought solution to distill a high-level strategy. The teacher is instructed to extract the overall approach, key theorems, and major branching decisions while excluding numerical computations, low-level derivations, and the final answer. The resulting (question, strategy, execution) triples are then manually verified before being added to the SFT corpus. The prompt template is shown in Figure 6.

C SRM Preference Data Construction Details

C.1 Data Pipeline and Statistics

We begin with 32,000 problems sampled from DeepMath and OpenThoughts. For each problem, we sample $K = 8$ candidate strategies from π_{SFT} and generate $M = 24$ independent execution rollouts per strategy, yielding 192 rollouts per problem. After computing empirical success rates and applying the Beta-Binomial confidence filter ($\mathbb{P}[q(s^+) > q(s^-)] \geq 0.9$), 53,464 preference pairs survive. These are then passed through the LLM judge consistency check (described below), and 48,413 pairs are retained after discarding judge–rollout disagreements. The teacher-constructed contrastive set contributes an additional 17,197 pairs. After deduplication and format filtering, the merged training set $\mathcal{D}_{\text{SRM}}$ contains **65,610** preference pairs (Table 5).

Table 5: SRM preference data statistics. Rollout-induced pairs are constructed from 32,000 problems (DeepMath + OpenThoughts) with $K = 8$ strategies and $M = 24$ rollouts each.

Source	Raw pairs	Retained
Rollout-induced (post confidence filter)	53,464	—
+ LLM judge consistency filter	—	48,413
Teacher-constructed	17,197	17,197
Total $\mathcal{D}_{\text{SRM}}$	—	**65,610**

Algorithm 1 formalizes the construction pipeline.

Algorithm 1 SRM Preference Data Construction

Require: SFT policy π_{SFT}, problems $\mathcal{D}_{\text{prob}}$, judge $\mathcal{J}$, teacher $\mathcal{T}$, group sizes K, M, confidence threshold τ

$\mathcal{D}_{\text{rollout}} \leftarrow \emptyset,\ \mathcal{D}_{\text{teacher}} \leftarrow \emptyset$
for each $x \in \mathcal{D}_{\text{prob}}$ **do** ▷ rollout-induced source
  Sample $\{s^{(k)}\}_{k=1}^{K} \sim \pi_{\text{SFT}}(\cdot \mid x)$
  for $k = 1, \ldots, K$ **do**
    Sample M executions conditioned on $s^{(k)}$; $c^{(k)} \leftarrow$ #correct
    $\hat{q}^{(k)} \leftarrow c^{(k)}/M$
  end for
  for each (i, j) with $\hat{q}^{(i)} > \hat{q}^{(j)}$ **do**
    $p_{ij} \leftarrow \mathbb{P}\big[q^{(i)} > q^{(j)} \mid c^{(i)}, c^{(j)}, M\big]$ ▷ Beta-Binomial, Eq. 3
    if $p_{ij} \geq \tau$ **and** $\mathcal{J}(x, s^{(i)}, s^{(j)}) = \textsc{Reasonable}$ **then**
      $\mathcal{D}_{\text{rollout}} \leftarrow \mathcal{D}_{\text{rollout}} \cup \{(x, s^{(i)}, s^{(j)})\}$
    end if
  end for
end for
for each $x \in \mathcal{D}_{\text{prob}}$ **do** ▷ teacher-constructed source
  $(s^{+}, s^{-}) \leftarrow \mathcal{T}(x)$
  if s^{+}, s^{-} both pass deterministic format check **then**
    $\mathcal{D}_{\text{teacher}} \leftarrow \mathcal{D}_{\text{teacher}} \cup \{(x, s^{+}, s^{-})\}$
  end if
end for
$\mathcal{D}_{\text{SRM}} \leftarrow \textsc{Dedup}(\mathcal{D}_{\text{rollout}} \cup \mathcal{D}_{\text{teacher}})$
return $\mathcal{D}_{\text{SRM}}$

C.2 Strategy Extraction and Deterministic Tag Checking

We use a deterministic parser to extract the `<strategy>` and `<think>` segments from each model output. The parser is responsible only for structural validation and segment extraction: it checks whether the output contains exactly one `<strategy>...</strategy>` block, followed by exactly one `<think>...</think>` block, followed by a `\boxed{}` expression containing the final answer. Format-invalid outputs are filtered out before SRM training and reward computation. The LLM judge described below is *not* used for tag checking or answer extraction. All tag validation is performed by deterministic rules.

C.3 LLM-Assisted Preference Reasonableness Verification

To reduce noisy or spurious preference labels, we use an LLM judge (Qwen3.5-397B-A17B) to verify whether a constructed strategy preference is reasonable, following the LLM-as-a-Judge paradigm (Zheng et al., 2023). Unlike a pairwise reranker that independently picks between two unlabeled strategies, our judge is given the proposed preference label directly. For each preference pair (x, s^{+}, s^{-}), the judge receives the problem x, the preferred strategy s^{+}, and the rejected strategy s^{-}, and decides whether $s^{+} \succ s^{-}$ is justified. The judge does not observe execution rollouts, empirical success rates, or final answers. It evaluates only whether the preferred strategy is mathematically more sound and more likely to lead to a correct solution.

For rollout-induced pairs, s^{+} is the strategy with higher estimated execution success after confidence-aware filtering. A pair is retained only if the judge outputs `REASONABLE`. Pairs labeled `UNREASONABLE`, with invalid labels, or with ambiguous responses are discarded. The prompt template is shown in Figure 7.

C.4 Teacher-Constructed Contrastive Prompt

To complement rollout-induced pairs, which are limited to strategies the SFT policy can generate, we use Qwen3.5-397B-A17B to directly construct positive and negative strategy pairs for each problem. The prompt template is shown in Figure 8. Generated pairs are subject to the same format-validity check applied to rollout-induced pairs. Teacher-constructed negatives provide contrastive signal for recognizable structural errors (wrong theorem, infeasible subgoal order) that may be underrepresented in rollout-induced pairs, where both strategies must be producible by the SFT policy.

D Implementation Details

All experiments are run on 4×A100 80GB GPUs in bfloat16 precision. Table 6 lists the full hyperparameter configuration for the three stages. Below we summarize the role of each stage.

Stage 1: Strategy-Guided SFT. The base policy (Qwen3-4B) is fully fine-tuned with FSDP2 to learn the structured `<strategy>`/`<think>` output format using the standard next-token prediction loss. This stage establishes the segmentation format rather than improving raw reasoning ability.

Stage 2: SRM Training. The SRM is initialized from Qwen2.5-7B-Instruct, adapted with LoRA, and trained with the margin-augmented Bradley–Terry loss with scale anchoring (Eq. 4). Training takes approximately 4.38 hours on the 4×A100

80GB setup. The trained SRM is frozen before Stage 3 and is shared across all GRPO experiments. Its training cost is not attributed to any individual run in the compute comparison of §4.6.

Stage 3: SRM-GRPO. GRPO is initialized from the Stage 1 SFT checkpoint and trained with full parameters under DeepSpeed ZeRO-2. Each rollout receives a composite reward combining correctness, format, and the frozen SRM score, with group-wise normalization and correctness gating. KL regularization follows Shao et al. (2024).

Inference and evaluation. All evaluation rollouts are sampled with temperature 0.6, top-p 0.95, and maximum new tokens 32,768 (matching the maximum sequence length, 32,768). We use a maximum batch size of 32 for inference. For each problem we draw 10 independent samples and report the mean pass@1. Final answers are extracted by a deterministic parser from the `\boxed{}` expression that follows the `</think>` block, and compared against the gold answer using a math-aware verifier.

E SRM-GRPO Training Loop

Algorithm 2 summarizes the full SRM-GRPO update. For each problem, G rollouts are sampled from the current policy. A deterministic parser extracts the strategy and execution segments, the verifier checks answer correctness, and the frozen SRM scores the strategy text. SRM scores are normalized within the group, gated by correctness, and combined with format and correctness rewards into a composite scalar that drives the standard GRPO update.

F Frozen-Executor Selection Results

G Qualitative Example

To illustrate what the strategy/execution decomposition looks like in practice, Figure 5 shows a complete (question, strategy, execution) instance from our SFT corpus. The strategy segment captures the high-level approach (choice of theorem, key transformation, ordering of subgoals) without any numerical computation. The execution segment then carries out the concrete derivation under the strategy prefix.

Algorithm 2 SRM-GRPO

Require: SFT policy π_{SFT}, frozen SRM r_ϕ, problems $\mathcal{D}_{\text{RL}}$, group size G, coefficients α, β, γ

$\pi_\theta \leftarrow \pi_{\text{SFT}}$
for each training step **do**
  Sample $x \sim \mathcal{D}_{\text{RL}}$
  Sample group $\{o_g\}_{g=1}^{G} \sim \pi_\theta(\cdot \mid x)$
  for $g = 1, \ldots, G$ **do**
    $(s_g, y_g, \text{ok}_g) \leftarrow \text{ParseTags}(o_g)$
    $R_{\text{format}}^{(g)} \leftarrow \mathbf{1}[\text{ok}_g]$
    $R_{\text{correct}}^{(g)} \leftarrow \mathbf{1}[\text{verify}(y_g)]$
    $R_{\text{SRM}}^{(g)} \leftarrow r_\phi(x, s_g)$ ▷ strategy only
  end for
  $\tilde{R}_{\text{SRM}}^{(g)} \leftarrow \text{GroupNorm}\big(\{R_{\text{SRM}}^{(g)}\}_{g=1}^{G}\big)$
  for $g = 1, \ldots, G$ **do**
    $\hat{R}_{\text{SRM}}^{(g)} \leftarrow R_{\text{correct}}^{(g)} \cdot \tilde{R}_{\text{SRM}}^{(g)}$ ▷ gating
    $R^{(g)} \leftarrow \alpha R_{\text{correct}}^{(g)} + \beta R_{\text{format}}^{(g)} + \gamma \hat{R}_{\text{SRM}}^{(g)}$
  end for
  Compute group-relative advantages $\hat{A}^{(g)}$ from $\{R^{(g)}\}$
  Update π_θ via clipped surrogate (Schulman et al., 2017) with KL penalty toward π_{SFT}
end for
return π_θ

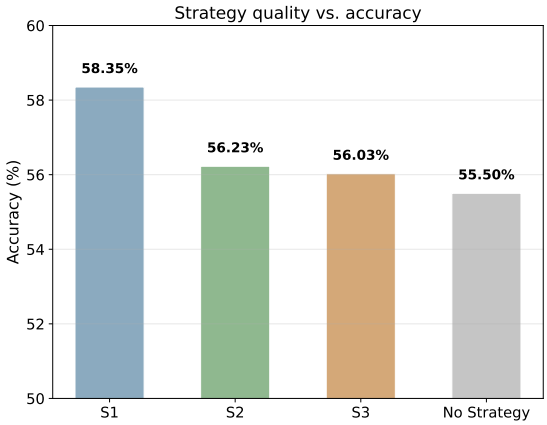

Figure 4: Frozen-executor pass@1 on 2,000 held-out problems with strategies selected at high, medium, and low SRM scores. Pass@1 tracks the SRM ranking, and all three levels exceed the no-strategy baseline.

H Dataset Licenses and Ethical Considerations

Datasets. We use three publicly released mathematical reasoning datasets, all employed within their intended research use: MATH (Hendrycks et al., 2021) is released under the MIT License; OpenThoughts (Guha et al., 2025) is released under the Apache 2.0 License; DeepMath-103K (He et al., 2025b) is released under the MIT License. For evaluation, AIME 2024, AIME 2025, and AMC 2023 problems are publicly released competition problems. MATH500 is the standard test split of MATH used by Lightman et al. (2024). SFT uses MATH-train, OpenThoughts, and Deep-

Table 6: Training hyperparameters for the three-stage SRM-GRPO pipeline on the Qwen3-4B backbone. The same configuration is applied to the Qwen3-8B and Llama-3.2-3B-Instruct backbones, except for the policy model and the matching Stage-1 SFT checkpoint used to initialize Stage 3.

Hyperparameter	**Stage 1: SFT**	**Stage 2: SRM**	**Stage 3: SRM-GRPO**
Model			
Base / policy model	Qwen3-4B	Qwen2.5-7B-Instruct	Stage-1 checkpoint
Reward model	–	–	Stage-2 SRM (frozen)
Training type	Full fine-tuning	LoRA	Full fine-tuning
LoRA rank / α / target	–	64 / 128 / all-linear	–
Reward / value head	–	Scalar value head	–
Data			
Training data size	17,467 triples	65,610 pairs	7,200 problems
Optimization			
Number of epochs	3	3	3
Learning rate	4×10^{-5}	1×10^{-4}	1×10^{-6}
Warmup ratio	0.05	0.05	–
Per-device batch size	1	1	2
Gradient accumulation	8	4	16
Effective global batch	32	16	128
Number of GPUs	4×A100 80GB	4×A100 80GB	4×A100 80GB
Precision	bfloat16	bfloat16	bfloat16
Distributed strategy	FSDP2	DeepSpeed ZeRO-2	DeepSpeed ZeRO-2
Gradient checkpointing	Enabled	–	–
Sequence length			
Maximum input length	8,192	2,048	4,095
Maximum completion length	–	–	4,096
Loss / Reward			
Loss function	Next-token prediction	Margin BT + scale anchor	GRPO clipped surrogate
Margin m	–	0.5	–
Scale regularization λ	–	0.02	–
Reward components	–	–	Correctness, format, SRM
Reward weights ($\alpha/\beta/\gamma$)	–	–	0.7 / 0.1 / 0.2
GRPO-specific			
GRPO group size G	–	–	8
Sampling temperature	–	–	0.9
KL coefficient	–	–	0.02
vLLM mode / GPU mem util	–	–	Colocate / 0.35

Math, sampled after deduplication against AIME 2024, AIME 2025, AMC 2023, and MATH500. The subset drawn from MATH uses only the training split, and MATH500 is never used in SFT, SRM training, or RL.

Models. The Qwen3-4B/8B, Qwen2.5-7B-Instruct, and Qwen2.5-Math-PRM-7B models (Yang et al., 2025, 2024a,b) are released under the Apache 2.0 License. Llama-3.2-3B-Instruct is released under the Meta Llama 3 Community License. The Qwen3.5-397B-A17B teacher model is accessed via its publicly available API for strategy generation and judge filtering, and we follow the provider's terms of service. All released model checkpoints in this work will be distributed under the Apache 2.0 License, consistent with the policies of the upstream models.

The full prompt templates referenced in earlier sections are collected here. Figure 6 shows the SFT strategy generation prompt (§B); Figure 7 shows the LLM-assisted preference verification prompt (§C.3); Figure 8 shows the teacher-constructed contrastive prompt (§C.4).

Example Strategy–Execution Output

Question:
A car travels 40 kph for 20 kilometers, 50 kph for 25 kilometers,60 kph for 45 minutes and 48 kph for 15 minutes. What is the average speed of the car, in kph?

<strategy>
Use average speed as total distance divided by total time. Convert all times to hours, compute the missing distance or time for each segment, then aggregate the totals before dividing.
</strategy>

<think>
Average speed should be computed from the whole trip, not by directly averaging the listed speeds. The first segment gives a distance of 20 km and a speed of 40 kph, so its time is 20/40 = 0.5 hours.
The second segment gives 25 km at 50 kph, so its time is 25/50 = 0.5 hours.

For the third segment, the time is given as 45 minutes, which is 45/60 = 0.75 hours. At 60 kph, this segment covers 60 × 0.75 = 45 km. For the fourth segment, 15 minutes is 0.25 hours, so at 48 kph the distance is 48 × 0.25 = 12 km.

Now the total distance is 20 + 25 + 45 + 12 = 102 km, and the total time is 0.5 + 0.5 + 0.75 + 0.25 = 2 hours. Therefore the average speed is 102 / 2 = 51 kph.
</think>

Final answer:
\boxed{51}

Figure 5: A complete (question, strategy, execution) example. The `<strategy>` block specifies the high-level route, while the `<think>` block performs the detailed derivation that produces the final answer.

SFT Strategy Generation Prompt

You are given a math problem and a detailed chain-of-thought solution.

Your task is to generate a concise high-level strategy for solving the problem based on the detailed solution.

Requirements:
- The strategy should describe the overall problem-solving approach.
- Include only major phases, key mathematical ideas, theorem choices, transformations, and important conditional branches.
- Do not include detailed derivations, numerical calculations, algebraic manipulations, or low-level intermediate steps.
- Do not reveal the final answer.
- The strategy should be concise, structured, and useful for guiding a later detailed solution.
- The strategy should be written as an actionable plan, not as a summary of what the solution already did.

Output format:
<strategy>...</strategy>

Problem:
{question}

Detailed solution:
{solution}

Figure 6: SFT strategy generation prompt used to distill high-level strategies from teacher chain-of-thought solutions.

LLM-Assisted Preference Reasonableness Verification Prompt

You are an expert mathematics problem-solving judge.

Your task is to determine whether a proposed preference between two high-level solution strategies is reasonable.

You will be given:
1. A math problem.
2. A preferred strategy.
3. A rejected strategy.

The preferred strategy is claimed to be more likely to lead to a correct solution than the rejected strategy. Judge whether this preference is reasonable.

A reasonable preference means that the preferred strategy is better than the rejected strategy in at least one important mathematical aspect, such as:
1. It chooses a more appropriate theorem, transformation, or method.
2. It proposes a more feasible order of subgoals.
3. It avoids a mathematical flaw present in the rejected strategy.
4. It includes a critical step that the rejected strategy omits.
5. It gives more useful high-level guidance for reaching a correct solution.

Do NOT judge based on writing style, length, or surface fluency.
Do NOT solve the problem in detail.
Do NOT use or infer the final answer.
Do NOT require the preferred strategy to be perfect;
it only needs to be meaningfully better than the rejected strategy.

Problem:
{problem}

Preferred strategy:
{positive_strategy}

Rejected strategy:
{negative_strategy}

Is the proposed preference reasonable?
Output only one of the following labels:
REASONABLE
UNREASONABLE

Figure 7: LLM-assisted preference reasonableness verification prompt: given a problem and a labeled preference (s^+, s^-), the judge decides whether the proposed preference is justified.

Teacher-Constructed Contrastive Prompt

You are an expert mathematics teacher constructing contrastive training data for a strategy reward model.

You are given a math problem and a detailed chain-of-thought solution.

Your task is to generate two high-level solution strategies:

1. A POSITIVE strategy:
   A concise high-level strategy distilled from the detailed solution. It should correctly identify the main mathematical approach, relevant theorems or transformations, key intermediate targets, and a feasible order of subgoals.

2. A NEGATIVE strategy:
   A plausible high-level strategy for the same problem that is meaningfully worse than the positive strategy.
   It should be constructed by modifying the positive strategy or the detailed solution at the strategy level, while keeping it natural and problem-specific.

The negative strategy should contain at least one important strategic flaw, such as:
- choosing an inappropriate theorem, transformation, or representation;
- pursuing the right idea in an infeasible or circular order;
- omitting a critical high-level step needed by the detailed solution;
- focusing on a misleading quantity or subproblem;
- giving guidance that is too vague to support a correct execution.

Requirements for both strategies:
- Include only high-level reasoning plans.
- Do not include detailed derivations, numerical calculations, algebraic manipulations, or low-level intermediate steps.
- Do not reveal the final answer.
- Do not copy long phrases from the detailed solution.
- Keep the positive and negative strategies similar in length and style, so that the preference depends on mathematical usefulness rather than fluency or detail.
- The positive strategy should be actionable and faithful to the detailed solution.
- The negative strategy should be plausible but strategically flawed; it should not be obviously nonsensical.

Output format:
<positive_strategy>
[Your positive strategy here]
</positive_strategy>
<negative_strategy>
[Your negative strategy here]
</negative_strategy>

Problem:
{problem}

Detailed solution:
{solution}

Figure 8: Teacher-constructed contrastive prompt: the teacher LLM generates a paired positive and negative strategy for each problem, exposing recognizable structural errors that may be underrepresented in rollout-induced data.