THE MODEL KNOWS ANOTHER WAY: STRATEGY SWITCHING FOR EFFECTIVE RLVR EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) is often limited by insufficient exploration: difficult problems can yield uniformly incorrect rollout groups and therefore little learning signal. We show that such failures need not reflect missing capability. Instead, finite sampling often concentrates on a problem-specific dominant reasoning strategy while leaving alternative strategies already supported by the model unexplored. Moreover, the accessibility of these strategies evolves during RL: some are internalized into autonomous behavior, while others become difficult to elicit before being absorbed. Motivated by these observations, we introduce **Problem–Strategy Rollout Allocation (PSRA)**, which treats unguided and strategy-conditioned prompts as competing exploration arms and uses Bayesian sequential allocation to direct a fixed rollout budget toward arms most likely to yield informative, non-saturated groups. A preservation objective keeps useful strategy-conditioned routes accessible while successful guided behaviors are transferred to the unguided policy. Across Qwen2.5 models from 1.5B to 7B and two RL training corpora, PSRA consistently improves reasoning performance, reduces dead saturation, strengthens out-of-distribution transfer, and maintains larger gains under increased inference budgets.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a central paradigm for improving the reasoning performance of large language models. Yet its gains remain strongly constrained by exploration. Growing evidence suggests that RL often sharpens reasoning trajectories already accessible to the base model: pass@1 can improve substantially while pass@k or broader solution coverage improve much less, while concentrating probability mass onto a narrower set of reasoning modes (Yue et al., 2025; Wu et al., 2025; Lee et al., 2026a). This limitation is particularly consequential on hard problems, where uniformly unsuccessful rollout groups provide little or no learning signal. Recent approaches therefore introduce solution prefixes, hints, privileged contexts, or strategy-level conditions to make successful trajectories reachable (Yan et al., 2025; Nath et al., 2025; Zhang et al., 2025a; Wang et al., 2025; Agrawal et al., 2026; Lee et al., 2026b). While effective, these methods leave open a more basic question: ***what should the model be guided toward when its default exploration fails?***

We study this question at the level of *reasoning strategies*: reusable, high-level solution procedures already supported by the model. Clustering internal representations of base-model rollouts reveals a compact repertoire of semantically coherent strategies, which can be reliably re-elicited through natural-language instructions. Crucially, their use is strongly problem-conditioned. For a given problem, finite autonomous sampling typically concentrates on one dominant strategy direction, leaving other members of the same repertoire weakly selected. This concentration becomes costly when the dominant route fails: on problems where the initial rollout group is entirely incorrect, allocating the same additional budget to another strategy recovers successful trajectories substantially more often than either continuing along the dominant strategy or simply sampling more from the default policy. Thus, an all-failure group need not indicate the absence of a viable solution procedure; it can instead reflect a finite-budget *strategy-selection failure*. This is consistent with evidence that RL primarily selects among reasoning patterns acquired beforehand (Krishnamurthy et al., 2026), and conditioning can expose modes neglected by naive sampling (Lee et al., 2026b; Xu et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Strategy-conditioned exploration, however, introduces a second challenge: the available exploration directions themselves evolve during RL. Tracking fixed strategy conditions across checkpoints reveals two distinct dynamics. Some strategies become increasingly expressed without guidance, indicating successful internalization, whereas others lose their ability to redirect the policy without a corresponding increase in autonomous use. A strategy is therefore not merely a one-time sampling scaffold, but a control handle whose usefulness must persist long enough to be exploited and transferred. This distinction matters for existing guided-RL methods. Without explicitly maintaining the conditional distribution, useful alternative routes may disappear before their behaviors are absorbed. Effective strategy-guided exploration must therefore address both *where to explore* and *how to keep those exploration directions available*.

Motivated by these observations, we introduce ***Problem–Strategy Rollout Allocation (PSRA)***, which treats the unguided prompt and each strategy-conditioned prompt as competing exploration arms. For each problem–strategy pair (q, z), PSRA first collects a small probe and constructs a hierarchical Bayesian posterior over its success probability, incorporating problem difficulty, global strategy effectiveness, and the predicted reasoning mass left uncovered by finite sampling. It then allocates the remaining fixed rollout budget to maximize the expected number of non-saturated groups, directing computation toward arms most likely to provide informative group-relative gradients. Because the unguided condition participates in the same objective, strategy guidance is selected only when it offers greater expected learning value than continued autonomous sampling, avoiding hand-designed failure triggers or fixed routing rules. Successful guided trajectories are transferred to the unguided policy through importance-corrected updates, while an on-context preservation term maintains the strategy-conditioned routes that made those trajectories reachable.

Experiments across models with varying scales and two RL training corpora consistently validate this design. PSRA recovers 26% of initially unsolved problems versus 17% for GRPO, reduces dead saturation while preserving higher policy entropy, and its advantage generally increases with larger pass@k budgets, indicating broader strategy coverage. Our contributions are summarized as:

- We provide a strategy-level account of RLVR exploration that LLMs contain a steerable repertoire of reasoning strategies, yet finite autonomous sampling concentrates strongly on problem-specific dominant routes, while the accessibility of alternative routes can either be internalized or selectively lost during RL.
- We introduce **PSRA**, a Bayesian allocation framework over problem–strategy arms that targets non-saturated learning signals while jointly transferring successful guided behaviors to the unguided policy and preserving useful strategy-conditioned exploration directions.
- Experiments across model scales and training distributions, showing consistent improvements over vanilla GRPO and guided-RL baselines, stronger recovery of initially unsolved problems, reduced saturation, and increasing advantages under larger inference budgets.

2 Related Work

Learning signals in RLVR. GRPO relies on within-group reward variation, so all-correct or all-incorrect groups yield zero relative advantages and provide no reward-driven update (Shao et al., 2024). DAPO alleviates this degeneracy through dynamic sampling that filters saturated groups and resamples prompts to maintain informative batches (Yu et al., 2025). Such methods improve the availability of learning signals, but an all-failure group is typically treated as an uninformative prompt rather than a recoverable exploration failure. PSRA instead leverage another reasoning strategy already available to the model to turn the problem into a non-saturated, learnable group.

Guided RLVR exploration. A growing line of work uses external guidance to recover successful reasoning trajectories when on-policy exploration fails. LUFFY mixes off-policy expert traces with on-policy rollouts (Yan et al., 2025); Guide adaptively introduces guidance on difficult problems (Nath et al., 2025); BREAD branches from partial expert anchors (Zhang et al., 2025b); and HINT uses abstract Meta-Hints with affinity-aware optimization to reduce mismatch between guidance and the policy (Wang et al., 2025). NudgeRL conditions rollouts on strategy-level contexts (Lee et al., 2026b), while Uniqueness-Aware RL upweights correct trajectories that instantiate rare observed strategies (Hu et al., 2026). These methods establish the value of guided or diverse reasoning, but largely treat the guidance as given or act on patterns already sampled. PSRA instead constructs a

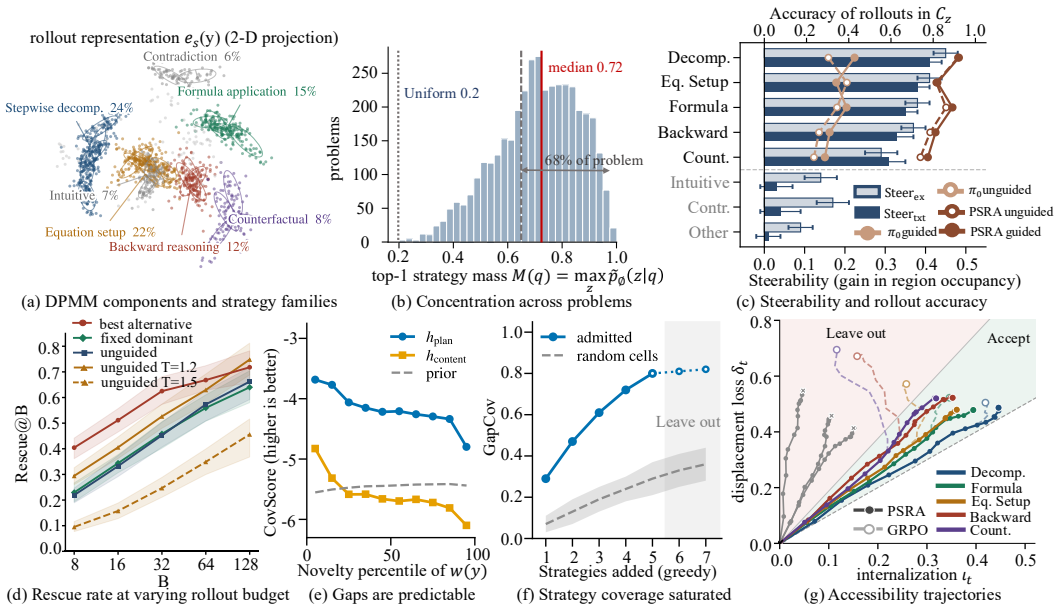


Figure 1: **Diagnosing strategy-level exploration in the base policy.** (a) Representation-space clustering reveals a compact set of reasoning strategies. (b) Autonomous rollouts are strongly concentrated. (c) Strategy conditioning reliably redirects generation toward distinct reasoning regions. (d) Switching to the best alternative strategy improves rescue rate over continued dominant-strategy or unguided sampling. (e) Pre-generation states predict reasoning directions missed by finite rollouts. (f) A small set of certified strategies covers most of the predicted exploration gaps. (g) During RL, strategy accessibility exhibits both internalization and displacement.

reusable strategy repertoire from the model itself, estimates its problem-specific exploration value online, and explicitly preserves useful strategy conditions as the policy evolves.

3 From Strategy to Problem-Conditioned Exploration

Pretraining equips the base policy with a compact repertoire of stable reasoning strategies. We first identify these strategies, then examine how finite-budget autonomous sampling selects among them and where alternative strategies provide useful exploration opportunities.

Reusable pretrained strategy repertoire. We re-encode base-model rollouts under their original problem context and pool intermediate hidden states into rollout representations $e_s(y)$. A Dirichlet process mixture model (DPMM) partitions these representations into fine-grained components, which are merged into semantic strategy candidates according to their solution structure (Fig. 1a). We behaviorally certify each candidate using both exemplar and natural-language instruction conditioning, retaining only strategies that consistently redirect rollouts toward their associated representation regions. Five of seven candidates satisfy this criterion and form the strategy set $\mathcal{Z}$; natural-language instructions retain 88% of the displacement induced by full exemplars, while matched placebo instructions have negligible effect (Fig. 1c).

For each strategy z, let $\mathcal{K}_z$ denote its associated DPMM components. We estimate intervention lift

$$\ell_z(k) = \frac{P(k(y) = k \mid z)}{P(k(y) = k \mid \varnothing)}, \tag{1}$$

and define its footprint $C_z \subseteq \mathcal{K}_z$ by accumulating the highest-lift components until they account for 50% of the strategy-associated guided mass.

Repeating the same analysis across Qwen2.5-1.5B, 3B, and 7B recovers the same strategy families across scales, with only slightly weaker elicitation stability on the 1.5B model, providing preliminary evidence that the discovered repertoire captures a shared and sufficiently complete strategy basis.

Problem-conditioned strategy selection. For a problem q, define the autonomous occupancy of strategy z as

$$p_{\varnothing}(z \mid q) = P(k(y) \in C_z \mid q, \varnothing), \qquad z_{\mathrm{dom}}(q) = \arg\max_{z \in \mathcal{Z}} \frac{p_{\varnothing}(z \mid q)}{\sum_{z' \in \mathcal{Z}} p_{\varnothing}(z' \mid q)}. \tag{2}$$

Autonomous generation is strongly concentrated at the problem level: most strategy-associated mass is typically assigned to a single dominant strategy (Fig. 1b), leaving alternative directions weakly explored. This becomes consequential when the dominant route fails. On problems for which the initial rollout group contains no correct solution, allocating the same additional budget to the dominant strategy or to unguided sampling recovers substantially fewer problems than switching to the best alternative strategy (Fig. 1d). Thus, a zero-reward group can reflect a finite-budget strategy-selection failure rather than the absence of a viable reasoning procedure.

Predicting missed strategy opportunities. The usefulness of a strategy depends on both the problem and what finite sampling has already covered. We therefore train a lightweight predictor G from the pre-generation planning state $h_{\mathrm{plan}}(q)$ to a distribution $\hat{q}(k \mid q)$ over frozen DPMM components. On held-out problems, we evaluate G with a novelty-weighted log score that emphasizes trajectories far from the observed rollout group; the planning representation substantially outperforms an early-layer content representation, including on the most novel trajectories (Fig. 1e). Refer to Appendix A.1 for more details and training procedure.

We define the high-probability components left unvisited by an observed rollout set O as

$$U_m(q; O) = \mathrm{Top}_m \left\{ \hat{q}(k \mid q) : k \notin \mathrm{hit}(O) \right\}, \tag{3}$$

and the remaining opportunity associated with strategy z as

$$\Omega(q, z; O) = \sum_{k \in U_m(q;O) \cap C_z} \hat{q}(k \mid q). \tag{4}$$

The five certified strategies account for most of the predicted gap mass, far exceeding matched random component sets, with coverage saturating rapidly as strategies are added (Fig. 1f). Hence, a small strategy basis captures most structured exploration opportunities exposed by finite sampling.

Strategy accessibility evolves during RL. Strategy-guided exploration additionally requires that a strategy remain an effective control handle as the policy changes. With the initial footprints $C_z^{(0)}$ frozen, we measure strategy-induced displacement as

$$D_t(z) = \mathrm{JS}\left(P_t(k \mid z) \,\|\, P_t(k \mid \varnothing) \right), \tag{5}$$

and decompose its reduction into

$$\delta_t(z) = 1 - \frac{D_t(z)}{D_0(z)}, \qquad \iota_t(z) = \frac{P_t(C_z^{(0)} \mid \varnothing) - P_0(C_z^{(0)} \mid \varnothing)}{P_0(C_z^{(0)} \mid z) - P_0(C_z^{(0)} \mid \varnothing)}, \qquad L_t(z) = \delta_t(z) - \iota_t(z). \tag{6}$$

Here ι_t measures internalization into autonomous behavior, while L_t captures residual loss of conditional accessibility. Both dynamics occur during standard RLVR: some strategies are progressively internalized, whereas others lose steerability before their behaviors are absorbed (Fig. 1g).

Together, these observations identify the problem–strategy pair (q, z) as the natural unit of structured exploration: $\Omega(q, z; O)$ captures the remaining coverage opportunity, $p_t(q, z)$ its current utility, and $L_t(z)$ whether the corresponding strategy remains accessible.

4 Problem–Strategy Rollout Allocation

Section 3 provides a fixed strategy set $\mathcal{Z}$ and its representation-space footprints. PSRA treats these strategies together with the unguided condition as exploration arms,

$$\mathcal{A} = \{\varnothing\} \cup \mathcal{Z}, \qquad p_t(q, z) = \Pr_{y \sim \pi_{\theta_t}(\cdot \mid c(q,z))}[r(q, y) = 1], \tag{7}$$

where $c(q, z)$ is the context for problem q under arm z. The goal is to allocate a fixed rollout budget across (q, z) pairs whose success probabilities are unknown and evolve during training.

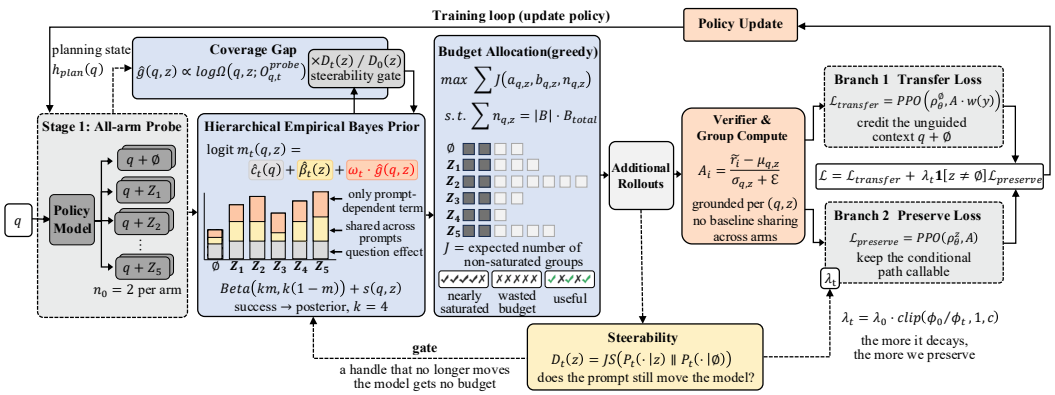


Figure 2: **Overview of PSRA Structure.** PSRA allocates rollout budget over unguided and strategy-conditioned arms, then transfers successful guided behaviors while preserving strategy accessibility.

4.1 Bayesian Allocation over Problem–Strategy Arms

Probe and posterior. At each step, PSRA draws n_0 rollouts from every active (q, z) arm and observes $s_{q,z}$ successes. We use an empirical-Bayes prior whose mean combines problem difficulty, global arm effectiveness, and the remaining strategy-addressable opportunity:

$$m_t(q,z) = \sigma\Big(\hat{c}_t^{-z}(q) + \hat{\beta}_t(z) + \omega_t \hat{g}_t(q,z)\Big). \tag{8}$$

Here $\hat{c}_t^{-z}(q)$ is a leave-one-arm-out estimate of problem difficulty, $\hat{\beta}_t(z)$ is a centered probe-only EMA of the global arm effect that tracks the global effect of strategy z, and $\hat{g}_t$ captures its uncovered problem-specific opportunity. $\omega_t \geq 0$ is an online coefficient learned from preceding probe batches; their estimators are given in Appendix A.1.7. After the all-arm probe, let

$$O_{q,t}^{\text{probe}} = \bigcup_{a \in \mathcal{A}} \text{probe}_t(q,a), \qquad \Omega_t(q,z) = \frac{D_t(z)}{D_0(z)} \Omega(q,z; O_{q,t}^{\text{probe}}), \quad z \neq \varnothing. \tag{9}$$

We set $\hat{g}_t(q,z)$ to the centered log transform of $\Omega_t(q,z)$ and $\hat{g}_t(q,\varnothing) = 0$. Thus, components reached by any probed arm are removed from the remaining opportunity, while $D_t(z)/D_0(z)$ discounts strategies whose steerability has degraded.

With prior strength κ, the prior and posterior are (with $n_0 = 2$ and $\kappa = 4$)

$$\begin{aligned} p_t(q,z) &\sim \text{Beta}(\kappa m_t(q,z),\ \kappa[1 - m_t(q,z)])\,, \\ a_{q,z} &= \kappa m_t(q,z) + s_{q,z}, \quad b_{q,z} = \kappa[1 - m_t(q,z)] + n_0 - s_{q,z}. \end{aligned} \tag{8}$$

Adaptive rollout allocation. Let $U_{q,z}(k)$ denote the posterior probability that arm (q, z) is non-saturated after receiving k additional rollouts. For binary rewards,

$$U_{q,z}(k) = \begin{cases} 1, & 0 < s_{q,z} < n_0, \\ 1 - \dfrac{B(a_{q,z},\, b_{q,z} + k)}{B(a_{q,z},\, b_{q,z})}, & s_{q,z} = 0, \\ 1 - \dfrac{B(a_{q,z} + k,\, b_{q,z})}{B(a_{q,z},\, b_{q,z})}, & s_{q,z} = n_0, \end{cases} \tag{9}$$

where $B(\cdot,\cdot)$ is the beta function. Mixed probes are already informative, whereas saturated arms are valuable only if their posterior assigns sufficient probability to the opposite outcome.

PSRA allocates the remaining budget by

$$\max_{\{k_{q,z}\}} \sum_{q \in \mathcal{B}} \sum_{z \in \mathcal{A}} U_{q,z}(k_{q,z}), \quad \sum_{q \in \mathcal{B}} \sum_{z \in \mathcal{A}} k_{q,z} \leq |\mathcal{B}|\, b_{\text{extra}}, \quad 0 \leq k_{q,z} \leq n_{\max} - n_0. \tag{10}$$

using greedy marginal gains $\Delta_{q,z}(k) = U_{q,z}(k+1) - U_{q,z}(k)$. Since $U_{q,z}$ is monotone and concave in integer k, this greedy allocation is optimal for the separable objective. We use $n_{\max} = 8$ and

$b_{\text{extra}} = 10$. With five strategies plus the unguided arm, this corresponds to 22 rollouts per prompt on average across the minibatch. Because $\varnothing$ competes under the same objective, guidance is selected only when it offers greater expected learning value than continued autonomous sampling.

Both probe and additionally allocated samples form the same reward group and contribute to the arm-wise advantage whenever the resulting group is non-saturated.

4.2 Learning from Strategy-Conditioned Rollouts

All rollouts from the same (q, z) arm form one reward group. We compute within-arm advantages,

$$A_i = \frac{r_i - \mu_{q,z}}{\sigma_{q,z} + \epsilon}, \tag{11}$$

and discard groups with zero reward variance. This avoids using a common baseline across strategy conditions whose success rates can differ substantially.

For $z = \varnothing$, the update reduces to GRPO. For $z \neq \varnothing$, the rollout was sampled from $\pi_{\theta_{\text{old}}}(\cdot \mid c(q, z))$ while deployment uses $c(q, \varnothing)$. We transfer guided trajectory to unguided policy by an off-context importance correction. Let the sequence-level behavior correction and token-level policy ratio be

$$w(y) = \frac{\pi_{\theta_{\text{old}}}(y \mid c(q, \varnothing))}{\pi_{\theta_{\text{old}}}(y \mid c(q, z))}, \quad \rho^{\varnothing}_{\theta,u} = \frac{\pi_\theta(y_u \mid c(q, \varnothing), y_{<u})}{\pi_{\theta_{\text{old}}}(y_u \mid c(q, \varnothing), y_{<u})}. \tag{12}$$

Here $w(y)$ corrects the guided behavior distribution at the sequence level and is clipped in log space, while $\rho^{\varnothing}_{\theta,u}$ is the standard token-level PPO ratio under the unguided context. The transfer loss is

$$\mathcal{L}_{\text{tr}} = -\frac{1}{|y|} \sum_u \min\Big(\rho^{\varnothing}_{\theta,u} A_i \, w(y), \; \text{clip}(\rho^{\varnothing}_{\theta,u}, 1 \pm \epsilon) \, A_i \, w(y)\Big). \tag{13}$$

This update uses strategy conditioning to obtain informative trajectories while optimizing the unguided policy that is used at inference time.

4.3 Maintaining Strategy Accessibility

The allocation mechanism assumes that the strategy instructions remain effective control handles. Section 3 shows that their steerability can decay during RL. For a guided rollout, we therefore retain a small on-context PPO term

$$\mathcal{L}_{\text{pres}} = -\frac{1}{|y|} \sum_u \min\big(\rho^{z}_{\theta,u} A_i, \; \text{clip}(\rho^{z}_{\theta,u}, 1 \pm \epsilon) \, A_i\big), \qquad \rho^{z}_{\theta,u} = \frac{\pi_\theta(y_u \mid c(q, z), y_{<u})}{\pi_{\theta_{\text{old}}}(y_u \mid c(q, z), y_{<u})}. \tag{14}$$

The final objective is

$$\mathcal{L}_{\text{PSRA}} = \mathcal{L}_{\text{tr}} + \lambda_t \, \mathbb{I}[z \neq \varnothing] \, \mathcal{L}_{\text{pres}}. \tag{15}$$

The preservation term is applied only to guided arms. Its coefficient is kept small and can be increased as the measured displacement $D_t(z)$ declines. Consequently, successful guided trajectories are transferred into autonomous behavior while the strategy conditions that expose alternative reasoning routes remain usable during subsequent allocation steps.

5 Experiment

5.1 Experimental Setup

Models and Datasets. We conduct experiments on Qwen2.5-7B-Instruct, Qwen2.5-3B-Instruct, and Qwen2.5-1.5B-Instruct (Yang et al., 2024). Unless otherwise specified, models are trained on DAPO-Math-17K-Processed (Yu et al., 2025) with Math-Verify for correctness rewards. To examine robustness, we additionally train Qwen2.5-7B-Instruct on DeepScaleR-40K (Luo et al., 2025).

Evaluation Benchmarks and Metrics. We evaluate in-distribution (ID) mathematical reasoning capability on AIME 2024/2025, AMC 2023, MATH-500 (Lightman et al., 2023), Minerva Math (Lewkowycz et al., 2022), and Omni-MATH (Gao et al., 2024). To test whether the learned reasoning improvements transfer beyond the mathematical training domain, we further evaluate out-of-distribution (OOD) generalization on StrategyQA (Geva et al., 2021), GPQA-Diamond (Rein et al.,

Table 1: Main results across model scales, training dataset, and evaluation distributions.

Model	In-Distribution Performance						Out-of-Distribution Performance			
	AIME 24/25	AMC	MATH-500	Minerva	OmniMath	Avg@8	StrategyQA	GPQA	MMLU-Pro	Avg@8
Qwen2.5-7B/3B/1.5B-Instruct on DAPO-Math-17K										
Qwen2.5-7B-Instruct	10.2/6.9	51.8	74.4	35.5	24.6	33.9	63.0	27.3	23.8	38.0
+Vanilla GRPO	10.8/6.9	53.3	75.2	36.7	25.1	34.6	63.0	27.0	22.1	37.4
LUFFY	12.2/6.6	54.2	75.1	36.3	25.3	35.0	63.4	29.6	25.4	39.5
BREAD	12.6/7.7	54.2	74.4	35.6	24.6	34.9	63.0	30.0	24.2	39.1
HINT	12.6/8.2	53.3	75.0	**39.2**	25.6	35.7	55.6	34.2	26.0	38.6
+Fixed Strategy	11.6/7.7	53.3	75.1	36.9	25.1	34.9	61.8	33.3	**26.1**	40.3
+PSRA	**12.8/9.8**	**57.6**	**76.5**	37.9	**26.6**	**36.9**	**64.3**	**34.7**	23.2	**40.7**
Qwen2.5-3B-Instruct	6.0/2.5	38.3	65.1	27.3	19.1	26.4	49.1	31.2	**37.4**	39.2
+Vanilla GRPO	**6.7**/3.1	41.0	65.8	27.6	19.2	27.2	50.1	30.7	35.9	38.9
+PSRA	6.5/**3.3**	**42.6**	**67.9**	**28.8**	**19.6**	**28.1**	**59.9**	**33.9**	37.1	**43.6**
Qwen2.5-1.5B-Instruct	3.3/1.5	27.9	52.4	16.9	14.1	19.3	0.1	14.9	8.3	7.8
+Vanilla GRPO	2.7/1.3	30.0	54.4	18.3	14.5	20.2	0.5	15.1	**8.9**	8.2
+PSRA	**3.6/3.3**	**32.3**	**56.2**	**19.5**	**16.2**	**21.9**	**2.1**	**15.9**	7.9	**8.6**
Qwen2.5-7B-Instruct on DeepScaleR-40k										
+Vanilla GRPO	11.2/5.8	52.5	74.7	36.2	25.2	34.3	65.4	**28.2**	25.8	39.8
+Fixed Strategy	9.6/5.6	52.9	75.9	36.3	26.0	34.4	66.6	27.3	26.7	40.2
+PSRA	**12.9/7.2**	**53.6**	**76.7**	**36.9**	**26.5**	**35.6**	**67.5**	27.1	**29.8**	**41.5**

2023), and MMLU-Pro (Wang et al., 2024), which emphasize implicit logical reasoning and broad knowledge-intensive reasoning. In the main results, we report PASS@1 and AVG@8. To measure reasoning coverage under larger inference budgets, we additionally report PASS@k for k up to 128.

Baselines and Ours. We compare against *Vanilla GRPO* (DeepSeek-AI, 2025), *LUFFY* (Yan et al., 2025), *BREAD* (Zhang et al., 2025b), and *HINT* (Wang et al., 2025), representing standard on-policy RLVR and representative guided or off-policy RLVR methods. We additionally consider *Fixed Strategy*, which uses the random 5 strategy anchors to isolate the effect of the inner discovered strategy. All methods are matched to the same total generation budget, equivalent to 22 policy rollouts per training prompt on average. Refer to Appendix A.4 for our training details.

5.2 MAIN RESULTS

Table 1 shows that PSRA consistently improves in-distribution reasoning across training corpora and model scales. On Qwen2.5-7B-Instruct, PSRA outperforms vanilla GRPO by 2.3 points on DAPO-Math-17K and 1.3 points on DeepScaleR-40K, while exceeding the strongest competing baseline by 1.2 points in both settings. The gains persist at smaller scales, with improvements of 0.9 and 1.7 points over vanilla GRPO on the 3B and 1.5B models. The larger gain on 1.5B is particularly notable, since smaller models exhibit substantially higher dead-saturation rates under vanilla GRPO (Figure 4); by switching toward alternative strategies already accessible to the model, PSRA recovers informative training signals on otherwise saturated problems.

PSRA also yields more consistent OOD transfer. While vanilla GRPO shows mixed behavior outside the mathematical training distribution, PSRA improves the OOD average across all settings, outperforming GRPO by 3.3, 4.7, and 0.4 points on the 7B, 3B, and 1.5B models trained on DAPO-Math-17K, and by 1.7 points on the 7B model trained on DeepScaleR-40K. This suggests that problem-conditioned strategy switching promotes a broader and more transferable reasoning repertoire. Moreover, Figure 3d shows that PSRA's advantage generally widens as the inference budget k increases, especially on AIME, indicating that the method improves not only single-sample accuracy but also preserves broader strategy coverage under larger test-time budgets.

Reasoning strategy analysis. We examine whether PSRA changes the strategy selected by the unguided policy. Let $\mathcal{Q}_{\text{fail}}$ denote held-out problems for which all $B{=}16$ base-model rollouts fail, and split them into $\mathcal{Q}^{+}_{\text{fail}}$ and $\mathcal{Q}^{-}_{\text{fail}}$ according to whether each trained policy subsequently solves them. PSRA recovers 26% of $\mathcal{Q}_{\text{fail}}$, compared with 17% for GRPO and 19% for *Fixed Strategy* (Fig. 3a,c). More importantly, its recovered problems are less concentrated on the base-dominant strategy, and 73% are first solved within an alternative strategy footprint, while strategy concentration remains similar across methods on $\mathcal{Q}_{\text{solve}}$. This indicates problem-conditioned switching rather than a uniform increase in diversity. Consistently, the unguided PSRA policy shifts substantially more mass toward admitted alternatives and approaches the base model's guided best-alternative

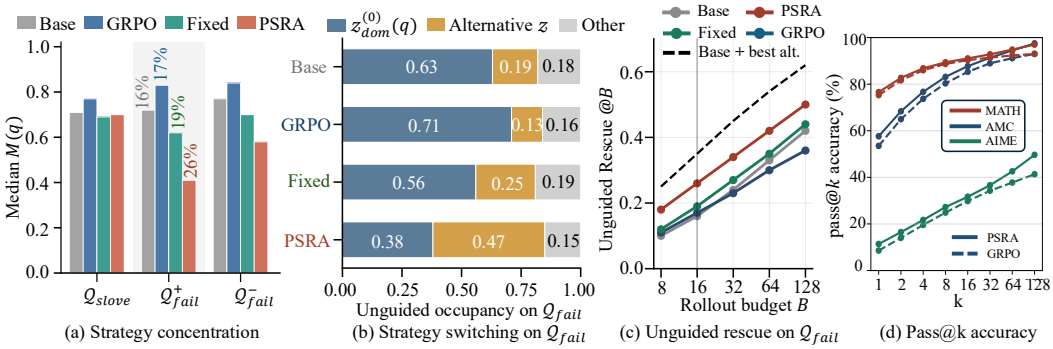
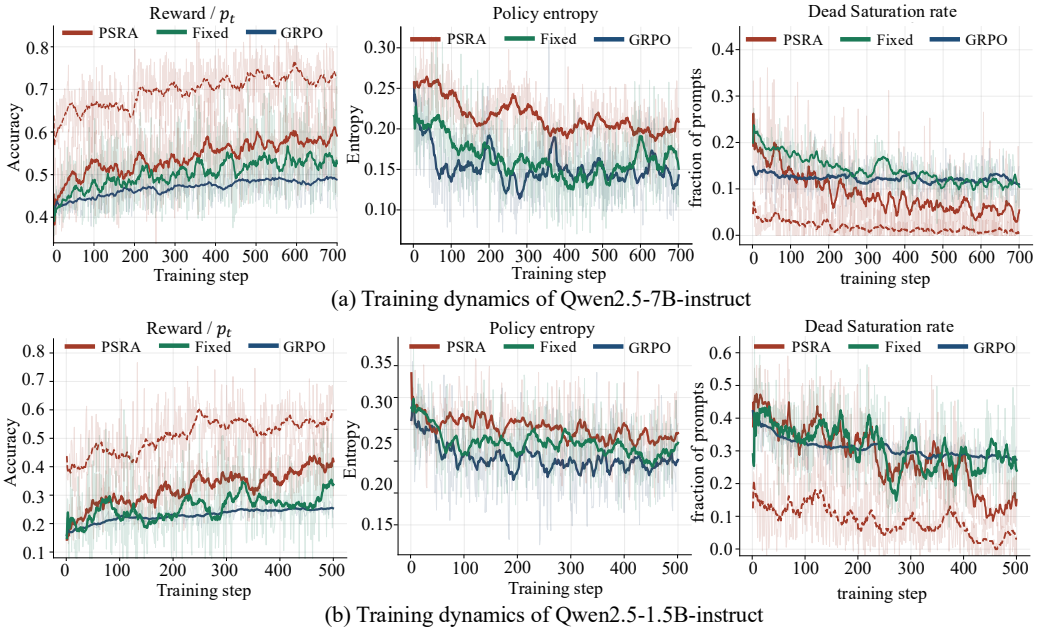


Figure 3: **Strategy-level effects of PSRA.** (a) PSRA successfully reduces strategy concentration. (b) Its unguided policy shifts probability mass from the base-dominant strategy toward alternative strategies. (c) This produces higher rescue rates on initially failed problems across rollout budgets. (d) PSRA maintains a growing advantage over vanilla GRPO at larger inference budgets.



(a) Training dynamics of Qwen2.5-7B-instruct

(b) Training dynamics of Qwen2.5-1.5B-instruct

Figure 4: **Training dynamics on Qwen2.5-7B-Instruct and Qwen2.5-1.5B-Instruct.** Under the same rollout budget, PSRA achieves faster and more stable reward improvement, maintains higher policy entropy, and reduces dead-saturated groups. The dashed curve denotes the best strategy arm, illustrating the remaining strategy-conditioned performance frontier.

rescue rate across budgets (Fig. 3b,c). Finally, Fig. 1g shows that under PSRA, reductions in conditional displacement are largely explained by internalization ι_t, whereas GRPO often loses strategy accessibility before absorption. Together, these results suggest that PSRA successfully internalized strategy-conditioned behaviors into autonomous reasoning.

Training Dynamics. Figure 4 shows that PSRA learns faster and more stably under the same rollout budget, while maintaining higher policy entropy and lower dead-saturation rates than GRPO. The reduction in saturation appears earlier on 7B than on 1.5B, suggesting that smaller models adapt more slowly to problem-conditioned strategy switching, but ultimately benefit from the same mechanism. Meanwhile, the best-strategy arm remains strong as the unguided policy steadily approaches it, consistent with preservation keeping useful routes callable while transfer progressively internalizes them. After accounting for the ≈ 0.20 optimistic noise floor, the remaining gap is small, indicating that much of the strategy-specific gain has been absorbed into autonomous reasoning.

Table 2: Ablation of strategy-set size and Bayesian rollout allocation. B denotes reasoning budget.

	Number of strategies				*Bayes allocation budget*				
Variant	#4	#3	#2	#1	Rand	$B0$	$B5$	$B15$	$B20$
IDP Avg.	$36.3_{\pm.4}$	$36.1_{\pm.3}$	$34.6_{\pm.6}$	$35.3_{\pm.5}$	$35.6_{\pm.5}$	$35.9_{\pm.7}$	$36.3_{\pm.7}$	$36.9_{\pm.5}$	$\mathbf{37.2}_{\pm.6}$
ODP Avg.	$40.3_{\pm.3}$	$40.3_{\pm.4}$	$38.0_{\pm.8}$	$38.1_{\pm.6}$	$38.2_{\pm.6}$	$38.0_{\pm.5}$	$38.8_{\pm.5}$	$40.3_{\pm.9}$	$\mathbf{40.7}_{\pm.6}$

5.3 Ablation Studies

Fixed Guidance. We first replace the discovered strategy set with a fixed set of randomly selected guidance prompts (see Appendix A.3). While such guidance can improve over vanilla GRPO in some settings, it consistently underperforms PSRA (Table 1) and recovers only 19% of initially unsolved problems, compared with 26% for PSRA (Figure 3). Figure 4 further shows higher saturation and weaker training progress, indicating that guidance alone is insufficient: effective exploration depends on steering toward strategy that are both accessible to the model and useful for the problem.

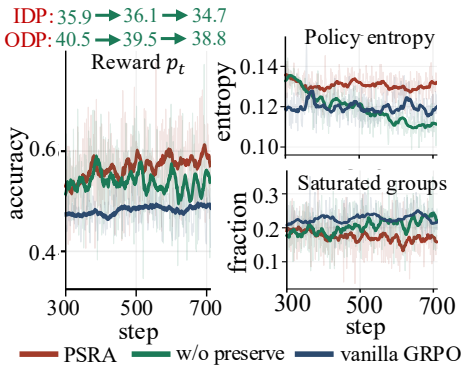


Figure 5: Strategy preservation ablation.

Number of Strategies. Table 2 shows that PSRA is relatively insensitive to moderate reductions in the strategy set: using four or three strategies yields performance close to the full configuration. Performance drops sharply with only two strategies, even falling below the single-strategy variant, suggesting that a small set of competing but insufficiently complementary directions can interfere with effective routing. Overall, the results support using a compact but sufficiently diverse strategy basis rather than maximizing the number of guidance arms.

Bayesian Allocation. Randomly assigning the additional rollouts performs worst among the allocation variants and even underperforms $B{=}0$ on ID performance, despite using more rollouts, showing that poorly targeted compute can introduce harmful training signals. In contrast, Bayesian allocation improves steadily as additional budget is introduced; increasing the budget beyond the default $B{=}10$ yields only marginal further gains while incurring proportionally higher rollout cost.

Strategy Preservation. Figure 5 further isolates the preservation objective. Removing it leads to faster entropy decay, more saturated groups, and weaker final performance, while full PSRA maintains stronger exploration and more stable optimization. This supports preservation as a necessary component for keeping strategy routes accessible to be transferred into autonomous reasoning.

6 Discussion and Limitations

Our experiments span multiple model scales and training corpora, but remain limited to the Qwen2.5 family up to 7B; broader model families and larger models remain important directions for future validation. Nevertheless, the shared strategy families observed across scales suggest that strategy conditioning provides a stable interface for RLVR exploration, while adapting PSRA to a new model requires only lightweight strategy discovery and calibration on a small sample relative to the cost of RL training. For complete examples of how PSRA improves reasoning through strategy switching, please refer to Appendix A.5. PSRA also introduces little online overhead and reaches comparable reward levels in markedly fewer training steps while maintaining stronger exploration (Figure 4).

7 Conclusion

We show that RLVR exploration failures can arise from finite-budget strategy selection rather than missing capability. Motivated by this view, we introduce PSRA, which adaptively allocates rollouts over problem–strategy arms while preserving useful strategy conditions and transferring successful guided behaviors into the unguided policy. Across model scales and training settings, PSRA improves reasoning performance, reduces saturation, and sustains broader exploration, highlighting pretrained strategy repertoires as an effective interface for RLVR exploration.

8 AI USE STATEMENT

We used LLMs to assist with writing and polishing the manuscript and to support retrieval and discovery, such as identifying related work. All AI-assisted content and suggested references were reviewed and verified. We take full responsibility for the final content of this paper, including the accuracy and integrity of its text, claims, and references.

9 REPRODUCIBILITY STATEMENT

To support the reproducibility of our work, we provide a detailed description of the proposed method in the main paper and have submitted the code as supplementary material. We have also documented the experimental setup, including the datasets, evaluation protocols, and implementation details, in the paper and the supplementary material.

REFERENCES

Priyank Agrawal, Ankur Samanta, Shervin Ghasemlou, Boris Vidolov, Jalaj Bhandari, Kavosh Asadi, Daniel Jiang, and Aditya Modi. Off-context grpo: Learning to reason on hard problems using privileged information. 2026. URL https://api.semanticscholar.org/CorpusID:290403118.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, abs/2501.12948, 2025. URL https://api.semanticscholar.org/CorpusID:284488789.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Dao-Guang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models. *ArXiv*, abs/2410.07985, 2024. URL https://api.semanticscholar.org/CorpusID:273233571.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021. doi: 10.1162/tacl_a_00370.

Zhiyuan Hu, Yucheng Wang, Yufei He, Jiaying Wu, Yilun Zhao, See Kiong Ng, Cynthia Breazeal, Luu Anh Tuan, Hae Won Park, and Bryan Hooi. Rewarding the rare: Uniqueness-aware rl for creative problem solving in llms. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 39765–39790, 2026.

Akshay Krishnamurthy, Audrey Huang, and Nived Rajaraman. Select and improve: Understanding the mechanics of post-training for reasoning. *ArXiv*, abs/2606.13125, 2026. URL https://api.semanticscholar.org/CorpusID:289186329.

Chanuk Lee, Minki Kang, and Sung Ju Hwang. Sage: Shaping anchors for guided exploration in rlvr of llms. *ArXiv*, abs/2605.18864, 2026a. URL https://api.semanticscholar.org/CorpusID:288655908.

Chanuk Lee, Sangwoo Park, Minki Kang, and Sung Ju Hwang. Nudging beyond the comfort zone: Efficient strategy-guided exploration for rlvr. *ArXiv*, abs/2605.15726, 2026b. URL https://api.semanticscholar.org/CorpusID:288392794.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

Vaskar Nath, Elaine Lau, Anisha Gunjal, Manasi Sharma, Nikhil Baharte, and Sean M. Hendryx. Adaptive guidance accelerates reinforcement learning of reasoning models. *ArXiv*, abs/2506.13923, 2025. URL https://api.semanticscholar.org/CorpusID:279410993.

Peter Rein, Fabian Balsiger, and et al. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300, 2024. URL https://api.semanticscholar.org/CorpusID:267412607.

Xinyi Wang, Jinyi Han, Zishang Jiang, Tingyun Li, Jiaqing Liang, Sihang Jiang, Zhaoqian Dai, Shuguang Ma, Fei Yu, and Yanghua Xiao. Don't tell the answer, truly guide the reasoning during rl rollouts. *Findings of the Association for Computational Linguistics: ACL 2026*, 2025. URL https://api.semanticscholar.org/CorpusID:282056186.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin. 2025. URL https://api.semanticscholar.org/CorpusID:281658257.

Yifan Xu, Junren Chen, and Yifan Chen. How you begin is how you reason: Driving exploration in rlvr via prefix-tuned priors. *ArXiv*, abs/2605.08817, 2026. URL https://api.semanticscholar.org/CorpusID:288257809.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *ArXiv*, abs/2504.14945, 2025. URL https://api.semanticscholar.org/CorpusID:277955249.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bo-Wen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhen-Ru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024. URL https://api.semanticscholar.org/CorpusID:274859421.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tian-Tian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Honglin Yu, Weinan Dai, Yuxuan Song, Xiang Wei, Haodong Zhou, Jingjing Liu, Wei Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yong-Xu Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *ArXiv*, abs/2503.14476, 2025. URL https://api.semanticscholar.org/CorpusID:277104124.

Yang Yue, Zhiqin Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *ArXiv*, abs/2504.13837, 2025. URL https://api.semanticscholar.org/CorpusID:277940134.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-grpo: Scaffolded group relative policy optimization for enhancing llm reasoning. *ArXiv*, abs/2510.19807, 2025a. URL https://api.semanticscholar.org/CorpusID:282272776.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chen-Shun Ni, Jia-Si Chen, and Samet Oymak. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. *ArXiv*, abs/2506.17211, 2025b. URL https://api.semanticscholar.org/CorpusID:279464619.

A Appendix

A.1 Detailed Strategy Discovery and Problem-Conditioned Exploration

This appendix provides additional details for the strategy discovery, behavioral certification, problem-conditioned rescue analysis, coverage predictor, and longitudinal accessibility analysis introduced in Section 3. Unless otherwise stated, all representation-space analyses use the frozen base-model coordinate system constructed before RL training.

A.1.1 Strategy Repertoire Discovery

Rollout representations. We first construct a repertoire discovery set D_{rep} consisting of problems and multiple autonomous rollouts generated by the frozen base policy. For every rollout y, we re-encode the complete problem–response sequence under the original problem context and extract hidden states from early and intermediate portions of the generated response. Token-level states within each region are pooled and concatenated to obtain a rollout-level representation

$$e_{\mathrm{s}}(y) \in \mathbb{R}^d. \tag{16}$$

All subsequent clustering and longitudinal measurements use this representation extractor frozen at the base checkpoint, preventing changes in the representation space itself from being confounded with changes in policy behavior.

DPMM clustering and semantic merging. We fit a Dirichlet process mixture model (DPMM) to $\{e_{\mathrm{s}}(y) : y \in D_{\mathrm{rep}}\}$. Unlike a fixed-$K$ clustering method, the DPMM allows the number of fine-grained components to be determined from the rollout distribution. Let

$$c(y) \in \{1, \ldots, K_{\mathrm{DPMM}}\} \tag{17}$$

denote the frozen component assignment of rollout y. We inspect representative correct trajectories from stable components and merge components exhibiting the same high-level solution procedure into semantic strategy candidates. This procedure yields seven candidates, five of which survive the behavioral certification described below and form the final strategy set $\mathcal{Z}$.

Importantly, the DPMM components themselves remain the fine-grained coordinate system throughout the paper. Semantic strategies are defined as unions of these components rather than as replacement cluster labels; this allows the same representation space to support both strategy-level analysis and finer-grained coverage estimation.

A.1.2 Behavioral Certification and Strategy Footprints

Representation-space coherence alone does not establish that a cluster corresponds to a controllable reasoning strategy. We therefore behaviorally certify each candidate on a disjoint calibration set D_{cal} using three conditions:

1. **Unguided**: the original problem prompt, denoted by $\varnothing$;
2. **Exemplar-conditioned**: the prompt is augmented with representative trajectories associated with candidate strategy z;
3. **Instruction-conditioned**: the exemplars are replaced by a short natural-language description of the same strategy.

We additionally use matched placebo instructions that preserve prompt length and instructional form without specifying the target reasoning procedure. A candidate is retained only when both exemplar and instruction conditioning consistently increase occupancy of its associated representation region relative to the unguided and placebo controls. This test distinguishes a genuinely steerable reasoning direction from a cluster that is merely descriptively coherent.

Let $\mathcal{K}_z$ be the set of DPMM components semantically associated with strategy z. For each component, we measure the intervention lift

$$\ell_z(k) = \frac{P(c(y) = k \mid z)}{P(c(y) = k \mid \varnothing)}, \qquad k \in \mathcal{K}_z. \tag{18}$$

We rank components in $\mathcal{K}_z$ by $\ell_z(k)$ and define the strategy footprint C_z as the smallest prefix of this ranking whose cumulative probability accounts for at least 50% of the strategy-conditioned mass within $\mathcal{K}_z$. The footprint therefore contains the components most causally associated with invoking z, rather than all components that happen to share its semantic label.

Natural-language instructions retain most of the representation displacement produced by exemplar conditioning, whereas matched placebo instructions produce negligible displacement (Fig. 1c). We consequently use natural-language instructions as the lightweight strategy handles during RL.

A.1.3 Problem-Conditioned Concentration and Rescue

For each problem q, we quantify how strongly autonomous sampling occupies the footprint of strategy z:

$$p_{\varnothing}(z \mid q) = P(c(y) \in C_z \mid q, \varnothing)\,. \tag{19}$$

Because a rollout may fall outside every admitted strategy footprint, we additionally normalize occupancy over the strategy-associated mass,

$$\widetilde{p}_{\varnothing}(z \mid q) = \frac{p_{\varnothing}(z \mid q)}{\sum_{z' \in \mathcal{Z}} p_{\varnothing}(z' \mid q)}, \tag{20}$$

and define the dominant autonomous strategy as

$$z_{\mathrm{dom}}(q) = \arg\max_{z \in \mathcal{Z}} p_{\varnothing}(z \mid q). \tag{21}$$

Rescue protocol. To determine whether an all-failure rollout group necessarily indicates missing capability, we isolate problems

$$\mathcal{Q}_{\mathrm{fail}} = \{q : r(y_i) = 0,\ \forall i \le B_{\mathrm{init}}\}\,, \tag{22}$$

for which the initial autonomous rollout group contains no correct response. Given an additional budget B, we define

$$\mathrm{Rescue@}B(a) = \Pr_{q \in \mathcal{Q}_{\mathrm{fail}}}\left[\exists\, i \le B : r\left(y_i^{(a)}\right) = 1\right], \tag{23}$$

where a denotes the sampling condition.

We compare continued unguided sampling, sampling under the base-dominant strategy $z_{\mathrm{dom}}(q)$, higher-temperature unguided sampling, and an oracle best-alternative strategy. The latter is defined only for analysis as

$$z^*_{\mathrm{alt}}(q) = \arg\max_{z \in \mathcal{Z} \setminus \{z_{\mathrm{dom}}(q)\}} P(r = 1 \mid q, z). \tag{24}$$

Importantly, z^*_{alt} is computed from evaluation rollouts and is *never* available to PSRA during training. It provides an oracle estimate of the headroom available from strategy switching.

Figure 1d reports Rescue@B over increasing budgets. The substantially higher rescue rate achieved by the oracle alternative shows that many all-failure groups arise from finite-budget strategy concentration rather than from a complete absence of successful behavior in the base policy.

A.1.4 Training the Pre-Generation Coverage Predictor

Section 3 introduces a predictor

$$G : h_{\mathrm{plan}}(q) \to \mathrm{PCA}_{128} \to \mathrm{MLP}_{128 \to 64 \to K_{\mathrm{DPMM}}} \to \widehat{q}(c \mid q), \tag{25}$$

which estimates the distribution of reasoning components that may be reached from problem q before any response tokens are generated.

Training data. For every training problem q, we extract the hidden state $h_{\text{plan}}(q)$ at the final prompt position from the frozen base model. We then independently sample multiple continuations

$$Y_q = \{y_{q,1}, \ldots, y_{q,n_q}\}$$

and assign every rollout to its frozen DPMM component $c(y)$. Thus, all trajectories for a given problem share the same predictor input but provide independent samples from the future reasoning distribution induced by that problem.

Problems used to fit G are disjoint from the analysis set D_{eval} used to report Figure 1e. The PCA projection is fitted using training problems only and then frozen together with the MLP for all subsequent evaluations and RL experiments.

Training objective. We train G by maximum likelihood over rollout-level DPMM assignments:

$$\mathcal{L}_G = -\sum_{q \in D_G^{\text{train}}} \frac{1}{|Y_q|} \sum_{y \in Y_q} \log \widehat{q}(c(y) \mid q)\,. \tag{26}$$

Equivalently, this minimizes the cross-entropy between $\widehat{q}(\cdot \mid q)$ and the empirical component-frequency distribution of future rollouts for q. No rollout outcome from the finite observed set used later by PSRA is provided as input to G; the prediction is based only on the pre-generation problem representation.

We select model checkpoints using held-out prediction likelihood. The early-layer baseline used in Figure 1e follows the same PCA–MLP architecture and training objective, differing only in the representation supplied to the predictor. This controls for predictor capacity and isolates the information contained in the planning representation.

Novelty-weighted evaluation. For each held-out problem $q \in D_{\text{eval}}$, we partition independently generated trajectories into a small observed set O_q and a larger held-out set H_q. For each $y \in H_q$, novelty relative to the finite observed group is

$$d_{\min}(y, O_q) = \min_{y' \in O_q} d(e_{\text{s}}(y), e_{\text{s}}(y'))\,, \tag{27}$$

where $d(\cdot,\cdot)$ is the representation-space distance used throughout the analysis. We convert these distances to rank-normalized weights

$$w(y) = \text{ranknorm}\left(d_{\min}(y, O_q)\right). \tag{28}$$

The coverage score is then

$$\text{CovScore}(G) = \frac{\sum_{q \in D_{\text{eval}}} \sum_{y \in H_q} w(y) \log \widehat{q}(c(y) \mid q)}{\sum_{q \in D_{\text{eval}}} \sum_{y \in H_q} w(y)}. \tag{29}$$

Higher-novelty trajectories receive larger weight, so this metric emphasizes the quality of G specifically on reasoning directions that are poorly represented by a finite autonomous sample.

This evaluation differs from ordinary component-prediction accuracy in an important way: a predictor cannot obtain a high score merely by assigning most probability to the dominant reasoning mode. It must also assign probability to plausible but under-sampled regions of the rollout distribution.

A.1.5 From Predicted Coverage to Strategy Opportunities

Given a finite observed rollout set O_q, let

$$\text{hit}(O_q) = \{c(y) : y \in O_q\} \tag{30}$$

be the set of DPMM components already observed. We define the coverage tail

$$U_m(q; O_q) = \text{Top}_m\left\{\widehat{q}(k \mid q) : k \notin \text{hit}(O_q)\right\} \tag{31}$$

as the m highest-probability predicted components that remain unvisited.

The strategy footprints then convert this representation-level prediction into executable exploration directions:

$$\Omega(q, z; O_q) = \sum_{k \in U_m(q; O_q) \cap C_z} \widehat{q}(k \mid q). \tag{32}$$

Thus, $\Omega(q, z; O_q)$ is large only when (i) the model predicts substantial probability mass in currently unobserved regions and (ii) those regions are reachable through strategy z.

Strategy coverage analysis. To evaluate how much of the predicted coverage tail is addressable by the discovered repertoire, we greedily add certified strategies according to the incremental gap mass covered by their footprints. For a strategy subset $\mathcal{S} \subseteq \mathcal{Z}$, define

$$\mathrm{GapCov}(\mathcal{S}) = \frac{\sum_q \sum_{k \in U_m(q;O_q)} \widehat{q}(k \mid q) \mathbb{I}\left[k \in \bigcup_{z \in \mathcal{S}} C_z\right]}{\sum_q \sum_{k \in U_m(q;O_q)} \widehat{q}(k \mid q)}. \tag{33}$$

At each step we add the strategy with the largest marginal increase in GapCov. We compare this curve against matched random subsets of DPMM components. As shown in Figure 1f, coverage rises rapidly and saturates with a small number of certified strategies, showing that the discovered repertoire captures most of the structured coverage opportunities rather than merely partitioning representation space arbitrarily.

A.1.6 Longitudinal Strategy Accessibility

To distinguish successful strategy internalization from loss of steerability, all longitudinal measurements are performed in the frozen base-model coordinate system. In particular, the initial DPMM and strategy footprints $C_z^{(0)}$ are never re-clustered after RL begins.

At checkpoint t, we generate matched unguided and strategy-conditioned rollouts and assign them to the frozen DPMM components. We define the conditional displacement of strategy z as

$$D_t(z) = \mathrm{JS}\left(P_t(c \mid z) \,\|\, P_t(c \mid \varnothing)\right), \tag{34}$$

and normalize its reduction relative to the base checkpoint:

$$\delta_t(z) = 1 - \frac{D_t(z)}{D_0(z)}. \tag{35}$$

A reduction in displacement is ambiguous by itself. It may occur because autonomous generation has moved into the strategy region, which is desirable internalization, or because the strategy instruction has simply stopped changing the model's behavior. We therefore measure autonomous occupancy of the initial footprint,

$$\iota_t(z) = \frac{P_t(C_z^{(0)} \mid \varnothing) - P_0(C_z^{(0)} \mid \varnothing)}{P_0(C_z^{(0)} \mid z) - P_0(C_z^{(0)} \mid \varnothing)}, \tag{36}$$

and define residual accessibility loss as

$$L_t(z) = \delta_t(z) - \iota_t(z). \tag{37}$$

When $\delta_t(z) \approx \iota_t(z)$, the shrinking conditional gap is largely explained by the unguided policy moving into the region previously exposed by z. We interpret this as *internalization*. In contrast, a large positive $L_t(z)$ indicates that strategy-conditioned displacement has disappeared without a corresponding increase in autonomous occupancy, which we interpret as *loss of conditional accessibility*. Figure 1g reports these two dynamics across RL checkpoints.

Connection to PSRA. These measurements play two different roles. The frozen footprints and the coverage predictor provide the representation-space interface used to estimate problem–strategy opportunity during allocation. The accessibility analysis additionally motivates the preservation objective in Section 4: a strategy that remains useful for exploration must stay callable long enough for its successful behaviors to be transferred to the unguided policy. No re-clustering of the reasoning space is performed during RL; only policy-dependent occupancy, success statistics, and strategy accessibility evolve.

A.1.7 Online Estimation of the PSRA Prior

We estimate the hierarchical prior in Eq. (8) using only the all-arm probe. Adaptively allocated rollouts are never used to update the prior statistics, preventing the allocation policy from reinforcing its own previous choices.

Leave-one-arm-out problem effect. For target arm z, let

$$S_{q,-z} = \sum_{a \in \mathcal{A} \setminus \{z\}} s_{q,a}.$$

We estimate the current difficulty of problem q using the probe outcomes of all other arms with Jeffreys smoothing:

$$\hat{c}_t^{-z}(q) = \operatorname{logit}\left(\frac{S_{q,-z} + \frac{1}{2}}{(|\mathcal{A}| - 1)n_0 + 1} \right). \tag{41}$$

Excluding $s_{q,z}$ prevents the outcome used in the posterior update of arm (q, z) from also entering its prior mean.

Global arm effect. Let

$$\ell_t(q, z) = \operatorname{logit}\left(\frac{s_{q,z} + \frac{1}{2}}{n_0 + 1} \right).$$

For each arm, we compute its mean residual success relative to problem difficulty,

$$r_t(z) = \frac{1}{|\mathcal{B}|} \sum_{q \in \mathcal{B}} \left[\ell_t(q, z) - \hat{c}_t^{-z}(q) \right], \tag{42}$$

and update its global effect by an exponential moving average,

$$\tilde{\beta}_{t+1}(z) = (1 - \eta_\beta)\hat{\beta}_t(z) + \eta_\beta r_t(z), \tag{38}$$

$$\hat{\beta}_{t+1}(z) = \tilde{\beta}_{t+1}(z) - \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \tilde{\beta}_{t+1}(a). \tag{43}$$

We initialize $\hat{\beta}_0(z) = 0$. The centering step imposes $\sum_{z \in \mathcal{A}} \hat{\beta}_t(z) = 0$ and separates the global arm effect from the problem intercept. Importantly, Eqs. (42)–(43) use only the n_0 probe rollouts and never the adaptively allocated samples.

Coverage feature. After probing all arms, we form

$$O_{q,t}^{\text{probe}} = \bigcup_{a \in \mathcal{A}} \text{probe}_t(q, a)$$

and compute $\Omega_t(q, z)$ as in Eq. (9). For guided arms, let

$$\tilde{g}_t(q, z) = \log(\Omega_t(q, z) + \epsilon_\Omega).$$

We remove its minibatch-wide offset,

$$\hat{g}_t(q, z) = \tilde{g}_t(q, z) - \frac{1}{|\mathcal{B}||\mathcal{Z}|} \sum_{q' \in \mathcal{B}} \sum_{z' \in \mathcal{Z}} \tilde{g}_t(q', z'), \qquad \hat{g}_t(q, \varnothing) = 0. \tag{44}$$

Centering makes ω_t capture whether relative differences in uncovered, strategy-addressable mass predict arm success rather than absorbing the global success-rate intercept.

Online coverage coefficient. We treat ω_t as a one-dimensional non-negative logistic coefficient. For a candidate value ω, define

$$m_{t,\omega}(q, z) = \sigma\left(\hat{c}_t^{-z}(q) + \hat{\beta}_t(z) + \omega \hat{g}_t(q, z) \right). \tag{45}$$

Using the current probe batch only to update the coefficient for the *next* training step, we perform projected stochastic gradient ascent on the regularized binomial log-likelihood:

$$\omega_{t+1} = \Pi_{[0, \omega_{\max}]} \left[\omega_t + \eta_\omega \left(\frac{1}{|\mathcal{B}||\mathcal{Z}|} \sum_{q \in \mathcal{B}} \sum_{z \in \mathcal{Z}} \hat{g}_t(q, z) \left[s_{q,z} - n_0 m_{t,\omega_t}(q, z) \right] - \lambda_\omega \omega_t \right) \right]. \tag{46}$$

We initialize $\omega_0 = 0$. Hence, when the predicted coverage opportunity carries little information about empirical success, its expected gradient vanishes and the shrinkage term drives ω_t toward zero; the prior then reduces to the problem and global-arm effects. At step t, both $\hat{\beta}_t$ and ω_t therefore depend only on preceding probe batches, while $\hat{c}_t^{-z}(q)$ and $\hat{g}_t(q, z)$ summarize the current all-arm probe.

A.2 Discovered Strategy Prompts

The following five natural-language instructions are used to instantiate the strategy-conditioned arms identified by our discovery and certification procedure. Each prompt specifies a reusable high-level reasoning procedure rather than providing problem-specific solution information.

Stepwise Decomposition.

> *Break the problem into explicit sequential steps or subgoals. Solve each step in order, verifying intermediate results by substitution or logical checks before proceeding. Combine the results to obtain the final answer.*

Equation Setup.

> *Define variables for unknowns, translate the problem into equations or formulas, then solve step-by-step, verifying the solution against the original conditions.*

Formula Application.

> *Identify the relevant formula or theorem, state it explicitly, substitute the given values, and compute step-by-step to obtain the final answer.*

Backward Reasoning.

> *Start from what the final answer must satisfy, work backwards to determine what the earlier quantities must be, then verify the chain forward.*

Counterfactual Reasoning.

> *Consider a nearby alternative to one of the problem conditions and reason through how the solution would change. Use this contrast to identify the quantities, constraints, or relationships that are essential, then return to the original conditions and solve the problem accordingly.*

A.3 Fixed Strategy Baseline

To disentangle the benefit of our discovered strategy repertoire from the generic effect of conditioning rollouts on diverse reasoning instructions, we construct a *Fixed Strategy* baseline using an externally specified, model-agnostic strategy set. Each strategy is represented only by a short natural-language instruction. We do not use demonstrations, few-shot examples, solution prefixes, problem-specific hints, or privileged solution information. The same five instructions are fixed across all problems, model scales, datasets, and training runs.

Intuitive reasoning.

> *Approach the problem by first building an intuitive understanding of the quantities, relationships, or structure involved. Use simple mental models and qualitative expectations to identify a plausible route to the solution, then turn that intuition into the necessary calculations or deductions.*

Planning.

> *Before carrying out detailed calculations, formulate a high-level plan for solving the problem. Identify the main intermediate goals and the order in which they should be addressed, then execute the plan systematically and revise it if necessary.*

Analogical reasoning.

> *Look for a simpler or more familiar problem with the same underlying structure. Reason through the analogous situation, identify which relationships carry over, and map those insights back to the original problem to obtain the solution.*

Socratic reasoning.

> *Guide the solution through a sequence of focused self-questions. At each stage, ask what is known, what must be determined next, and what follows from the current information; answer each question before proceeding to the next deduction.*

Contrastive reasoning.

> *Consider multiple plausible solution paths, interpretations, or candidate answers rather than committing immediately to one. Compare them using the constraints of the problem, eliminate inconsistent alternatives, and continue with the candidate best supported by the evidence.*

For all five strategies, we use the same strategy-conditioning template:

```
{question}

Use the following approach as your primary reasoning
strategy:

{strategy_text}

Apply the approach naturally; do not mention this
instruction.
Put your final answer in \boxed{}.
```

A.4 Training and Implementation Details

Models and parameterization. We evaluate PSRA with Qwen2.5-Instruct models at three scales: 1.5B, 3B, and 7B parameters. All models are trained using LoRA with rank 32, scaling factor $\alpha = 64$, and adapters applied to all linear layers. Model computation uses bfloat16 precision, SDPA attention, and gradient checkpointing. We use AdamW with learning rate 3×10^{-5}, gradient-norm clipping at 1.0, and random seed 42. Unless otherwise specified, training runs for at most 1000 optimization steps.

The model- and dataset-specific configurations are summarized below:

	7B / DeepScaleR	7B / DAPO	3B / DAPO	1.5B / DAPO
Backbone	Qwen2.5-7B	Qwen2.5-7B	Qwen2.5-3B	Qwen2.5-1.5B
Training data	DeepScaleR-40K	DAPO-Math-17K	DAPO-Math-17K	DAPO-Math-17K
Training split	7B pass@8 mixed	7B pass@8 mixed	3B pass@8 mixed	1.5B pass@8 mixed
GPUs	4× GH200	4× GH200	4× GH200	4× GH200
Prompts / rank / step	32	8	8	8
Global prompts / step	128	32	32	32
LoRA rank / α	32 / 64	32 / 64	32 / 64	32 / 64
Learning rate	3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}
Maximum steps	1000	1000	1000	1000

Table 3: Training configurations across model scales and training corpora. All PSRA-specific hyperparameters and per-prompt rollout budgets are shared across settings.

Problem–strategy rollout allocation. The final strategy repertoire contains five certified strategies. Together with the unguided condition, PSRA therefore operates over

$$|\mathcal{Z}| = 5, \qquad |\mathcal{A}| = |\mathcal{Z}| + 1 = 6$$

arms for each problem. Every active arm first receives $n_0 = 2$ probe rollouts, resulting in a fixed probe cost of

$$|\mathcal{A}|n_0 = 12$$

rollouts per prompt. We use prior strength $\kappa = 4$ and update the global arm effect using an EMA coefficient $\eta_\beta = 0.05$.

After probing, PSRA allocates an additional per-prompt-equivalent budget $b_{\text{extra}} = 10$ across problem–arm pairs, subject to a maximum of $n_{\max} = 8$ rollouts per arm. For minibatch $\mathcal{B}$,

$$\sum_{q\in\mathcal{B}}\sum_{z\in\mathcal{A}} k_{q,z} \le |\mathcal{B}|b_{\text{extra}}.$$

The total generation budget is therefore equivalent to

$$|\mathcal{A}|n_0 + b_{\text{extra}} = 6 \times 2 + 10 = 22$$

policy rollouts per training prompt on average. The additional budget is shared across (q, z) pairs rather than assigned independently to each prompt.

Empirical-Bayes allocation prior. For each problem–arm pair, the prior mean is

$$m_t(q, z) = \sigma\Big(\hat{c}_t^{-z}(q) + \hat{\beta}_t(z) + \omega_t\hat{g}_t(q, z)\Big),$$

where $\hat{c}_t^{-z}(q)$ is the leave-one-arm-out problem effect, $\hat{\beta}_t(z)$ is the centered probe-only global arm effect, and $\hat{g}_t(q, z)$ is the centered log coverage feature. The estimators and the online update of the non-negative coverage coefficient ω_t are given in Appendix A.1.7. Adaptively allocated rollouts are not used to update these prior statistics.

Policy optimization. Within each (q, z) arm, advantages are normalized using rollouts from that arm. PPO clipping uses $\epsilon = 0.2$. For strategy-conditioned rollouts, the off-context importance weight is clipped symmetrically in log space with $w_{\max} = 2$, corresponding to

$$w(y) \in [1/2, 2].$$

The main experiments use a fixed preservation coefficient

$$\lambda_t = \lambda_0 = 0.3,$$

and adaptive preservation is disabled.

All model scales use the same KL regularization toward the frozen base policy, with coefficient $\beta_{\text{KL}} = 0.04$. For LoRA training, the reference policy is obtained by disabling the LoRA adapters, and KL is estimated using the k_3 estimator.

Sampling and sequence limits. All training rollouts are sampled with temperature 1.0 and top-$p = 1.0$. We use a maximum prompt length of 1024 tokens and a maximum completion length of 2048, giving a maximum sequence length of 3072. No explicit length shaping is used ($\alpha_{\text{len}} = 0$). Responses terminated by the maximum completion length receive a truncation penalty of 0.2 in the policy-training score. The Bayesian allocator itself uses binary answer correctness when computing the success counts $s_{q,z}$.

Distributed training and rollout budgets. All experiments use four GH200 GPUs with data parallelism, tensor-parallel size 1, and colocated vLLM rollout generation. Each rank independently samples its local prompt minibatch, while gradients are averaged across all four ranks.

For the 7B DeepScaleR-40K setting, we use 32 prompts per rank, giving a global minibatch of 128 prompts. Under the average budget of 22 rollouts per prompt, this corresponds to

$$128 \times 22 = 2816$$

rollout slots per optimization step, or 704 per rank.

For the 7B DAPO-Math-17K setting and the 3B and 1.5B scale-study runs, we use a split-batch configuration with 8 prompts per rank and 32 prompts globally. The corresponding generation budget is

$$32 \times 22 = 704$$

rollout slots per optimization step, or 176 per rank. Thus, all settings share the same per-prompt exploration budget and PSRA hyperparameters, while the DeepScaleR 7B run uses a larger global prompt batch.

The sufficient statistics used for the global arm effect and online coverage coefficient are synchronized across data-parallel workers before updating the shared allocator state.

Baseline reproduction and compute matching. For Vanilla GRPO, we use the same backbone, training data, LoRA parameterization, optimizer, learning rate, sampling configuration, sequence limits, KL regularization, global prompt batch, and number of optimization steps as the corresponding PSRA run. The only differences are therefore the PSRA-specific strategy conditioning, rollout allocation, transfer correction, and preservation objective. In particular, GRPO is matched to the same total generation budget, equivalent to 22 policy rollouts per training prompt on average.

For the remaining baselines, we reproduce each method using the method-specific hyperparameters and optimization settings reported in its public implementation or paper whenever available. To ensure a compute-matched comparison, we adjust the total rollout budget and the number of optimizer updates to match the corresponding PSRA setting, while keeping the backbone, training corpus, evaluation protocol, and inference budget identical. Because guided-RL baselines can be sensitive to implementation-specific hyperparameters and exhibit substantial run-to-run training variation, we perform three tuning runs for each reproduced baseline and report the configuration selected by the best held-out validation performance.

Evaluation setup. We evaluate all models with the same decoding and answer-verification pipeline. Each problem is formatted as a user-only chat message with the instruction `Put your final answer in \boxed{}`, and generation uses temperature 0.6 and top-$p = 0.95$. Mathematical answers are extracted only from the final `\boxed{}` expression and verified with Math-Verify against normalized gold LaTeX answers; no bare-expression fallback is used. All evaluations use deterministic question- and sample-specific random seeds.

For the main evaluation, we draw 16 independent responses per problem and estimate Pass@1 by the mean single-sample correctness. Pass@k is computed using the standard unbiased estimator. Generation is capped at 4096 new tokens for MATH-500, AMC 2023, Minerva Math, StrategyQA, GPQA-Diamond, and MMLU-Pro, and at 8192 tokens for AIME and Omni-MATH. For the inference-budget analysis, we additionally sample 128 responses per problem on AIME 2024/2025, AMC 2023, and MATH-500, and report $k \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. All methods use identical prompts, decoding parameters, evaluation subsets, generation limits, and scoring code.

A.5 Qualitative Examples of Strategy Switching

We provide two representative examples in which the default reasoning route fails while an alternative strategy yields a correct solution. For each problem, we first show the original rollouts and then summarize their strategy-level reasoning structure using the five strategy families identified in Section 3.

A.5.1 Example 1: Formula Application to Stepwise Decomposition

Problem. Paul and Jesse each choose a number at random from the first six primes. What is the probability that the sum of the numbers they choose is even?

Failed rollout: `formula_application`.

> To determine the probability that the sum of the numbers chosen by Paul and Jesse is even, we first need to identify the first six prime numbers. These are: $2, 3, 5, 7, 11$, and 13.
>
> The sum of two numbers is even if both numbers are either even or both are odd. Among the first six prime numbers, only 2 is even, and the rest $(3, 5, 7, 11, 13)$ are odd. Therefore, the only way to get an even sum is if both Paul and Jesse choose the number 2. The probability that Paul chooses 2 is $\frac{1}{6}$. The probability that Jesse chooses 2 is $\frac{1}{6}$.
>
> Since the choices are independent, the probability that both Paul and Jesse choose 2 is
>
> $$\frac{1}{6} \times \frac{1}{6} = \frac{1}{36}.$$

Thus, the probability that the sum of the numbers they choose is even is

$$\boxed{\frac{1}{36}}.$$

Successful alternative rollout: `stepwise_decomposition`.

To determine the probability that the sum of the numbers chosen by Paul and Jesse is even, we first need to identify the first six prime numbers. These are: $2, 3, 5, 7, 11$, and 13.

The sum of two numbers is even if both numbers are either even or both are odd. Among the first six prime numbers, only one number is even (2), and the rest are odd ($3, 5, 7, 11, 13$).

Let's calculate the total number of possible outcomes. Since each person chooses a number independently from the six primes, there are

$$6 \times 6 = 36$$

possible outcomes.

Next, we count the favorable outcomes where the sum is even. This can happen in two scenarios:

1. Both numbers chosen are even.
2. Both numbers chosen are odd.

For the first scenario, the only even number is 2. Hence there is only one favorable outcome.

For the second scenario, Paul and Jesse can each choose from the five odd numbers, giving

$$5 \times 5 = 25$$

favorable outcomes.

Thus, the total number of favorable outcomes is

$$1 + 25 = 26,$$

and the probability is

$$\frac{26}{36} = \frac{13}{18}.$$

Therefore,

$$\boxed{\frac{13}{18}}.$$

Strategy-level structure.

Failed: identify parity rule $\rightarrow$ focus on the unique even prime $\rightarrow P(2, 2) \rightarrow \frac{1}{36}$,

Successful: decompose the target event $\rightarrow$ {even–even, odd–odd} $\rightarrow 1 + 25 \rightarrow \frac{13}{18}$.

The failed route applies the parity condition directly but prematurely collapses the event to the even–even case. The alternative strategy explicitly decomposes the target event into exhaustive subcases before combining their contributions.

A.5.2 Example 2: Equation Setup to Formula Application

Problem. In triangle ABC, $AB = AC = 5$ and $BC = 6$. Let O be the circumcenter of triangle ABC. Find the area of triangle OBC. The answer is in the form $\frac{m}{n}$, where $\gcd(m, n) = 1$; report $m + n$.

Failed rollout: `equation_setup`.

To find the area of triangle OBC, we start by determining the coordinates of the vertices of triangle ABC and the circumcenter O.

First, place B at $(-3, 0)$ and C at $(3, 0)$. Since $AB = AC = 5$, the coordinates of A are $(0, 4)$.

Next, we find the circumcenter O. The circumcenter is the intersection of the perpendicular bisectors of the sides of the triangle. The perpendicular bisector of BC is the vertical line

$$x = 0.$$

The midpoint of AB is $(-1.5, 2)$ and the slope of AB is $-\frac{4}{3}$, so the slope of its perpendicular bisector is $\frac{3}{4}$. Its equation is

$$y - 2 = \frac{3}{4}(x + 1.5)$$

and hence

$$y = \frac{3}{4}x + \frac{15}{8} + 2 = \frac{3}{4}x + \frac{23}{8}.$$

Setting $x = 0$, we obtain

$$y = \frac{23}{8},$$

so

$$O = \left(0, \frac{23}{8}\right).$$

The base BC is 6, and the height from O to BC is $\frac{23}{8}$. Therefore,

$$[OBC] = \frac{1}{2} \cdot 6 \cdot \frac{23}{8} = \frac{69}{8}.$$

Thus,

$$m + n = 69 + 8 = \boxed{77}.$$

Successful alternative rollout: `formula_application`.

To find the area of triangle OBC, we start by determining the circumradius R of triangle ABC. Since $AB = AC = 5$ and $BC = 6$, triangle ABC is isosceles.
Let D be the midpoint of BC. Then

$$BD = DC = 3.$$

In right triangle ABD,

$$AD^2 + BD^2 = AB^2,$$

so

$$AD^2 + 3^2 = 5^2 \quad \Longrightarrow \quad AD = 4.$$

Therefore, the area of triangle ABC is

$$K = \frac{1}{2} \cdot 6 \cdot 4 = 12.$$

Using the circumradius formula

$$R = \frac{abc}{4K},$$

we obtain

$$R = \frac{6 \cdot 5 \cdot 5}{4 \cdot 12} = \frac{25}{8}.$$

Since O is the circumcenter, $OB = R$. In right triangle OBD,

$$OB^2 = OD^2 + BD^2,$$

so

$$\left(\frac{25}{8}\right)^2 = OD^2 + 3^2.$$

Hence

$$OD^2 = \frac{625}{64} - \frac{576}{64} = \frac{49}{64}, \qquad OD = \frac{7}{8}.$$

Finally,

$$[OBC] = \frac{1}{2} \cdot BC \cdot OD = \frac{1}{2} \cdot 6 \cdot \frac{7}{8} = \frac{21}{8}.$$

Thus,

$$m + n = 21 + 8 = \boxed{29}.$$

Strategy-level structure.

Failed: choose coordinates $\rightarrow$ construct perpendicular bisectors $\rightarrow$ solve for $O \rightarrow [OBC]$,

Successful: $AD \rightarrow [ABC] \rightarrow R = \frac{abc}{4K} \rightarrow OD = \sqrt{R^2 - BD^2} \rightarrow [OBC]$.

Here the unsuccessful route translates the geometry into a coordinate system and explicitly solves for the circumcenter, whereas the alternative route avoids recovering O altogether and instead composes standard geometric formulas for the circumradius and the distance from the circumcenter to the chord.

Together, the two examples illustrate that a failed rollout need not indicate the absence of a viable solution procedure. Switching the strategy changes how the same problem is organized—from direct formula use to explicit case decomposition in the first example, and from coordinate-based equation setup to theorem-driven formula application in the second—exposing successful trajectories that remain accessible to the same model.