

Beyond Reweighting: Can RLVR Discover?

Research Exploration

Rohan Sikand · Stanford University

March 2026

The Uncomfortable Truth About RLVR

RLVR (GRPO, DeepSeek-R1) is the dominant paradigm for reasoning post-training.

But what does it actually *do* to the policy?

Claim: RLVR does two things and only two things:

1. **Sharpens** — concentrates probability mass on already-sampled correct solutions
2. **Reweights** — upweights solutions the verifier likes, downweights those it doesn't

It does **not**:

3. Discover genuinely new solution strategies the model couldn't already sample
4. Explore outside the support of the initial policy's generation distribution

RLVR optimizes Pass@1 by collapsing Pass@k — exploitation, not exploration.

Pass@k Sharpening: What's Actually Happening

Consider a model with $\text{Pass@64} = 80\%$ on a problem. Among 64 samples, several distinct correct approaches exist.

Before RLVR: Diverse solution distribution. Multiple strategies sampled.

After RLVR: Policy concentrates on 1–2 strategies. Pass@1 goes up. Pass@64 may go *down*. Diversity collapses.

What we measure:

- $\text{Pass@1} \uparrow$ (the metric we optimize) -
- Solution diversity $\downarrow$ (never measured) -
- Novel strategies found: 0 (never measured)

What this means:

- RLVR doesn't make the model *smarter* -
- It makes the model *more decisive* - It narrows, not broadens - Gains are bounded by pre-training coverage

The Reweighting Ceiling

GRPO computes advantages: $\hat{A}_i = (r_i - \mu)/\sigma$

This can only move probability mass **between existing samples**. It cannot:

- Assign mass to solutions the policy has never generated
- Escape the support of $\pi_\theta(\cdot | s)$
- Bridge across modes the initial distribution doesn't connect

Consequence: If the correct solution requires an approach the model has never sampled (probability ≈ 0 under π_θ), no amount of RLVR training will find it.

This is a **fundamental limitation of policy gradients on discrete sequences**. The gradient can sharpen peaks but cannot create new ones. The model's pre-training determines the ceiling; RLVR just pushes you toward it faster.

Three Regimes of RLVR Behavior

Regime	Condition	RLVR behavior	Failure mode
Easy	High Pass@k	Sharpens quickly	Diversity collapse
Medium	Moderate Pass@k	Slow sharpening	Verifier overfitting
Hard	Near-zero Pass@k	Stalls completely	Zero gradient signal

The **hard regime** is where the real problem lives. If the model can't sample a correct solution in G rollouts, GRPO gets zero signal. No reweighting can help because there's nothing correct to reweight *toward*.

Dense reward (EGSR) partially helps — it provides gradient within failure space. But it still can't inject new solution strategies into the sampling distribution. You climb the hill more efficiently, but if the correct solution is on a different hill entirely, you're stuck.

Can we make RLVR *discover* rather than just *reweight*?

Three sub-questions:

1. **Measurement:** How do we even know if a training method discovers vs. reweights? What metrics distinguish them?
2. **Mechanism:** What algorithmic modifications to RLVR could enable genuine exploration in solution space?
3. **Limits:** Is there a theoretical boundary between what reweighting can achieve and what requires discovery?

Part 1: Measuring Discovery

We need metrics that distinguish “the model found a strategy it already knew” from “the model found a genuinely new strategy.”

Metric 1: Solution family diversity

- Cluster correct solutions by approach (embedding similarity, AST structure for code, proof structure for math)
- Track number of distinct clusters over training
- **Reweight**ing: cluster count decreases (mode collapse)
- **Discovery**: cluster count stays stable or increases

Measuring Discovery (continued)

Metric 2: Novel solution rate

- At step N , sample 1K correct solutions
- Compute minimum edit distance to any correct solution sampled at step 0
- Novel solution = correct and far from all step-0 correct solutions
- $\text{NovelRate}(N) = \frac{|\{a: R(a)=1 \wedge d(a, \mathcal{C}_0) > \tau\}|}{|\{a: R(a)=1\}|}$

Metric 3: Coverage

- Use a strong model (Opus) to enumerate K distinct correct approaches for each problem
- Track what fraction the policy can produce at each checkpoint
- $\text{Coverage}(N) = \frac{|\text{approaches found by step } N|}{K}$
- **Reweighting:** coverage flat or decreasing
- **Discovery:** coverage increasing

Measuring Discovery (continued)

Metric 4: Capability frontier

- Partition problems by difficulty
- Track the hardest problem tier the model can solve
- Reweighting improves Pass@1 on problems the model could already solve sometimes
- Discovery extends to problems the model *never* solved pre-RLVR

Metric 5: Cross-entropy gap

- Compute $H(\pi_N, \pi_0)$ — cross-entropy of step- N policy under step-0 policy
- High cross-entropy = the model assigns mass where it previously didn't
- Reweighting: moderate cross-entropy (reshuffling existing mass)
- Discovery: high cross-entropy (mass in previously-zero regions)

First experiment: Apply all five metrics to standard RLVR training. If the diagnosis is

Part 2: Mechanisms for Discovery

Five candidate methods, ordered by how fundamentally they depart from standard RLVR:

Method	Core idea	Departure from RLVR
Novelty bonus	Reward solutions far from current mode	Reward modification
Hindsight relabeling	Strong model "fixes" failures $\rightarrow$ SFT	Data injection
Latent perturbation	Noise in activations, not tokens	Generation modification
Population training	K independent policies, cross-pollinate	Architecture change
Backtracking search	Tree search during generation	Inference modification

Method 1: Novelty-Rewarded RLVR

Idea: Add an intrinsic reward for solutions that are distant from the current policy's mode.

$$R_{\text{total}}(s, a) = R_{\text{env}}(s, a) + \alpha \cdot d(a, \hat{\mu}_{\theta}(s))$$

where $\hat{\mu}_{\theta}(s)$ is the centroid of recent correct solutions (in embedding space) and d measures deviation.

Pros:

- Minimal change to RLVR — just an additive reward term
- Directly counteracts mode collapse
- Compatible with GRPO

Method 2: Hindsight Relabeling

Idea: When the policy fails on a hard problem, use a strong model to generate a correct solution *using a different approach than the policy attempted*. SFT on these, then continue RLVR.

RL step → collect failures → strong model fixes with novel approach → SFT injection → RL

Pros:

- Actually injects new solution families into the sampling distribution
- Targeted: only fires on problems where the policy is stuck
- The strong model can be prompted to use a *different* approach

Cons:

Method 3: Latent-Space Perturbation

Idea: Exploration in token space is discrete and high-dimensional. Exploration in activation space is continuous and lower-dimensional.

Add structured noise to hidden states at generation time:

$$h'_t = h_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2 I)$$

Use RL to optimize which perturbations lead to novel correct solutions. Anneal σ over training.

Pros:

- Can reach solutions with near-zero sampling probability under π_θ
- Smooth perturbation $\rightarrow$ smooth exploration (vs. discrete token jumps)

Method 4: Population-Based RLVR

Idea: Maintain K copies of the policy, each trained independently on different problem subsets. Periodically cross-pollinate by distilling diverse solutions.

$$\{\pi_{\theta_1}, \pi_{\theta_2}, \dots, \pi_{\theta_K}\} \xrightarrow{\text{independent RLVR}} \xrightarrow{\text{cross-distillation}} \pi_{\theta_{\text{merged}}}$$

Pros:

- Different random seeds $\rightarrow$ different modes found $\rightarrow$ genuine diversity
- Cross-pollination injects each copy's discoveries into the others
- Theoretically well-motivated (population-based training, evolutionary strategies)

Cons:

- $K \times$ compute cost

Method 5: Backtracking Search During Generation

Idea: Instead of sampling complete rollouts, do tree search during generation. When the model hits a dead end (detected by a value head or the self-judge), backtrack and try a different continuation.

Related to: MCTS for LLMs, tree-of-thought, but integrated into RL training.

Pros:

- Can discover solutions that are correct compositions of good sub-steps, even if the full sequence was never sampled
- Search is the classical answer to exploration in discrete spaces
- Self-judge (EGSR) provides the value signal for backtracking

Cons:

Which Method Attacks Which Problem?

	Novelty	Hindsight	Latent	Population	Search
Prevents mode collapse	✓			✓	
Injects new strategies		✓	?	✓	✓
Reaches zero-prob regions		✓	✓		✓
Self-contained (no oracle)	✓		✓	✓	✓
Low overhead	✓				

My ranking for tractability $\times$ impact:

1. **Hindsight relabeling** — most likely to work, clearest mechanism
2. **Novelty bonus** — easiest to implement, modest expected impact
3. **Population-based** — strong theoretical motivation, expensive
4. **Latent perturbation** — most novel, highest uncertainty
5. **Backtracking search** — most powerful but hardest engineering

Proposed Paper Structure

Part 1: The Diagnosis (confirm the problem exists)

- Train standard GRPO on code/math
- Apply all five discovery metrics at checkpoints
- Show: diversity collapses, novel rate is near zero, coverage is flat
- *“RLVR is pure reweighting”*

Part 2: Interventions

- Implement 2–3 methods (novelty bonus, hindsight relabeling, one wild card)
- Apply same metrics: does diversity increase? Does coverage grow? Does the capability frontier extend?

Part 3: Analysis

- Reweighting vs. discovery decomposition: how much of Pass@1 improvement comes

The Theoretical Question

Can we formalize the boundary between what reweighting can achieve and what requires discovery?

Sketch: Define the *reweighting ceiling* for a policy π_0 on a problem distribution:

$$\text{Ceiling}(\pi_0) = \sup_{\pi \in \Delta(\pi_0)} \mathbb{E}_s[\text{Pass@1}_\pi(s)]$$

where $\Delta(\pi_0) = \{\pi : \text{supp}(\pi) \subseteq \text{supp}(\pi_0)\}$ — the set of all policies supported on the same sequences as π_0 .

Any policy gradient method (GRPO, PPO, REINFORCE) converges to some $\pi^* \in \Delta(\pi_0)$.

$\text{Ceiling}(\pi_0)$ is the best you can do without discovery. The gap

Connection to EGSR

EGSR occupies a specific position in this landscape:

- EGSR makes reweighting **more efficient** — dense reward provides better gradient signal, so you reach the reweighting ceiling faster and with less compute
- EGSR provides **soft exploration within failure space** — failure-mode gradients point toward near-correct solutions. But it's still within the support
- EGSR's self-judge could serve as the **backtracking signal** in Method 5 — the teacher evaluates partial solutions and decides when to abandon a strategy. Natural synergy.
- EGSR's feedback prediction ($\mathcal{L}_{\text{pred}}$) builds an internal execution model that could **guide latent perturbation** in Method 3 — perturb in directions the execution model predicts will change the outcome.

Experimental Plan

Phase 1: Diagnosis (3 weeks)

- Train GRPO (Qwen3-4B) on LeetCode, 400 steps
- Compute all five metrics at checkpoints every 50 steps
- Confirm: pure reweighting signature (diversity $\downarrow$, novel rate ≈ 0 , coverage flat)

Phase 2: Interventions (5 weeks)

- Novelty-rewarded RLVR (α sweep)
- Hindsight relabeling (Opus as oracle, interleaved SFT+RL)
- Latent perturbation (σ and layer sweep)
- Same metrics, same checkpoints

Phase 3: Analysis (3 weeks)

- Reweighting ceiling estimation for each problem

Risks

Risk	Mitigation
RLVR already discovers (diagnosis fails)	Measure carefully; if true, that's also a result
Hindsight relabeling = just SFT	Compare SFT-only vs. SFT+RL to isolate RL contribution
Novelty bonus destroys performance	α sweep + annealing schedule
No method beats the ceiling	Proving the ceiling is tight is itself a contribution
Metrics don't distinguish reweighting vs. discovery	Multiple complementary metrics reduce risk
Compute budget for population training	Start with $K=4$, smallest model (4B)

Why This Matters

If RLVR is fundamentally reweighting:

- The community is optimizing a metric (Pass@1) that masks a ceiling
- Scaling RLVR compute has diminishing returns after the reweighting ceiling
- We need different methods for the hard regime (problems the model can't already solve)

If we can enable discovery:

- RLVR becomes a genuine learning algorithm, not just a ranking algorithm
- The model can solve problems it couldn't solve before training, not just solve faster
- This is the difference between the model got more confident'' and the model got smarter''

The question isn't "can RLVR improve Pass@1?" — of course it can.

Summary

Thesis: RLVR fundamentally reweights existing samples rather than discovering new solution strategies. We lack the metrics to detect this and the methods to fix it.

Contributions (proposed):

1. **Diagnosis:** Five metrics that distinguish reweighting from discovery, applied to standard RLVR training
2. **Theory:** Formal definition of the reweighting ceiling and discovery gap
3. **Methods:** 2–3 interventions that push beyond the ceiling (hindsight relabeling, novelty bonus, latent perturbation)
4. **Analysis:** Decomposition of Pass@1 gains into sharpening vs. discovery components

Relationship to EGSR: EGSR is the best reweighting can do (efficient, dense, grounded). This paper asks what lies beyond the reweighting ceiling.