From AI Assistant to AI Scientist: Autonomous Discovery of LLM-RL Algorithms with LLM Agents

Anonymous ACL submission

Abstract

Discovering improved policy optimization algorithms for language models remains a costly manual process requiring repeated mechanism-level modification and validation. Unlike simple combinatorial code search, this problem requires searching over algorithmic mechanisms tightly coupled with training dynamics while reusing empirical evidence across iterations. We propose POISE, a closed-loop framework for automated discovery of policy optimization algorithms for language models. POISE maintains an archive that links each candidate design with its implementation, evaluation evidence, and analysis summary, enabling later iterations to build on both successful and failed designs. In mathematical reasoning experiments starting from GRPO, POISE evaluates 64 candidate algorithms and discovers improved mechanisms, including analytic-variance scaling and validity masking. The best variant improves weighted Overall from 47.8 to 52.5 (+4.6) and increases AIME25 `pass@32` from 26.7% to 43.3%, demonstrating the feasibility of automated policy optimization discovery while supporting interpretable design principles.

1 Introduction

Reinforcement Learning (RL) has become an important post-training paradigm for Large Language Models (LLMs) (Ouyang et al., 2022; Shao et al., 2024). However, discovering improved policy optimization algorithms for language models remains a costly manual process. Designing an effective policy optimization algorithm involves more than hyperparameter tuning: researchers repeatedly modify and validate core mechanisms, including loss design, advantage estimation, and regularization. Each new hypothesis requires time-consuming training and standardized evaluation, demanding substantial computational resources. LLMs have shown advanced reasoning and planning capabilities (Yang et al., 2025; Liu et al., 2025a; Erdogan et al.), and recent studies have begun developing LLM-driven systems for automated research (Lu et al., 2024; Schmidgall et al., 2025; Schmidgall and Moor, 2025). This raises a question: **can LLMs discover new policy optimization algorithms for language models?**

Although automated research has made remarkable progress in tasks such as machine-learning research automation (Liu et al., 2025c), program optimization (Fawzi and Paredes, 2023), and model architecture discovery (Liu et al., 2025b), discovering new policy optimization algorithms for language models still presents unique challenges. Unlike tasks framed as combinatorial searches over discrete program components, discovering policy optimization algorithms for language models requires searching over mechanisms tightly coupled with training dynamics. For example, small design choices, such as introducing a clipping objective as in PPO (Schulman et al., 2017), adding a KL penalty, or modifying advantage normalization, can substantially alter optimization behavior and training stability. Moreover, discovering a new algorithm requires more than generating a candidate design: it also requires identifying which mechanisms contribute to performance gains, which fail under empirical evaluation, and how accumulated evidence should inform the next testable hypothesis. This calls for a search process that converts empirical feedback into reusable evidence and carries that evidence across iterations.

To address this challenge, we propose POISE (**P**olicy **O**ptimization through **I**terative **S**earch and **E**vidence), a closed-loop framework for the automated discovery of policy optimization algorithms for language models. POISE organizes discovery as an iterative search process in which candidate designs are proposed, implemented, trained, evaluated, analyzed, and stored for later use. The framework maintains an archive of evaluated candidates that links each design with its implementa-

tion, evaluation evidence, analysis summary, and derivation history, allowing later proposals to build on accumulated evidence instead of isolated trial-and-error. In mathematical reasoning experiments starting from a GRPO baseline (Shao et al., 2024), POISE evaluates 64 algorithms and discovers improved policy optimization mechanisms, including analytic-variance scaling and validity masking. The best variant improves weighted Overall from 47.8 to 52.5 (+4.6) and increases AIME25 `pass@32` from 26.7% to 43.3%, while the resulting lineages yield interpretable design principles.

Our main contributions are as follows:

- **Closed-Loop Discovery Framework:** We introduce POISE for automated discovery of policy optimization algorithms for language models. An evidence-preserving archive connects proposal generation, evaluation, result analysis, and archive update, allowing later iterations to reuse accumulated empirical evidence.
- **Empirical Validation of Automated Algorithm Discovery:** Starting from GRPO, POISE evaluates 64 candidate algorithms and discovers mechanisms that yield substantial gains on mathematical reasoning benchmarks. These results demonstrate the feasibility of automated policy optimization discovery for language models.
- **Interpretable Design Principles:** Beyond discovering stronger algorithms, our analysis extracts design principles from the resulting lineages, including signal decoupling, conditional normalization, and correctness-first efficiency shaping. These principles clarify how and when such mechanisms help in sparse-reward language model policy optimization.

2 Related Work

LLM-driven Scientific Discovery LLM-driven scientific discovery leverages the knowledge, reasoning, and execution capabilities of Large Language Models (LLMs) to automate parts of the research process. This paradigm has evolved from experimental automation (Boiko et al., 2023) to algorithm design (Fawzi and Paredes, 2023; Shi et al., 2025). More recently, "AI Scientist" agents have emerged, aiming for end-to-end pipelines that encompass literature review, coding, and experimentation (Lu et al., 2024; Yamada et al., 2025; Schmidgall et al., 2025; Gottweis et al., 2025; Mitchener et al., 2025). Progress has also been made in quality control for idea generation (Si et al.; Li et al., 2024; Keya et al., 2025). LLMs have further demonstrated effective exploration within well-defined search spaces, including program optimization (Novikov et al., 2025), theorem proving (Trinh et al., 2024; Chervonyi et al., 2025), architecture search (Liu et al., 2025b; Cheng et al., 2025), automated machine learning engineering (Chan et al.; Jiang et al., 2025; Liu et al., 2025c), and automated AI research systems (Xu et al., 2026). However, extending automated discovery to RL algorithm design for language model policy optimization is more challenging because the search space is compositional, feedback is delayed and stochastic, and verification requires expensive distributed training runs. To address this setting, POISE organizes automated algorithm discovery as an iterative search process that proposes, implements, evaluates, analyzes, and refines policy-optimization algorithms using accumulated empirical evidence.

Policy Optimization for Large Language Models Policy optimization for Large Language Models aims to enhance reasoning capabilities by training LLMs with algorithms such as PPO (Schulman et al., 2017) and its computationally efficient variant GRPO (Shao et al., 2024). Recent studies have improved these baselines by refining specific algorithmic components, such as decoupled clipping in DAPO (Yu et al., 2025), soft adaptive gating in SAPO (Gao et al., 2025), asymmetric weighting in ASPO (Wang et al., 2025), and geometric mean aggregation in GMPO (Zhao et al., 2025). While these studies demonstrate the value of targeted component-level refinements, exploring the broader design space of such mechanisms still largely relies on manual iteration. Unlike prior work, which manually refines individual components, POISE automatically searches for and composes policy-optimization mechanisms under a fixed training and evaluation protocol.

3 Method

3.1 Problem Modeling

We formulate automated discovery of policy-optimization algorithms for language models as a search problem under a fixed training and evaluation setting. Across candidates, we keep the base model, training data, evaluation procedure, and main training configuration fixed, while modifying mechanisms inside the training loop, such as advantage estimation, reward or validity masking, loss weighting, and stabilization terms.

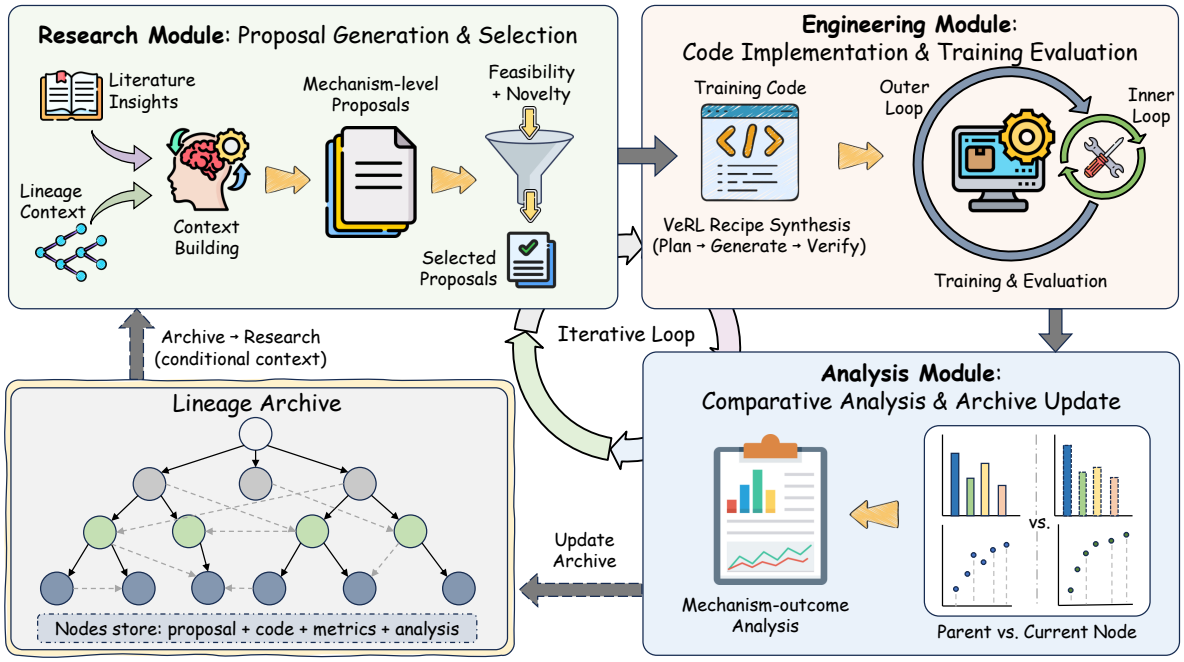

Figure 1: Overview of POISE. Phase I selects parent nodes and generates candidate proposals; Phase II implements selected proposals, trains models, and evaluates them under a fixed protocol; Phase III analyzes each candidate outcome and updates the candidate archive.

Let $\mathcal{Z}$ denote the resulting search space. Each candidate $z \in \mathcal{Z}$ specifies a complete policy-optimization variant, including the modified mechanism and its mechanism-specific hyperparameters. For any candidate z, the standardized training and evaluation procedure returns a training trajectory τ and evaluation scores $\mathbf{y} \in \mathbb{R}^K$:

$$(\tau, \mathbf{y}) = \mathcal{O}(z).$$

Here, τ records scalar training signals, such as reward, KL, entropy, and response length, while $\mathbf{y}$ contains the standardized metrics from the evaluation suite.

The search aims to find candidates with high overall evaluation scores:

$$z^* = \arg\max_{z \in \mathcal{Z}} \; U\big(\mathbf{y}(z)\big), \tag{1}$$

where $U(\cdot)$ aggregates the evaluation scores into the overall score used for search and reporting. In our experiments, $U(\cdot)$ is a fixed weighted average of evaluation metrics.

3.2 Lineage Archive and Iterative Loop

Lineage Archive. POISE searches over $\mathcal{Z}$ by growing a lineage archive of evaluated candidates, as shown in Figure 1. At iteration t, the archive is denoted by $\mathcal{M}_t = \{e_1, \ldots, e_t\}$. Each archive node e_i corresponds to one evaluated candidate:

$$e_i = (z_i, \rho_i, \tau_i, \mathbf{y}_i, r_i, p_i),$$

where z_i is the candidate, ρ_i its executable implementation, τ_i the training trajectory, $\mathbf{y}_i$ the evaluation scores, r_i the analysis summary, and p_i the parent pointer.

Iterative Loop. Each iteration extends the archive through three phases. First, **Phase I** selects a parent node from the current archive and generates candidate policy-optimization proposals using the selected parent, the current archive, and literature priors. Second, **Phase II** implements the selected proposal, runs training, and evaluates the trained model under the fixed protocol. Third, **Phase III** analyzes the candidate outcome and adds the evaluated candidate as a new node to the archive. This loop allows later iterations to use both final evaluation scores and training behavior from earlier candidates, rather than relying only on isolated trial-and-error.

3.3 Phase I: Parent Selection and Proposal Generation

This phase selects a parent node from the lineage archive and generates the next candidate policy-optimization proposal.

Parent Selection. At each iteration, POISE selects the parent node with the highest estimated potential for producing stronger descendants, rather than simply expanding the node with the highest current evaluation score. Each evaluated node $e_i \in \mathcal{M}_t$ receives a score:

$$s(e_i) = w_{\text{pareto}}\, q_{\text{pareto}}(e_i) + w_{\text{perf}}\, q_{\text{perf}}(e_i) + w_{\text{div}}\, q_{\text{div}}(e_i) + w_{\text{gp}}\, \alpha_{\text{GP}}(e_i).$$

The node with the highest score is selected as the parent node:

$$e_p = \arg\max_{e_i \in \mathcal{M}_t} s(e_i).$$

Here, $q_{\text{perf}}(e_i)$ is the normalized version of the overall score $U(\mathbf{y}_i)$. The Pareto term $q_{\text{pareto}}(e_i)$ favors nodes that remain competitive across multiple evaluation metrics. The diversity term $q_{\text{div}}(e_i)$ favors different structural mechanism patterns, such as advantage estimation, masking, loss weighting, or regularization. The GP term $\alpha_{\text{GP}}(e_i)$ estimates the future potential of expanding e_i.

After each archive update, the GP is refitted to predict parent potential from node features. For nodes with descendants, the target is the normalized discounted top-K descendant gain:

$$\tilde{g}(e_i) = \frac{1}{|\text{TopK}(e_i)|} \sum_{c \in \text{TopK}(e_i)} \left[\frac{1}{2} + \frac{1}{2} \gamma^{\Delta(c, e_i)} \cdot \tanh\left(\frac{U(\mathbf{y}_c) - U(\mathbf{y}_i)}{\beta} \right) \right],$$

where $\text{TopK}(e_i)$ contains the top-K evaluated descendants of e_i, $\Delta(c, e_i)$ is the descendant depth relative to e_i, γ discounts deeper descendants, and β controls the gain scale. For newer nodes without enough descendant evidence, the target falls back to the normalized current score. The GP contribution is computed with a UCB score:

$$\alpha_{\text{GP}}(e_i) = \mu_{\text{GP}}(e_i) + \lambda \sigma_{\text{GP}}(e_i),$$

where $\mu_{\text{GP}}(e_i)$ is the predicted parent potential and $\sigma_{\text{GP}}(e_i)$ is predictive uncertainty. This score balances exploiting nodes with high predicted potential and exploring uncertain nodes that may still lead to strong descendants.

Proposal Generation. Given the selected parent node e_p, the LLM generates N candidate policy-optimization proposals using the parent, the current archive, and literature priors $\mathcal{L}$:

$$e_p, \mathcal{M}_t, \mathcal{L} \longrightarrow \{z_{t+1}^{(n)}\}_{n=1}^{N}.$$

The archive is provided through compact summaries of evaluated candidates, including their proposals, outcomes, and analysis summaries. Each proposal describes a policy-optimization change within the fixed search setting. The LLM ranks proposals by clarity, feasibility, novelty relative to archived candidates, and design plausibility. The selected proposal z_{t+1} is passed to the implementation, training, and evaluation phase.

3.4 Phase II: Candidate Implementation, Training, and Evaluation

After a proposal z_{t+1} is selected, POISE turns it into an executable implementation ρ_{t+1}, trains the candidate, and evaluates the trained model under the fixed protocol.

Candidate Implementation. POISE implements the selected proposal in VERL by modifying policy-optimization components such as advantage computation, loss weighting, masking, or regularization. The selected proposal is translated into the shared VERL implementation interface, so implementation focuses on the proposed policy-optimization mechanism while the surrounding experimental setting remains unchanged.

Training and Evaluation. After implementation, POISE trains the candidate under the fixed protocol. If execution fails or training is clearly abnormal, the implementation is revised and rerun. Training produces a trajectory τ_{t+1} that records the candidate's behavior. The trained model is then evaluated under the same protocol as other archive nodes, producing scores $\mathbf{y}_{t+1}$:

$$z_{t+1} \xrightarrow{\mathcal{O}} (\tau_{t+1}, \mathbf{y}_{t+1}).$$

Together, τ_{t+1} and $\mathbf{y}_{t+1}$ are used to compare the candidate with its parent and support the subsequent analysis.

3.5 Phase III: Result Analysis and Archive Update

Result Analysis. After training and evaluation, POISE compares the new candidate with its selected parent. Given the candidate z_{t+1}, its executable implementation ρ_{t+1}, training trajectory τ_{t+1}, evaluation scores $\mathbf{y}_{t+1}$, the selected parent node e_p, and the current archive $\mathcal{M}_t$, the LLM produces a structured mechanism-level analysis summary:

$$z_{t+1}, \rho_{t+1}, \tau_{t+1}, \mathbf{y}_{t+1}, e_p, \mathcal{M}_t \longrightarrow r_{t+1}.$$

The summary records what changed relative to the parent, which metrics improved or declined, how the training behavior changed, and whether the observed behavior is consistent with the intended mechanism. This turns the candidate outcome into structured evidence for later search.

Archive Update. After the analysis, POISE creates a new archive node for the evaluated candidate and adds it to the lineage archive:

$$e_{t+1} = (z_{t+1}, \rho_{t+1}, \tau_{t+1}, \mathbf{y}_{t+1}, r_{t+1}, p_{t+1}),$$
$$\mathcal{M}_{t+1} = \mathcal{M}_t \cup \{e_{t+1}\}.$$

Here, e_{t+1} is the new archive node and p_{t+1} points to the selected parent node e_p. This update makes the full candidate outcome available to later search steps, rather than only the final evaluation scores. [1]

4 Experiments

4.1 Experimental Setup

Training Setup. We implement candidate algorithms in VERL (Sheng et al., 2024) and train them on its distributed policy-optimization backend. The main search starts from the **GRPO** baseline (Shao et al., 2024), with **Gemini-3-pro-preview** (Google DeepMind) serving as the reasoning engine for the LLM-based stages. Following Zhao et al. (2025), we use **Qwen2.5-Math-1.5B** (Yang et al., 2024) as the policy model for rapid iteration. For training data, we sample a 5000-example subset from MATH Levels 3–5 (Hendrycks et al., 2021). Hyperparameters remain fixed across iterations: learning rate 10^{-6}, global batch size 256, group size $G = 8$, and 8 training epochs. Maximum prompt and response lengths are set to 1024 and 3000, respectively. All runs use 8× 80GB A100 GPUs.

Evaluation Protocol. We evaluate on six reasoning benchmarks: AIME24 (MAA, 2024), AIME25 (MAA, 2025), AMC (MAA, 2023), MATH-500 (Hendrycks et al., 2021), Minerva (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). Inference uses vLLM (Kwon et al., 2023) with a maximum generation length of 4096. For AIME24, AIME25, and AMC, we report pass@32 under temperature 1.0. For the larger datasets MATH-500, Minerva, and OlympiadBench, we report standard acc@1 using deterministic generation. To align with the search objective, we compute a weighted **Overall** that assigns larger weights to harder benchmarks: 0.2 for each AIME dataset and 0.15 for each remaining dataset.

[1] Implementation details and parameter settings are provided in Section A.

4.2 Results

Main Results. In the main search run, we evaluate 64 algorithms: GRPO and 63 generated variants. Table 1 reports representative generated variants together with three recent human-designed baselines, DAPO (Yu et al., 2025), Dr.GRPO (Liu et al., 2025d), and GMPO (Zhao et al., 2025), all evaluated under the same protocol; full results are provided in Section B.1.

The strongest discovered variant is VM-AV-GRPO, which achieves an Overall of 52.5, outperforming GRPO by +4.6. Among the human-designed baselines, the strongest result is Dr.GRPO with an Overall of 49.8; VM-AV-GRPO improves over it by +2.7. Thus, the discovered variant outperforms both the original GRPO baseline and competitive manually designed GRPO-style algorithms under the same protocol. Its largest gain appears on AIME25, where performance rises from 26.7 to 43.3, while AIME24, AMC, and MATH also improve over baseline. The search also uncovers benchmark-specific variants: AV-GRPO attains the best AIME24 result (56.7), MSA-GRPO matches the best AIME25 score (43.3), and SVE-LNA-GRPO achieves the highest AMC score (89.2). These results show that the search improves overall performance while discovering variants with benchmark-specific strengths.

Mechanism Families. Across the discovered lineages, four recurrent mechanism families emerge. **Core Signal Modeling** (*e.g.*, AV-GRPO and VM-AV-GRPO) changes how sparse successes are scaled and compared. **Signal Routing and Conditional Normalization** (*e.g.*, PR-GRPO and CDA-GRPO) separates solved, failed, and mixed regimes and reduces interference among correctness, format, and efficiency signals. **Failure-side Control and Failure Reshaping** (*e.g.*, FA-GRPO and DFR-GRPO) provides directional learning signals even when all samples in a group are wrong. **Stabilization and Constraint Control** (*e.g.*, RF-GRPO and BN-GRPO) dampens update shocks and policy drift. This last family mainly acts as a stabilizing precursor rather than the strongest final design; Sections 5 and E further explain how all four families interact across the lineage.

Algorithm	AIME24 pass@32	AIME25 pass@32	AMC pass@32	MATH acc@1	Minerva acc@1	Olympiad acc@1	Overall
GRPO	50.0	26.7	84.3	72.4	24.6	35.4	47.8
Human-Designed Baselines							
DAPO (Yu et al., 2025)	46.7 $_{-3.3}$	30.0 $_{+3.3}$	83.1 $_{-1.2}$	71.8 $_{-0.6}$	25.7 $_{+1.1}$	34.1 $_{-1.3}$	47.5 $_{-0.3}$
Dr.GRPO (Liu et al., 2025d)	53.3 $_{+3.3}$	36.7 $_{+10.0}$	86.7 $_{+2.4}$	68.4 $_{-4.0}$	22.8 $_{-1.8}$	33.9 $_{-1.5}$	49.8 $_{+2.0}$
GMPO (Zhao et al., 2025)	46.7 $_{-3.3}$	30.0 $_{+3.3}$	81.9 $_{-2.4}$	70.8 $_{-1.6}$	25.4 $_{+0.8}$	34.8 $_{-0.6}$	47.3 $_{-0.5}$
Core Signal Modeling							
VM-AV-GRPO	53.3 $_{+3.3}$	**43.3** $_{+16.7}$	88.0 $_{+3.6}$	**73.6** $_{+1.2}$	24.3 $_{-0.4}$	35.3 $_{-0.1}$	**52.5** $_{+4.6}$
AV-GRPO	**56.7** $_{+6.7}$	36.7 $_{+10.0}$	81.9 $_{-2.4}$	71.0 $_{-1.4}$	**26.8** $_{+2.2}$	35.3 $_{-0.1}$	50.9 $_{+3.1}$
High-Performing Specializations							
MSA-GRPO	50.0 $_{+0.0}$	**43.3** $_{+16.7}$	84.3 $_{+0.0}$	69.4 $_{-3.0}$	24.3 $_{-0.4}$	35.3 $_{-0.1}$	50.7 $_{+2.8}$
SVE-LNA-GRPO	50.0 $_{+0.0}$	30.0 $_{+3.3}$	**89.2** $_{+4.8}$	73.0 $_{+0.6}$	23.2 $_{-1.5}$	32.7 $_{-2.7}$	48.7 $_{+0.9}$
Failure-side and Routing Variants							
FA-GRPO	53.3 $_{+3.3}$	36.7 $_{+10.0}$	83.1 $_{-1.2}$	72.2 $_{-0.2}$	22.4 $_{-2.2}$	35.0 $_{-0.4}$	49.9 $_{+2.1}$
PR-GRPO	53.3 $_{+3.3}$	36.7 $_{+10.0}$	80.7 $_{-3.6}$	71.2 $_{-1.2}$	23.9 $_{-0.7}$	33.2 $_{-2.2}$	49.4 $_{+1.5}$
CDA-GRPO	50.0 $_{+0.0}$	30.0 $_{+3.3}$	85.5 $_{+1.2}$	72.6 $_{+0.2}$	25.0 $_{+0.4}$	**36.9** $_{+1.5}$	49.0 $_{+1.2}$

Table 1: Reasoning evaluation results (percentage points). We report a compact set of strong and mechanism-representative variants, along with human-designed baselines; full results are provided in the Appendix. Small colored numbers denote Δ vs. GRPO. Best results are in bold and second-best results are underlined.

Figure 2: Training dynamics in the main experiment for GRPO and representative generated variants. The plots compare entropy, reward, and response length over training.

Figure 2 compares training dynamics of GRPO and representative variants. Relative to the baseline, stronger variants show more stable entropy and reward trajectories, with different response-length behavior. These patterns suggest that the final-score gains are accompanied by distinct optimization dynamics.

4.3 Controllability Under Length-Compression Constraints

To test whether natural-language directives can steer the search, we introduce a **length-compression constraint**: candidate algorithms are favored when they substantially reduce reasoning output length while preserving overall accuracy.

Under this constraint, 6 of the 10 generated variants reduce mean response length. Mean length is computed as the unweighted mean word count over the six evaluation datasets; 4 of these variants (DACE-GRPO, CAG-GRPO, DRE-GRPO, PAS-GRPO) simultaneously achieve shorter responses and higher Overall than GRPO. The strongest compression results arise from correctness-first optimization combined with conditional efficiency shaping. DACE-GRPO gates the length-efficiency term on correct samples by group pass rate, while CAG-GRPO reformulates efficiency as a penalty-only cost gated by success rate. Variants with poorly calibrated efficiency constraints exhibit weaker accuracy-length trade-offs, suggesting that compression pressure is more effective when introduced conditionally rather than indiscriminately.

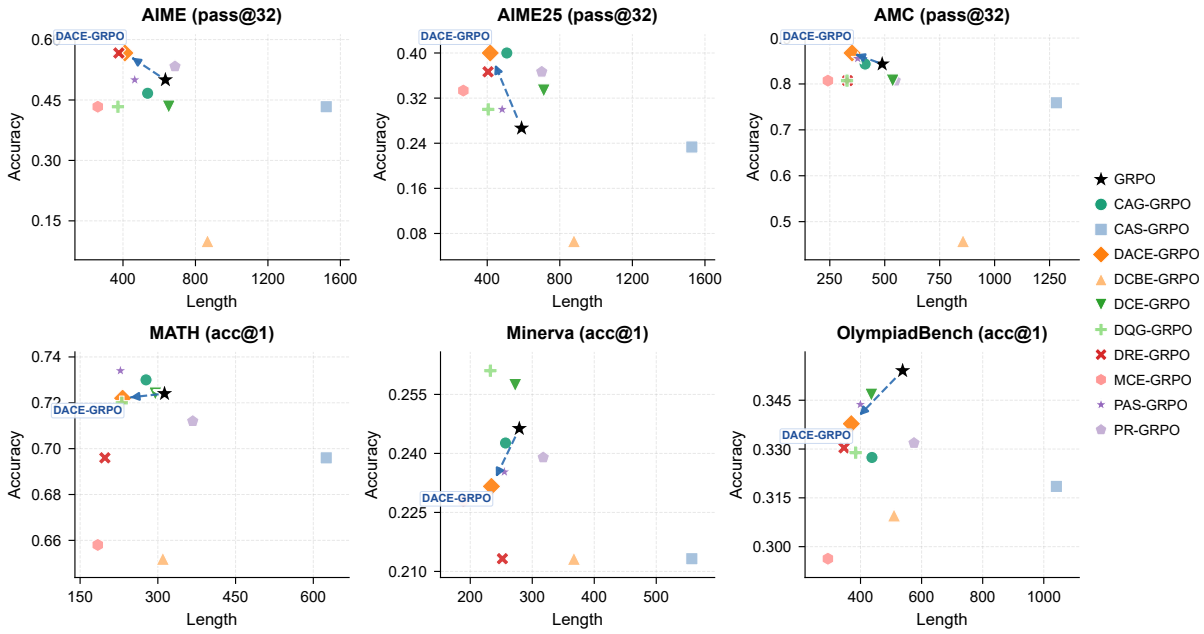

Figure 3: Accuracy-length trade-offs under the length-compression constraint.

Section F and Table 4 summarize the corresponding dynamics and the full quantitative results for all 11 nodes in the branch.

Figure 3 and Table 4 show that DACE-GRPO provides the best balance, improving Overall from 47.8 to 51.7 (+3.9) while reducing mean output length from 473.6 to 335.7 words (−29.1%); CAG-GRPO achieves stable compression (length ratio 0.854) with positive Overall gain. DRE-GRPO and PAS-GRPO show moderate gains but follow the same direction of difficulty- and correctness-conditioned efficiency shaping. At the dataset level, MATH and AMC retain substantial compression headroom without sacrificing accuracy: for example, DACE-GRPO reaches 86.7 on AMC pass@32 while reducing mean length by 28.1%. OlympiadBench is more compression-sensitive, where shorter variants generally incur mild accuracy drops, indicating a stronger dependence on longer reasoning chains. Overall, these results indicate that natural-language directives can meaningfully steer the search in a constrained accuracy-length trade-off space, and that correctness-first, conditionally shaped efficiency signals provide a viable route to effective compression.

5 Analysis

We begin by summarizing the full lineage by tree depth and then examine representative mechanism chains. Figure 4 shows that depth-wise averages are not monotonic, since deeper generations still include exploratory failures. Nevertheless, the frontier moves beyond the root: the cumulative best Overall increases from 47.8 for GRPO to 49.9 at depth 1, 50.9 at depth 3, and 52.5 at depth 4. Later depths still contain competitive descendants such as MSA-GRPO and CDA-GRPO. This suggests that the search remains productive even after the best observed node has already appeared. A fuller depth-wise view is provided in Figure 6.

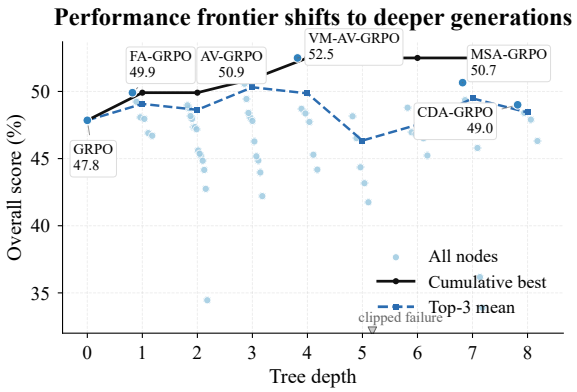

Figure 4: Performance frontier across tree depth in the base-branch lineage. Dots denote algorithms; solid and dashed lines show cumulative best Overall and top-3 mean Overall at each depth. The frontier improves beyond the root despite exploratory failures.

The lineage also helps assess whether retaining multiple parents is useful. Across 21 comparable parent-selection rounds, each selecting 3 nodes for parallel expansion, the parent yielding

the strongest descendant is not the highest-Overall parent in 11 rounds. One representative round is as follows (numbers in parentheses denote Overall scores): ZA-GRPO (47.3), BN-GRPO (44.2), and Anchor-GRPO (45.2) are retained together. In that round, the strongest later descendant arises from BN-GRPO via VM-AV-GRPO (52.5), above the ZA-GRPO branch peak of 48.4. This supports retaining diverse parents rather than filtering only by current Overall score. We provide a fuller round-level account and representative parent-descendant gains in Section E.1.

5.1 Mechanistic Analysis: Core Signal Modeling

A natural question is whether the search produces interpretable mechanism patterns rather than merely tuning hyperparameters. The lineage from GRPO to VM-AV-GRPO illustrates this in the **Core Signal Modeling** family, supported by training dynamics (Figure 2) and detailed lineage traces in Sections C, D and E.2. Baseline GRPO underperforms on sparse-reward benchmarks (*e.g.*, AIME25): failure-dominated group statistics can dilute the learning signal from rare successes, while high-variance updates can hasten entropy collapse.

To address this bottleneck, the lineage introduces **Analytic-variance scaling** (AV-GRPO). Instead of relying on empirical batch variance, it anchors advantage normalization to the analytic variance of the reward distribution so that rarer successes can yield stronger positive updates. A subsequent refinement then exposes another failure mode: when "format-correct but reasoning-wrong" samples enter the statistics, they can receive positive advantages. In response, the lineage introduces **Validity Masking** (VM-AV-GRPO), which restores a strict gradient hierarchy (invalid $\ll$ valid-wrong $<$ valid-correct) by computing statistics exclusively on the valid subset. Together, analytic-variance scaling and validity masking are associated with a substantial AIME25 improvement from GRPO to VM-AV-GRPO, while Sections C and E.2 trace the intermediate turning points and the corresponding algorithm details behind this refinement.

5.2 Learning from Negative Evidence

Beyond discovering improved algorithms, the search also uses negative evidence to understand why some designs fail. This is evident in the lineage targeting all-fail stagnation (GRPO $\rightarrow$ FA-GRPO $\rightarrow$ DFR-GRPO), a representative instance of **Failure-side Control and Failure Reshaping**; Sections C and E.2 provide the extended trajectory and algorithm-level formulations.

FA-GRPO first injects gradients into all-fail groups through fixed scalar penalties. However, analysis of training dynamics suggests that mixing such fixed penalties with normalized advantages causes a scale mismatch that destabilizes formatting. Furthermore, applying uniform negative advantages acts as "directionless repulsion," triggering policy drift. DFR-GRPO then refines this design by separating the signals. It decouples format and reasoning into independent streams and replaces uniform repulsion with a **zero-centered signal** derived from token entropy, so that more informative failures are compared against less informative ones rather than being pushed equally. This design provides a more directional signal even when all candidates are incorrect, alleviating directionless drift, albeit with trade-offs on the hardest benchmarks. Together with the **Signal Routing and Conditional Normalization** cases detailed in Sections E.2 and E.3, this cross-generational refinement highlights three recurring principles. Rare successes should be amplified through calibrated signal modeling, negative evidence should be used to revise or avoid flawed hypotheses, and secondary objectives should be introduced through signal decoupling under controlled conditions.

6 Conclusion

In this work, we show that the discovery of policy optimization algorithms for language models can be organized as a closed-loop search process of candidate generation, implementation, verification, training, evaluation, and result analysis, rather than treated solely as manual trial and error. POISE discovers stronger policy-optimization variants from a GRPO baseline, identifies mechanisms such as analytic-variance scaling and validity masking, and further shows through the length-compression setting that natural-language directives can steer the search toward a specified region of the accuracy-length trade-off space. The resulting lineages reveal recurring principles such as signal decoupling, conditional normalization, and correctness-first efficiency shaping, showing that automated search can not only discover stronger variants but also produce reusable evidence for understanding and improving policy-optimization design.

Limitations

Our study demonstrates the promise of automated mechanism discovery, but limitations remain. First, the closed-loop search process is still computationally intensive, since each candidate requires end-to-end training and standardized evaluation; under a fixed budget, this limits both the breadth of explored candidates and the number of controlled follow-up trials. Second, the empirical evidence is concentrated on mathematical reasoning benchmarks and a limited range of model settings, so the transferability of the discovered mechanisms to broader regimes, such as open-domain dialogue, code generation, and tool-use tasks, remains to be established. Third, although lineage analysis, training dynamics, and result analysis support mechanism-level interpretations, these interpretations should be treated as evidence-grounded hypotheses rather than as controlled causal attribution of each gain to a specific component. Future work should focus on improving search efficiency, broadening cross-task and cross-scale validation, and developing targeted ablations and causal evaluation protocols.

Ethics Statement

We confirm that all authors are aware of the ACL Code of Ethics and commit to conducting this research in accordance with responsible research practices.

Data Use and Privacy. Our experiments use publicly available mathematical reasoning benchmarks (*e.g.*, AIME, AMC, MATH, Minerva, and OlympiadBench) and do not involve private or personally identifiable information. The training subset and evaluation data are used under the respective dataset terms and licenses. We do not collect user-level behavioral data or deploy systems that interact with human subjects.

Automation Risks and Responsible Use. This work studies automated algorithm discovery for LLM-RL. Although our setting is focused on mathematical reasoning, automated search systems can, in principle, generate unstable or misaligned optimization strategies if applied without safeguards. To reduce this risk, we use bounded verification loops, standardized evaluation, and human oversight in the research pipeline, and we report limitations regarding generalization and causal interpretation. We do not claim that the discovered mechanisms are universally safe or optimal outside the evaluated scope.

Use of AI Assistants. LLMs are used as research assistants within the proposed framework (for proposal generation, implementation support, and result analysis) and for manuscript polishing. Final scientific claims, experimental decisions, and interpretation of results are made and verified by the authors.

References

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. 2023. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, and 1 others. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*.

Junyan Cheng, Peter Clark, and Kyle Richardson. 2025. Language modeling by language models. *arXiv preprint arXiv:2506.20249*.

Yuri Chervonyi, Trieu H Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang H Nguyen, Marcelo Menegali, Junehyuk Jung, Junsu Kim, Vikas Verma, Quoc V Le, and 1 others. 2025. Gold-medalist performance in solving olympiad geometry with alphageometry2. *Journal of Machine Learning Research*, 26(241):1–39.

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

Alhussein Fawzi and Bernardino Romera Paredes. 2023. Funsearch: Making new discoveries in mathematical sciences using large language models. *DeepMind Google*.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. 2025. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*.

Google DeepMind. https://deepmind.google/models/gemini/pro/.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, and 1 others. 2025. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

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

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*.

Farhana Keya, Gollam Rabby, Prasenjit Mitra, Sahar Vahdati, Sören Auer, and Yaser Jaradeh. 2025. Sci-idea: Context-aware scientific ideation using token and sentence embeddings. *arXiv preprint arXiv:2503.19257*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

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

Ruochen Li, Liqiang Jing, and Xinya Du. 2024. Learning to generate research idea with dynamic control. In *2nd AI4Research Workshop: Towards a Knowledge-grounded Scientific Research Lifecycle*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025a. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Yixiu Liu, Yang Nan, Weixian Xu, Xiangkun Hu, Lyumanshan Ye, Zhen Qin, and Pengfei Liu. 2025b. Alphago moment for model architecture discovery. *arXiv preprint arXiv:2507.18074*.

Zexi Liu, Yuzhu Cai, Xinyu Zhu, Yujie Zheng, Runkun Chen, Ying Wen, Yanfeng Wang, Siheng Chen, and 1 others. 2025c. Ml-master: Towards ai-for-ai via integration of exploration and reasoning. *arXiv preprint arXiv:2506.16499*.

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

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob N Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *CoRR*.

MAA. 2023. American mathematics contest 12 (amc 12).

MAA. 2024. American invitational mathematics examination (aime).

MAA. 2025. American invitational mathematics examination (aime).

Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C Landsness, Daniel L Barabasi, Siddharth Narayanan, Nicky Evans, and 1 others. 2025. Kosmos: An ai scientist for autonomous discovery. *arXiv preprint arXiv:2511.02824*.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, and 1 others. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

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.

Samuel Schmidgall and Michael Moor. 2025. Agentrxiv: Towards collaborative autonomous research. *arXiv preprint arXiv:2503.18102*.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*.

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, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Yu-Zhe Shi, Mingchen Liu, Fanxu Meng, Qiao Xu, Zhangqian Bi, Kun He, Lecheng Ruan, and Qining Wang. 2025. Hierarchically encapsulated representation for protocol design in self-driving labs. In *The Thirteenth International Conference on Learning Representations*.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. In *The Thirteenth International Conference on Learning Representations*.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.

Jiakang Wang, Runze Liu, Lei Lin, Wenping Hu, Xiu Li, Fuzheng Zhang, Guorui Zhou, and Kun Gai. 2025. Aspo: Asymmetric importance sampling policy optimization. *arXiv preprint arXiv:2510.06062*.

Weixian Xu, Tiantian Mi, Yixiu Liu, Yang Nan, Zhimeng Zhou, Lyumanshan Ye, Lin Zhang, Yu Qiao, and Pengfei Liu. 2026. Asi-evolve: Ai accelerates ai. *arXiv preprint arXiv:2603.29640*.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *CoRR*.

Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shaohan Huang, Lei Cui, Qixiang Ye, and 1 others. 2025. Geometric-mean policy optimization. *arXiv preprint arXiv:2507.20673*.

A Implementation and Reproducibility Details

A.1 Search Space and Fixed Protocol

The implementation instantiates POISE as a fixed-protocol search over policy-optimization variants for LLM-RL. Each candidate z specifies a complete policy-optimization variant under this protocol, including a natural-language proposal, a mechanism family, mechanism-specific hyperparameters, expected effects, and an executable implementation ρ in VERL. The search is restricted to policy-optimization mechanisms inside the training loop. It does not modify the base model architecture, training corpus, evaluation benchmarks, answer parser, prompt format, rollout group size, decoding temperature, or total training length.

The fixed and searchable parts of the system are as follows. The base model and training data are fixed to Qwen2.5-Math-1.5B and a 5000-example subset from MATH Levels 3–5. The evaluation protocol is fixed across AIME24, AIME25, AMC, MATH-500, Minerva, and OlympiadBench, with unchanged metric definitions. The rollout protocol uses group size $G = 8$, vLLM rollout, and fixed decoding settings for each benchmark group. The training budget is fixed, including learning rate 10^{-6}, global batch size 256, 8 epochs, and fixed prompt and response lengths. The searchable components are policy-optimization mechanisms, including advantage estimation, reward or validity masking, loss weighting, signal routing, regularization, and stabilization terms. Candidate-specific hyperparameters are specified together with the proposed mechanism.

Within VERL, candidate implementations modify recipe-level policy-optimization components. Editable components include reward preprocessing, advantage computation, advantage postprocessing, sample- or group-level weighting, auxiliary regularization terms, and mechanism-specific training signals. Model architecture, data loading, evaluation scripts, reward-model training, and distributed runtime infrastructure remain outside the search space.

A.2 Stage Inputs and Outputs

Each stage in the search loop receives the information needed for its role rather than an unfiltered interaction history. For LLM-based stages, this information is provided as agent context; for non-LLM stages, it is used by the selector, executor, or archive updater. The information flow is organized as follows.

Parent selection.

Input: the current archive, evaluation metrics, trajectory summaries, derived structural features, and parent-child relations.

Output: a selected parent e_p.

Evidence construction.

Input: the selected parent, parent analysis, archive summaries, lineage evidence, and literature retrieval results.

Output: compact context for proposal generation.

Proposal generation.

Input: the selected parent, compact archive summaries, regressive or failed-design cases, search constraints, and literature priors.

Output: candidate policy-optimization proposals.

Proposal ranking.

Input: generated proposals, the search scope, and novelty and feasibility criteria.

Output: the selected proposal z_{t+1}.

Implementation planning.

Input: the selected proposal, VERL interface information, reference implementations, and the allowed modification scope.

Output: an implementation plan specifying where the mechanism enters the training computation.

Implementation.

Input: the selected proposal, the implementation plan, a parent implementation or template, and VERL interface information.

Output: an executable implementation ρ_{t+1}.

Verification and correction.

Input: the current implementation, proposal intent, verification feedback, error traces, or abnormal-run summaries.

Output: a verified implementation, a revised implementation, or a failure record.

Training and evaluation.

Input: the executable implementation and the fixed training and evaluation protocol.

Output: trajectory evidence τ_{t+1} and evaluation metrics $\mathbf{y}_{t+1}$.

Result analysis.

Input: the proposal, implementation summary, trajectory evidence, evaluation metrics, and parent/reference comparisons.

Output: a structured analysis summary r_{t+1}.

Archive update.

Input: the candidate, implementation, trajectory evidence, metrics, analysis summary, and parent relation.

Output: the updated lineage archive $\mathcal{M}_{t+1}$.

The proposal stage asks the LLM to specify the candidate name, parent algorithm, key mechanism, algorithmic change, mechanism-specific hyperparameters, and expected effect. The proposal checker filters candidates that fall outside the fixed search scope, require changes to the training or evaluation protocol, duplicate existing archive nodes, or lack a testable mechanism.

A.3 Archive Records and Evidence Flow

The lineage archive stores the information needed to reuse an evaluated candidate in later search steps. For each evaluated candidate, the archive records the proposal z_i, the executable implementation ρ_i, the trajectory evidence τ_i, the evaluation metrics $\mathbf{y}_i$, the analysis summary r_i, and the parent relation when the candidate is derived from an earlier node. These records correspond to the archive entry $e_i = (z_i, \rho_i, \tau_i, \mathbf{y}_i, r_i)$ in Section 3.

The archive supports two uses. First, it provides evidence for parent selection, including evaluation scores, trajectory summaries, structural features derived from proposals and implementations, and parent-child relations. Second, it provides compact context for proposal generation and result analysis, including successful candidates, regressive candidates, and prior analysis summaries. Completed candidates are retained even when their scores regress, because they identify failure modes and mechanisms that later proposals should avoid.

A.4 Parent Selection and GP Scoring

Parent selection combines current performance, multi-metric competitiveness, structural diversity, and a Gaussian-process estimate of parent potential. The acquisition score is

$$s(d) = w_{\text{pareto}} q_{\text{pareto}}(d) + w_{\text{perf}} q_{\text{perf}}(d) + w_{\text{div}} q_{\text{div}}(d) + w_{\text{gp}} \alpha_{\text{GP}}(d).$$

The term q_{perf} is computed from normalized evaluation performance. The term q_{pareto} is based on Pareto-front membership computed over performance, validity, learning-signal aliveness, stability, efficiency, and diversity objectives. The term q_{div} is computed from structural mechanism indicators inferred from the candidate name and implementation. The term α_{GP} is a GP-UCB score.

In the main search, the base parent-selection weights are $w_{\text{pareto}} = 0.2$, $w_{\text{gp}} = 0.2$, $w_{\text{div}} = 0.1$, and $w_{\text{perf}} = 0.5$. The GP component uses a small exploration weight to preserve uncertainty-sensitive selection, a top-K descendant target with $K = 3$, a depth discount $\gamma = 0.7$, and a gain scale $\beta = 0.02$. We use an RBF-kernel Gaussian process with observation-noise parameter 10^{-4}, a maximum descendant depth of 3, and three optimizer restarts. These values are fixed for the main search.

The selector uses a dynamic-weight schedule around these base weights. In early rounds (≤ 5), the diversity weight is increased and the performance weight is reduced before normalization. In late rounds (> 15), the diversity weight is reduced and the performance weight is increased before normalization. This schedule gives more weight to exploration early and more weight to performance later.

Node features. The GP input is a 20-dimensional archived-node feature vector. The feature vector includes performance features, stability features, efficiency features, structural mechanism indicators, and rollout-derived features. The performance features include weighted performance, an overall pass/accuracy proxy, and task coverage. The stability features include entropy instability, policy-loss convergence, and KL instability. The efficiency features include convergence speed and sample efficiency. The structural features indicate whether the candidate uses mechanisms such as KL control, groupwise processing, entropy regularization, advantage normalization, reference update, PPO clipping, value clipping, or custom-loss computation. The rollout-derived features include mixed-rate, boxed-answer rate, refusal rate, and response-length variation. Missing rollout-derived features are set to zero.

GP target and fallback. When a node has evaluated descendants within the configured depth, the GP target is a normalized discounted top-K descen-

dant gain:

$$\tilde{g}(d) = \frac{1}{|\mathrm{TopK}(d)|} \sum_{c \in \mathrm{TopK}(d)} \left[\frac{1}{2} + \frac{1}{2} \gamma^{\Delta(c,d)} \cdot \tanh\left(\frac{U(\mathbf{y}_c) - U(\mathbf{y}_d)}{\beta} \right) \right],$$

where $\Delta(c, d)$ denotes the depth of descendant c relative to node d. Descendants include children and deeper descendants up to the maximum configured depth. If a node does not yet have enough descendant evidence, the selector uses the normalized current overall score $U(\mathbf{y}_d)$ as a fallback target. The GP is refit after new evaluated nodes are added to the archive. For already-observed nodes, the selector supports leave-one-out GP scoring so that predictive uncertainty remains meaningful instead of collapsing at training points.

A.5 Proposal Generation and Selection

The proposal generator samples multiple candidate proposals before selecting one for implementation. The main multi-proposal setting uses $N = 5$ candidates per proposal step. The proposal stage uses the selected parent, the parent lineage, compact archive evidence, reference nodes, prior analysis summaries, and literature priors.

Each proposal includes a candidate name, parent algorithm, key innovation, motivation, algorithmic change, pseudocode, mechanism-specific hyperparameters, and expected benefit. The proposal is ranked by clarity, compatibility with the fixed search scope, novelty relative to archived candidates, design plausibility, and testability through training and evaluation signals.

A.6 Implementation, Verification, and Correction

The selected proposal is first converted into an implementation plan. The planning stage inspects the proposal, the parent implementation or template when available, and the VERL interfaces that define the editable policy-optimization components. It identifies which part of the training computation should be modified and checks whether the proposed mechanism can be expressed within the allowed implementation boundary. Dedicated checking agents examine plan completeness, consistency between the plan and proposal, feasibility under VERL, and whether additional clarification or revision is required before code generation.

The implementation agent then generates the candidate implementation within a bounded workspace. It can inspect VERL source code, search existing implementations for similar patterns, read interface definitions, and write candidate-specific implementation files. It is not allowed to modify the base model, training data, evaluation scripts, or distributed runtime infrastructure. After code generation, validation agents check whether the implementation follows the proposal and plan and whether the fixed protocol remains unchanged.

A local execution harness performs syntax, import, configuration, and lightweight execution checks. When execution fails, the failure trace is converted into a structured correction context. The correction agent can inspect the current candidate files, consult read-only framework references, apply bounded edits within the candidate workspace, and rerun lightweight checks. The training-fix loop uses a bounded correction budget; in our configuration, the maximum number of correction attempts is 30. The executor records error summaries, correction attempts, and correction outcomes. Candidates that cannot be corrected to a trainable implementation are not counted as successful evaluated candidates.

Training can also reveal semantic implementation problems that are not exposed by local checks. For example, a metric may remain zero because a mechanism is not wired into the training computation, or a training signal may be inconsistent with the intended design. In such cases, the training-results analysis stage can trigger a targeted correction when the issue is attributable to implementation rather than to the algorithmic idea itself.

A.7 Training and Evaluation Configuration

All candidates are trained and evaluated under the same protocol. The base policy model is Qwen2.5-Math-1.5B, and the initial baseline is GRPO. The training data is a 5000-example subset from MATH Levels 3–5. Training uses the VERL policy-optimization backend with learning rate 10^{-6}, global batch size 256, rollout group size $G = 8$, and 8 epochs. The maximum prompt length is 1024, and the maximum training response length is 3000. Each full training run uses $8\times$ 80GB A100 GPUs.

The evaluation suite includes AIME24, AIME25, AMC, MATH-500, Minerva, and OlympiadBench. Evaluation uses vLLM with maximum generation

length 4096. AIME and AMC are evaluated with pass@32 under temperature 1.0. MATH-500, Minerva, and OlympiadBench use deterministic acc@1. The overall score is a fixed weighted average: AIME24 and AIME25 each receive weight 0.2, and AMC, MATH-500, Minerva, and OlympiadBench each receive weight 0.15.

A.8 Trajectory Evidence

Trajectory evidence τ is represented by sanitized training and evaluation summaries rather than raw unfiltered logs. The pipeline removes timing, memory, GPU, throughput, and other system-housekeeping fields from the full training logs, and then renders a compact text representation for the analysis stage. For training trajectories, the summary is organized by sampled training steps. For evaluation outputs, the summary lists task metrics after filtering metadata.

The trajectory summary can include reward mean and variability, policy loss, clip fraction, KL, entropy, gradient norm, response length, prompt/response clipping indicators, advantage mean and variance, validity or boxed-answer proxies, rollout-level indicators such as mixed rate and refusal rate, and abnormal-run flags such as divergence or NaN. The agent-facing representation is textual and derived from these sanitized summaries.

A.9 Result Analysis

After a candidate is trained and evaluated, the analysis stage compares it with its parent and selected reference nodes. The LLM receives the proposal, an implementation summary, trajectory evidence, evaluation metrics, and parent/reference comparisons. The resulting analysis summary records the main differences from the parent, metric improvements or regressions, changes in training behavior, consistency with the intended mechanism, and implications for later proposals.

This summary is stored in the archive and used as evidence in later parent selection and proposal generation. When the available evidence is incomplete, the analysis is required to state the missing information rather than infer unsupported mechanisms. The summary is treated as evidence-grounded interpretation for later search, not as standalone causal proof.

A.10 Computational Cost and Run Accounting

The reported search evaluates 64 algorithms, including the GRPO baseline and evolved variants. Each full training run takes approximately 6 hours on 8× A100 GPUs in our environment, corresponding to about 48 GPU-hours. Successful candidates also incur correction and rerun overhead; in our run accounting, this averages about 0.8 failed or corrected attempts per successful variant. Including this overhead, the main search cost is approximately 5,500 GPU-hours. Proposal generation, analysis, and GP-based parent scoring account for a small fraction of the full training cost.

B Additional Results and Lineage Analysis

B.1 Full Main-Experiment Results and Lineage

We provide the full evaluation results for all evaluated variants in Table 2. To show how candidates are derived across the main search run, we also visualize the complete lineage structure in Figure 5.

C Detailed Algorithm Descriptions (Representative Subset)

Due to space constraints, we do not enumerate all evaluated variants in this section. Instead, we describe a representative subset that supports the main interpretation, including improving cases, regressive cases, and response-compression cases. For each algorithm, we report its **motivation**, **method** with key equations, and **results**. AIME24, AIME25, and AMC use pass@32; MATH, Minerva, and OlympiadBench use acc@1; Overall follows the weighted score in the main text.

C.1 GRPO Baseline

GRPO

Motivation. Under sparse correctness rewards, GRPO provides a stable relative learning signal without introducing a value network.

Method. For each prompt, GRPO samples G

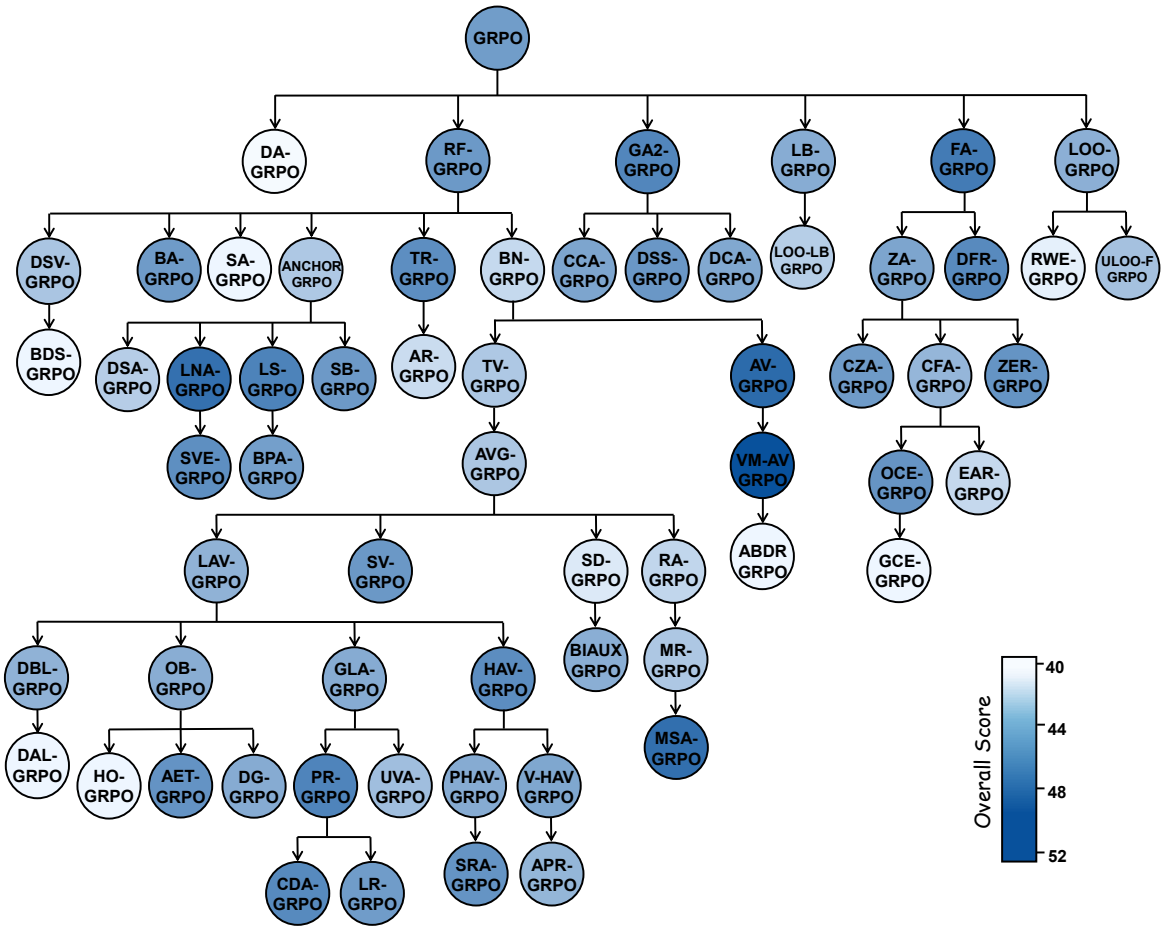

Figure 5: **Lineage of GRPO variants in the main run.** Nodes denote evaluated algorithms, arrows denote parent-child relations, and node color encodes the weighted Overall score, with lighter colors indicating lower scores.

Algorithm	AIME24 pass@32	AIME25 pass@32	AMC pass@32	MATH acc@1	Minerva acc@1	Olympiad acc@1	Overall
GRPO	50.0	26.7	84.3	72.4	24.6	35.4	47.8
VM-AV-GRPO	53.3 +3.3	43.3 +16.7	88.0 +3.6	73.6 +1.2	24.3 -0.4	35.3 -0.1	52.5 +4.6
AV-GRPO	56.7 +6.7	36.7 +10.0	81.9 -2.4	71.0 -1.4	26.8 +2.2	35.3 -0.1	50.9 +3.1
MSA-GRPO	50.0 +0.0	43.3 +16.7	84.3 +0.0	69.4 -3.0	24.3 -0.4	35.3 -0.1	50.7 +2.8
LNA-GRPO	56.7 +6.7	33.3 +6.7	84.3 +0.0	73.6 +1.2	24.3 -0.4	35.0 -0.4	50.6 +2.7
FA-GRPO	53.3 +3.3	36.7 +10.0	83.1 -1.2	72.2 -0.2	22.4 -2.2	35.0 -0.4	49.9 +2.1
LS-GRPO	46.7 -3.3	36.7 +10.0	86.7 +2.4	71.4 -1.0	24.6 +0.0	35.7 +0.3	49.4 +1.6
PR-GRPO	53.3 +3.3	36.7 +10.0	80.7 -3.6	71.2 -1.2	23.9 -0.7	33.2 -2.2	49.4 +1.5
GA2-GRPO	53.3 +3.3	33.3 +6.7	83.1 -1.2	70.6 -1.8	24.3 -0.4	34.7 -0.7	49.2 +1.4
CDA-GRPO	50.0 +0.0	30.0 +3.3	85.5 +1.2	72.6 +0.2	25.0 +0.4	36.9 +1.5	49.0 +1.2
DFR-GRPO	46.7 -3.3	33.3 +6.7	85.5 +1.2	73.4 +1.0	25.7 +1.1	35.1 -0.3	49.0 +1.1
HAV-GRPO	43.3 -6.7	36.7 +10.0	88.0 +3.6	72.6 +0.2	22.8 -1.8	35.3 -0.1	48.8 +0.9
TR-GRPO	50.0 +0.0	33.3 +6.7	83.1 -1.2	72.4 +0.0	25.4 +0.7	33.0 -2.4	48.8 +0.9
SVE-LNA-GRPO	50.0 +0.0	30.0 +3.3	89.2 +4.8	73.0 +0.6	23.2 -1.5	32.7 -2.7	48.7 +0.9
AET-GRPO	46.7 -3.3	36.7 +10.0	84.3 +0.0	71.2 -1.2	23.5 -1.1	32.7 -2.7	48.4 +0.6
SRA-GRPO	46.7 -3.3	33.3 +6.7	86.7 +2.4	72.0 -0.4	23.5 -1.1	33.8 -1.6	48.4 +0.6
ZER-GRPO	50.0 +0.0	33.3 +6.7	80.7 -3.6	72.2 -0.2	24.3 -0.4	34.4 -1.0	48.4 +0.6
OCE-GRPO	46.7 -3.3	33.3 +6.7	84.3 +0.0	71.0 -1.4	24.3 -0.4	36.1 +0.7	48.4 +0.5
DSS-GRPO	53.3 +3.3	26.7 +0.0	83.1 -1.2	72.4 +0.0	23.5 -1.1	35.4 +0.0	48.2 +0.3
SV-GRPO	46.7 -3.3	30.0 +3.3	86.7 +2.4	72.6 +0.2	26.1 +1.5	33.3 -2.1	48.2 +0.3
RF-GRPO	50.0 +0.0	33.3 +6.7	80.7 -3.6	71.2 -1.2	23.5 -1.1	33.9 -1.5	48.1 +0.2
SB-GRPO	50.0 +0.0	30.0 +3.3	88.0 +3.6	71.6 -0.8	21.7 -2.9	32.4 -3.0	48.1 +0.2
DA-GRPO	53.3 +3.3	30.0 +3.3	79.5 -4.8	71.4 -1.0	23.9 -0.7	33.8 -1.6	48.0 +0.1
BA-GRPO	46.7 -3.3	33.3 +6.7	83.1 -1.2	71.4 -1.0	24.3 -0.4	34.2 -1.2	48.0 +0.1
LR-GRPO	46.7 -3.3	36.7 +10.0	84.3 +0.0	70.2 -2.2	21.7 -2.9	32.0 -3.4	47.9 +0.1
CZA-GRPO	43.3 -6.7	36.7 +10.0	83.1 -1.2	74.6 +2.2	20.2 -4.4	34.1 -1.3	47.8 +0.0
BPA-GRPO	50.0 +0.0	33.3 +6.7	83.1 -1.2	71.0 -1.4	18.8 -5.9	34.2 -1.2	47.7 -0.1
ZA-GRPO	50.0 +0.0	26.7 +0.0	81.9 -2.4	73.0 +0.6	24.3 -0.4	34.2 -1.2	47.3 -0.5
CCA-GRPO	50.0 +0.0	26.7 +0.0	81.9 -2.4	71.8 -0.6	25.0 +0.4	34.7 -0.7	47.3 -0.5
V-HAV-GRPO	43.3 -6.7	33.3 +6.7	85.5 +1.2	68.2 -4.2	23.5 -1.1	35.4 +0.0	47.2 -0.6
DCA-GRPO	43.3 -6.7	36.7 +10.0	81.9 -2.4	69.4 -3.0	23.5 -1.1	33.0 -2.4	47.2 -0.7
DG-GRPO	43.3 -6.7	30.0 +3.3	86.7 +2.4	71.8 -0.6	23.2 -1.5	33.6 -1.8	47.0 -0.9
PHAV-GRPO	46.7 -3.3	26.7 +0.0	84.3 +0.0	72.4 +0.0	24.6 +0.0	33.9 -1.5	47.0 -0.9
GLA-GRPO	40.0 -10.0	33.3 +6.7	86.7 +2.4	71.8 -0.6	22.1 -2.6	34.7 -0.7	47.0 -0.9
LB-GRPO	53.3 +3.3	23.3 -3.3	83.1 -1.2	69.0 -3.4	23.2 -1.5	35.1 -0.3	46.9 -1.0
OB-GRPO	53.3 +3.3	26.7 +0.0	78.3 -6.0	69.4 -3.0	22.8 -1.8	35.0 -0.4	46.8 -1.0
BIAUX-GRPO	50.0 +0.0	30.0 +3.3	83.1 -1.2	69.2 -3.2	20.6 -4.0	32.3 -3.1	46.8 -1.1
LOO-GRPO	53.3 +3.3	16.7 -10.0	85.5 +1.2	72.8 +0.4	24.3 -0.4	35.4 +0.0	46.7 -1.1
LAV-GRPO	50.0 +0.0	33.3 +6.7	79.5 -4.8	67.6 -4.8	21.0 -3.7	31.0 -4.4	46.5 -1.3
DBL-GRPO	43.3 -6.7	33.3 +6.7	80.7 -3.6	71.0 -1.4	22.8 -1.8	33.3 -2.1	46.5 -1.3
APR-GRPO	43.3 -6.7	26.7 +0.0	83.1 -1.2	71.2 -1.2	26.1 +1.5	35.0 -0.4	46.3 -1.5
CFA-GRPO	33.3 -16.7	33.3 +6.7	85.5 +1.2	75.0 +2.6	24.3 -0.4	34.8 -0.6	46.3 -1.6
UVA-GRPO	46.7 -3.3	20.0 -6.7	86.7 +2.4	73.0 +0.6	23.2 -1.5	33.5 -1.9	45.8 -2.1
ULOO-F-GRPO	36.7 -13.3	40.0 +13.3	77.1 -7.2	70.4 -2.0	20.2 -4.4	34.1 -1.3	45.6 -2.2
DSV-GRPO	43.3 -6.7	30.0 +3.3	80.7 -3.6	69.2 -3.2	23.2 -1.5	31.6 -3.9	45.4 -2.5
AVG-GRPO	43.3 -6.7	30.0 +3.3	83.1 -1.2	68.8 -3.6	21.0 -3.7	31.3 -4.1	45.3 -2.6
MR-GRPO	46.7 -3.3	30.0 +3.3	79.5 -4.8	67.8 -4.6	21.3 -3.3	30.7 -4.7	45.2 -2.6
TV-GRPO	40.0 -10.0	33.3 +6.7	78.3 -6.0	71.0 -1.4	21.3 -3.3	32.7 -2.7	45.2 -2.7
ANCHOR-GRPO	43.3 -6.7	26.7 +0.0	79.5 -4.8	72.4 +0.0	23.2 -1.5	32.7 -2.7	45.2 -2.7
DSA-GRPO	46.7 -3.3	23.3 -3.3	75.9 -8.4	73.6 +1.2	23.2 -1.5	33.0 -2.4	44.9 -3.0
LOO-LB-GRPO	43.3 -6.7	26.7 +0.0	79.5 -4.8	70.4 -2.0	22.8 -1.8	32.9 -2.5	44.8 -3.0
RA-GRPO	40.0 -10.0	26.7 +0.0	84.3 +0.0	69.8 -2.6	21.0 -3.7	31.7 -3.7	44.4 -3.5
EAR-GRPO	36.7 -13.3	26.7 +0.0	80.7 -3.6	72.2 -0.2	22.8 -1.8	34.4 -1.0	44.2 -3.7
BN-GRPO	40.0 -10.0	26.7 +0.0	77.1 -7.2	72.4 +0.0	22.1 -2.6	33.9 -1.5	44.2 -3.7
AR-GRPO	40.0 -10.0	23.3 -3.3	80.7 -3.6	71.6 -0.8	23.2 -1.5	33.2 -2.2	44.0 -3.9
SD-GRPO	33.3 -16.7	26.7 +0.0	81.9 -2.4	71.2 -1.2	22.8 -1.8	31.9 -3.6	43.2 -4.7
RWE-GRPO	36.7 -13.3	20.0 -6.7	77.1 -7.2	72.8 +0.4	24.3 -0.4	35.3 -0.1	42.7 -5.1
BDS-GRPO	36.7 -13.3	23.3 -3.3	75.9 -8.4	71.4 -1.0	21.3 -3.3	32.7 -2.7	42.2 -5.6
ABDR-GRPO	33.3 -16.7	20.0 -6.7	78.3 -6.0	72.4 +0.0	22.4 -2.2	34.1 -1.3	41.7 -6.1
DAL-GRPO	16.7 -33.3	26.7 +0.0	55.4 -28.9	70.8 -1.6	22.1 -2.6	35.1 -0.3	36.2 -11.7
GAA-GRPO	20.0 -30.0	10.0 -16.7	63.9 -20.5	71.2 -1.2	27.6 +2.9	35.0 -0.4	35.6 -12.2
SA-GRPO	20.0 -30.0	13.3 -13.3	54.2 -30.1	73.2 +0.8	23.9 -0.7	33.9 -1.5	34.5 -13.4
HO-GRPO	26.7 -23.3	6.7 -20.0	63.9 -20.5	67.2 -5.2	17.3 -7.4	33.2 -2.2	33.9 -14.0
GCE-GRPO	0.0 -50.0	0.0 -26.7	1.2 -83.1	53.0 -19.4	15.1 -9.6	24.4 -11.0	14.1 -33.8

Table 2: Full held-out reasoning evaluation results for all evaluated variants (percentage points). GRPO (first row) is the baseline. Small colored numbers denote Δ vs. GRPO (green: improvement; red: regression). "Overall" is the fixed weighted average used by the evolutionary selector (weights in Section 4.1). "Olympiad" denotes OlympiadBench.

responses and computes group-relative advantages:

$$\mu_g = \frac{1}{G}\sum_{j=1}^{G} r_j,$$

$$\sigma_g = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(r_j - \mu_g)^2 + \epsilon},$$

$$A_i = \frac{r_i - \mu_g}{\sigma_g}.$$

The policy is then updated with a PPO-style objective and KL regularization against a fixed reference policy.

Results. Overall 47.8, AIME24 50.0, AIME25 26.7, AMC 84.3, MATH 72.4, Minerva 24.6, OlympiadBench 35.4.

C.2 Main Experiment: Representative Improving Algorithms

This subset focuses on representative improving and regressive cases needed for interpreting the main search results. AV-GRPO and VM-AV-GRPO belong to **Core Signal Modeling**; FA-GRPO and DFR-GRPO belong to **Failure-side Control and Failure Reshaping**; and BN-GRPO serves as a representative **Stabilization and Constraint Control** case. MSA-GRPO is included as an additional high-performing variant. The **Signal Routing and Conditional Normalization** family is analyzed in more detail in Section E.

AV-GRPO

Motivation. The preceding analysis suggests that batch-level scaling in BN-GRPO improves stability but can dilute sparse hard-task success signals.

Method. AV-GRPO replaces empirical scaling with analytic-variance scaling and stricter format constraints:

$$\mu_g = \frac{1}{G}\sum_i R_i,$$

$$\mu_{2,g} = \frac{1}{G}\sum_i R_i^2,$$

$$\sigma_{\text{analytic}} = \sqrt{\mu_{2,g} - \mu_g^2},$$

$$A_i = \frac{R_i - \mu_g}{\max(\sigma_{\text{analytic}}, \sigma_{\min})}.$$

For binary rewards, this reduces to $\sigma = \sqrt{\mu(1-\mu)}$, providing a more adaptive scale in rare-success regimes. This scaling-centered redesign is paired with a strict format penalty and a lower entropy coefficient, so the empirical gain should be interpreted as the effect of a coordinated design rather than a single isolated substitution.

Results. Overall 50.9, AIME24 56.7, AIME25 36.7, with clear gains over GRPO.

VM-AV-GRPO

Motivation. The analysis of AV-GRPO identifies a residual mismatch: mixing invalid and valid samples in shared statistics can create false-positive incentives for valid-but-wrong outputs.

Method. VM-AV-GRPO applies validity masking with Bayesian-smoothed valid-subset statistics, computed only on the format-valid subset, together with an explicit floor penalty for invalid outputs:

$$\mu_{\text{valid}} = \frac{\sum_{i\in V} R_i + \alpha}{|V| + \alpha + \beta},$$

$$\sigma_{\text{valid}} = \sqrt{\mu_{\text{valid}}(1 - \mu_{\text{valid}})}.$$

The resulting advantage is

$$A_i = \begin{cases} \dfrac{R_i - \mu_{\text{valid}}}{\max(\sigma_{\text{valid}}, \sigma_{\text{floor}})}, & i \in V, \\ A_{\text{floor}}, & i \notin V. \end{cases}$$

It further applies validity-gated length normalization to negative valid advantages:

$$L_{\text{rel}} = \text{clip}\left(\frac{L_i}{\bar{L}_{\text{valid}}}, 0.5, 2.0\right),$$

$$A_i^{\text{final}} = \begin{cases} A_i / L_{\text{rel}}, & i \in V \wedge A_i < 0, \\ A_i, & \text{otherwise}. \end{cases}$$

Results. Overall 52.5, AIME25 43.3, AMC 88.0; higher than AV-GRPO on AMC (81.9).

MSA-GRPO

Motivation. MSA-GRPO separates high-information mixed groups from low-information uniform groups, since they should not drive updates with the same magnitude.

Method. MSA-GRPO introduces regime-aware anchor selection and scaling, where mixed groups contain both correct and incorrect samples, while uniform groups contain a single outcome:

$$\sigma_{\text{anchor}} = \begin{cases} \sigma_{\text{outcome}}, & \sigma_{\text{outcome}} > \epsilon, \\ \sigma_{\text{total}}, & \text{otherwise}. \end{cases}$$

$$A_{\text{scaled}} = A_{\text{raw}} \cdot \begin{cases} 1.0, & \text{Mixed}, \\ \alpha_{\text{uniform}}, & \text{Uniform}. \end{cases}$$

It then applies multiplicative length scaling:

$$\text{ratio}_i = \frac{L_i}{\bar{L}_{\text{group}}},$$

$$A_{\text{final}} = \begin{cases} A_{\text{scaled}} \cdot \text{ratio}_i^{\lambda}, & A_{\text{scaled}} > 0, \\ A_{\text{scaled}} \cdot \text{ratio}_i^{-\lambda}, & A_{\text{scaled}} < 0. \end{cases}$$

Results. Overall 50.7, AIME25 43.3.

FA-GRPO

Motivation. FA-GRPO targets all-fail stagnation under standard GRPO by adding explicit signal shaping in all-fail groups.

Method. FA-GRPO uses a two-regime advantage:

$$A_{\text{final}} = \begin{cases} \text{clip}\left(\frac{R_i - \mu_g}{\sigma_g + \epsilon}, -3, 3\right), & R_{\max} > 0, \\ -2.0, & \text{Inv}, \\ -0.5, & \text{Wrong}. \end{cases}$$

Here, Inv denotes invalid-format outputs and Wrong denotes valid but incorrect outputs. This design separates structural errors from reasoning failures through different penalty magnitudes.

Results. Overall 49.9, AIME25 36.7.

DFR-GRPO

Motivation. Training-dynamics analysis of FA-GRPO suggests that fixed penalties can destabilize formatting when mixed directly with normalized reasoning advantages, and that uniform negative penalties in all-fail groups create directionless repulsion rather than directional correction.

Method. DFR-GRPO decouples format control and reasoning quality into independent advantage streams:

$$A_i^{\text{total}} = A_i^{\text{reason}} + \lambda_{\text{fmt}} A_i^{\text{format}},$$

where A_i^{format} tracks format compliance separately from reasoning quality. In all-fail groups, the reasoning stream no longer uses a uniform constant penalty; instead, it assigns a zero-centered signal derived from token entropy, so that more informative failures are compared with less informative ones rather than being pushed equally. This design changes failure-side updates from undirected repulsion toward directional comparison among failures.

Results. Overall 49.0, AIME25 33.3, AMC 85.5. Relative to FA-GRPO, this revision mainly regularizes the failure-side signal path, while incurring trade-offs on some hard benchmarks.

C.3 Main Experiment: Stabilization and Regression Cases

BN-GRPO

Motivation. BN-GRPO aims to suppress noise amplification through group centering and batch scaling, while moving KL and entropy terms into the loss.

Method.

$$A_{\text{raw},ij} = R_{ij} - \mu_i,$$

$$\sigma_{\text{batch}} = \sqrt{\frac{1}{B} \sum_{i,j} A_{\text{raw},ij}^2},$$

$$A_{\text{final},ij} = \text{clip}\left(\frac{A_{\text{raw},ij}}{\sigma_{\text{batch}} + \epsilon}, -3, 3\right).$$

$$L(\theta) = \mathbb{E}\left[L^{\text{CLIP}}(\theta) - \beta_{\text{KL}} D_{\text{KL}} + \beta_{\text{ent}} H(\pi)\right].$$

Results. Overall 44.2, AIME25 26.7. In this stage, the stabilization objective does not translate into hard-task gains.

SA-GRPO

Motivation. SA-GRPO uses fixed scaling and a dual-anchor baseline to avoid extreme advantages in low-variance groups.

Method.

$$\mu_{\text{anchor}} = \alpha \mu_{\text{group}} + (1 - \alpha) \mu_{\text{batch}},$$

$$A_{ij} = \frac{R_{ij} - \mu_{\text{anchor}}}{\sigma_{\text{fixed}}},$$

with decoupled regularization in the loss:

$$L_{\text{total}} = L_{\text{CLIP}} - \beta_{\text{KL}} D_{\text{KL}} + \gamma_{\text{Ent}} H(\pi).$$

Results. Overall 34.5, AIME25 13.3, representing a regression case.

C.4 Response-Length Compression Experiment: Success and Failure

DACE-GRPO (Success)

Motivation. DACE-GRPO aims to compress response length while preserving hard-task correctness, avoiding "compress-first, regress-later" behavior.

Method. Difficulty-aware reweighting is applied via group pass rate:

$$\text{PassRate}_i = \frac{1}{G} \sum_{j=1}^{G} \mathbb{I}(r_{ij} > 0.5).$$

$$A_{\text{acc}}^{\text{new}} = A_{\text{acc}}\big(1 + \alpha(1 - \text{PassRate}_i)\big),$$
$$A_{\text{eff}}^{\text{new}} = A_{\text{eff}} \cdot \text{PassRate}_i.$$

$$A_{\text{total}} = A_{\text{acc}}^{\text{new}} + \beta A_{\text{eff}}^{\text{new}} \cdot \mathbb{I}(\text{Correct}_i).$$

Here, A_{eff} is computed from z-scored negative response lengths within the correct subset, so the compression signal only acts among correct samples and is further gated by group pass rate. An annealed entropy target $H_{\text{target}}(t)$ supports exploration early and consolidation later.

Results. Overall 51.7 (+3.9), mean length 473.6 → 335.7 words (ratio 0.709); AIME25 40.0, AMC 86.7.

CAG-GRPO (Success)

Motivation. CAG-GRPO aims to prevent length collapse driven by disproportionate gradients from overly short solutions by reformulating efficiency as a cost term rather than a positive bonus.

Method.

$$A_{\text{eff}} = \min\left(\frac{\mu_L - L_i}{\mu_L}, 0\right),$$
$$\beta = \beta_{\text{base}} \cdot SR^2,$$
$$A_{\text{total}} = A_{\text{acc}} + \beta A_{\text{eff}} \cdot \mathbb{I}(\text{Correct}_i).$$

Here, μ_L is computed over the correct subset within a prompt group. The non-positive efficiency adjustment removes explicit bonuses for being shorter than the mean, reducing the risk of over-compression.

Results. Overall 49.5 (+1.6), mean length ratio 0.854; AIME25 40.0, AMC 84.3.

DCBE-GRPO (Failure)

Motivation. DCBE-GRPO introduces a relative-length-improvement efficiency term and difficulty gating, aiming for stable compression without accuracy collapse.

Method.

$$RLI_i = \frac{\mu_L - L_i}{\mu_L + \epsilon},$$
$$\alpha_{\text{diff}} = \left(\frac{N_{\text{correct}}}{N_{\text{total}}}\right)^2.$$

$$A_{\text{total}} = A_{\text{acc}} + \lambda\, \alpha_{\text{diff}}\, RLI_i \cdot \mathbb{I}(\text{Correct}_i).$$

Here, μ_L denotes the mean length computed over the correct subset within a prompt group. In the implemented recipe, this efficiency path is paired with a dynamic entropy controller.

Results. Overall 27.8 (−20.0), mean length ratio 1.333, with increased rather than reduced length; AIME25 6.7, AMC 45.8. This serves as a negative case for the combined design, where the RLI-style efficiency term alone does not explain the observed failure.

C.5 Summary

Across these representative cases, improvements are associated with finer signal hierarchy design, such as validity separation and difficulty-aware reweighting, rather than stronger penalties alone. Failure cases indicate that stabilization or efficiency objectives can degrade hard reasoning when they are not coordinated with task difficulty and signal structure.

D Case Study of the Search Workflow

To illustrate how the framework accumulates and reuses evidence, we present a detailed trace of one lineage, visualized in Figure 5 and grounded in the full held-out results in Table 2. This case study shows how stronger variants emerge under a fixed protocol, and how failures, revisions, and follow-up hypotheses are linked across iterations.

The selected lineage follows the path: GRPO → RF-GRPO → BN-GRPO → AV-GRPO → VM-AV-GRPO. This path captures two informative phases of the search process. In the mechanism-family grouping used in Section 4.2, it begins with **Stabilization and Constraint Control** (RF-GRPO, BN-GRPO) and culminates in **Core Signal Modeling** (AV-GRPO, VM-AV-GRPO):

1. **Performance gain.** Starting from insufficient use of failure-group signals and unstable sparse-success amplification, the lineage passes through stabilization attempts and a regression before reaching analytic-variance scaling, which is associated with a clear performance gain.

2. **Follow-up correction.** After that gain, the analysis identifies a residual false-positive incentive on valid-but-wrong samples, which motivates validity masking, validity-gated length normalization, and adaptive entropy control.

Root: GRPO Baseline

Status: Baseline (Overall: 47.8, AIME25: 26.7%)

The lineage begins with standard GRPO. While GRPO is competitive across the broader benchmark

suite, the analysis identifies a bottleneck on sparse-success benchmarks such as AIME25.

- **Observed issue:** Under sparse success, the training signal faces two structural challenges. First, when only a few samples succeed in a group, the within-group variance can be small, inflating advantage scale and inducing update shocks or distribution collapse. Second, in all-wrong groups, rewards provide little discriminative relative structure, yielding weak directional learning signals. These issues make it difficult to balance exploration with the consolidation of rare successful trajectories on hard prompts.

Attempt 1: Reference Fallback and Damped Normalization (RF-GRPO)

Status: Targeted gain (Overall: 48.1, AIME25: 26.7% → 33.3%)

RF-GRPO targets sparse-success shocks and failure-mode lock-in.

- **Mechanism:** It introduces damped advantage normalization to cap outlier updates, performance-gated KL fallback to the reference policy for all-fail groups, and hierarchical format rewards to provide denser structural feedback.
- **Outcome:** AIME25 improves over the baseline, but the overall gain remains limited, and failure-group noise can still interfere with learning on the hardest problems.
- **Analysis:** In this setting, the stabilizing controls improve training stability, but a remaining issue persists: if failure-group noise is normalized without distinction, signal contamination can still hinder hard-task learning.

Stabilization Attempt: Batch-Level Scaling (BN-GRPO)

Status: Stable but regressive (Overall: 44.2, AIME25: 26.7%)

BN-GRPO further attempts to reduce failure-noise contamination by decoupling group centering and batch-level scaling, while moving KL and entropy regularization explicitly into the optimization objective.

- **Outcome:** Although motivated by stability, evaluation shows a regression in aggregate performance and no improvement over the baseline on AIME25.
- **Analysis:** Stabilization alone is not equivalent to hard-task progress. If the advantage scale is not aligned with success rarity, suppressing drift can also attenuate the positive signal needed for rare-success consolidation.

Main Gain: Analytic-Variance Scaling (AV-GRPO)

Status: Clear gain (Overall: 50.9, AIME25: 26.7% → **36.7%**)

Based on the preceding failure and stability analysis, AV-GRPO replaces empirical-scale heuristics with analytic-variance scaling for advantage normalization.

- **Mechanism 1: Analytic-variance advantage scaling.** Advantages are normalized by an analytic variance term, so that rarer successes can produce stronger positive updates.
- **Mechanism 2: Stricter format constraints and conservative entropy regularization.** These changes reduce invalid exploration noise while preserving effective sparse-signal learning.
- **Evidence:** This iteration yields a clear jump on both aggregate and hard-task metrics, including an AIME24 score of 56.7, consistent with the view that calibrated rare-signal amplification helps difficult reasoning.

Follow-up Refinement: Validity-Masked Correction (VM-AV-GRPO)

Status: Refinement gain (Overall: 52.5, AIME25: 36.7% → 43.3%)

After the AV-GRPO gain, the analysis identifies a residual mismatch: when invalid and valid samples enter the same statistics, valid-but-wrong responses can be over-rewarded.

- **Mechanism:** VM-AV-GRPO applies validity masking with Bayesian-smoothed valid-subset statistics, assigns a floor penalty to invalid outputs, and combines validity-gated length normalization with adaptive entropy control.
- **Outcome:** The update improves Overall and AIME25 while also yielding a concurrent gain on AMC (AV-GRPO: 81.9 → VM-AV-GRPO: 88.0).
- **Analysis:** This pattern is consistent with the view that separating structural validity from reasoning quality produces a cleaner sparse-reward signal hierarchy. The step links an empirical gain to a residual mismatch and a targeted mechanism revision, rather than treating the first improvement as the end of the search.

E Detailed Search Trajectories and Evidence Accumulation

This section expands the concise analysis in Section 5 by making the intermediate turning points, failed hypotheses, and cross-lineage transfer patterns more explicit.

E.1 Retrospective Evidence on Parent Retention

Across the 21 comparable parent-selection rounds in the base branch, three parents are retained in each round, yielding 63 retained-parent decisions in total. These retained sets cover about 1.8 top-level branches on average, and 14 rounds span at least two branches.

A parent's score at selection time and its later contribution are not equivalent. Among the 62 retained-parent decisions whose descendants can be observed later in the lineage, 43 yield a descendant with higher Overall than the selected parent itself, with a median gain of $+2.6$ Overall; 27 improve by more than 3 points, and 12 improve by more than 5 points. In particular, 27 of the 33 selections whose parent is below the GRPO baseline eventually lead to descendants that surpass that baseline.

The evidence becomes sharper when we compare parents selected in the same round. In more than half of the comparable rounds ($11/21$), the parent that eventually produces the strongest descendant is not the selected parent with the highest Overall score at selection time; across all comparable within-round parent pairs, $26/50$ reverse order between Overall score at selection and eventual best-descendant score. One representative round is as follows (numbers in parentheses denote Overall scores): ZA-GRPO (47.3), BN-GRPO (44.2), and Anchor-GRPO (45.2) are retained together; the strongest later descendant is VM-AV-GRPO (52.5) from BN-GRPO, whereas the ZA-GRPO branch peaks at 48.4. Another representative round is as follows: LAV-GRPO (46.5), GA2-GRPO (49.2), and RA-GRPO (44.4) are retained together; here the selected parent with the lowest Overall score, RA-GRPO, later reaches MSA-GRPO (50.7), exceeding the GA2-GRPO branch peak of 48.2. Representative cases are listed in Table 3. These observations support retaining parents for future improvement rather than filtering solely by current Overall score; they constitute retrospective rather than controlled causal evidence.

Table 3: Representative selected parents whose strongest descendants substantially outperform them. Scores are Overall. Several cases arise in rounds where a lower-scoring selected parent later outperforms the branch of an initially stronger selected parent.

Parent	Overall	Best descendant	Gain
BN-GRPO	44.2	VM-AV-GRPO (52.5)	+8.3
Anchor-GRPO	45.2	LNA-GRPO (50.6)	+5.4
RA-GRPO	44.4	MSA-GRPO (50.7)	+6.3
GLA-GRPO	47.0	PR-GRPO (49.4)	+2.4
CFA-GRPO	46.3	OCE-GRPO (48.4)	+2.1

E.2 Search Dynamics: Exploration and Refinement in Mechanism Space

The search trajectory reveals a stepwise refinement process: each generation typically introduces a localized mechanism revision, or a compact bundle of coordinated changes, on top of its parent and is then filtered by the evaluation protocol. To explain these turning points, we refer to three core training signals: (i) **advantage**, which governs the relative weight of rare successes versus failures in updates; (ii) **KL divergence**, which constrains deviation from a reference model and controls update size; and (iii) **entropy**, which helps diagnose premature collapse or excessive drift.

The traces indicate that performance jumps often coincide with restructuring and decoupling learning-signal pathways rather than simply increasing a single signal magnitude, together with recalibrating penalties on failures and stability constraints. We trace three representative lineages that align with three of the four mechanism families in the main text; the remaining family, **Stabilization and Constraint Control**, appears as an important precursor within the first lineage.

(i) Core Signal Modeling with Stabilization and Constraint-Control Precursors. GRPO (Overall 47.8, AIME25 26.7) $\rightarrow$ RF-GRPO (Overall 48.1, AIME25 33.3) $\rightarrow$ BN-GRPO (Overall 44.2, AIME25 26.7) $\rightarrow$ AV-GRPO (Overall 50.9, AIME25 36.7) $\rightarrow$ VM-AV-GRPO (Overall 52.5, AIME25 43.3). **Main transition:** This lineage refines how sparse successes are safely amplified. The GRPO baseline suffers from gradient shocks associated with rare successes and provides no usable relative signal in all-fail groups. RF-GRPO introduces stabilizing controls: damped normalization caps outlier advantages, dynamic reference fallback mitigates deterministic failure modes, and hierarchical rewards provide denser format feed-

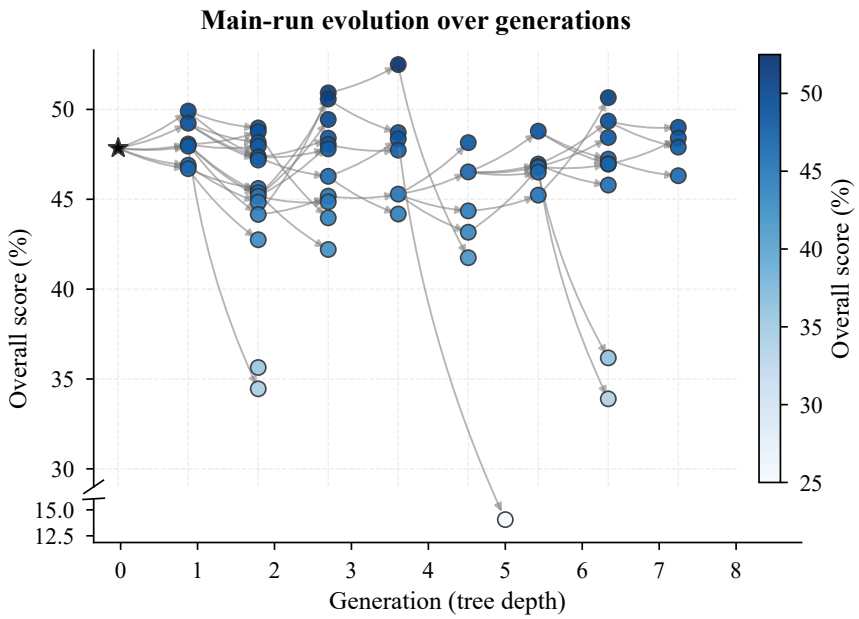

Figure 6: Complete view of the full base-branch lineage by tree depth. Nodes are plotted by tree depth and Overall score, and arrows indicate parent-child relations. Relative to the summary frontier view in Figure 4, this figure makes the underlying structure explicit: deeper generations still contain exploratory failures, and several strong descendants emerge only after intermediate regressions or branch-specific refinements.

back. BN-GRPO further exposes a side effect of per-group whitening: failure groups dominated by format noise can be amplified to the same magnitude as success groups, diluting the rare-success signal needed for hard tasks. To address this bottleneck, AV-GRPO shifts to explicit rare-event modeling by anchoring normalization to analytic variance so that rarer successes can yield stronger positive updates. VM-AV-GRPO then addresses a false-positive failure mode: when invalid samples enter the statistics, valid-but-wrong outputs can be assigned positive advantages. It restores the strict gradient hierarchy (invalid $\ll$ valid-wrong $<$ valid-correct) via validity masking with valid-subset statistics, combining this with validity-gated length normalization and an adaptive entropy controller to further improve AIME25 and Overall, while still navigating trade-offs on some hard metrics.

(ii) Failure-side Control and Failure Reshaping. GRPO (Overall 47.8, AIME25 26.7) $\rightarrow$ FA-GRPO (Overall 49.9, AIME25 36.7) $\rightarrow$ DFR-GRPO (Overall 49.0, AIME25 33.3). **Main transition:** This lineage targets all-fail stagnation and related failure modes. FA-GRPO attempts to break the zero-gradient issue by branching on group success: it assigns hierarchical fixed penalties in all-fail groups, with milder penalties for logic errors and stronger penalties for format errors. Subsequent evidence reveals two pitfalls of fixed penalties: mixing scalars with normalized advantages causes scale mismatch that destabilizes formatting, and applying uniform negative advantages acts as "directionless repulsion," often triggering drift and verbose failures. DFR-GRPO addresses this by separating the signals: it decouples format and reasoning into two independent advantage streams, and replaces uniform repulsion with a zero-centered signal derived from token entropy. This signal compares more informative failures against less informative ones, so the policy receives a directional gradient even when all candidates are incorrect, providing more robust learning signals in all-fail groups, albeit with trade-offs on the hardest benchmarks.

(iii) Signal Routing and Conditional Normalization. GRPO (Overall 47.8, AIME25 26.7) $\rightarrow$ GLA-GRPO (Overall 47.0, AIME25 33.3) $\rightarrow$ PR-GRPO (Overall 49.4, AIME25 36.7) $\rightarrow$ CDA-GRPO (Overall 49.0, AIME25 30.0). **Main transition:** This lineage illustrates the pitfalls of scalarizing multiple signals too early. While GLA-GRPO improves stability, it leaves all-fail groups with little usable learning signal. PR-GRPO makes routing explicit by partitioning groups into solved, failed, and mixed regimes: it silences mastered

groups, penalizes all-fail groups, and normalizes only mixed-group residuals to improve signal fidelity. CDA-GRPO addresses a complementary bottleneck: aggregating outcome, format, and efficiency into a single scalar can cause interference, and hard-silencing solved groups creates an efficiency "deadzone." To address this, CDA-GRPO splits advantages into independent streams and conditionally normalizes efficiency only within the correct subset: first be correct, then be short. It replaces brittle regime cliffs with a continuous optimistic baseline, which helps suppress short-and-wrong tendencies caused by signal conflict. This supports a core heuristic: secondary objectives should be introduced through signal decoupling and conditional normalization so that they complement, rather than dilute, the primary correctness signal.

E.3 Evidence Accumulation and Cross-Lineage Transfer

Based on standardized evaluation across all 64 nodes, we view each iteration as a mechanism-level experiment: each candidate is associated with an anticipated bottleneck and a corrective hypothesis, while evaluation results and key training signals, including advantage, KL, and entropy, provide feedback for later revisions. This feedback does not remain isolated within a single path. In later iterations, negative outcomes help mark practical design limits, while components that repeatedly correlate with gains are reused in new combinations. We summarize three primary modes of cross-lineage knowledge transfer.

Negative-evidence-driven path pruning. Negative examples are often fed back into later proposals through analysis summaries of failed or regressive designs, such as the FA→DFR progression in Section E.2. FA-GRPO attempts to inject gradients into all-fail groups through fixed scalar penalties, but the later analysis indicates that mixing fixed penalties with normalized advantages causes scale mismatch, and applying identical negative signals to all failures lacks directionality and can induce policy drift. Consequently, later designs tend to decouple format control from reasoning quality into comparable independent streams and introduce zero-centered token-entropy signals in all-fail groups to reduce directionless repulsion.

Horizontal diffusion of stability mechanisms. Stability components frequently diffuse across paths. For example, advantage clipping and explicit KL regularization are reused across multiple representative paths, including BN-GRPO, DFR-GRPO, and later variants, to suppress excessively large updates when amplifying rare successes. Later variants incorporate these regularizers into the loss function and pair them with clipping to control drift, shifting stability control from a single global hyperparameter to more fine-grained local constraints. This reuse reduces the overhead of exploring new algorithmic combinations, allowing more of the search budget to focus on signal-structure changes rather than repeated recovery from unstable training.

Recurring empirical regularities. As the search proceeds, recurring heuristics across branches gradually consolidate into empirical rules. First, in success-scarce regimes, penalties on failures should generally avoid coupling with length to prevent a bias toward short but uninformative answers. Second, secondary objectives are better introduced via signal decoupling and conditional normalization so that the primary signal does not overwhelm finer-grained objectives. VM-AV-GRPO's fix for false-positive advantages on valid-but-wrong samples and CDA-GRPO's first-be-correct-then-be-short conditional logic follow the same pattern: establish comparable, routable learning signals first, then introduce secondary preferences under controlled conditions.

The lineage structure can therefore be viewed as a process of evidence accumulation. Negative examples mark exploration boundaries, component reuse lowers exploratory overhead, and recurring regularities give later proposals a stronger starting point. This helps improve search efficiency and the quality of algorithm discovery under a fixed computational budget.

F Training Dynamics under Length-Compression Constraints

Table 4 summarizes all 11 nodes in the length-compression branch: the GRPO baseline and 10 generated variants. For readability, the table reports four representative dataset columns explicitly, while Overall and mean length are computed over the full six-dataset evaluation suite. Mean length is the unweighted mean word count across the six evaluation datasets.

Figure 7 shows the training dynamics of representative successful variants under the length-

Algorithm	AIME24 pass@32	AIME25 pass@32	AMC pass@32	MATH acc@1	Overall	Mean Length	Length Ratio
GRPO	50.0	26.7	84.3	72.4	47.8	473.6	1.000
DACE-GRPO	56.7	40.0	86.7	72.2	51.7	335.7	0.709
CAG-GRPO	46.7	40.0	84.3	73.0	49.5	404.4	0.854
DRE-GRPO	56.7	36.7	80.7	69.6	49.4	318.0	0.671
PR-GRPO	53.3	36.7	80.7	71.2	49.4	532.5	1.124
PAS-GRPO	50.0	30.0	85.5	73.4	48.5	367.7	0.776
DCE-GRPO	43.3	33.3	80.7	72.4	47.4	484.0	1.022
DQG-GRPO	43.3	30.0	80.7	72.0	46.4	325.5	0.687
MCE-GRPO	43.3	33.3	80.7	65.8	45.2	239.5	0.506
CAS-GRPO	43.3	23.3	75.9	69.6	43.1	1092.3	2.306
DCBE-GRPO	10.0	6.7	45.8	65.2	27.8	631.2	1.333

Table 4: Quantitative summary of the length-compression branch. The table includes the GRPO baseline and 10 generated variants. Four representative datasets are shown explicitly; Overall and mean length are computed over the full six-dataset evaluation suite. Length ratio is computed relative to the GRPO baseline.

compression constraint. Together with the failure cases summarized in Section C, these traces are consistent with the view that effective compression requires controlled length reduction while maintaining comparatively stable entropy and reward trajectories, rather than applying an indiscriminately stronger length penalty.

Figure 7: Training dynamics under the length-compression constraint for representative successful variants. The plots compare entropy, reward, and response length over training.