AUDITING VERIFIER CANCELLATION IN GRPO WITH MIXED REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Group-relative policy optimization (GRPO) subtracts each sampled group's mean reward. A binary verifier therefore contributes nothing to the advantages when every completion fails. A varying task reward can preserve nonzero advantages in the same group. We audit this known cancellation identity and examine its relationship to reward weighting and training recovery. Numerical replay reproduces task-only advantages in all 300 of 300 sampled all-violating groups after a constant verdict penalty is added, up to numerical rounding. We train a 0.5B policy on two fixed synthetic constrained tasks and compare reward designs and four remedy components at matched sampling budgets. The main comparisons omit KL regularization. Across the tasks' hard levels, a verdict below its crossover weight reaches final feasibility on 2 of 8 seeds. Task-only rewards reach it on 0 of 8 seeds. A verdict far above the crossover reaches it on 6 of 8 seeds. A graded residual, whose penalty also exceeds the crossover, reaches it on 7 of 8 seeds. A residual whose first-violation penalty equals the light verdict's reaches it on 7 of 8 seeds. That residual still orders every mixed group, and its penalty exceeds the light verdict's from the second violation unit. The stronger verdict acts through mixed groups that contain a feasible completion. Across the capacity task's hard-level reward and remedy arms, 23 of 24 feasible runs produce only parsed empty selections and yield no task value. We classify remedies by their inputs and show that filtering identical rewards retains every all-violating group with reward variation. Below the crossover on the speed task, the verifier filter and NGRPO with a feasible anchor reach final feasibility on 1 of 3 and 0 of 3 seeds. The identical-reward filter and Dr. GRPO reach it on 0 of 3 and 0 of 3 seeds. This is the predicted ordering, but it depends on that 1 verifier-filter seed. The remedy comparisons do not establish a stable efficacy ranking, and 0 of 50 capacity runs at weights below and above the crossover have every output useful and feasible. The main experiments reuse training prompts for evaluation and test components within a simplified training loop. Our findings support reporting constraint satisfaction, task value, and output validity together. Acting on the cancelled groups does not settle whether training recovers.

1 INTRODUCTION

Constrained reinforcement learning often combines a task reward with a verifier penalty. Group-relative policy optimization (GRPO) subtracts the sampled group's mean reward before updating the policy. A verifier term that is constant within the group therefore cancels, even when another reward component varies. We ask how this loss of verifier information relates to penalty weight, remedy coverage, and useful recovery during training.

The cancellation identity is known. Binary-reward studies describe zero advantages in groups whose members all fail (Nie et al., 2026). Wang (2026) studies mixed rewards in all-failing groups, where a common return disappears and dense shaping remains active. Our contribution is a controlled audit that separates fixed-group cancellation, mixed-group reward ordering, and observed training outcomes. We first check the reward components on fixed groups. Adding a verdict to the task term reproduces the task-only advantages in all 300 of 300 sampled all-violating groups. A residual that varies with violation severity can change those advantages, depending on its scale. A residual matched to the verdict's first-violation penalty reaches feasibility on 7 of 8 hard-level seeds. We then compare reward designs on a speed task and a capacity task. Across their hard levels, a

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

verdict far above its crossover weight reaches final feasibility on 6 of 8 seeds. A verdict below that weight reaches it on 2 of 8. The weight changes ordering in mixed groups but cannot restore the verdict within an all-failing group.

Finally, we classify remedies by the information they use and test selected components under a shared training loop. Every run without a remedy or KL penalty logs between 877 and 1,794 identical-reward groups out of 1,800 groups. Successful hard-level speed runs without a remedy have at least 85 times fewer all-violating groups than failed runs. Successful and failed capacity runs overlap on that count. The speed-task remedy ordering depends on one verifier-filter seed, so the comparisons do not establish a stable efficacy ranking.

The capacity results also separate feasibility from useful recovery. At the hard level, 23 of 24 feasible runs produce only parsed empty selections. We therefore assess constraint satisfaction together with task value and parsing behaviour. The study characterizes a reward-design mechanism on fixed synthetic tasks, without establishing generalization or a superior training method. Figure 1 summarizes the audit.

2 Background and related work

Group-relative policy optimization (Shao et al., 2024) samples K completions for a prompt, scores each, and turns the scores into advantages by subtracting the group mean and dividing by the group's own spread,

$$A_k = \frac{R_k - \mu_K}{\sigma_K + \epsilon}. \tag{1}$$

Here μ_K is the mean of the group's rewards, σ_K is their sample standard deviation with $K-1$ in the denominator, and $\epsilon = 10^{-6}$. Two established properties of Eq. (1) matter here. Subtracting μ_K removes any term that takes one value across the group. The magnitude $|A_k|$ cannot exceed $(K-1)/\sqrt{K}$, which is 1.5 at $K = 4$ (Shiffler, 1988).

A zero advantage removes a group's policy-gradient contribution. Optimizer state or a separate KL penalty can still move the parameters.

Homogeneous rewards and normalization: When a verifiable reward is the only reward, a group in which every completion fails gives every completion the same score, so its gradient is zero. Salmani-Zarchi et al. (2026) describe three failure modes of this normalization that motivate their method. Under zero-variance collapse, every completion in a group satisfies the same subset of constraints. Under mean-centering blindness, normalization discards how well a group did in absolute terms. Under low-variance amplification, a small but nonzero spread turns a negligible reward difference into a large advantage. Nie et al. (2026) measure 54.75 percent all-failing and 14.50 percent all-passing groups at $K = 4$ in one logged training run, and prove the rate exceeds an independence estimate. Garg et al. (2025) observe that a failed completion can still draw a positive advantage by exceeding a low group average.

Remedies: The published responses change which groups enter the update, what the advantage is measured against, its divisor, the sampler, or the reward. Section 6 sorts them by their inputs, because a remedy's input determines which groups it can act on. Normalization also helps. Ge et al. (2026) show that, under mild conditions, dividing by the group's spread implements an adaptive gradient with a better convergence rate than unnormalized REINFORCE, and Bay & Yearick (2026) characterize what that choice controls.

Mixed rewards and the scope of this study: Published constrained-search recipes combine reward components. Li et al. (2026) add format and feasibility terms, Liu et al. (2025a) distinguish malformed from well-formed wrong answers, and Cai et al. (2026) combine validity with a continuous score. Wang (2026) explicitly centres away a common return in all-failing groups and studies how normalization amplifies the remaining shaping signal. This study audits those components, the mixed-group crossover weight and the limits of remedy coverage in controlled tasks.

Relation to critic-based methods: Proximal policy optimization (PPO) estimates advantages with a learned value function (Schulman et al., 2017). It does not impose subtraction of the current group's mean, so the same exact within-group cancellation need not hold. A critic can still give weak advantages when returns vary little. Wang (2026, App. K) reports a limited PPO comparison with generalized advantage estimation. We train PPO only in the strict protocol of Appendix D.2, where it parsed 0 of its 5760 held-out completions, so we cannot establish whether a critic improves recovery. Appendix E.2 extends this section with further related work, grouped by theme.

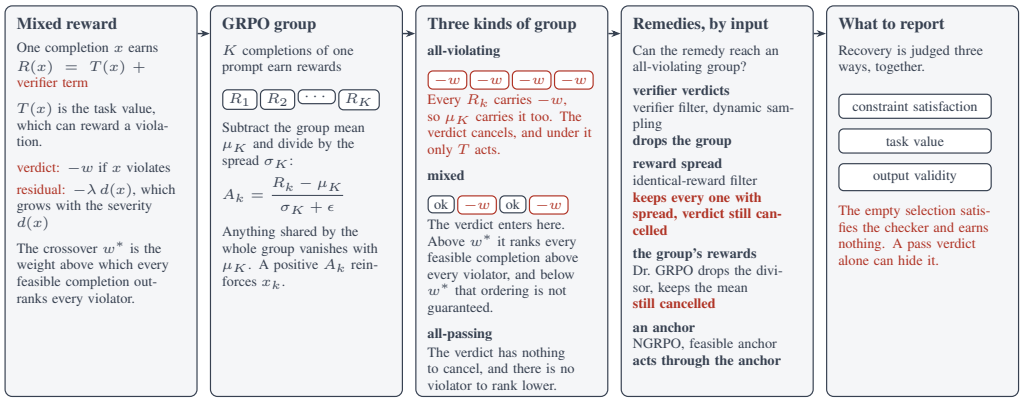

Figure 1: The five panels summarize the audit. GRPO subtracts each group's mean reward, so a constant verdict survives only in a group that holds both a feasible member and a violator, and it is guaranteed to rank every feasible completion above every violator only above the crossover weight. A remedy can restore the verdict inside an all-violating group only if its input reaches that group. The last panel lists the three quantities to report together, because an empty answer can satisfy the checker and earn nothing.

3 Two placements of one verifier

We use tasks whose reward components can be computed exactly. A completion names a speed s, and the verifier accepts it when $|s| \le v_{\max}$. Its task value is $T(s) = \min(|s|, 200)/10$. The verdict-plus-task reward is $R(s) = T(s) - w\,\mathbb{I}[|s| > v_{\max}]$. The residual-plus-task reward replaces the verdict penalty with $-w \cdot 100\, d(s)$. Here $d(s) = \max(|s|/v_{\max} - 1,\, 0)$ is the violation severity, which Appendix B.1 writes as r. Appendix B.1 defines the treatment of unparseable outputs. The later capacity task uses the same decomposition.

Constant-verdict groups: Every member of an all-violating group receives the same verdict penalty. Subtracting the group mean removes that penalty at every weight. The task-only and verdict-plus-task advantages are therefore identical on a fixed group, up to numerical rounding. The identity also holds in all-passing groups. A varying residual remains after centering, although its presence alone does not guarantee a useful update.

Mixed groups and the crossover weight: A verdict can affect an advantage when the group contains both a feasible member and a violator. Every feasible completion outranks every violating completion on every prompt exactly when the penalty exceeds

$$w^* = \max_{\text{prompts}} \Big(\max_{v \text{ violates}} T(v) - \min_{f \text{ feasible}} T(f) \Big). \tag{2}$$

The maxima exist because our task rewards are bounded, and an unbounded task reward need not admit a finite crossover. The crossover is 20 at both speed limits (Appendix A.4). The condition guarantees ordering on the fixed prompt set. It does not guarantee that training converges. At large weight the advantages approach those of the penalty alone (Appendix A.3). A distributional constraint objective would require a separate analysis.

Residual ordering depends on scale: A residual penalty $-\lambda d$ ranks a less-violating completion x above y exactly when $\lambda[d(y) - d(x)] > T(y) - T(x)$. Our speed-task residual satisfies that condition outside the envelope at our weights. The factor 100 inside it makes its slope 100 times the nominal weight, so at a matched weight the residual and verdict arms differ in penalty scale as well as in shape (Appendix B.1). Across 200,000 random groups of four speeds under the graded residual, a violator outranked a feasible completion 0 times in 97,786 mixed groups. Appendix A.2 separates this ordering from the positive advantage a violator can receive within its group.

Reward-component checks without training: A constant verdict reward gives every member of an all-violating group the same score and zero advantage. A varying task term preserves reward spread and nonzero advantages, with the constant verdict removed. Table 4 in Appendix A.1 counts both over 300 groups per condition, sampled from the second task of Section 5. A verdict-only

design has spread in 0 of the all-violating groups, and a residual-only design has spread in 300. A verdict added to a task term matches adding nothing in 300.

4 SPEED-TASK EVALUATION

The speed task tests how penalty weight affects recovery when the verifier cancels in all-failing groups. We train Qwen2.5-0.5B-Instruct (Qwen Team, 2024) as a full fine-tune under four reward designs at a matched sampling budget. The arms differ only in the penalty term and its weight. Everything else is shared, including the prompts, the group size $K = 4$, 6 groups per step, 300 steps, the learning rate, the sampler, the device type and 3 seeds, and Table 9 reports the shared settings. The objective carries no KL penalty in these main comparisons. Section 5 reports a separate KL ablation. Training and evaluation use the same 6 fixed speed prompts, with fresh evaluation samples from a fixed random seed. The experiment measures adaptation to those prompts. It does not measure held-out generalization.

The trainer sums token log-probabilities within each completion and takes at most one optimizer update per sampled batch. The importance ratio equals one before that update, so ratio clipping is inactive. We vary advantage construction and group filtering within this common loop. The remedy runs are component ablations. They do not reproduce the complete DAPO, Dr. GRPO or NGRPO recipes (Appendices B.2 and D.1).

Two regimes set how often a group contains no feasible member. At the hard limit $v_{\max} = 5$, a fraction 0.83 of groups start with none, and at the easy limit $v_{\max} = 30$, that fraction is 0.17. The light verdict at weight 5 sits *below* the crossover of 20, and the heavy verdict at 5,000 sits *above* it. The rest of the paper calls the light verdict the added verdict or the plain verdict, and it calls the task-only design the no-penalty arm. A plain verdict at another weight is stated with that weight.

Group diagnostics: Table 1 reports, per run, how many of the 1,800 sampled groups had no feasible member, how many gave every member the same reward, and how many were all-violating while still carrying reward spread. The first column spans a factor of 450 across the 12 runs. The second spans a factor of 1.02, from 1,763 to 1,794, so every design has nearly the same count.

Table 1: Group counts per run, out of 1,800 groups (300 steps by 6 groups), 3 seeds per arm, hard regime. The third column counts all-violating groups that still carry reward spread, where the task term acts with the verdict removed. The last counts groups holding both a feasible member and a violator, the only groups in which a verdict enters the advantage of Eq. (1).

Reward design	all-violating	zero-variance	all-violating, spread	mixed
residual + task	5, 4, 8	1,777, 1,777, 1,778	5, 4, 7	5, 8, 6
verdict + task	1,799, 1,797, 1,799	1,794, 1,791, 1,793	5, 6, 6	1, 3, 1
verdict $\times 1000$ + task	19, 4, 18	1,778, 1,787, 1,763	8, 4, 8	10, 9, 23
task only	1,799, 1,798, 1,799	1,794, 1,793, 1,793	5, 5, 6	1, 2, 1

A filter that discards groups whose members all earn the same reward would drop 1,777, 1,777, 1,778 of 1,800 groups per seed in the arm that converged to the feasible boundary. It would drop 1,794, 1,791, 1,793 in the arm that ended 5 orders of magnitude outside it. In that failing arm it would keep 6, 9, 7 groups. Of those, 5, 6, 6 are all-violating with spread, where the verdict cancels and the task term acts alone. Groups that hold both a feasible member and a violator number 1, 3, 1 per seed, and only those let the verdict enter the advantage.

The zero-variance count does order the arms, because every run that reaches feasibility has a lower zero-variance count than every run that does not, by at least 4 groups in the hard regime and 31 in the easy one. The zero-variance margin is small, and we did not validate a predictive threshold before observing the outcomes. On the all-violating count the same successful runs sit at least 94 times below the failed ones.

Training dynamics: For the added verdict in the hard regime, the all-violating groups with spread number 5 to 6 per run, and 14 of the 17 lie in the first step. Within each all-violating group with spread, the verdict cancels and the task term determines the group's advantages. The policy moves into the region where the task term is flat, and from then on every all-violating group has identical rewards and zero gradient.

Almost all of the few mixed groups these runs sample fall in the first steps. A verdict below the crossover does not guarantee that a feasible member outranks the violators in a mixed group.

In the speed task's hard-regime runs, the light verdict's rewards rank a violator above a feasible completion in 4 of its 5 logged mixed groups (Appendix B.1). The recorded runs do not recover, but the traces do not isolate the contribution of underweighting from cancellation. On seed 2 the added-verdict run and the no-penalty run sample identical answers at all 300 steps (Appendix A.1).

The outputs continue to change after the sampled rewards lose their variation. In the added-verdict run at seed 0 in the hard regime, the last group with any reward spread is in step 1, where the mean residual is 18. It reaches 58,228 at the next step and is within one percent of its run maximum of 192,139 by step 47. Every later policy gradient is zero, and the optimizer still takes a step each time. The continued output changes are consistent with retained optimizer momentum. The trace does not isolate that mechanism or measure the direction of parameter movement. Table 2 gives the outcome.

Table 2: Final evaluation in the hard regime, mean and standard deviation over 3 seeds. The reference log-ratio is measured at the last training step and summed over completion tokens, in nats per completion, so it scales with answer length (Appendix B.3). Task value saturates at 20 and is a mean over the completions that parsed.

Reward design	violation	residual	task value	unparseable	log-ratio
residual + task	0.000 ± 0.000	0.000 ± 0.000	0.50 ± 0.00	0.00 ± 0.00	7.19 ± 0.52
verdict + task	1.000 ± 0.000	$144{,}053 \pm 34{,}416$	20.00 ± 0.00	0.00 ± 0.00	20.86 ± 1.24
verdict $\times 1000$ + task	0.000 ± 0.000	0.000 ± 0.000	0.23 ± 0.25	0.00 ± 0.00	17.17 ± 4.02
task only	1.000 ± 0.000	$135{,}143 \pm 42{,}884$	20.00 ± 0.00	0.00 ± 0.00	20.87 ± 1.36

Task value after feasibility: Violation rate measures constraint satisfaction, while task value measures the quality of the parsed answers. The residual-plus-task arm finished at task values of $0.50, 0.50, 0.50$ across its seeds, which is the most any feasible answer can earn at this limit. The heavy-verdict-plus-task arm finished at $0.50, 0.00, 0.20$. The verdict records feasibility. The residual also records violation severity, at a penalty scale the verdict arms do not match (Appendix B.1).

Predictions: We wrote four predictions for this task in a local planning file before the first training run, and Appendix E.1 lists each with its outcome and quotes each as first written.

P1, exact cancellation in every all-violating group, held, with 0 failures in 21,687 groups.

P2 said the light and heavy verdicts would give identical advantages in every group and identical curves. It failed, because they differ in mixed groups and only the heavy verdict reaches feasibility.

P3 said only the residual would lower the mean residual and the violation rate, and it failed, because the heavy verdict reached zero violation by evaluation steps $10, 10, 10$.

P4 said the light verdict would lower the violation rate in the easy regime, and it failed, because that rate rose from 0.46 at step zero to 1.000 at the end. The light verdict sits *below* the crossover in both regimes, so on some prompts a violator can outrank a feasible member.

Replay consistency: The logged parsed groups determine every reward, including the fallback for unparseable outputs. We therefore recompute what each design would have scored on the groups a run sampled. Across 43,200 logged groups from the 24 speed-task runs at both limits, 21,687 were all-violating and 211 were mixed. The cancellation identity failed 0 times and the weight identity in constant-verdict groups failed 0 times. The largest disagreements were 1.11×10^{-11} and 1.11×10^{-11}. The recomputation also reproduces each run's own logged advantages exactly.

Reference-policy log-ratios at feasibility: At the hard limit, the residual-plus-task arm has a lower mean reference log-ratio at first feasibility than the heavy verdict. The two arms also differ in penalty scale, which this comparison does not control (Appendix B.1). At the easy limit, variation across residual seeds exceeds the difference between the means. The diagnostic uses nucleus-sampled completions and precedes the update tested by the evaluation, so it is not an unbiased KL estimate or a matched-state comparison. Appendix B.3 gives the values and measurement details, and Figure 2 in Appendix C.1 shows individual trajectories.

5 Capacity-task evaluation

The capacity task tests whether the same reward comparison yields useful feasible selections in a bounded answer space. The policy selects from 16 items with fixed values and weights. The selected weight must fit a capacity that the prompt does not reveal. The empty selection is therefore the one

answer that is feasible whatever the capacity is. The answer space contains 65,536 subsets, and selecting every item maximizes violation. Completions may contain 128 new tokens, compared with 6 on the speed task. Training and evaluation share the same 6 instances. We measure task value alongside feasibility because the empty selection earns no value.

The recorded parser treats text with no bracketed span as unparseable and maps a bracketed span with no valid index to the empty selection. We therefore use "empty selection" to mean the parsed selection throughout. The logs retain parsed selections, not raw completions, so strict format validity cannot be checked retrospectively (Appendix B.1).

Reward components: The verdict is whether the chosen weight fits. The residual is the excess weight as a fraction of the capacity. The task value is the chosen value divided by the exact optimum for that capacity, so a feasible answer earns at most one and an overfull answer can earn more, which rewards violating the constraint. An unparseable completion earns the task floor -10 plus the arm's penalty on a stand-in selection targeted to the group's worst residual. Its reward therefore depends on its siblings before centering. The discrete fallback can miss that target (Appendix B.1).

Constraint levels and reward ablations: The capacity is a fraction of each instance's total weight. At the hard level it is 0.03 and a fraction 0.97 of groups start with no feasible member, and at the easy level it is 0.2 and a fraction 0.29 start with none. We set both levels by sampling the untrained base policy. Two arms remove the task term.

Verdict only is the zero-signal case the published remedies are built for, and *residual only* separates what the residual contributes from what the task term contributes. The light verdict sits $below$ the hard-level crossover of 12.50, the largest across the 6 instances (Table 8). It sits $above$ the easy-level crossover of 2.54, the largest at that level. Its hard-level outcome therefore cannot separate cancellation from a weight too small to order a mixed group.

Final feasibility and exact cancellation: At the hard level the added verdict reaches feasibility on 2 of 5 seeds, and adding no penalty reaches it on 0. Each feasible verdict seed ends with only parsed empty selections, and each other verdict seed ends either violating on every parsed completion or answering almost nothing that parses.

The added verdict reaches zero violation at an evaluation on 1 of 5 seeds and ends above zero at the final evaluation. Adding no penalty does the same on 1 of 5 seeds. These runs return parsed empty selections for most of training, and they count as failures because feasibility is read at the final evaluation.

The heavy verdict sits above the crossover and reaches feasibility on 3 of 5 seeds, and the residual reaches it on 4. On the verdict plus task arm's seed 3 the verdict enters the update through only 1 of that run's 1,800 groups, because every other group holds no feasible member. Changing the verdict weight on those other recorded groups cannot alter their advantages. That replay isolates the fixed-group identity, and it does not establish what a new above-crossover training run on that seed would do. This seed's verdict plus task run fails as the no-penalty run does (Appendix C.2).

Table 3: Capacity task, hard level: for each reward design, the seeds that reach feasibility, the seeds that end with nothing parseable, and the median task value over the seeds that parse. Task value is the chosen value over the exact optimum, averaged over each seed's parsed answers, including infeasible ones. Table 14 gives every seed, and Table 17 gives the feasible value over all completions.

Reward design	feasible seeds	unparseable seeds	median task value
residual + task	4 of 5	1 of 5	0.00
verdict + task	2 of 5	1 of 5	0.50
verdict $\times 1000$ + task	3 of 5	1 of 5	0.00
task only	0 of 5	0 of 5	9.53
verdict only	4 of 5	1 of 5	0.00
residual only	5 of 5	0 of 5	0.00

Group diagnostics: Table 12 reports the capacity-task group counts. Counting groups with no feasible member does not separate the runs here. At the hard level the successful runs log up to 579 such groups, and a failed run logs as few as 30. The empty selection is feasible, so a run that returns it for most of training logs few such groups even when it ends infeasible.

Counting groups whose members all earn the same score does not separate feasible and infeasible runs. At the hard level the runs that reach feasibility span 1,399 to 1,793 and the runs that fail span 877 to 1,777, so the two ranges overlap.

Feasibility without useful task value: We predicted that the residual would finish at a higher task value than the heavy verdict at equal violation, and it does not. At the hard level every seed of either design that reaches feasibility answers with the empty selection and earns no task value. At the easy level the residual ends at task values of $0.52, 0.52, 0.52, –, 0.52$ per seed, and the heavy verdict ends at $0.52, 0.52, 0.36, 0.36, 0.52$. A dash marks a seed with nothing parseable.

In the observed empty-selection state, sampled rewards are equal and the policy-gradient term is zero. Reward shaping alone supplies no local update in such a group. Future samples or optimizer momentum can still change the policy, so the state is not a proven absorbing state.

Parsing failures during training: 7 reward-design runs end unable to produce a parseable answer, 4 at the hard level and 3 at the easy level, and they include residual arms. Every run begins at a parse-failure rate of 0.021, so complete parsing failure develops during training. On the hard-level residual seed that fails, one low-spread group produced an advantage close to the normalization bound, and nothing parsed afterward. Salmani-Zarchi et al. (2026) describe this amplification, and Appendix C.2 gives the trace.

KL regularization changes the result: The main comparisons omit a KL penalty, as do DAPO and Dr. GRPO (Yu et al., 2025; Liu et al., 2025b). The capacity ablation adds DeepSeekMath's coefficient (Shao et al., 2024). At the final evaluation, 0 of 9 penalized runs end fully unparseable, compared with 2 of 9 unpenalized runs. With the penalty, the verdict plus task run on seed 1 becomes fully unparseable at earlier evaluations and later parses again. The residual reaches final feasibility on 0 of 3 seeds with the penalty, compared with 2 of 3 without it. Improved final parsing therefore does not imply useful recovery (Appendix C.3).

6 Remedy coverage and training outcomes

The verdict is removed from every all-violating group with reward spread, so the task term determines that group's advantages. Other groups and optimizer state can still affect the parameter update. Such groups and groups with identical rewards occur in the same steps, so a count of identical-reward groups cannot mark when the verdict stopped acting. A remedy can restore the verdict only if it acts on the groups that keep their spread, and the signal a remedy uses determines whether it can. Figure 1 sorts the tested remedies by that property, and Table 18 in Appendix D.1 classifies every published remedy by its input, the groups it acts on and its source.

Identical-reward filters: A filter on identical rewards drops a group only when every member earns the same reward. Every all-violating group with spread passes it, and the verdict has cancelled in each of them, so the filter cannot restore the verdict at any step. A diagnostic that counts zero-variance groups measures the same quantity. Under a pure pass-or-fail reward this filter and an accuracy filter drop the same groups, and the two differ once a task term varies among failures.

Verifier-based filters: DAPO's dynamic sampling drops a prompt whose group is entirely correct or entirely wrong by the verifier, and it keeps sampling until the batch is full (Yu et al., 2025, Sec. 3.2). DAPO's own reward adds an overlong penalty to its correctness score, so an all-wrong group there can carry spread and is still dropped. Our verifier filter uses the same drop rule without refilling the batch, so it removes every all-violating group from the first step.

Fully filtered batches take no optimizer step. The verifier filter at weight 5 took $2, 1, 2$ of 300 optimizer steps per seed on the speed task and $4, 13, 6$ on the capacity task. The filter arms therefore take few updates and do not test dynamic resampling. Matched sampling does not isolate group selection from update frequency or effective batch size.

A control that kept the same number of groups at the verifier filter's own update steps, chosen without reading rewards or verdicts, reached feasibility on 0 of 6 capacity runs against the filter's 3 of 3 seeds. The control copies the filter's update steps from the filter's own runs. On these seeds the filter's capacity result depends on which groups it keeps, and not on its update count alone (Appendix D.1). In single-seed agent runs, Wang (2026) found that a filter on all-failing groups kept training from collapsing (Sec. 5.3), without stating when the filter started, and that fixes applied after the policy was stuck recovered nothing (Sec. 5.2).

External references: Nie et al. (2026) replace the group mean with a fixed reference, which gives the sign advantage $2r - 1$ for a binary reward r and has no rule for a task term. Nan et al. (2025) calibrate every group against a virtual sample with the maximum reward. Salmani-Zarchi et al. (2026) blend a goal-aware anchor into every sample's advantage after a tanh. Under NGRPO's augmented-group normalization, a fixed anchor retains sensitivity to a group-constant verdict when the real rewards differ, so the speed task tests two NGRPO anchors (Appendix D.1).

Remedy outcomes on the speed task: We trained four remedy components on the verdict plus task design at the speed task's hard limit, with the verdict weight 5 below the crossover. Their predictions were committed before the runs, and Table 19 in Appendix D.1 gives every seed. Dr. GRPO reached feasibility on 0 of 3 seeds, and the identical-reward filter reached it on 0. The verifier filter reached feasibility on 1 of 3 seeds, and NGRPO with the feasible anchor, set to the best feasible reward, reached it on 0 of 3.

Every remedy arm ended below the plain verdict's residual on every seed. At five times the weight, which is above the crossover, the plain verdict, the verifier filter and NGRPO reached feasibility on 3, 3 and 3 of 3 seeds. Below the crossover, the difference in final feasibility depends on 1 verifier-filter seed. The counts do not establish a stable efficacy ranking. For a verdict below its crossover, acting from the first step did not reliably yield feasibility.

Remedy outcomes on the capacity task: We trained the same four remedies on the same design at the capacity task's hard level, where $w^* = 12.50$ and the verdict weight again sits below it. Table 22 in Appendix D.1 gives every seed. The plain verdict reached feasibility on 2 of 5 seeds, and the verifier filter reached it on 3 of 3. Dr. GRPO reached it on 2 of 3, and the identical-reward filter reached it on 1 of 3. Our predictions, committed before the runs, said both would reach it on none. The verdict still cancels under both, and Appendix D.1 counts the groups where it does.

The verdict also takes one value in a group whose members all pass, so under both remedies it entered the update only through mixed groups, which hold a feasible member and a violator. Dr. GRPO logged $21, 35, 137$ mixed groups per seed, and the identical-reward filter kept $0, 5, 29$ mixed groups per seed. Removing the divisor may change update magnitudes and avoid amplification of small reward differences. The mixed groups also retain a verdict contribution. We have not isolated these explanations experimentally, and the feasible Dr. GRPO runs yield empty selections. NGRPO with the feasible anchor reached feasibility on 0 of 3 seeds. It answered $93, 93, 90$ of the 96 evaluation completions per seed with no parsed item, and its other answers failed to parse, which counts as violating. These counts are from the final evaluation. The plain verdict lost an earlier zero violation on 1 of its seeds, and NGRPO did so on 1.

At this level, 23 of the 24 runs that reached feasibility, with or without a remedy, answered all 96 evaluation completions with no parsed item. A residual of zero in those runs accompanies parsed empty selections with zero task value, as discussed in Section 7. The exception is the identical-reward filter's feasible seed. It answered 32 completions with no parsed item and selected 0.67 items per answer on average. The best feasible answer at this level holds 1 or 2 items, and that seed earned 0.39 of the optimal task value. On this task, the signal a remedy uses does not settle whether training recovers.

7 Limitations and interpretation

Scope and statistical uncertainty: The main comparisons use two synthetic tasks, one model size, one group size, one unclipped optimizer update per batch, and fixed training prompts that also appear in evaluation. The strict protocol of Appendix D.2 adds held-out instances, a second group size and multi-epoch clipped updates on the capacity task, and Appendix D.4 adds one reasoning task. None of them tests larger models or long-horizon agents.

On GSM8K (Cobbe et al., 2021), 15 of our 15 grade-school-math runs, including the verdict-only arm, ended at zero accuracy on a random held-out set where the base policy scored 43.5 percent. Nearly every completion had stopped parsing (Appendix D.4), so those runs do not test the mechanism there, and we do not test MATH (Hendrycks et al., 2021). The speed task has 3 seeds per arm. The capacity task has 5 per main reward design and 3 per remedy, with 3 and 3 for the easy-level penalty-only arms. Counts from so few seeds are descriptive and cannot establish remedy superiority. We reran the 36 original capacity seeds 0 to 2 at one commit, and 30 of them ended with the same feasibility label (Appendix D.3).

The capacity reward-design, remedy, KL and audit tables hold runs from that one commit. The speed remedy table combines runs from two commits that the fingerprint check of Appendix B.4 ad-

mits, and the capacity below-threshold table combines runs from two commits under the drift waiver of Appendix B.4. Repeated evaluation completions are not independent training replicates. Undefined task values are reported alongside parse failures, and the appendix shows individual seeds. Appendix C.4 adds seed uncertainty and useful feasible value.

The penalty weight remains a confound: The capacity remedies all use a verdict weight below the hard-level crossover. Their outcomes therefore combine exact cancellation in homogeneous groups with possible underweighting in mixed groups. The speed task repeats selected remedies above the crossover. On the capacity task, Appendix D.2 retrains the five algorithms at weight 25, above the crossover, with 5 seeds each. They reach feasibility on 4, 4, 2, 4 and 0 seeds for plain GRPO, the verifier filter, the identical-reward filter, Dr. GRPO and NGRPO. Of the 50 runs at the two weights, 0 have every output useful and feasible, and 24 of the 25 feasible seeds answer every prompt with an empty selection. The residual's smallest nonzero penalty at the trained weight exceeds both hard-level crossovers, as the heavy verdict's penalty does (Appendix B.1).

The two designs reach hard-level feasibility on 7 of 8 and 6 of 8 seeds. Replayed at a matched weight whose first-violation penalty does not exceed the light verdict's (Appendix B.1), the logged groups of the hard-level residual runs keep the same advantage ranking in $14{,}400$ of $14{,}400$ groups. A residual whose first-violation penalty equals the light verdict's reaches it on 7 of 8 seeds. That residual still orders every mixed group, its penalty exceeds the light verdict's from the second violation unit, and its advantages agree with the trained residual's in $1{,}781$ to $1{,}800$ of $1{,}800$ logged groups per run, so the runs test the first-violation penalty scale and not the graded shape. The match holds at the first violation unit only, and the fallback is not matched.

A residual at weight 0.01, below its own ordering threshold of 0.03, reaches feasibility at the speed task's hard limit on 3 of 3 seeds. Its first-violation penalty is 0.20 against the light verdict's 5, its advantages agree with the trained residual's within the tolerance stated in Appendix B.1 in every logged all-violating group whose members parse and lie on one side of the task cap, and it ranks a violator above a feasible completion in 0 of its 18 logged mixed groups.

A violator ties a feasible completion in 4 of those groups, so the runs tested the dropped ordering only weakly. On these seeds the residual recovered with a first-violation penalty below the light verdict's and without ranking every feasible completion above every violator.

The best feasible completion still outranks every violator at that weight, so the runs do not test recovery without that property. Table 20 gives every seed. The page limit moved the sentences that the capacity below-threshold entry fixed for this section to Appendix B.1, where they stand as written. They report the residual at weight 0.04 on the capacity task.

Feasibility is not task success: At the capacity hard level, 23 of 24 feasible runs produce only parsed empty selections. Those outcomes satisfy the checker but achieve no task value. The KL ablation improves final parsing without producing reliable feasibility (Section 5). The permissive parser further limits claims about output validity. The strict protocol of Appendix D.2 no longer counts a parsed empty selection as feasible. It requires a valid, nonempty and feasible selection, and it ends with a held-out useful value of zero in 4 of its 7 cells.

Algebra, coverage, and recovery are distinct claims: The cancellation identity concerns a fixed group's advantages. A failure of numerical replay would challenge its implementation, while a remedy's recovery rate cannot refute the identity. Several outcome predictions did fail, including those for Dr. GRPO and the identical-reward filter on the capacity task. The speed-task ordering depends on one verifier-filter seed. Zero-variance counts separate the original speed arms by only 4 groups at the hard limit and 31 at the easy limit. Successful and failed capacity runs overlap on both counts (Section 5). We did not validate either count as an outcome predictor, and coverage should not be read as an ordering of training performance.

8 Conclusion and future work

A constant verifier term cancels under group-mean centering even when another reward component preserves nonzero advantages. Our audit separates this known identity from mixed-group weighting and training recovery. A remedy's input determines whether it can act on an all-failing group, but useful recovery also depends on optimization, reward scale, and the task. In the capacity task, almost every feasible hard-level run returns only parsed empty selections. Practical evaluations should therefore report constraint satisfaction, task value, and output validity together. A stated capacity in the prompt, multi-epoch clipped training with a KL penalty, and a reasoning benchmark on which training keeps the answer format remain untested.

AI USE STATEMENT

We used generative AI tools to edit the manuscript's language, including its wording and sentence structure. We also used Grammarly to improve readability and AI tools to ensure consistent reference formatting throughout the manuscript. We reviewed all AI-assisted work to ensure that the edits preserved the intended meaning and accurately reflected our research. We take full responsibility for the final content of this work, including its text, claims, references, and any artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

Code and records are available at `https://anonymous.4open.science/r/group-relative-verifier-placement-1F79/`. The supplement contains the training loop, both tasks, the analysis, the test suite and the prediction record, and it does not contain the code of the extension or of the grade-school-math block. Table 9 lists the recorded training settings, and each run of the main comparisons records its configuration, commit hash, device and library versions. The speed remedy runs come from a later commit than their baselines and pass the fingerprint check of Appendix B.4. The capacity below-threshold runs come from a later commit than their baselines under a drift waiver that does not establish statistical equivalence between commits (Appendix B.4), and the capacity seeds 0 to 2 were rerun at one commit (Appendix D.3). The runs were not bit-reproducible across GPU jobs. Appendix E.1 lists every prediction we wrote, with its timing relative to the runs it concerns and its outcome. `PREDICTIONS.md` in the supplement gives the later blocks' predictions as first written, and Appendix E.1 quotes the earlier ones.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 22–31, 2017.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 29304–29320, 2021.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024. doi: 10.18653/v1/2024.acl-long.662.

Eitan Altman. *Constrained Markov Decision Processes*. Stochastic Modeling Series. Chapman and Hall/CRC, 1999.

Yong Yi Bay and Kathleen A. Yearick. GRPO, Dr. GRPO, and DAPO are three operations on one number: The group-standard-deviation identity. *arXiv preprint arXiv:2607.00152*, 2026.

Pengfei Cai, Utkarsh Utkarsh, Alan Edelman, Christopher Vincent Rackauckas, and Rafael Gomez-Bombarelli. Reinforcement learning with verifiable physics: Post-training LLMs with continuous rewards. *arXiv preprint arXiv:2607.10474*, 2026.

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

Anisha Garg, Claire Zhang, Nishit Neema, David Bick, Ganesh Venkatesh, and Joel Hestness. CoRPO: Adding a correctness bias to GRPO improves generalization. *arXiv preprint arXiv:2511.04439*, 2025.

Cheng Ge, Caitlyn Heqi Yin, Hao Liang, and Jiawei Zhang. Why GRPO needs normalization: A local-curvature perspective on adaptive gradients. *arXiv preprint arXiv:2601.23135*, 2026.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 3207–3214, 2018. doi: 10.1609/aaai.v32i1.11694.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1904.09751.

Minsu Kim and Se-Young Yun. Process-verified reinforcement learning for theorem proving via Lean. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2606.20068.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE samples, get a baseline for free! In *ICLR 2019 Workshop on Deep Reinforcement Learning Meets Structured Prediction*, 2019. OpenReview id r1lgTGL5DE.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Xiaozhe Li, Xinyu Fang, Shengyuan Ding, Yang Li, Linyang Li, Haodong Duan, Qingwen Liu, and Kai Chen. Forge: Quality-aware reinforcement learning for NP-hard optimization in LLMs. *arXiv preprint arXiv:2605.08905*, 2026.

Huanyu Liu, Ge Li, Jia Li, Hao Zhu, Kechi Zhang, and Yihong Dong. SATURN: SAT-based reinforcement learning to unleash LLMs reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Conference on Language Modeling*, 2025b. arXiv:2503.20783.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1711.05101.

Gongrui Nan, Siye Chen, Jing Huang, Mengyu Lu, Dexun Wang, Chunmei Xie, Weiqi Xiong, Xianzhou Zeng, Qixuan Zhou, Yadong Li, and Xingzhong Xu. NGRPO: Negative-enhanced group relative policy optimization. *arXiv preprint arXiv:2509.18851*, 2025.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pp. 278–287, 1999.

Wenhua Nie, Jianan Wu, Junlin Liu, Ziwei Li, Zheng Lin, Zijian Zhang, Yilong Fan, Haoran Zheng, and Jyh-Shing Roger Jang. Gradient starvation in binary-reward GRPO: Why group-mean centering fails and why the simplest fix works. *arXiv preprint arXiv:2605.07689*, 2026.

Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606, 2018. doi: 10.1073/pnas.1708274114.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 53728–53741, 2023.

Mohammad Mahdi Salmani-Zarchi, Zahra Rahimi, Heshaam Faili, and Mohammad Javad Dousti. MDP-GRPO: Stabilized group relative policy optimization for multi-constraint instruction following. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016. arXiv:1506.02438.

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

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

Ronald E. Shiffler. Maximum Z scores and outliers. *The American Statistician*, 42(1):79–80, 1988.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 9460–9471, 2022. The proceedings PDF and arXiv:2209.13085 carry the title "Defining and Characterizing Reward Hacking".

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1805.11074.

Yu Wang. The dark room in the reward channel: Dense prediction rewards collapse GRPO-trained LLM agents, and the channel, not the content, decides what works. *arXiv preprint arXiv:2607.21273*, 2026.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Reward algebra and the crossover weight

A.1 Reward identities on sampled groups

The groups in Table 4 are random draws from the capacity task. We checked the identities on 240 groups of speeds sampled from an untrained language model at $v_{\max} = 5$, with no training in the

loop, and the identities held to 8.8×10^{-12}. Completions that named no integer counted as violations at a rate of 0.189. In 0 of the 59 mixed groups, the graded residual let a violator outrank every feasible completion, and a binary verdict at the same weight did so in 31. The cause is arithmetic. The task term spans a wider range than a verdict of that weight contributes, so below the crossover weight of Eq. (2) the verdict does not preserve the ordering of feasible completions above violators. Above the crossover it preserves the ordering, and then it is the term that cancels in the all-violating groups. The residual's penalty at the same nominal weight is larger, because its factor of 100 sits inside the penalty (Appendix B.1). This comparison therefore does not match penalty scales.

Table 4: Whether a group has reward variation and nonzero advantages, by reward design. The groups are random selections on the capacity task with a capacity of 0.5 of each instance's total weight, and no training run used this level. The first row is the published shape. A term that is constant across the group gives no gradient there, whether the reward is binary or graded.

Reward design	**all-violating groups with any reward spread**	**mixed groups with any reward spread**
verdict only (published shape)	0 of 300	300 of 300
residual only	300 of 300	300 of 300
verdict + task term	300 of 300	300 of 300
residual + task term	300 of 300	300 of 300
verdict $\times 1000$ + task	300 of 300	300 of 300
task term only	300 of 300	300 of 300

On seed 2 of the speed task's hard regime, the added-verdict run and the no-penalty run sample identical answers at all 300 steps, and their advantages never differ by more than 5.1×10^{-8}. Exactly 1 of that run's $1{,}800$ groups holds a feasible member, and its three violators sit above the task term's ceiling and tie. A group of four with three tied members has the same advantage vector whenever the lone member sits below the tie, up to the ϵ in Eq. (1). The feasible member sits below the tie in both runs, so even there the verdict moved the rewards and did not change the update.

A.2 The decomposition of the sampled groups

Table 5 is easy to misread. A violator received a positive advantage in 85.6 percent of the $199{,}901$ groups that held one, which suggests that normalization favours infeasible completions. The table shows why it does not. Of the $171{,}131$ groups that reinforced a violator, $102{,}115$ had no feasible member, which is 59.7 percent of them. In such a group the advantages sum to zero, so some member is reinforced unless the rewards all tie, and in $20{,}000$ of $20{,}000$ sampled all-violating groups the most reinforced member was the one closest to feasibility. The other $69{,}016$ reinforcing groups were mixed, which is 70.6 percent of the $97{,}786$ mixed groups. In every one of them each feasible member also received a positive advantage and ranked above every violator. Across all mixed groups a violator outranked a feasible completion 0 times.

Table 5: Groups of four random speeds under a graded residual, out of $200{,}000$ sampled. The last column distinguishes positive advantages from infeasible-over-feasible ordering.

group type	groups	a violator reinforced	a violator outranks every feasible member
every member violates	102,115	102,115	–
mixed	97,786	69,016	**0**
all groups holding a violator	199,901	171,131	0

A.3 The penalty weight and the advantage cap

Table 6 shows one group of four speeds under a graded residual at weight 5, with $v_{\max} = 30$, and in Table 7 the weight rises through six orders of magnitude. The violating completion's advantage moves by only 0.000335. The z-score in Eq. (1) does not change when every reward is multiplied

by the same positive number, up to its ϵ. As the weight grows, the penalty dominates each reward's distance from the group mean, so the advantages approach the z-scores of the penalty alone. Under a verdict, the same limit gives every feasible member of a mixed group one common advantage and every violator another. The task differences among the feasible members then contribute nothing to their advantages. The bound $(K-1)/\sqrt{K}$ plays no part here, because every advantage in Table 7 stays well inside it. The bound says only that no advantage exceeds it in magnitude at any weight.

Table 6: One group of four speeds under a graded residual at weight 5, with $v_{\max} = 30$. The third completion violates the envelope and still receives a positive advantage. It is the less severe of the two violators, and it ranks third in its group.

speed	reward	advantage	status
20	2.00	$+0.6151$	feasible, reinforced
25	2.50	$+0.6155$	feasible, reinforced
60	-494.00	$+0.2465$	violates, reinforced
200	-2813.33	-1.4772	violates, suppressed

Table 7: The advantage of the violating completion in Table 6 as the penalty weight grows. It changes by 0.000335 as the weight spans six orders of magnitude.

penalty weight	advantage of the 60 completion
5	$+0.246518$
50	$+0.246216$
5,000	$+0.246183$
5,000,000	$+0.246183$

A.4 THE CROSSOVER WEIGHT

A verdict adds $-w$ to a violating completion and nothing to a feasible one. A feasible completion f then outranks a violating completion v of the same prompt exactly when $T(f) > T(v) - w$, which gives Eq. (2). Unparseable completions are left out. Their task floor sits below every feasible task value, and no penalty is positive. With a task term, every feasible completion therefore outranks them at any weight. On the speed task every prompt admits a feasible completion and T runs from zero to 20 on every prompt, so $w^* = 20$ at both limits. On the capacity task every item value is positive, so the empty selection is the feasible answer with the smallest task value, and taking every item, which violates at both levels, has the largest. The crossover for an instance is therefore its total value over its optimum. Table 8 gives it per instance, and the largest is 12.50 at the hard level and 2.54 at the easy level. The generator asserts both structural facts before it writes these values. The light verdict's weight 5 sits $below$ the speed crossover, $below$ the capacity hard crossover, and $above$ the capacity easy crossover.

Table 8: The crossover weight per capacity instance, as total value over the optimum at each level.

		hard level		**easy level**	
instance	**total value**	**optimum**	w^*	**optimum**	w^*
0	173	20	8.65	77	2.25
1	175	14	12.50	69	2.54
2	170	26	6.54	71	2.39
3	200	22	9.09	79	2.53
4	150	17	8.82	63	2.38
5	208	18	11.56	87	2.39

B TASKS, REWARDS, SETTINGS AND PROVENANCE

B.1 THE REWARDS IN CLOSED FORM AND THE REPLAYS AT OTHER WEIGHTS

Speed task: A parsed speed s earns $R = T(s) + w\,P(s)$ with $T(s) = \min(|s|, 200)/10$ and residual $r(s) = \max(|s|/v_{\max} - 1,\, 0)$, which Section 3 writes as d. The verdict penalty is $P(s) = -1$ when $|s| > v_{\max}$ and zero otherwise. The residual penalty is $P(s) = -100\,r(s)$. The factor 100 sits inside the penalty, so at a matched weight the residual's slope is 100 times the nominal weight w. The parser returns integers, so at $v_{\max} = 5$ the smallest violation is one unit and the smallest nonzero residual penalty has magnitude $w\,100/v_{\max}$. At $w = 5$ that magnitude is $100/v_{\max}$ times the verdict penalty w, and it already exceeds the task term's full range, which is the crossover 20. The residual and verdict arms therefore differ in penalty scale as well as in shape. A completion with no integer earns -10 plus the arm's penalty at the speed $v_{\max}(1 + r_{\max})$, where $r_{\max}$ is the largest residual r among the group's parsed members and is taken as one when no member parsed.

Capacity task: Instance i at level ρ has capacity $C_i = \max\left(1, \mathrm{round}(\rho \sum_j u_{ij})\right)$, where u_{ij} are the item weights and the rounding is Python's. The prompt lists every item's value and weight and does not state C_i. The empty selection is feasible at every capacity, so withholding the capacity leaves it the one answer that is feasible whatever the capacity is. A selection with weight U and value V earns $R = w_{\text{task}}\,V/\mathrm{OPT}_i + w\,P$, where OPT_i is the exact optimum at capacity C_i, found by dynamic programming. The verdict penalty is $P = -1$ when $U > C_i$ and zero otherwise. A selection's residual is $r = \max(U/C_i - 1,\, 0)$, and the residual penalty is $P = -100\,r$. A completion with no bracketed list earns w_{task} times -10 plus the arm's penalty at a stand-in selection. The fallback always uses the first fixed instance in Table 8, regardless of the prompt. It adds items in descending weight order until the residual reaches the largest among the group's parsed members or all items are used. The target is one when no member parsed. Discrete item weights can overshoot the target, and the fixed instance's maximum residual can fall below it. The reward uses the stand-in's actual residual. We have not isolated the fallback's effect across arms. No heaviest-first prefix of the first instance weighs exactly its capacity at either level, so the stand-in is infeasible at every target, including zero. The stand-in's largest reachable residual is 31.00 at the hard level and 4.05 at the easy level. The largest take-all residual across the instances is 32.20 at the hard level and 4.05 at the easy level. A parsed violator can exceed the stand-in's reach only where the second value exceeds the first. A residual arm can then rank an unparseable member above that violator. A verdict arm cannot, because the stand-in always carries the verdict penalty and the task floor sits below every parsed task value. The raw reward of an unparseable member therefore depends on its parsed siblings before the group mean is subtracted. The same unparseable text earns different rewards in groups whose parsed members differ, and centering then subtracts a mean containing that value. Across the residual and residual-only arms at both levels, 4,542 logged groups held an unparseable member. In 0 of them a parsed residual exceeded the stand-in's reach, and in 0 an unparseable member's reward exceeded a parsed violator's. The residual arm with the KL penalty adds 247 such groups, of which 0 rank an unparseable member above a parsed violator. In the verdict-only and residual-only arms, 175 logged groups held an unparseable member beside parsed members that were all feasible. Of these, 0 had zero reward spread. The verdict-only and residual-only arms set $w_{\text{task}} = 0$. The 6 instances are fixed literals generated once, offline, by a seeded script that is in the supplement, so creating them takes no draws from the random stream of a training run. The smallest nonzero residual penalty at weight 5 is 83.33 at the hard level and 12.82 at the easy level. These sit *above* the hard crossover 12.50 and *above* the easy crossover 2.54.

Replay at a matched weight: At the trained weight 5 the residual's smallest nonzero penalty exceeds the light verdict's penalty of 5 on both tasks. A weight of 0.25 makes that smallest penalty equal 5.00 at the speed task's hard limit, which is the light verdict's penalty. At the capacity task's hard level the weight 0.25 gives a smallest nonzero penalty between 4.17 and 5.00 across the 6 instances. These smallest penalties therefore never exceed the light verdict's penalty on that task. At either residual weight, every feasible answer outranks every violator on every prompt of both tasks at their hard levels. The light verdict below its crossover lacks this property. We replayed every logged training group of the 3 hard-level residual runs on the speed task and of the 5 hard-level residual runs without the KL penalty on the capacity task. For each group we recomputed the residual reward at the trained weight and at the matched weight and took the advantages of Eq. (1). The recomputed rewards at the trained weight equal the logged rewards in every group. The advantage ranking is the

same at both weights in $5{,}400$ of $5{,}400$ speed groups and in $9{,}000$ of $9{,}000$ capacity groups. The advantages agree within 10^{-2} in $5{,}396$ of the speed groups and $8{,}962$ of the capacity groups. The largest absolute difference is 0.190 on the speed task and 0.356 on the capacity task. The trainer would therefore have received the same ordering at the matched weight on the groups these runs sampled. The replay does not show which groups a policy trained at the matched weight would sample. It also does not identify the source of the residual's recovery. That recovery may come from its ordering of violators inside all-violating groups or from its ordering of feasible answers above violators in mixed groups. We then trained the residual at the matched weight 0.25 on both tasks, 8 runs in all, whose predictions were committed before the runs (Appendix E.1). The matched residual reached feasibility on 3 of 3 speed seeds and 4 of 5 capacity seeds, and 0 of the runs ended unparseable. The match holds at the first violation unit only, and the fallback is not matched. Of the $14{,}400$ logged groups of the matched runs, 277 have rewards that are not all equal, and the advantages at the two weights share their ranking in 277 of those groups. Per run, the advantages at the two weights agree to within 10^{-2} in $1{,}781$ to $1{,}800$ of $1{,}800$ groups.

Replay below the ordering threshold: The residual reward decreases with $|s|$ on each side of the task cap 200 exactly when the weight exceeds 0.005. The residual ranks every feasible completion above every violator exactly when the weight exceeds 0.03. The lower bound follows from the slope of the reward below the task cap, and the upper bound comes from enumerating every parseable speed. The window between them is open at both ends, and the weight 0.01 lies inside it. At that weight the smallest nonzero penalty is 0.20 against the light verdict's 5. The two penalties are equal at $|s| = 30$, and the residual's penalty reaches the crossover 20 at $|s| = 105$. At this weight the violators from $|s| = 6$ to $|s| = 9$ outrank some feasible completion, and no violator outranks the best feasible completion. The light verdict ranks every violator with $|s| \geq 56$ above the best feasible completion. The light verdict lets a violator outrank the best feasible completion, and the residual at this weight lets a violator outrank only feasible completions with a lower task value. The best feasible completion outranks every violator at $w = 0.01$, so this design cannot show that the ordering inside all-violating groups alone produces recovery.

When no member of a group parses, the fallback reward is -11 at this weight. It is -15 under the light verdict and -510 under the residual at $w = 5$. The fallback at this weight is therefore closer to the light verdict's than to the trained residual's. On this task an unparseable member never earns more than a parsed member of its group under any of these three arms.

The identity class holds every logged group that meets three conditions. Every member of such a group parses, and at least one member is above the limit. The parsed magnitudes either all lie between the limit and the task cap inclusive or all lie at or above the cap. All-feasible groups lie outside the class. The residual reward is affine and decreasing in $|s|$ on each of those ranges at both weights. Group normalization therefore gives the same advantages at $w = 0.01$ and at $w = 5$, up to the 10^{-6} term of the normalizer. That term is larger relative to the reward spread at the smaller weight, so we compare the advantages within 10^{-4}. The limit itself lies inside the first range, so the class also holds mixed groups whose feasible members all sit at exactly 5.

The three new runs (Table 20) logged $1{,}800$ training groups each. The class holds $7, 6, 9$ groups in the three runs in turn, which is 22 in all. Of these, 14 are all-violating groups and 8 are mixed groups whose feasible members all sit at exactly 5. The advantages at the two weights agree within 10^{-4} in $7, 6, 9$ of those groups in turn, and the largest difference is 3.0×10^{-5}. The advantage ranking at the two weights is the same in $1{,}798, 1{,}799, 1{,}799$ of $1{,}800$ groups in the three runs in turn. It differs in 4 groups in all, and 4 of these are mixed groups in which a violator ties a feasible completion at $w = 0.01$.

Across the three runs, 18 logged groups are mixed. A violator ranks above a feasible completion in 0 of them. A violator ties a feasible completion in 4 of them. The same replay finds a violator above a feasible completion in 3 of the 19 mixed groups of the existing residual runs and in 0 of the 5 mixed groups of the light-verdict runs. The light verdict's own rewards rank a violator above a feasible completion in 4 of those 5 groups. The committed entry in `PREDICTIONS.md` stated both prior counts before the new runs were submitted, and Table 28 records B6 as failed.

On the capacity task the residual's mixed-group ordering threshold differs by instance. The weight above which the residual ranks every feasible answer above every violator is $0.0960, 0.0600, 0.0298, 0.0432, 0.0847, 0.0361$ on the six instances in turn. Adding

any item to an overfull selection lowers the reward exactly when the weight exceeds $0.0200, 0.0120, 0.0183, 0.0114, 0.0200, 0.0111$ per instance. The best feasible answer outranks every violator exactly when the weight exceeds $0.0360, 0.0129, 0.0058, 0.0068, 0.0247, 0.0111$. The weight that keeps the trained residual's order of every violating pair is $0.2820, 0.3471, 0.1731, 0.1682, 0.3212, 0.2806$, and it lies above the threshold on every instance, so no weight that drops the mixed-group ordering on an instance keeps that order there. Two windows were candidates. Weights between 0.0200 and 0.0298 lower the reward for every added item on every instance and drop the mixed-group ordering everywhere, and weights between 0.0360 and 0.0432 keep the best feasible answer on top everywhere and drop that ordering on 4 instances. No weight does both everywhere. The committed weight 0.04 is twice the inclusion bound of 0.02, which is the speed task's rule applied to this reward, and it lies in the second window, so the residual no longer ranks every feasible answer above every violator on instances $0, 1, 3, 4$. The mixed-group ordering still holds on the other two instances. At that weight 5 violating selections outrank the empty selection. They are items 0 and 10 and items 0 and 12 on instance 0, item 0 on instance 1, items 14 and 15 on instance 3, and items 9 and 10 on instance 4. Only these selections outrank the empty selection at this weight, so the design departs from the trained residual's ordering of mixed groups only in groups that hold one of them. The light verdict ranks $5{,}901, 35{,}845, 43, 7{,}997, 7{,}468, 28{,}689$ violating selections above the best feasible answer on the six instances. The first-violation penalty is 0.67 on the instances of capacity 6 and 0.80 on those of capacity 5, against the light verdict's 5. Over the parsed completions of the first evaluation, the mean penalty at this weight is 39.1, and the mean over its violating completions alone is 39.53. With no parsed member the unparseable fallback earns -19.33 under this residual, -15.00 under the light verdict and $-1{,}176.67$ under the residual at weight 5.

The gate D0 of this block has two parts, and both held. Every logged reward of the five runs equals the reward the residual at weight 0.04 gives on the logged selections, and every logged advantage reproduces. The pool of all-violating groups with spread from the five runs holds 56 groups, above the floor of 30, and the advantages at weight 0.04 share the ranking of the advantages at weight 5 in 54 of them, above the required 85 percent. Step 1 samples the initial policy and holds 30 of those groups, of which 29 agree, and the 26 groups after step 1 agree in 25. Across the five runs, 109 logged groups are mixed, and the residual at weight 0.04 ranks a violator above a feasible completion in 3 of them, each through the selection item 0 on instance 1. The same replay at that weight finds 48 mixed groups in the original mixed-cohort runs of the light verdict, which the committed entry named as the comparison runs, and it inverts 1 of them. Under its own reward the light verdict inverts 7 of them. Table 21 gives every seed. On 2 of the 5 seeds the final evaluation holds selections that outrank the empty selection at this weight, $16, 16, 0, 0, 0$ per seed, so D8 failed.

The sentences the committed entry fixed for Section 7 follow as written, and they stand here because the page limit moved them. A residual at weight 0.04, below its ordering threshold on 4 of the 6 capacity instances, reaches feasibility at the capacity task's hard level on 3 of 5 seeds. Its first-violation penalty is 0.67 or 0.80 against the light verdict's 5. Its advantages share the trained residual's ranking in 54 of 56 logged all-violating groups with spread, and it ranks a violator above a feasible completion in 3 of its 109 logged mixed groups. On these seeds the residual recovered with a first-violation penalty below the light verdict's and without ranking every feasible answer above every violator on 4 instances. The best feasible answer still outranks every violator on every instance at that weight. Its mean penalty at the first evaluation is 39.1 against the light verdict's 5. The runs therefore do not test recovery at the light verdict's penalty scale or without the best feasible answer on top (Table 21).

Prompts: Each of the 6 speed prompts asks for one speed as a single integer, in its own wording, and none states the limit $v_{\max}$. The capacity prompt lists the 16 items with their values and weights and asks for the chosen indices as a bracketed list. It says that the total weight must not exceed the capacity, and it does not state C_i. The exact strings are the `PROMPTS` lists in `speed_task.py` and `capacity_task.py` in the supplement.

Parsing and evaluation: The speed parser extracts the first signed integer from the generated text. The capacity parser reads the first bracketed substring, extracts digit sequences, retains valid indices, and removes duplicates. It does not validate a strict list grammar. A bracketed nonnumeric string or a list containing only out-of-range indices can become an empty feasible selection. A

decimal can become several integer indices. The reported outcomes use this parser unchanged, and the logs do not retain the raw completions needed for stricter rescoring. A parse failure counts as a violation. Task value and residual averages exclude parse failures, so they must be read with the parse-failure rate. The evaluation samples multiple completions for each fixed training prompt, with a fixed evaluation random seed. When no member of a group parses, every member earns the same fallback reward on either task, so the group has zero reward spread. Under Eq. (1) or the Dr. GRPO rule it then has zero advantages and contributes nothing to the policy gradient, and under an NGRPO anchor it keeps the tied-group advantage described in Appendix D.1. Under the first two rules, a run whose groups all stop parsing can then change only through optimizer state, or through the KL term in the ablation runs.

B.2 Training settings

Table 9 reports the shared trainer settings and each arm's recorded variations. The policy loss averages completion-level terms, with token log-probabilities summed inside each completion. It does not divide by completion length. Dr. GRPO removes that length divisor as well as the standard-deviation divisor (Liu et al., 2025b). Every arm here therefore shares one length convention, and the Dr. GRPO arm differs from plain GRPO only in the advantage divisor. Each retained batch receives one optimizer update, so the old and current policies agree when the importance ratio is evaluated. The nominal ratio clip is therefore inactive, so the update is unclipped REINFORCE with group-normalized advantages, or with the remedy arm's own advantage rule. The NGRPO arm tests virtual-anchor calibration within this loop and does not reproduce the full method's asymmetric clipping. The sampler draws every completion by nucleus sampling (Holtzman et al., 2020) at a temperature of one, with the top-p value in Table 9 and top-k off. Every arm trains the full model with AdamW (Loshchilov & Hutter, 2019) at the library defaults for (β_1, β_2) and ϵ, with weight decay set to zero.

B.3 The reference-policy log-ratio diagnostic

We compare the sampled reference-policy log-ratio near the first evaluation with zero violations. The diagnostic sums $\log \pi - \log \pi_{\text{ref}}$ over completion tokens. Samples come from nucleus sampling, so the quantity is not an unbiased estimate of full-policy KL divergence. The logged value also precedes the update tested by the evaluation. We use it as a descriptive measure of policy change.

At the speed task's hard limit, the residual-plus-task arm records 6.89 ± 0.00, compared with 19.38 ± 11.25 for the heavy verdict. The per-seed values are $6.89, 6.89, 6.89$ and $11.27, 14.64, 32.23$, respectively. The residual values coincide at the reported precision. The two unsuccessful arms end at 20.86 ± 1.24 and 20.87 ± 1.36, but unrecorded completion lengths may confound that comparison.

At the easy limit, the residual records 11.57 ± 10.48 and the heavy verdict records 16.19 ± 9.16. The residual's values, $23.65, 6.23, 4.83$, vary more across seeds than the difference between the means. The lower log-ratio at the hard limit therefore supports only a task-specific observation. The residual and the heavy verdict also differ in penalty scale, so the observation does not isolate the residual's shape (Appendix B.1). Figure 2 shows the individual training trajectories.

The trainer logs the sum of $\log \pi(y_t \mid y_{<t}, x) - \log \pi_{\text{ref}}(y_t \mid y_{<t}, x)$ over generated tokens and averages it across sampled completions. The reference is the frozen initial policy. The original log key is `kl_seq_mean`, but the samples come from top-p truncation while the probabilities are taken from the untruncated models. The resulting expectation is under the sampling distribution. It is not an unbiased KL estimate between the full policies or between their truncated sampling distributions. We call it the reference log-ratio throughout the reported results.

The diagnostic is recorded before each optimizer update. Evaluation occurs after the update, so the value paired with first feasible evaluation is from the preceding pre-update batch. It is a nearby diagnostic, not a measurement on that evaluation's completions. Raw token sequences and intermediate checkpoints are not retained in these logs, which prevents retrospective correction of the sampling and timing mismatch.

The units are nats per completion, with token contributions summed. Longer answers therefore affect comparisons. At the speed task's hard limit, feasible arms yield single-digit parsed speeds, while failed arms mostly yield six-digit parsed speeds. The original logs omit raw completions and

Table 9: Training settings, read from each run's configuration and provenance, the shared configuration and trainer, and the installed optimizer's defaults. A setting that defines an arm, a level or a seed lists the values the runs used.

Setting	speed task	capacity task
base model	`Qwen/Qwen2.5-0.5B-Instruct`	
group size K	4	
groups per step	6	
sampling steps	300	
learning rate	10^{-5}	
weight decay	0	
gradient norm clip	1	
ratio clip, inert with at most one update per step	0.2	
sampling temperature	1	
top-p	0.95	
top-k (zero turns it off)	0	
repetition penalty	1	
new tokens per completion	6	128
evaluation interval in steps	10	
evaluation samples per prompt	16	
constraint level	5, 30	0.03, 0.2
penalty weight w	0, 0.01, 5, 25, 5,000	0, 5, 5,000
task-term weight	1	0, 1
advantage	Dr. GRPO, GRPO, NGRPO feasible, NGRPO literal	Dr. GRPO, GRPO, NGRPO feasible
group filter	none, identical reward, verifier	
KL coefficient β	0	0, 0.04
seeds	0, 1, 2	0, 1, 2, 3, 4
learning-rate schedule	constant, no warm-up or decay	
precision	`float32`	
full fine-tune	yes	
optimizer	AdamW, library defaults	
AdamW (β_1, β_2)	$(0.9, 0.999)$	
AdamW ϵ	10^{-8}	
GPU model	`NVIDIA A100-SXM4-40GB`	
PyTorch version	`2.14.0+cu130`	

token counts, so equal completion lengths cannot be verified. Answer length remains a possible confound alongside sampling and timing. The KL-regularized runs optimize the separate k3 expression described in Appendix C.3.

B.4 Full provenance output

The supplement contains the run-analysis code. The archive includes the task-quality audit generator and summary records, but not the full set of table generators or raw training logs. Running it over a run directory writes the complete check output, the per-run and per-arm final tables, the group counts per run, the per-seed training and evaluation curves, the trace of every run that stops producing a parseable answer, and the step-by-step agreement between the added-verdict run and the no-penalty run. Separate calibration scripts in the supplement sample the base policy with no gradient. They fixed both capacity levels and the speed task's hard limit. Every standard deviation reported as uncertainty is the sample deviation across training seeds. Every shaded band in a figure shows one sample standard deviation across seeds, clipped to a rate's valid range where applicable. The bands are descriptive and are not confidence intervals. The capacity figures show individual seeds without bands.

Commits, fingerprints and the drift waiver: The analysis refuses to build a comparison when any configuration field differs between runs, apart from 10 exempt fields. They name the arm, its weights, the constraint level, the seed and the output path, and the advantage variant, group filter and KL weight that define the remedy and ablation arms. The generation length is not among the exempt fields, so runs made at two generation limits are never averaged together. The speed remedy runs were made at a later commit than the speed baselines, so Table 19 combines runs from more than one commit. The analysis admits runs from commits that differ in a training source only when a fingerprint run at the later commit reproduces an earlier run's rewards and advantages. On the speed task the two match at every step. On the capacity task they matched only until their sampled answers first differed, and a rerun of the earlier code drifted from the original run in the same way, so the original capacity runs failed the gate committed in `PREDICTIONS.md`. That gate also requires the run at the later commit to equal the rerun, and those two runs differed as well. We wrote a drift waiver after the later run diverged but before we read the rerun, and an earlier draft's capacity comparisons relied on it. We then reran the capacity seeds 0 to 2 at the commit of the remedy, KL and seed 3 and 4 runs (Appendix D.3). The capacity reward-design, remedy, KL and audit tables and figures therefore hold runs from that one commit, admitted without a waiver. Only the capacity below-threshold comparison of Appendix B.1 still admits its commit pair through the waiver. The runs were not bit-reproducible across GPU jobs, and we did not isolate the source of the drift. The analysis also refuses to report a check that examined no cases. The logged parsed groups determine every reward, including group-dependent fallback rewards, so the analysis recomputes each arm's advantages as a consistency check.

Predictions committed before the runs: The earlier predictions are in local planning files that are not released, so a reader cannot verify their dates. The predictions for the remedy, KL and added-seed runs, the below-threshold residual on both tasks, the scale-matched residual, the single-commit rerun, the filter controls and the grade-school-math block were committed before those runs and are in the supplement as `PREDICTIONS.md`. The anonymized copy is a single commit, so none of those dates can be checked during review either. Future predictions need an independently timestamped record fixed before execution.

C Results per seed

C.1 Speed-task trajectories and the easy regime

The main text reports the hard regime, where a fraction 0.83 of groups start with no feasible member. The appendix reports the easy regime, where that fraction is 0.17. Table 10 and Table 11 give its group counts and final evaluation, and Figure 3 plots the training curves in both regimes. Figure 2 plots each seed's trajectory at the hard limit. The all-violating column separates the reward designs across orders of magnitude, and the zero-variance column orders them by 31 groups out of $1{,}800$. The residual finishes at a task value of 2.97 ± 0.06, and the heavy verdict finishes at 2.00 ± 1.73.

One row in these tables is not an independent run. The speed task's term does not depend on the constraint, so its no-penalty arm receives exactly the same reward at both levels. Its runs behind Table 10 therefore repeat its runs behind Table 1 bit for bit. We checked this match instead of assuming it, and the final parameter checksums of the two sets agree on all 3 seeds. Both levels use the same policy, but the all-violating count and the count of all-violating groups with spread are judged against each level's own limit. The first falls from $1{,}799, 1{,}798, 1{,}799$ at the hard limit to $1{,}795, 1{,}794, 1{,}795$ at the easy one, and the second falls from $5, 5, 6$ to $1, 1, 2$. The zero-variance count does not move. It is $1{,}794, 1{,}793, 1{,}793$ at the hard limit and $1{,}794, 1{,}793, 1{,}793$ at the easy one, because it counts groups in which every member earns the same reward, and it never uses the constraint.

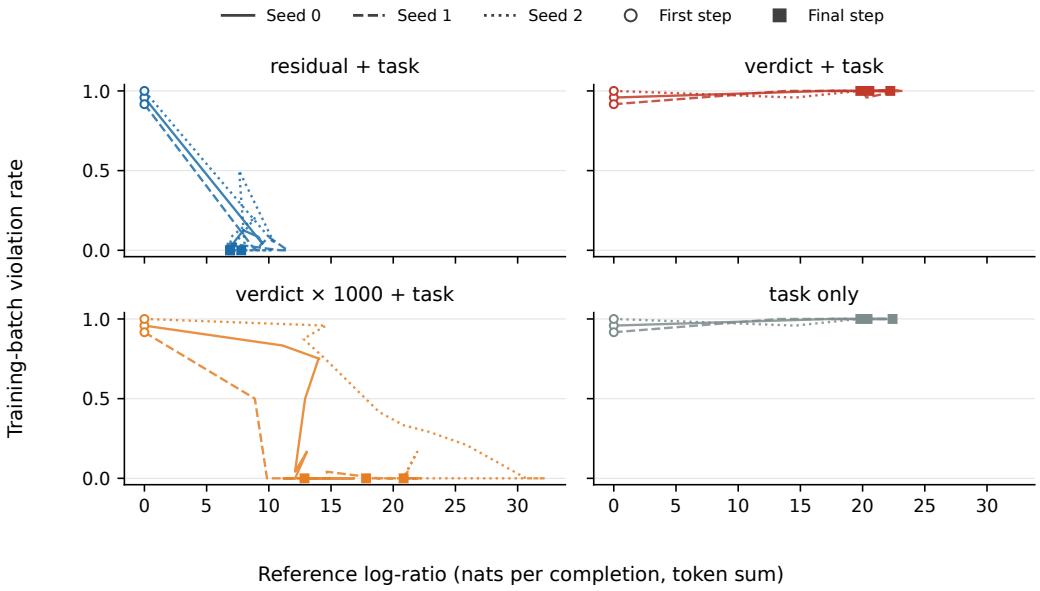

Figure 2: Speed task at the hard limit. Each panel shows one reward design, and line styles identify individual seeds. The vertical axis uses training-batch violations. Circles and squares mark the first and last recorded steps. The horizontal axis is the pre-update reference log-ratio, not an unbiased KL estimate (Appendix B.3). The panels separate overlapping trajectories without averaging across seeds.

Table 10: Group counts per run in the easy regime, out of $1{,}800$ groups (300 steps by 6 groups), 3 seeds per arm. The task-only rows come from the same runs as Table 1, and their columns differ only where a count is judged against each level's own limit.

Reward design	**all-violating**	**zero-variance**	**all-violating, spread**	**mixed**
residual + task	1, 4, 3	1,743, 1,737, 1,744	1, 3, 1	18, 27, 12
verdict + task	1,795, 1,794, 1,792	1,794, 1,788, 1,789	1, 6, 3	3, 5, 6
verdict $\times 1000$ + task	3, 0, 62	1,757, 1,755, 1,754	1, 0, 2	10, 16, 33
task only	1,795, 1,794, 1,795	1,794, 1,793, 1,793	1, 1, 2	3, 5, 3

C.2 The capacity task in detail

On seed 3 at the hard level the verdict enters the update through only 1 of the $1{,}800$ groups. Every other group in that run has no feasible member, so the verdict takes the same value inside each of them. The added-verdict run's advantages agree with the no-penalty run's to 2.2×10^{-16} for as long as the two sample the same answers. They sample the same answers on 285 of 300 steps and first differ at step 7. Numerical differences accompany this divergence, but the logs do not isolate its cause. All 5 seeds of every arm come from one commit (Appendix D.3), so no commit boundary separates the seeds of Table 12.

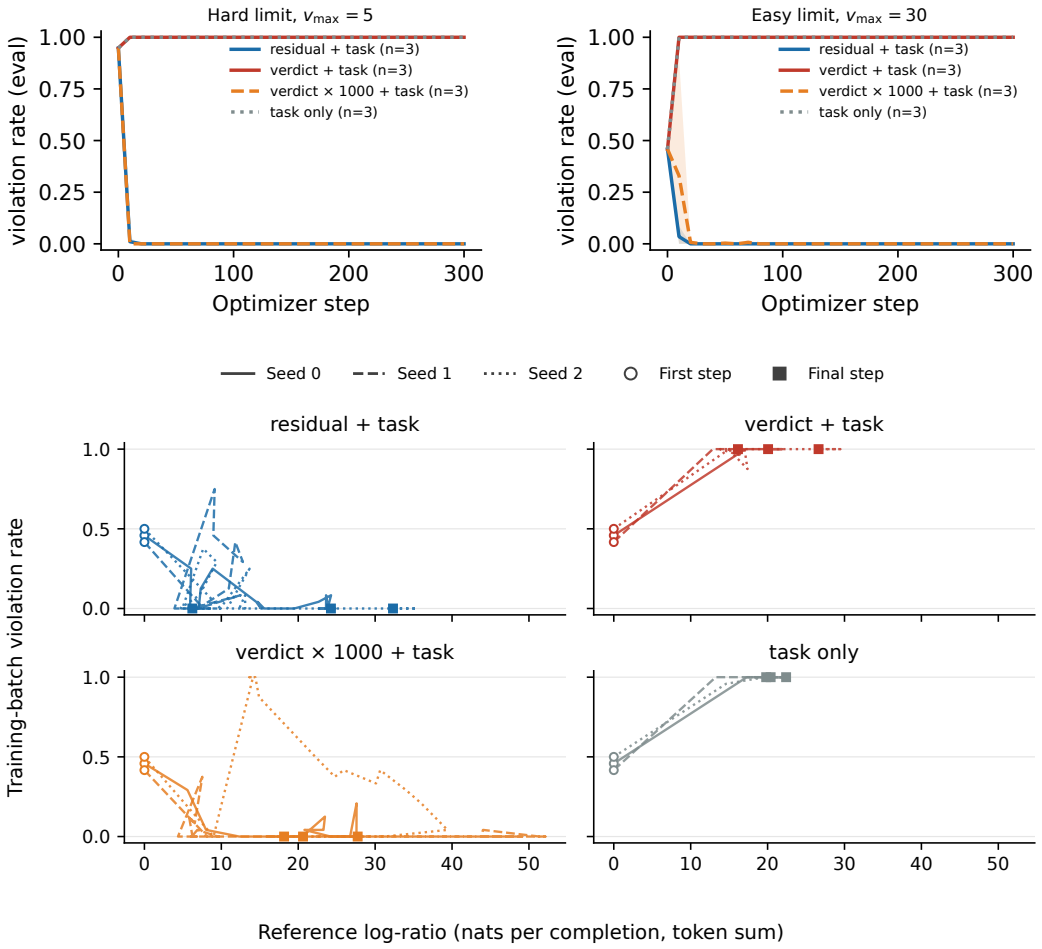

Figure 3: Speed-task trajectories. The top panels show evaluation violation rates at the hard and easy limits, averaged over 3 seeds. Shading shows one sample standard deviation, clipped to the valid rate range, and is not a confidence interval. At the hard limit, the residual first reaches zero violation at steps $20, 10, 10$, and the heavy verdict does so at $10, 10, 10$. The reported step sequences follow seed order. The lower panels show individual seeds at the easy limit, with training-batch violation against the reference log-ratio. Circles mark the first step and squares mark the last.

Table 11: Final evaluation in the easy regime, mean and standard deviation over 3 seeds. Task value is read as in Table 2. The log-ratio is summed over completion tokens, in nats per completion, so it scales with answer length (Appendix B.3).

Reward design	**violation**	**residual**	**task value**	**unparseable**	**log-ratio**
residual + task	0.000 ± 0.000	0.000 ± 0.000	2.97 ± 0.06	0.00 ± 0.00	20.94 ± 13.36
verdict + task	1.000 ± 0.000	$15{,}241 \pm 12{,}695$	20.00 ± 0.00	0.00 ± 0.00	20.95 ± 5.30
verdict $\times 1000$ + task	0.000 ± 0.000	0.000 ± 0.000	2.00 ± 1.73	0.00 ± 0.00	22.17 ± 4.97
task only	1.000 ± 0.000	$22{,}523 \pm 7{,}147$	20.00 ± 0.00	0.00 ± 0.00	20.87 ± 1.36

On the residual seed that fails at the hard level, the first steps work as designed and drive the mean excess weight from 8.85 capacities above the limit to 0.00. By step 6 every group has exactly zero spread, because every answer is the empty selection. At step 9 one group carries a spread of 0.0909, and its advantages round to -0.5, -0.5, $+1.5$ and -0.5. Of the 4 completions in that group, 0 received the unparseable fallback, and the spread comes from parsed selections that differ in value. Dividing by the group's own deviation keeps the ordering and discards the size, so that gap produces an advantage close to the normalization bound. At the next step the reference log-ratio moves from 13.14 to 41.02, and nothing parses again (Appendix B.1).

At the easy level every design carrying both a task term and a penalty reaches feasibility on every seed that keeps parsing. Table 15 counts those seeds and gives the task value that each seed ends at. The no-penalty baseline ends on a different kind of answer from seed to seed. On seed 0 it overfills, on seeds 1 and 2 it reaches feasibility, on seed 3 it stops parsing, and on seed 4 it ends just short of feasibility. The two arms carrying a penalty and no task term reach feasibility with nearly empty selections on every seed that keeps parsing, and the residual-only arm stops parsing on 1 of 3 seeds. So the capacity task's easy level distinguishes a penalty from no penalty. It does not distinguish a verdict from a residual when both carry a task term, because the light verdict's weight sits $above$ that level's crossover. The speed task's easy level does distinguish them, because there the same weight sits $below$ the crossover and the verdict fails. At the same nominal weight the residual's penalty carries the factor 100 (Appendix B.1), so both comparisons differ in penalty scale as well as in shape. At the easy level the zero-variance counts of runs that reach feasibility span $1{,}369$ to $1{,}794$, and those of runs that fail span $1{,}673$ to $1{,}766$. The successful runs there log up to 87 groups with no feasible member, and a failed run logs as few as 30 (Table 13). Figures 4 and 5 show each seed at both levels.

C.3 The KL-penalty ablation

On seeds 0 to 2, the ablation trains new runs of the residual arm, the verdict plus task arm and the no-penalty arm at the capacity hard level. Each loss gains a per-token KL penalty toward the frozen initial policy with DeepSeekMath's coefficient (Shao et al., 2024). The penalty uses the k3 estimator $r - \log r - 1$, where r is a token's probability under the initial policy divided by its probability under the current policy. It is summed over each completion's tokens, the same way the policy-gradient term is summed. Its value and gradient are zero at the first step, so a run starts exactly where its unpenalized counterpart starts. Table 16 gives every seed, and Appendix E.1 lists the predictions committed for these runs with their outcomes.

With the penalty, 0 of the 9 runs end unparseable at the final evaluation, against 2 of their 9 unpenalized counterparts. By the rule that K3 fixed before these runs, the KL penalty prevented the collapse in these cells. A cell here is a reward design in Table 16, and a run collapses when it ends unparseable at the final evaluation. The penalty did not keep every run parseable during training. The worst evaluation parse-failure rates of the verdict plus task arm with the penalty were $0.83, 1.00, 0.22$ on seeds 0 to 2, and no seed ended unparseable.

The residual with the penalty reaches feasibility on 0 of 3 seeds, against 2 of 3 without it. The residual with the penalty lost an earlier zero violation on 1 of its seeds. On seeds 0 to 2, its final violation rates are $0.05, 0.02, 0.27$. Each seed is scored on 96 evaluation completions, and the residual with the penalty answers $89, 93, 60$ of them with no parsed item. Without the penalty, the residual answers $96, 96$ of them with no parsed item on seeds 0 and 1, and its seed 2 gives no answer that parses.

With the penalty, the verdict plus task arm reaches feasibility on 0 of 3 seeds, and the no-penalty arm reaches it on 0 of 3. Without the penalty, the verdict plus task arm reaches it on 2 of 3 seeds, and the no-penalty arm reaches it on 0 of 3. The penalty lowers the final reference log-ratio in 8 of 9 pairs of runs. The exception is the residual at seed 2, whose unpenalized counterpart stops producing a parseable answer at step 10. Every other unpenalized counterpart that ends unparseable still has a higher final reference log-ratio than its penalized run.

The reference penalty may limit policy shifts that improve checker feasibility while suppressing task value. That interpretation is a hypothesis. These runs do not isolate that mechanism from other optimization effects, and useful recovery under KL regularization remains unestablished.

Table 12: Capacity task, hard level: group counts per run, out of 1,800 groups (300 steps by 6 groups), 5 seeds per arm. The columns mean what they mean in Table 1. An unparseable answer counts as violating, so a seed that stops producing a parseable answer, such as the residual arm's seed 2 at step 10, has a first column that resembles a failing design's. Seeds 0 to 2 are the single-commit reruns of Appendix D.3, at the commit of seeds 3 and 4 of every arm.

Reward design	**seed**	**all-violating**	**zero-variance**	**all-violating, spread**	**mixed**
residual + task	0	15	1,784	12	4
	1	16	1,747	14	38
	2	1,761	1,777	12	10
	3	25	1,746	13	37
	4	53	1,779	14	7
verdict + task	0	431	1,698	82	20
	1	1,795	1,736	59	5
	2	579	1,399	348	52
	3	1,799	1,744	55	1
	4	30	1,761	7	32
verdict $\times 1000$ + task	0	1,800	1,684	116	0
	1	56	1,671	42	78
	2	58	1,727	36	37
	3	1,800	1,766	34	0
	4	24	1,768	7	25
task only	0	1,797	1,740	57	3
	1	1,791	1,691	100	9
	2	1,645	877	848	73
	3	1,800	1,766	34	0
	4	37	1,670	13	109
verdict only	0	120	1,791	0	9
	1	128	1,720	0	80
	2	89	1,793	0	7
	3	96	1,771	0	29
	4	1,742	1,766	0	34
residual only	0	15	1,784	12	4
	1	16	1,772	14	14
	2	15	1,773	12	15
	3	13	1,767	13	20
	4	53	1,779	14	7

Table 13: Capacity task, easy level: group counts per run, out of 1,800 groups (300 steps by 6 groups). Each arm has 5 seeds, except the verdict-only and residual-only arms. The compute ceiling cut seeds 3 and 4 from those two arms, so the verdict-only arm keeps 3 seeds and the residual-only arm keeps 3. Measured at the first optimizer step, a fraction 0.27 of groups has no feasible member, against 1.00 at the hard level. The fractions in Section 5 come from sampling the untrained policy instead.

Reward design	**seed**	**all-violating**	**zero-variance**	**all-violating, spread**	**mixed**
residual + task	0	2	1,782	2	10
	1	2	1,785	2	5
	2	4	1,781	1	11
	3	1,243	1,749	3	17
	4	8	1,531	8	126
verdict + task	0	3	1,786	3	10
	1	2	1,781	2	7
	2	1	1,771	1	15
	3	87	1,476	9	120
	4	1	1,782	1	11
verdict $\times 1000$ + task	0	2	1,788	2	10
	1	2	1,788	2	4
	2	5	1,764	5	15
	3	3	1,369	2	210
	4	1	1,783	1	9
task only	0	1,771	1,693	78	28
	1	25	1,770	19	11
	2	11	1,749	6	32
	3	1,742	1,732	26	24
	4	30	1,673	1	118
verdict only	0	24	1,769	0	31
	1	2	1,787	0	13
	2	1	1,767	0	33
residual only	0	1,372	1,766	2	32
	1	2	1,794	2	4
	2	1	1,789	1	10

Table 14: Capacity task, hard level: final evaluation per seed. Task value is the chosen value over the exact optimum. The residual and the task value are means over the completions that parsed. The unparseable column gives the fraction of each seed's evaluation completions that did not parse. A dash marks a seed where no completion parsed, so its residual and task value are undefined. Repeated maximum task values occur when the saved final parsed selections contain every item. Those selections violate the capacity constraint. The log-ratio is summed over completion tokens, in nats per completion, so it scales with answer length (Appendix B.3). Seeds 0 to 2 are the single-commit reruns of Appendix D.3, at the commit of seeds 3 and 4 of every arm.

Reward design	**seed**	**violation**	**residual**	**task value**	**unparseable**	**log-ratio**
residual + task	0	0.00	0.00	0.00	0.00	89.8
	1	0.00	0.00	0.00	0.00	266.7
	2	1.00	–	–	1.00	32.4
	3	0.00	0.00	0.00	0.00	23.5
	4	0.00	0.00	0.00	0.00	147.7
verdict + task	0	0.00	0.00	0.00	0.00	33.8
	1	1.00	–	–	1.00	77.7
	2	0.00	0.00	0.00	0.00	240.9
	3	1.00	31.32	9.53	0.00	11.7
	4	0.99	0.00	1.00	0.99	157.6
verdict $\times 1000$ + task	0	1.00	–	–	1.00	125.8
	1	0.00	0.00	0.00	0.00	67.0
	2	0.00	0.00	0.00	0.00	181.5
	3	1.00	31.32	9.53	0.00	45.3
	4	0.00	0.00	0.00	0.00	184.9
task only	0	1.00	31.32	9.53	0.00	186.0
	1	1.00	31.32	9.53	0.00	79.8
	2	1.00	23.19	7.36	0.00	177.9
	3	1.00	31.32	9.53	0.00	46.1
	4	0.46	0.53	0.42	0.18	121.2
verdict only	0	0.00	0.00	0.00	0.00	38.2
	1	0.00	0.00	0.00	0.00	120.6
	2	0.00	0.00	0.00	0.00	7.0
	3	0.00	0.00	0.00	0.00	159.5
	4	1.00	–	–	1.00	366.2
residual only	0	0.00	0.00	0.00	0.00	83.2
	1	0.00	0.00	0.00	0.00	128.8
	2	0.00	0.00	0.00	0.00	119.2
	3	0.00	0.00	0.00	0.00	94.3
	4	0.00	0.00	0.00	0.00	97.7

Table 15: Capacity task, easy level: final evaluation per seed. The columns mean what they mean in Table 14. The compute ceiling cut seeds 3 and 4 of the verdict-only and residual-only arms.

Reward design	**seed**	**violation**	**residual**	**task value**	**unparseable**	**log-ratio**
residual + task	0	0.00	0.00	0.52	0.00	25.1
	1	0.00	0.00	0.52	0.00	3.4
	2	0.00	0.00	0.52	0.00	143.9
	3	1.00	–	–	1.00	247.0
	4	0.00	0.00	0.52	0.00	102.0
verdict + task	0	0.00	0.00	0.52	0.00	34.7
	1	0.00	0.00	0.52	0.00	18.0
	2	0.00	0.00	0.52	0.00	110.8
	3	0.00	0.00	0.14	0.00	50.7
	4	0.00	0.00	0.52	0.00	3.5
verdict $\times 1000$ + task	0	0.00	0.00	0.52	0.00	3.0
	1	0.00	0.00	0.52	0.00	3.0
	2	0.00	0.00	0.36	0.00	155.5
	3	0.00	0.00	0.36	0.00	210.7
	4	0.00	0.00	0.52	0.00	9.9
task only	0	1.00	4.00	2.41	0.00	138.0
	1	0.00	0.00	0.00	0.00	77.8
	2	0.00	0.00	0.52	0.00	130.0
	3	1.00	–	–	1.00	51.5
	4	0.02	0.00	0.09	0.00	256.6
verdict only	0	0.00	0.00	0.17	0.00	17.2
	1	0.00	0.00	0.00	0.00	49.9
	2	0.00	0.00	0.00	0.00	71.5
residual only	0	1.00	–	–	1.00	39.0
	1	0.00	0.00	0.18	0.00	6.2
	2	0.00	0.00	0.00	0.00	25.4

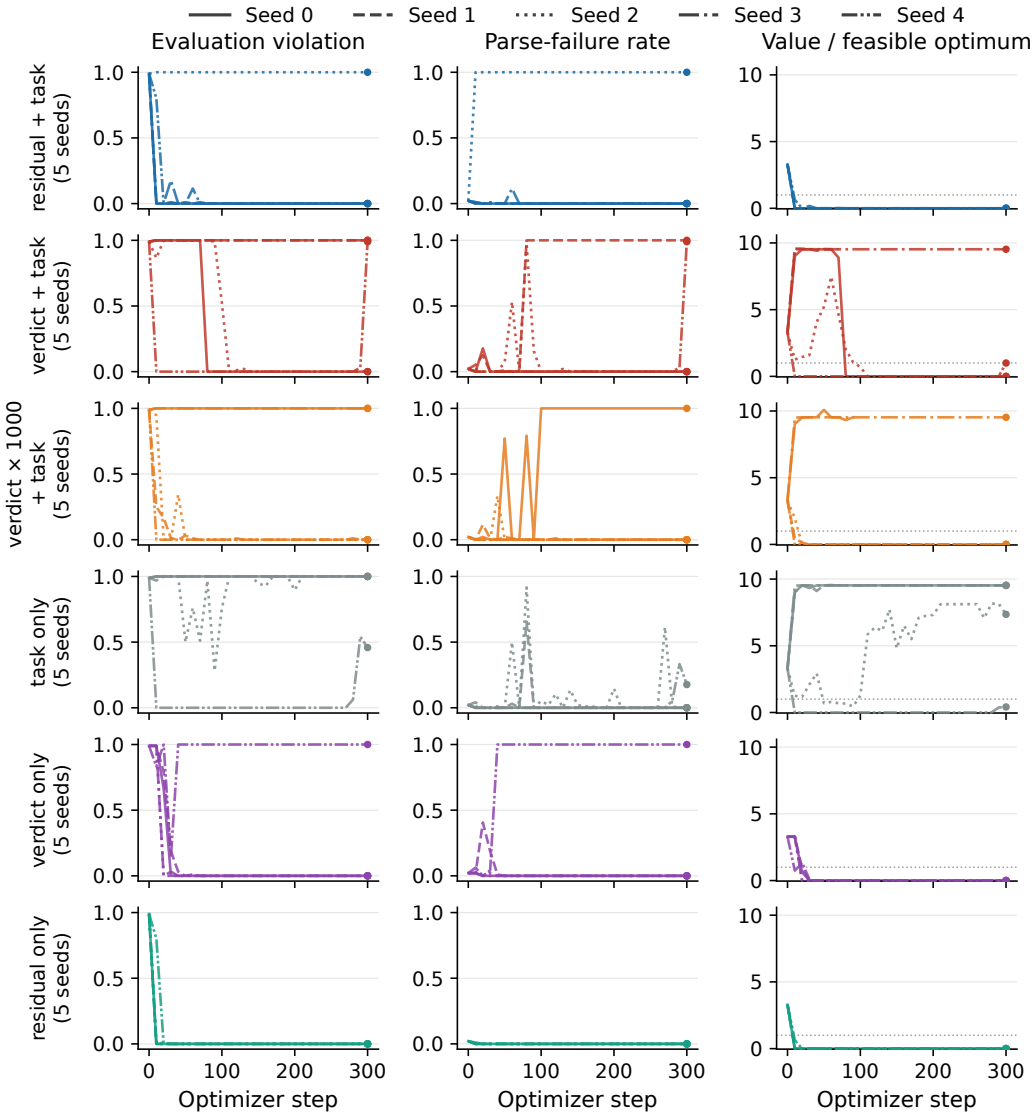

Figure 4: Capacity task at the hard level, with one row per reward design and one trace per seed. The columns show evaluation violation rate, parse-failure rate, and normalized task value. Task values average only parsed answers at each evaluation, and a gap means none parsed. The middle column shows that loss of observations. Values above the dotted optimum line come from infeasible selections. The figure displays individual runs and no uncertainty interval. Exact overlaps can conceal individual seed traces. Feasibility together with zero task value identifies parsed empty selections.

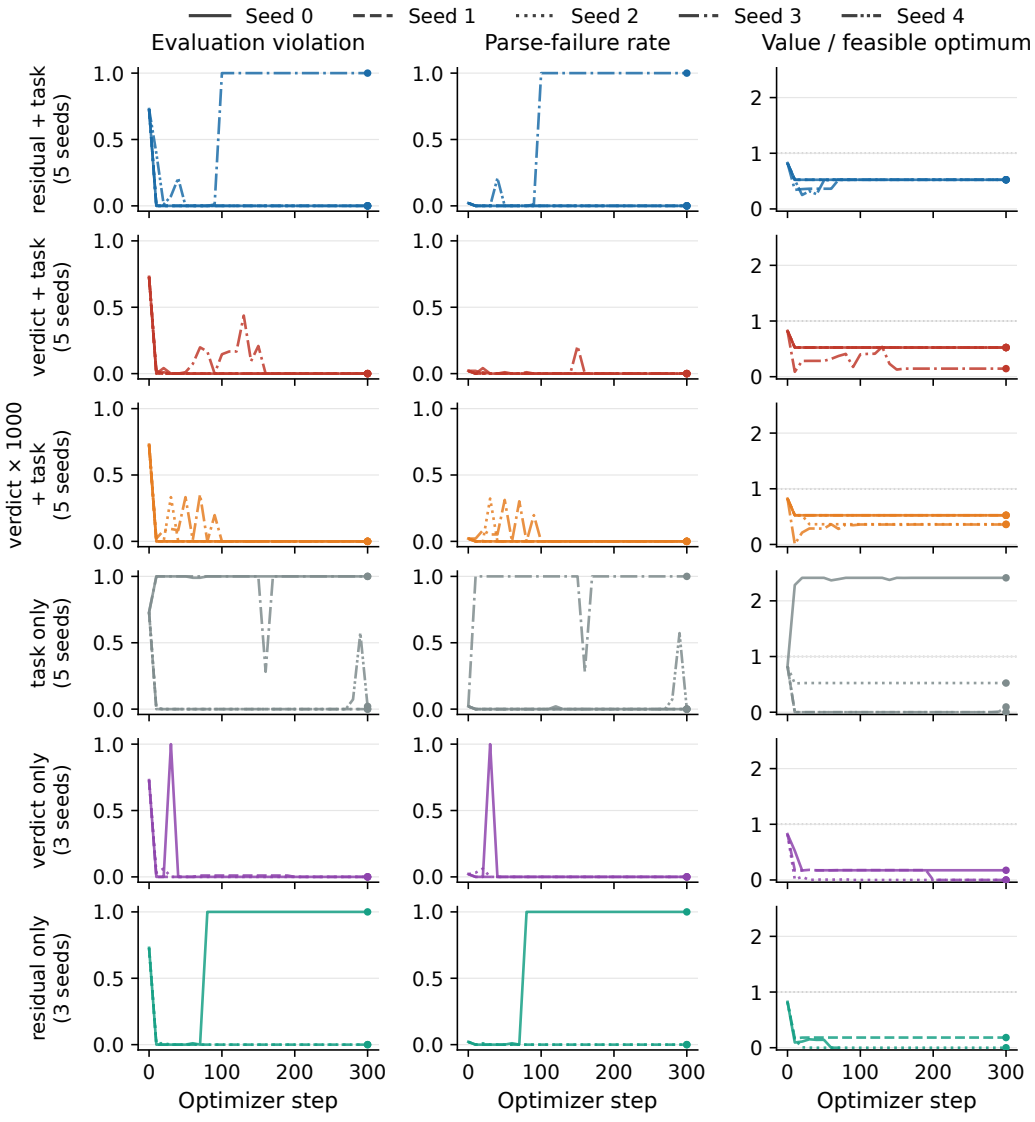

Figure 5: Capacity task at the easy level, with the same metrics and conventions as Figure 4. Each main reward arm has 5 seeds. The verdict-only and residual-only arms have 3 and 3 because the remaining runs were not performed. Individual traces separate constraint satisfaction from parsing failures and task value.

Table 16: Capacity task, hard level: each reward design trained with and without the KL penalty, per seed over 3 seeds. The rows labelled *task only* are the no-penalty arm. Violation, residual, task value and unparseable come from the final evaluation and mean what they mean in Table 14, where a dash marks a seed with no parseable completion. The log-ratio is the reference diagnostic logged at the last training step, as defined in Appendix B.3. It is summed over completion tokens, in nats per completion, so it scales with answer length. The penalty itself uses the k3 expression, which gives a different value.

Reward design	seed	violation	residual	task value	unparseable	log-ratio
verdict + task	0	0.00	0.00	0.00	0.00	33.8
	1	1.00	–	–	1.00	77.7
	2	0.00	0.00	0.00	0.00	240.9
task only	0	1.00	31.32	9.53	0.00	186.0
	1	1.00	31.32	9.53	0.00	79.8
	2	1.00	23.19	7.36	0.00	177.9
residual + task	0	0.00	0.00	0.00	0.00	89.8
	1	0.00	0.00	0.00	0.00	266.7
	2	1.00	–	–	1.00	32.4
residual + task, KL $\beta = 0.04$	0	0.05	0.14	0.05	0.00	48.4
	1	0.02	0.03	0.04	0.00	21.9
	2	0.27	0.42	0.32	0.00	36.7
verdict + task, KL $\beta = 0.04$	0	1.00	29.94	9.13	0.00	7.3
	1	0.99	30.49	9.26	0.02	7.7
	2	0.98	29.66	8.97	0.03	9.8
task only, KL $\beta = 0.04$	0	1.00	31.30	9.52	0.00	18.2
	1	1.00	31.28	9.52	0.00	25.9
	2	1.00	31.32	9.53	0.00	6.6

C.4 ADDITIONAL AUDITS OF RECORDED RUNS

We reanalyse 51 recorded runs at the capacity hard level without new training. The runs cover 13 arms, which are the reward designs, the remedy arms and the KL-penalized arms of Appendix C.3. Table 17 distinguishes a seed with zero final violations from useful feasible output. A completion earns normalized feasible value only when its parsed selection is feasible. Unparseable and infeasible completions contribute zero, so every evaluation completion remains in the denominator. The nonempty-feasible fraction uses the same denominator. These are post-hoc summaries and were not preregistered endpoints.

Of 24 fully feasible runs, 23 produce only parsed empty selections. An empty selection here is the parsed selection as Section 5 defines it, and the recorded parser can map a bracketed span with no valid index to it (Appendix B.1). The remaining run, the identical-reward filter's feasible seed, produces 64 nonempty feasible selections among 96 completions and normalized feasible value 0.3858. An arm's average includes all its seeds, so its value differs from that one successful seed's value. The residual arm with the KL penalty reaches zero final violation on 0 of 3 seeds. Its nonempty-feasible fraction of 0.0451 and feasible value of 0.0184 come from seeds that end with a nonzero violation rate. The verdict plus task arm with the penalty has a feasible value of 0.0017, and the task-only arm with the penalty has 0.0000.

Comparing point estimates from a handful of training runs ignores their statistical uncertainty, so Agarwal et al. (2021) advocate reporting interval estimates of aggregate performance. The 95% Wilson intervals treat each training seed as an independent observation. Repeated evaluation completions within a run are not independent training replicates. Each interval is the score interval of Wilson (1927), computed from an arm's feasible-seed count and its total seed count. They describe sampling uncertainty conditional on this setup. They do not account for task selection, parser limitations, or shared implementation choices. Their width is another reason that these runs cannot establish a general remedy ranking.

Table 17: Capacity-hard outcomes and useful task value from the saved final evaluations. Feasible seeds have zero violations across all 96 evaluation completions. Brackets after the feasible-seed count give descriptive 95% Wilson intervals over seeds. Brackets after the nonempty-feasible fraction and the feasible value give the per-seed minimum and maximum. The last three rows are the KL-penalized arms of Appendix C.3, trained with the coefficient stated there. The nonempty-feasible fraction and normalized feasible value average over every evaluation completion from every seed. Infeasible and unparseable outputs contribute zero. The table uses the recorded parser and cannot certify strict raw-output validity.

Reward design and component	feasible seeds [95% interval]	nonempty feasible	feasible value
residual + task	4/5 [0.38, 0.96]	0.0000 [0.00, 0.00]	0.0000 [0.00, 0.00]
verdict + task	2/5 [0.12, 0.77]	0.0021 [0.00, 0.01]	0.0021 [0.00, 0.01]
verdict $\times 1000$ + task	3/5 [0.23, 0.88]	0.0000 [0.00, 0.00]	0.0000 [0.00, 0.00]
task only	0/5 [0.00, 0.43]	0.0354 [0.00, 0.18]	0.0185 [0.00, 0.09]
verdict only	4/5 [0.38, 0.96]	0.0000 [0.00, 0.00]	0.0000 [0.00, 0.00]
residual only	5/5 [0.57, 1.00]	0.0000 [0.00, 0.00]	0.0000 [0.00, 0.00]
verdict + task, Dr. GRPO	2/3 [0.21, 0.94]	0.0000 [0.00, 0.00]	0.0000 [0.00, 0.00]
verdict + task, NGRPO, feasible anchor	0/3 [0.00, 0.56]	0.0000 [0.00, 0.00]	0.0000 [0.00, 0.00]
verdict + task, verifier filter	3/3 [0.44, 1.00]	0.0000 [0.00, 0.00]	0.0000 [0.00, 0.00]
verdict + task, identical-reward filter	1/3 [0.06, 0.79]	0.2222 [0.00, 0.67]	0.1286 [0.00, 0.39]
residual + task, KL penalty	0/3 [0.00, 0.56]	0.0451 [0.01, 0.10]	0.0184 [0.00, 0.05]
verdict + task, KL penalty	0/3 [0.00, 0.56]	0.0035 [0.00, 0.01]	0.0017 [0.00, 0.01]
task only, KL penalty	0/3 [0.00, 0.56]	0.0000 [0.00, 0.00]	0.0000 [0.00, 0.00]

The released audit also rescores recorded mixed groups at the original verdict weight, the exact crossover, and an integer strictly above it. The replay tests reward ordering on fixed samples. It does not predict the trajectories of a policy trained at a different weight. Separate CPU checks reproduce the NGRPO population-deviation convention, counterexamples to overly broad baseline and residual claims, and the configured residual ordering across all fixed capacity subsets. The supplement includes the audit code and sanitized summary records.

D REMEDIES, CONTROLS, THE EXTENSION, THE RERUN AND GRADE-SCHOOL MATH

D.1 THE REMEDY ARMS

Every remedy arm uses the verdict plus task design at weight 5 and runs at the hard level of each task. On the speed task the verifier filter and NGRPO with the feasible anchor also run at five times that weight, above the crossover. Table 19 gives the speed task, Table 20 adds the residual below its ordering threshold at the speed task's hard limit, Table 21 adds the residual below its threshold at the capacity task's hard level, and Table 22 gives the capacity task. The verifier filter keeps a group only when at least one member passes the verifier and at least one fails, and an unparseable completion fails. DAPO keeps sampling until the batch is full (Yu et al., 2025, Sec. 3.2). Our filter drops groups without refilling. At the capacity hard level no first-step group holds a feasible member on any seed, so a refill loop there would need many rounds of sampling to fill one batch. One rule runs on both tasks, so the sample budget stays matched. A step in which a filter drops every group skips the backward pass, the gradient clip and the optimizer step. The loss of a step that keeps some groups is the mean over the kept completions. These arms match sampling budgets, but neither optimizer-update counts nor contributing batch sizes are matched. Their outcomes cannot separate the filter's selection rule from those optimization differences.

Table 18: Remedy mechanisms classified by their inputs. The last data column shows whether a verdict that is constant across an all-violating group cancels out of that group's advantages.

Remedy	Input	Acts on	Verdict cancels	Source
identical-reward filter	reward spread	zero-spread groups only	yes, in every kept group	this paper, Section 6
dynamic sampling	verifier accuracy	homogeneous-accuracy groups, from step one, refilling the batch	not applicable, group is dropped	Yu et al., 2025, Sec. 3.2
sign advantage	a fixed reference	every group	no, binary rewards only	Nie et al., 2026, Sec. 3.2
advantage calibration	a virtual best sample	every group	no with a feasible anchor, yes with an infeasible one	Nan et al., 2025, Sec. 4.1
dual anchor	a goal anchor	every sample, after a tanh	only in the z-score part	Salmani-Zarchi et al., 2026, Sec. 4.3–4.4
Dr. GRPO	the group's rewards	every group	yes	Liu et al., 2025b, Sec. 3.2
mixed temperatures	the sampler	before scoring	yes, if every member fails	Salmani-Zarchi et al., 2026, Sec. 4.2

NGRPO appends a virtual sample to every group and takes the mean and the population standard deviation over the $K+1$ rewards (Nan et al., 2025, Sec. 4.1). With a task term, the maximum reward is a design choice, so we train two anchors on the speed task. We train the feasible anchor alone on the capacity task. The feasible anchor is the task weight times the best feasible task value. The literal anchor is the design's own maximum reward, found by enumeration, and at both hard levels it belongs to an infeasible answer. Unlike the sample deviation in Eq. (1), this population deviation divides by $K+1$. When the K real rewards equal $r \neq a$, every member's advantage approaches $\operatorname{sign}(r-a)/\sqrt{K}$ as ϵ vanishes. It is zero when $r = a$. A verdict can change the sign by moving r across the feasible anchor. At finite ϵ, it can also change the magnitude without crossing the anchor. With the literal anchor both r and a carry the same penalty, so the verdict cancels entirely. The identical-reward filter keeps a group when its reward range exceeds the tolerance the analysis uses for a zero-variance group. Dr. GRPO subtracts the group mean and does not divide by the group deviation.

We recomputed the advantages of every all-violating group that the speed task's runs at the hard limit logged, with the verdict at weight 5. These runs logged 36,210 such groups. The verdict cancels in 36,210 of them under plain GRPO, in 36,210 under Dr. GRPO and in 36,210 under NGRPO with the literal anchor. Under NGRPO with the feasible anchor, the verdict changes the advantage in 665

groups. Of these, 662 have reward spread and 3 have a tied reward that crosses the anchor. In the remaining 35,545 groups the added verdict and no penalty differ only through ϵ. Replayed on the same groups, the identical-reward filter would keep 662 of them, which are exactly the groups with spread, and plain GRPO cancels the verdict in every one. The verifier filter would keep 0, because no member of an all-violating group passes the verifier. These replayed counts differ from the kept groups in Table 19, which count what each filter let into its own training run.

On the speed task, NGRPO with the literal anchor, under which the verdict cancels in every all-violating group, reached feasibility on 0 of 3 seeds. The identical-reward filter and the verifier filter skipped almost every optimizer step. The identical-reward filter took $2, 5, 8$ of its 300 steps per seed, and the verifier filter took $2, 1, 2$. The verifier filter kept $2, 2, 3$ of 1,800 groups per seed, and it does not refill the batch. It drops a group with no feasible member and also a group in which every member passes. Per seed, $1{,}798, 1{,}798, 10$ groups had no feasible member and $0, 0, 1{,}787$ had every member pass. The plain verdict at five times the weight reached feasibility on 3 of 3 seeds, and per seed it logged $11, 7, 21$ groups with no feasible member. Every failed run's all-violating count is at least 85 times every successful run's once these runs join those of Table 1. The four designs of that table alone give 94 times, and Section 1 reports the lower figure.

On the capacity task we recomputed the same quantities on every all-violating group that the runs at the hard level logged. These runs logged 26,924 such groups. The verdict cancels in 26,924 of them under plain GRPO and in 26,924 under Dr. GRPO. The identical-reward filter would keep 3,389 of them, which are exactly the groups with spread, and the verifier filter would keep 0. NGRPO with the feasible anchor was trained only on the design with a task term, and the runs with a task term logged 24,637 of these groups. In them the verdict changes the NGRPO advantage in 3,977 groups. Of these, 3,389 have reward spread and 588 have a tied reward that crosses the anchor. In the remaining 20,660 groups the added verdict and no penalty differ only through ϵ. These replayed counts differ from the kept groups in Table 22, which count what each filter let into its own training run.

The literal anchor was not trained on this task. It depends on the instance, so we report it by instance. The literal anchor of the design with a task term takes the values $3.65, 7.50, 1.54, 4.09, 3.82, 6.56$ on the 6 instances, in the fixed instance order of Table 8, when the verdict has weight 5. Each value is exactly 5 below the anchor with no penalty, because each instance's highest reward belongs to an infeasible answer. In the runs with a task term, the verdict cancels in every one of the 24,637 all-violating groups. The group counts on the 6 instances are $4{,}045, 4{,}140, 4{,}058, 4{,}171, 4{,}180, 4{,}043$. The anchor of the design with no task term does not move with the verdict. The verdict changes the advantage in every one of the 2,287 all-violating groups in the runs with no task term.

On the capacity task the verifier filter took $4, 13, 6$ optimizer steps per seed, and its policy did not change on any other step. Every group it kept held both a feasible member and a violator. The identical-reward filter took $4, 14, 32$ steps per seed. Among these remedy runs, the count of groups with no feasible member does not separate success from failure. NGRPO with the feasible anchor logged $41, 48, 47$ such groups per seed and ended infeasible on every seed. The verifier filter logged $122, 735, 94$ and ended feasible on every seed. Feasibility here is read at the final evaluation. The plain verdict first reached zero violation at evaluation steps $80, –, 110, –, 10$ per seed, and NGRPO with the feasible anchor first reached it at steps $240, –, –$. A dash marks a seed that never did. The plain verdict on seed 4 and NGRPO on seed 0 then ended above zero violation.

Dropping the divisor keeps the mean: Liu et al. (2025b) remove σ_K from Eq. (1) and keep μ_K. Mean-centering alone removes a group-constant term, so under Dr. GRPO the verdict still cancels in every all-violating group.

The NGRPO anchors: For tied rewards r and a fixed anchor a, the real members' common advantage under NGRPO's augmented-group normalization approaches $\mathrm{sign}(r-a)/\sqrt{K}$ as ϵ vanishes. The advantage is zero when $r = a$. A verdict changes its sign when it moves r across a. At finite ϵ, a verdict can also change the magnitude without crossing the anchor. The speed task therefore tests two anchors, the feasible anchor set to the best feasible reward and the literal anchor.

Alternative constraint mechanisms: A multiplicative gate multiplies the task term by the verdict, so an all-violating group earns one constant and has zero advantages. We trained this variant only

Table 19: Speed task, hard regime: the remedy arms and the reward designs of Section 4, per seed over 3 seeds. Violation, residual, task value and unparseable come from the final evaluation. Residual and task value average only parsed completions, and a dash means none parsed. Capacity task value divides by the exact feasible optimum, while speed task value uses the clipped speed reward. Parsed task value differs from the all-completion feasible value in Appendix C.4. Kept groups counts the groups that entered the update, out of $1{,}800$, and skipped steps counts the optimizer steps a filter skipped, out of 300. An arm without a filter keeps every group and skips no step. In the verdict $\times 5$ rows the verdict weight is five times larger, above the crossover of 20. Where NGRPO with the feasible anchor repeats a residual across seeds, those seeds give the same answer to every evaluation completion.

Reward design	seed	violation	residual	task value	unparseable	kept groups	skipped steps
verdict + task	0	1.00	183,692	20.00	0.00	1,800	0
	1	1.00	121,775	20.00	0.00	1,800	0
	2	1.00	126,694	20.00	0.00	1,800	0
task only	0	1.00	181,622	20.00	0.00	1,800	0
	1	1.00	97,112	20.00	0.00	1,800	0
	2	1.00	126,694	20.00	0.00	1,800	0
residual + task	0	0.00	0.00	0.50	0.00	1,800	0
	1	0.00	0.00	0.50	0.00	1,800	0
	2	0.00	0.00	0.50	0.00	1,800	0
verdict $\times 1000$ + task	0	0.00	0.00	0.50	0.00	1,800	0
	1	0.00	0.00	0.00	0.00	1,800	0
	2	0.00	0.00	0.20	0.00	1,800	0
verdict + task, Dr. GRPO	0	1.00	59,751	20.00	0.00	1,800	0
	1	1.00	59,732	20.00	0.00	1,800	0
	2	1.00	90,406	20.00	0.00	1,800	0
verdict + task, NGRPO, feasible anchor	0	1.00	19,999	20.00	0.00	1,800	0
	1	1.00	428	20.00	0.00	1,800	0
	2	1.00	19,999	20.00	0.00	1,800	0
verdict + task, NGRPO, literal anchor	0	1.00	97,044	20.00	0.00	1,800	0
	1	1.00	64,501	20.00	0.00	1,800	0
	2	1.00	32,291	20.00	0.00	1,800	0
verdict + task, verifier filter	0	1.00	85,420	20.00	0.00	2	298
	1	1.00	4,839	19.69	0.00	2	299
	2	0.00	0.00	0.11	0.00	3	298
verdict + task, identical-reward filter	0	1.00	32,745	20.00	0.00	7	298
	1	1.00	103,626	20.00	0.00	10	295
	2	1.00	51,529	20.00	0.00	13	292
verdict $\times 5$ + task	0	0.00	0.00	0.50	0.00	1,800	0
	1	0.00	0.00	0.00	0.00	1,800	0
	2	0.00	0.00	0.20	0.00	1,800	0
verdict $\times 5$ + task, NGRPO, feasible anchor	0	0.00	0.00	0.50	0.00	1,800	0
	1	0.00	0.00	0.50	0.00	1,800	0
	2	0.00	0.00	0.50	0.00	1,800	0
verdict $\times 5$ + task, verifier filter	0	0.00	0.00	0.05	0.00	4	297
	1	0.00	0.00	0.00	0.00	7	298
	2	0.00	0.00	0.19	0.00	4	297

Table 20: Speed task, hard regime: the residual at weight 0.01, below its ordering threshold, beside the residual at weight 5 and the verdict plus task design at weight 5, per seed over 3 seeds. The columns mean what they mean in Table 19.

Reward design	seed	violation	residual	task value	unparseable	kept groups	skipped steps
verdict + task	0	1.00	183,692	20.00	0.00	1,800	0
	1	1.00	121,775	20.00	0.00	1,800	0
	2	1.00	126,694	20.00	0.00	1,800	0
residual + task	0	0.00	0.00	0.50	0.00	1,800	0
	1	0.00	0.00	0.50	0.00	1,800	0
	2	0.00	0.00	0.50	0.00	1,800	0
residual + task, below the ordering threshold	0	0.00	0.00	0.50	0.00	1,800	0
	1	0.00	0.00	0.50	0.00	1,800	0
	2	0.00	0.00	0.50	0.00	1,800	0

Table 21: Capacity task, hard level: the residual at weight 0.04, below its mixed-group ordering threshold on 4 of the 6 instances, beside the residual at weight 5 and the verdict plus task design at weight 5, per seed over 5 seeds. The columns mean what they mean in Table 22, and the light-verdict and residual runs are the originals of the mixed cohort, which the committed entry named as the comparison runs.

Reward design	seed	violation	residual	task value	unparseable	kept groups	skipped steps
verdict + task	0	1.00	31.32	9.53	0.00	1,800	0
	1	1.00	31.32	9.53	0.00	1,800	0
	2	1.00	–	–	1.00	1,800	0
	3	1.00	31.32	9.53	0.00	1,800	0
	4	0.99	0.00	1.00	0.99	1,800	0
residual + task	0	0.00	0.00	0.00	0.00	1,800	0
	1	0.00	0.00	0.00	0.00	1,800	0
	2	1.00	–	–	1.00	1,800	0
	3	0.00	0.00	0.00	0.00	1,800	0
	4	0.00	0.00	0.00	0.00	1,800	0
residual + task, below the ordering threshold on four instances	0	0.50	0.34	0.69	0.00	1,800	0
	1	0.50	0.34	0.69	0.00	1,800	0
	2	0.00	0.00	0.00	0.00	1,800	0
	3	0.00	0.00	0.00	0.00	1,800	0
	4	0.00	0.00	0.00	0.00	1,800	0

Table 22: Capacity task, hard level: the remedy arms and the capacity task's reward designs, per seed, with 5 seeds for each reward design and 3 for each remedy arm. The columns mean what they mean in Table 19. The verdict weight 5 sits below the crossover of 12.50. The paragraph on remedy outcomes on the capacity task in Section 6 gives the answers behind the remedy arms' residuals of zero. The verdict plus task run on seed 4 also shows a residual of zero. Almost none of its answers parse, and its residual and task value average only the answers that do. Every other reward-design run with a residual of zero answers all 96 evaluation completions with no parsed item. That statement excludes the remedy arms. Seeds 0 to 2 of the reward designs are the single-commit reruns of Appendix D.3, at the commit of seeds 3 and 4 and of the remedy runs.

Reward design	seed	violation	residual	task value	unparseable	kept groups	skipped steps
verdict + task	0	0.00	0.00	0.00	0.00	1,800	0
	1	1.00	–	–	1.00	1,800	0
	2	0.00	0.00	0.00	0.00	1,800	0
	3	1.00	31.32	9.53	0.00	1,800	0
	4	0.99	0.00	1.00	0.99	1,800	0
task only	0	1.00	31.32	9.53	0.00	1,800	0
	1	1.00	31.32	9.53	0.00	1,800	0
	2	1.00	23.19	7.36	0.00	1,800	0
	3	1.00	31.32	9.53	0.00	1,800	0
	4	0.46	0.53	0.42	0.18	1,800	0
residual + task	0	0.00	0.00	0.00	0.00	1,800	0
	1	0.00	0.00	0.00	0.00	1,800	0
	2	1.00	–	–	1.00	1,800	0
	3	0.00	0.00	0.00	0.00	1,800	0
	4	0.00	0.00	0.00	0.00	1,800	0
verdict $\times 1000$ + task	0	1.00	–	–	1.00	1,800	0
	1	0.00	0.00	0.00	0.00	1,800	0
	2	0.00	0.00	0.00	0.00	1,800	0
	3	1.00	31.32	9.53	0.00	1,800	0
	4	0.00	0.00	0.00	0.00	1,800	0
verdict only	0	0.00	0.00	0.00	0.00	1,800	0
	1	0.00	0.00	0.00	0.00	1,800	0
	2	0.00	0.00	0.00	0.00	1,800	0
	3	0.00	0.00	0.00	0.00	1,800	0
	4	1.00	–	–	1.00	1,800	0
residual only	0	0.00	0.00	0.00	0.00	1,800	0
	1	0.00	0.00	0.00	0.00	1,800	0
	2	0.00	0.00	0.00	0.00	1,800	0
	3	0.00	0.00	0.00	0.00	1,800	0
	4	0.00	0.00	0.00	0.00	1,800	0
verdict + task, Dr. GRPO	0	0.00	0.00	0.00	0.00	1,800	0
	1	1.00	–	–	1.00	1,800	0
	2	0.00	0.00	0.00	0.00	1,800	0
verdict + task, NGRPO, feasible anchor	0	0.03	0.00	0.00	0.03	1,800	0
	1	0.03	0.00	0.00	0.03	1,800	0
	2	0.06	0.00	0.00	0.06	1,800	0
verdict + task, verifier filter	0	0.00	0.00	0.00	0.00	10	296
	1	0.00	0.00	0.00	0.00	27	287
	2	0.00	0.00	0.00	0.00	13	294
verdict + task, identical-reward filter	0	1.00	31.32	9.53	0.00	23	296
	1	1.00	31.32	9.53	0.00	49	286
	2	0.00	0.00	0.39	0.00	85	268

on the grade-school-math task of Appendix D.4, where every arm ended at zero accuracy on the random held-out set. Constrained decoding (Geng et al., 2023) enforces a decidable constraint at the sampler, so no completion can violate it, and it needs a constraint the sampler can check on every prefix. Kim & Yun (2026) keep the outcome advantage from a Lean checker, add a tactic-level process advantage from the same checker, and report that this outperforms outcome-only baselines in most settings.

Filter controls that match optimizer updates: The verifier filter takes few optimizer steps, so its recovery could come from its update schedule alone. For each verifier-filter run we trained control runs that copy the filter's own update steps and keep the same number of groups at each step, with the kept prompts drawn without reading rewards or verdicts. There are 42 controls in all, 6 draws per seed on the speed task at weights 5 and 25 and 2 draws per seed on the capacity task at weight 5. The predictions Y0 to Y6 were committed before the runs and are listed in Appendix E.1. Every control took its steps at exactly the listed steps and kept exactly the listed prompts (0 failures), and on the speed task every control logged the same answers, rewards and advantages as its filter run through its first update step (0 failures). On the capacity task the controls reached feasibility on 0 of 6 runs, with final violation rates of $1.0, 0.927, 0.156, 0.0104, 1.0, 0.552$, against the filter's 3 of 3 seeds. On the speed task they reached feasibility on 2 of 18 runs at weight 5 and on 5 of 18 at weight 25. By seed, the feasible controls number $0, 0, 2$ at the lower weight and $0, 3, 2$ at the higher one, which are seed 2, draws 3 and 5 at the lower weight and seed 1, draws 2, 3 and 5, and seed 2, draws 3 and 4 at the higher one. Of the 22 controls whose first update step keeps no mixed group, 1 reached feasibility, against a committed prediction of none. That run, at weight 25 on the speed task, took its first update at step 1 on 2 groups and ended with a final violation rate of 0.0. Of the 42 controls, 24 took at least one update step with an exactly zero gradient, and there were 40 such steps. The controls retained 928 completions and $34{,}463$ tokens in all, which is 528 completions and $2{,}476$ tokens on the speed task and 400 completions and $31{,}987$ tokens on the capacity task. The identical-reward filter needs no such control. It drops only groups whose advantages are exactly zero, so at the same weights and samples its clipped gradient equals the plain arm's whenever both gradient norms exceed the clip norm, and it differs from the plain arm only by skipping the steps at which the plain arm's gradient is exactly zero. That difference is an update-count difference by construction. The design cannot show that keeping mixed groups is sufficient in general, because the controls copy the filter's own update steps and 3 filter seeds carry the comparison.

D.2 Capacity remedies above the crossover and a strict held-out protocol

This extension was requested after the earlier results and reviewer feedback were known, and its analysis plan was recorded in `PREDICTIONS.md` before any new policy sampling. The plan predicted no direction of recovery. The 68 new runs of one array all completed, and the complete-matrix analysis validated them together with 17 reused earlier runs, with 5 seeds per condition. A second scorer that could not see the analysis recomputed every run's endpoints from the raw run files and agreed on every compared value. The runs keep their own records, and none of them enters the capacity tables of Section 5 or Appendix D.1.

The original protocol above the crossover: On the capacity task's hard level, five algorithms were trained at verdict weights 5 and 25. The second weight lies above the crossover of Appendix A.4. The seeds that reach feasibility at the two weights are 0 and 4 for plain GRPO, 5 and 4 for the verifier filter, 2 and 2 for the identical-reward filter, 4 and 4 for Dr. GRPO, and 0 and 0 for NGRPO with the feasible anchor, each of 5. Of the 25 feasible seeds, 24 answer every evaluation prompt with an empty selection. Of the 50 runs, 0 have every output useful and feasible, and the largest condition mean of the useful feasible rate is 0.133. Table 23 gives every cell.

The strict protocol: The strict protocol uses the 6 original training instances and 24 disjoint held-out instances with 16 samples each. It fixes the binary weights 5 and 25 and a reward divisor of 26 from the training setup, and a completion earns useful value only when it is valid, nonempty and feasible. Four cells compare completion-level GRPO and PPO at the two weights at group size four, a fifth cell trains graded GRPO at weight 5, and two cells compare GRPO and PPO at weight 25 and group size eight. Every run samples 7200 training completions with 4 actor optimization epochs per rollout, and PPO uses a learned table of prompt values captured before the current rewards are

observed. The mean normalized useful feasible value on the held-out instances is exactly zero in 4 of the 7 cells and at most 0.043 in the others, and the smallest nonzero cell mean rounds to zero at the precision of Table 24. Each nonzero cell mean comes from 1 seed, and the two largest come from one seed each that gave the same one-item answer to every held-out prompt. The PPO cells parsed 0 of their 5760 held-out completions, and 85.8 percent of all strict held-out completions failed to parse. Table 24 gives every cell, Table 25 gives every seed, and Figure 6 shows them.

5 seeds support descriptive comparisons only. The original protocol's verdict passes an empty parsed selection. The weight contrast pairs new runs with reused records from earlier arrays, and their initial checkpoints cannot be shown equal. The strict and original protocols differ in several settings at once, and PPO and GRPO differ in both baseline and normalization, so neither comparison isolates one factor as the cause of a difference.

Table 23: The historical capacity comparison uses five training seeds per method and weight. Pass seeds satisfy the recorded verdict on every final evaluation completion. The historical parser admits empty selections. Useful value is normalized by each instance's optimum and assigns zero to unparseable, empty, and infeasible outputs. Every evaluation completion enters the denominator. The runs use the same six training instances, one actor update per rollout, and 7200 training completions. Values show means and sample standard deviations across seeds.

	$w = 5$		$w = 25$	
Method	Pass seeds	Useful value	Pass seeds	Useful value
GRPO	0/5	0.002 ± 0.005	4/5	0.000 ± 0.000
Dr. GRPO	4/5	0.000 ± 0.000	4/5	0.059 ± 0.132
Verifier filter	5/5	0.000 ± 0.000	4/5	0.011 ± 0.024
Identical-reward filter	2/5	0.077 ± 0.173	2/5	0.059 ± 0.132
NGRPO (feasible anchor)	0/5	0.000 ± 0.000	0/5	0.000 ± 0.000

The historical verdict admits parsed empty selections. Useful value assigns zero to unparseable, empty, and infeasible outputs. Values show the mean and sample SD across five training seeds.

Table 24: The strict comparison reports final performance on 24 disjoint held-out instances with 16 samples per instance. Useful value assigns zero to invalid, empty, and capacity-violating outputs. Useful feasibility requires a valid nonempty capacity-feasible selection. Every rate includes all completions in its denominator. Each condition contains 5 training seeds. Values show means and sample standard deviations.

Condition (w, K)	Useful value	Useful fraction
GRPO $(5, 4)$	0.000 ± 0.000	0.000 ± 0.000
GRPO $(25, 4)$	0.000 ± 0.000	0.001 ± 0.001
PPO $(5, 4)$	0.000 ± 0.000	0.000 ± 0.000
PPO $(25, 4)$	0.000 ± 0.000	0.000 ± 0.000
Graded GRPO $(5, 4)$	0.043 ± 0.097	0.067 ± 0.149
GRPO $(25, 8)$	0.043 ± 0.097	0.067 ± 0.149
PPO $(25, 8)$	0.000 ± 0.000	0.000 ± 0.000

Condition (w, K)	Parse failure	Empty	Over capacity	Tokens
GRPO $(5, 4)$	0.600 ± 0.548	0.000 ± 0.000	0.400 ± 0.548	69.0 ± 56.5
GRPO $(25, 4)$	0.999 ± 0.002	0.000 ± 0.000	0.001 ± 0.001	83.8 ± 53.9
PPO $(5, 4)$	1.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	112.0 ± 35.5
PPO $(25, 4)$	1.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	127.3 ± 1.1
Graded GRPO $(5, 4)$	0.605 ± 0.541	0.000 ± 0.000	0.328 ± 0.462	37.2 ± 54.0
GRPO $(25, 8)$	0.800 ± 0.447	0.000 ± 0.000	0.133 ± 0.298	71.9 ± 64.5
PPO $(25, 8)$	1.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	127.7 ± 0.6

The pairs give weight and group size. Rows use binary rewards unless labeled graded. Every rate uses all held-out completions as its denominator. Over capacity counts only parsed selections that exceed capacity. Tokens include a terminating stop token. Values show the mean and sample SD across five training seeds.

Table 25: The table lists every seed of the strict protocol at the final evaluation on the 24 held-out instances. Useful value and useful fraction assign zero to invalid, empty and capacity-violating outputs, and parse failure and empty are rates over all held-out completions.

Condition (w, K)	Seed	Useful value	Useful fraction	Parse failure	Empty
GRPO $(5, 4)$	0	0.000	0.000	0.000	0.000
GRPO $(5, 4)$	1	0.000	0.000	1.000	0.000
GRPO $(5, 4)$	2	0.000	0.000	1.000	0.000
GRPO $(5, 4)$	3	0.000	0.000	0.000	0.000
GRPO $(5, 4)$	4	0.000	0.000	1.000	0.000
GRPO $(25, 4)$	0	0.000	0.000	1.000	0.000
GRPO $(25, 4)$	1	0.000	0.000	1.000	0.000
GRPO $(25, 4)$	2	0.000	0.000	1.000	0.000
GRPO $(25, 4)$	3	0.001	0.003	0.995	0.000
GRPO $(25, 4)$	4	0.000	0.000	1.000	0.000
PPO $(5, 4)$	0	0.000	0.000	1.000	0.000
PPO $(5, 4)$	1	0.000	0.000	1.000	0.000
PPO $(5, 4)$	2	0.000	0.000	1.000	0.000
PPO $(5, 4)$	3	0.000	0.000	1.000	0.000
PPO $(5, 4)$	4	0.000	0.000	1.000	0.000
PPO $(25, 4)$	0	0.000	0.000	1.000	0.000
PPO $(25, 4)$	1	0.000	0.000	1.000	0.000
PPO $(25, 4)$	2	0.000	0.000	1.000	0.000
PPO $(25, 4)$	3	0.000	0.000	1.000	0.000
PPO $(25, 4)$	4	0.000	0.000	1.000	0.000
Graded GRPO $(5, 4)$	0	0.000	0.000	1.000	0.000
Graded GRPO $(5, 4)$	1	0.000	0.000	1.000	0.000
Graded GRPO $(5, 4)$	2	0.216	0.333	0.000	0.000
Graded GRPO $(5, 4)$	3	0.000	0.000	1.000	0.000
Graded GRPO $(5, 4)$	4	0.000	0.000	0.026	0.000
GRPO $(25, 8)$	0	0.000	0.000	1.000	0.000
GRPO $(25, 8)$	1	0.000	0.000	1.000	0.000
GRPO $(25, 8)$	2	0.216	0.333	0.000	0.000
GRPO $(25, 8)$	3	0.000	0.000	1.000	0.000
GRPO $(25, 8)$	4	0.000	0.000	1.000	0.000
PPO $(25, 8)$	0	0.000	0.000	1.000	0.000
PPO $(25, 8)$	1	0.000	0.000	1.000	0.000
PPO $(25, 8)$	2	0.000	0.000	1.000	0.000
PPO $(25, 8)$	3	0.000	0.000	1.000	0.000
PPO $(25, 8)$	4	0.000	0.000	1.000	0.000

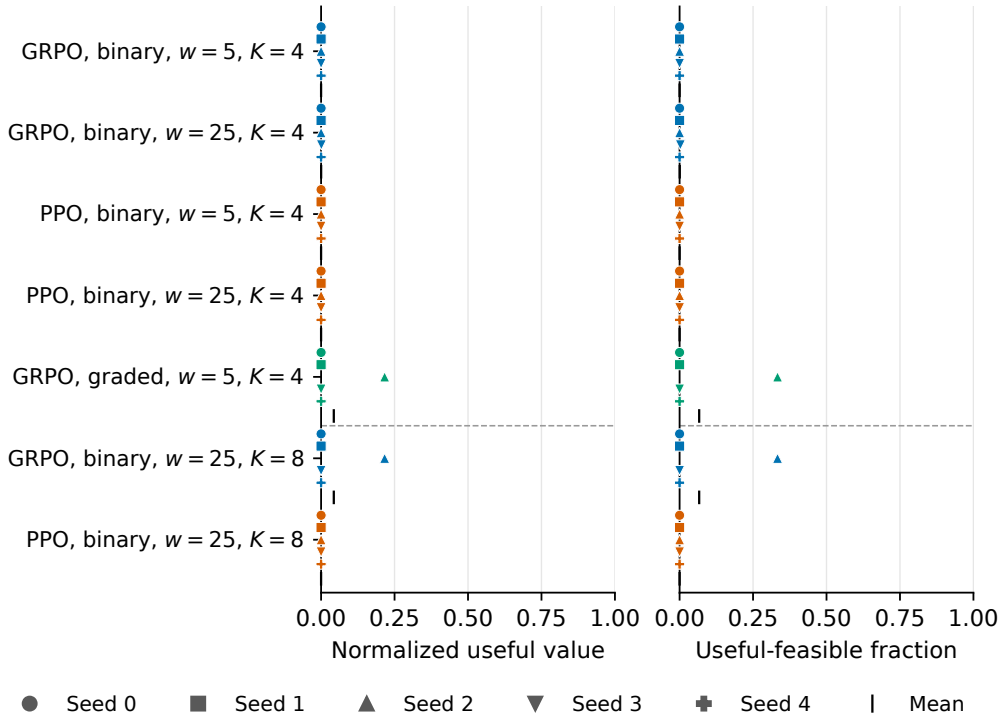

Figure 6: Each marker shows one training seed's final held-out normalized useful value or useful-feasible fraction. Invalid, empty, and capacity-violating outputs receive zero useful value. Vertical offsets distinguish all five seeds even when values coincide. Black ticks show seed means. Every final policy uses the same 24 held-out instances. The rows below the divider use group size eight and halve the rollout and actor-update counts while keeping 7200 training completions.

D.3 SINGLE-COMMIT RERUN OF THE CAPACITY SEEDS

The original capacity runs of seeds 0 to 2 were made at two earlier commits, and the remedy, KL and seed 3 and 4 runs at a later one, so an earlier draft's capacity tables combined runs from more than one commit under the drift waiver of Appendix B.4. We reran all 36 original runs of seeds 0 to 2, the six reward designs at both levels, from a clean clone at commit `364ad77e`, whose training-source digest is `d5ea8063`. Every rerun finished, no seed was added, and 0 rows were left unrun. The predictions SC0 to SC6 for this rerun were committed before it was submitted and are listed in Appendix E.1. Every capacity table, figure and generated count built from the reward-design, remedy, KL and audit records and from the capacity runs of the matched-weight replay comes from the single-commit cohort, in which every run records commit `364ad77e`. The below-threshold comparison of Table 21 keeps the original light-verdict and residual runs that its committed entry named as the comparison runs, and the extension of Appendix D.2 keeps its own records.

In every pair the logged answers, rewards and advantages of the 6 groups at step 1 are identical (36 of 36 pairs), and no pair logs identical answers with different rewards or advantages (0 pairs). The final parameter checksums are equal in 0 pairs. In 16 pairs the logged training answers never differ over the run, and in the other 20 pairs the first difference falls between steps 6 and 73. Table 26 lists every pair.

A pair agrees on a label when the original and the rerun end with the same label at the final evaluation. Feasibility agrees in 30 of 36 pairs, the unparseable label agrees in 29 pairs, and the outcome class agrees in 27 pairs. Each count is reported beside two reference points that the committed entry fixed. Full reproducibility gives agreement in every pair. Independent draws give each pair the share of its arm's other recorded seeds at the same level that carry the original's label. Over the 36 pairs they give an expected 25.75 agreements on feasibility with a central ninety-five percent range of 22 to 29, 23.25 on the unparseable label with a range of 19 to 27, and 18.75 on the outcome class with a range of 14 to 23. The ranges rest on three to five seeds per arm, so they are descriptive reference points and not tests with known error rates. Band R is agreement in every pair, band M lies above the range and below full agreement, band X lies inside the range, and band B lies below it. The three counts fall in bands M, M and M. On feasibility, 5 pairs went from infeasible to feasible and 1 went the other way. On the unparseable label, 2 pairs went from parseable to unparseable and 5 went the other way. These counts are recorded without a test. The rerun shows that the seed does not fix every outcome class across GPU jobs, and it cannot name the cause, because no rerun at the old commit exists.

D.4 MIXED-REWARD GRPO ON GRADE-SCHOOL MATH

We trained Qwen2.5-0.5B-Instruct on GSM8K (Cobbe et al., 2021) with a mixed reward to test whether the mechanism extends beyond the synthetic tasks. A completion earns $w_{\text{task}}\,T - wF$. Here F is one when the strict parser finds no boxed answer or the answer differs from the reference, and zero otherwise. T is one minus the completion length over the generation limit, w is 2, and w_{task} is one in every arm except the verdict-only arm, where it is zero. T lies between zero and one, so every correct completion outranks every incorrect one in every group, and every arm sits above the crossover on every prompt. The arms are plain GRPO, the verifier filter, Dr. GRPO, a gate whose reward is $w_{\text{task}}\,T(1-F) - wF$, and the verdict alone with w_{task} at zero. The predictions W0 to W7 were committed before any run, together with calibration rules that fixed every setting from base-policy samples only. The training pool holds 387 problems chosen by those rules, the hard held-out set holds 200 test problems, and the random held-out set holds 200. Each run takes 300 steps of 6 groups of 4 completions at a generation limit of 512 tokens and a learning rate of 10^{-5}, with 3 seeds per arm and 15 runs in all. The larger of the two planned model sizes failed the entry's calibration floors and was not run.

Before training, the policy scored 43.5 percent on the random held-out set and 16.9 percent on the hard set. At the final evaluation, 15 of the 15 runs score zero on the random set and 14 score zero on the hard set, where the largest final count of correct answers is 1 of 800 completions. At least 0.993 of every run's final held-out completions fail the strict parse, and 9 runs fail it on every completion. Plain GRPO and Dr. GRPO end with a mean completion length of 1.0 to 2.0 tokens, and the gate arm ends at 507.9 to 512.0 tokens, at the generation limit. Table 27 lists every run.

Table 26: The table lists the original capacity runs of seeds 0 to 2 beside their single-commit reruns. The class at the final evaluation is FE for a run whose completions all pass with only empty selections, FI for a feasible run with at least one selected item, U for a run whose completions all fail to parse, and V for every other run. The last column gives the first training step at which the logged answers of the two runs differ, and a dash marks a pair whose answers never differ in 300 steps. Feasibility agrees in 30 of 36 pairs (band M), the unparseable label in 29 (band M) and the outcome class in 27 (band M).

level	arm	seed	original class	rerun class	first differing step
hard	`binary`	0	V	FE	18
		1	V	U	6
		2	U	FE	31
hard	`binary_1000x`	0	FE	U	18
		1	FE	FE	8
		2	FE	FE	12
hard	`graded`	0	FE	FE	–
		1	FE	FE	73
		2	U	U	–
hard	`graded_no_task`	0	FE	FE	–
		1	FE	FE	–
		2	FE	FE	–
hard	`none`	0	V	V	20
		1	U	V	7
		2	V	V	15
hard	`verdict_only`	0	FE	FE	–
		1	FE	FE	29
		2	FE	FE	–
easy	`binary`	0	FI	FI	–
		1	FI	FI	24
		2	U	FI	23
easy	`binary_1000x`	0	FI	FI	–
		1	FI	FI	–
		2	U	FI	29
easy	`graded`	0	FI	FI	–
		1	FI	FI	–
		2	FI	FI	–
easy	`graded_no_task`	0	U	U	58
		1	FI	FI	–
		2	FE	FE	–
easy	`none`	0	V	V	13
		1	FE	FE	–
		2	U	FI	10
easy	`verdict_only`	0	FI	FI	26
		1	FE	FE	67
		2	FI	FE	12

The outcome rule compares the gate with plain GRPO on final held-out hard accuracy over the seeds on which both finish. Outcome Z applies, with 0 of 3 seeds on which the gate ends above plain GRPO. The verifier filter ends above plain GRPO on 1 of 3 seeds and Dr. GRPO on 0 of 3. The gate and plain GRPO both end at zero on the hard set on every seed, so the outcome is a tie at zero. These runs therefore do not test whether removing the length term from incorrect answers raises accuracy. Two records explain the tie. In the plain runs, every group became all-fail once the answer format was lost. Only 1.6 to 3.0 percent of the all-fail groups kept any reward spread, against the committed threshold of 99 percent, so the length term was the only signal left and the completions shrank to one or two tokens. The gate arm zeroes every all-fail group, and it had a nonzero gradient on only 12 to 17 of its 300 optimizer steps. The trainer calls the optimizer whenever any group is kept, so on the other 283 to 288 steps only the optimizer's stored momentum moved the weights. The verifier filter skipped those steps and took 58 to 69 steps. The scope is one task, one task term, one size of one model family and 3 seeds. The gate also ties the incorrect members of mixed groups, the verifier filter skips steps, and Dr. GRPO still cancels the verdict. A test of the gate on this task would need a design that keeps the answer format, and it would need a new committed entry.

Table 27: The table lists every run of the grade-school-math block at the final evaluation. Hard and random accuracy are over the two held-out sets, length is the mean completion length in tokens over the final held-out completions, and parse failure is the strict parser's failure rate over them.

Arm	seed	hard accuracy	random accuracy	mean length	strict parse failure
plain GRPO	0	0.000	0.000	1.0	1.000
	1	0.000	0.000	2.0	1.000
	2	0.000	0.000	2.0	1.000
verifier filter	0	0.000	0.000	498.2	0.999
	1	0.000	0.000	291.9	0.998
	2	0.001	0.000	512.0	0.993
Dr. GRPO	0	0.000	0.000	2.0	1.000
	1	0.000	0.000	1.0	1.000
	2	0.000	0.000	1.0	1.000
gate	0	0.000	0.000	512.0	0.994
	1	0.000	0.000	507.9	1.000
	2	0.000	0.000	511.2	1.000
verdict only	0	0.000	0.000	503.8	0.999
	1	0.000	0.000	159.7	1.000
	2	0.000	0.000	189.0	0.998

E PREDICTIONS, OUTCOMES AND FURTHER RELATED WORK

E.1 EVERY PREDICTION AND HOW IT CAME OUT

We wrote every prediction for this paper before the runs it concerns, and Table 28 lists each with its timing and its outcome. Preregistration fixes the research questions and the analysis plan before the outcomes are observed, and it separates predictions from postdictions (Nosek et al., 2018). The exact wording of the predictions of the later blocks, with the rules that score them, is in `PREDICTIONS.md` in the supplement. The speed-task and capacity-task predictions P1 to P4 and P3-K to P7-K come from local planning files that are not released, so this appendix quotes them as first written. The sections and tables named in the last column carry the generated counts that score each prediction. Timing is stated relative to the runs, because the planning files are local and not released. The anonymized copy of the record is a single commit, so its dates cannot be checked during review either. Regime A is the speed task's hard limit, and Regime B is its easy one. A gate is a check that must hold before the block's other predictions are read.

One note qualifies the table. C1-4 held on the original capacity runs and failed on the single-commit cohort that the paper now reports, because the rerun plain verdict reaches the same residual as both remedies on the comparable seeds.

Each of the nine earliest predictions is given below as first written, with arm letters replaced by arm names and semicolons replaced by periods, and with its outcome.

Speed task, written before the first training run: P1: "In every all-violating group, [verdict + task] advantages equal [task only] advantages to float64 precision." The replay passed the analysis tolerance of 1.00×10^{-9}, with 0 failures in 21,687 all-violating groups and a largest disagreement of 1.11×10^{-11}. That discrepancy exceeds the binary64 machine epsilon of 2.22×10^{-16}, so the quoted wording should not be read as equality at machine precision. P2: "[verdict + task] and [verdict $\times 1000$ + task] give identical advantage vectors in every group and identical curves under the same seed." It failed. The two arms agree in every constant-verdict group, with 0 failures, and differ in mixed groups. The heavy verdict reaches feasibility on 3 of 3 hard-regime seeds, and the light verdict reaches it on 0. P3: "In Regime A, [residual + task] lowers mean residual and violation rate. [verdict + task], [task only] and [verdict $\times 1000$ + task] do not." It failed, because the heavy verdict reached zero violation by evaluation steps $10, 10, 10$. P4: "In Regime B, [verdict + task] does lower the violation rate." It failed, because that rate rose from 0.46 at step zero to 1.000 at the end.

Speed task, the wording committed later: The project's ledger restates P1 to P4 in wording committed after the runs. That wording limits P2 to groups that share one verdict and predicts no direction for the light verdict in P4. Under it P2 holds, and the ledger records P4 as partly failed. The main text reports the first wording.

Capacity task, written after the speed runs began and before any capacity run: P3-K: "At the hardest level the heavy verdict does not reach feasibility within 300 steps and the graded residual does. At the easiest level both do, as in Regime A." It failed. At the hard level the heavy verdict reaches feasibility on 3 of 5 seeds and the residual reaches it on 4. At the easy level the residual reaches feasibility on 4 of 5 seeds, and the heavy verdict reaches it on 5 of 5. The residual ends unparseable on its remaining seed. Two calibration notes were recorded before these runs. The first expected the heavy verdict to succeed at the hard level. A corrected second note found far fewer mixed groups at that level and withdrew the expectation. P4-K: "The KL at which violation reaches zero rises with hardness for the heavy verdict and stays flat for the residual." Averaged over the seeds that reach zero violation, the heavy verdict reaches it at a reference log-ratio of 42.84 at the easy level and 152.93 at the hard level. The residual reaches it at log-ratios of 19.58 and 31.83. The heavy verdict's values from seed 0 to seed 4 are $3.02, 3.03, 36.69, 168.12, 3.34$ at the easy level and $–, 198.42, 144.61, –, 115.76$ at the hard level. The residual's values are $12.64, 3.48, 2.99, 11.98, 66.81$ and $10.44, 44.34, –, 22.45, 50.08$, and a dash marks a seed that never reaches zero violation. Both means rise with hardness, so on the means the heavy verdict's half held and the residual's half failed. The values of both designs also overlap across the two levels, so 5 seeds settle neither half. P5-K: "The graded arm lands at higher task reward than the heavy verdict at equal violation, because the residual points to the capacity boundary where value is maximized." It failed (Section 5). P6-K, first version: "In every all-violating group the group reward variance is strictly positive under all four arms, so the zero-variance diagnostic [...] never fires, while the verdict is nonetheless removed exactly. Log the per-step group reward variance and the fraction of groups a DAPO-style all-same-reward filter would drop. Predict that fraction is near zero while the verdict's contribution to the advantage is exactly zero." It failed on both tasks. Of the 21,687 all-violating groups the speed runs sampled, 21,596 had zero reward variance, and on the capacity task 23,799 of 25,946 did. P6-K, rewritten before any capacity run: "The zero-variance group fraction over a whole run will be high and similar across every arm, while the all-violating fraction separates them by an order of magnitude, so a DAPO-style all-same-reward filter cannot tell the successful arm from the failed one." It failed on the five-seed cohort. The zero-variance counts of successful and failed runs still overlap at both levels, as predicted. The all-violating count no longer separates those runs. At the hard level a successful run logs up to 579 such groups, and a failed run logs as few as 30. DAPO's own filter uses verifier accuracy (Section 6). P7-K: "Count the all-violating groups that still have strictly positive reward variance [...]. Predict the window is longer for a 32-token subset than for a 6-token integer, because the output space is richer and the policy mode-collapses more slowly." It held. The capacity window holds 7 to 348 groups per run and closes by step 108, against 5 to 6 by step 4 on the speed task. The prediction assumed a 32-token answer, and the runs allowed 128 new tokens.

Table 28: The table lists every prediction, when it was written, and how it came out. The identifiers of the later blocks follow `PREDICTIONS.md`, and P1 to P4 and P3-K to P7-K follow the planning files quoted above. A failed prediction is reported as failed wherever the paper scores it.

Block, written before	**Outcome**	**Scored in**
Speed task, before the first training run	P1 held. P2, P3 and P4 failed.	Section 4
Capacity task, before any capacity run	P3-K, P5-K and both versions of P6-K failed. P4-K split. P7-K held.	Section 5, Appendix C.2
Speed remedies, before the remedy runs	S1, the gate, held with a correction committed before the runs. S2 to S8 held.	Section 6, Appendix D.1, Table 19
Capacity remedies, before the remedy runs	C1-1, the gate, held. C1-2 and C1-3 failed. C1-4 held on the original runs and failed on the single-commit cohort.	Section 6, Appendix D.1, Table 22
Added capacity seeds, before those runs	R1, R3, R4, R5, R6 and R7 held. R2 failed.	Section 5, Appendix C.2
KL ablation, before those runs	K1 and K2 held. K0, K3 and K4 failed, and K3's failure triggers the committed sentence that the penalty prevented the collapse in these cells.	Section 5, Appendix C.3
Speed residual below its threshold, before the runs	B0, the gate, held. B1 to B5 and B7 held. B6 failed.	Section 7, Appendix B.1, Table 20
Scale-matched residual, before the runs	M0 and M0′, the gates, held. M1 to M5 held. Outcome S.	abstract, Section 1, Section 7, Appendix B.1
Single-commit rerun, before the reruns	SC0, the gate, held. SC1, SC2, SC5 and SC6 held. SC3 is a reporting rule, and the three counts fell in bands M, M and M. SC4 failed.	Section 7, Appendix D.3
Capacity residual below its threshold, before the runs	D0, the gate, held. D1 to D7 held. D8 failed. Outcome S.	Appendix B.1, Table 21
Filter controls, before the control runs	Y0, the gate, held. Y1, Y2, Y3, Y5 and Y6 held. Y4 failed. Outcome I.	Section 6, Appendix D.1
Grade-school math, before the calibration runs	W0, the gate, held. W1, W3, W5 and W6 held. W2 and W4 failed. W7 was not scored. Outcome Z.	Section 7, Appendix D.4
The extension, before any new sampling	No direction was predicted, so nothing is scored.	Appendix D.2

Inclusion of the later blocks in this paper: The entries for the scale-matched residual, the single-commit rerun, the below-threshold residual on the capacity task and the filter controls each said that nothing from the block would enter the paper uploaded at the submission deadline and that their sentences would govern a later revision. A dated entry of `PREDICTIONS.md`, recorded after every outcome above was read, reverses that and puts every block into this paper, with no block selected or omitted by its outcome. It fixes three terms. Each entry's sentences enter with the same wording and generated values, no prescribed sentence is shortened, and the main text made room only by cutting sentences that proposed the experiments these blocks performed and by moving background to an appendix. A correction entry that follows it fixes one clause about the grade-school-math block and adds the failure of W2 to the list of outcomes that cut against the submitted claims, and a later entry records the regrouping of the appendices. A reader should weigh every sentence these blocks added as a choice made after the outcomes were known.

E.2 Extended related work

Baselines and variance reduction: The gradient theorem for REINFORCE (Williams, 1992) assumes a baseline that is conditionally independent of the sampled output given the weights and input. Kool et al. (2019) draw several samples per input, so that the other samples supply each one with a variance-reducing baseline at no extra cost. Ahmadian et al. (2024) train language models with a leave-one-out baseline, in which the other $K - 1$ completions of a prompt give an unbiased estimate of its expected return. A different line of work learns a baseline instead of sampling one. Generalized advantage estimation uses a learned value function to lower the variance of the policy gradient at the cost of some bias (Schulman et al., 2016).

Reinforcement learning for language models and KL control: Reinforcement learning methods for language models form the wider setting of this paper. Most of them add a KL penalty that our main comparisons omit. Guo et al. (2025) train DeepSeek-R1-Zero with GRPO on a rule-based reward that adds an accuracy reward and a format reward with equal weight. Lambert et al. (2024) introduce reinforcement learning with verifiable rewards, in which a completion receives a constant reward only when it is verified correct. Ziegler et al. (2019) add a penalty with expectation $\beta \, \mathrm{KL}(\pi, \pi_{\mathrm{ref}})$ to the reward so that the fine-tuned policy π stays near its initial policy π_{ref}. Rafailov et al. (2023) fit the policy to human preference data with a simple classification loss, which removes the need to sample from the language model during fine-tuning.

Constrained reinforcement learning and reward design: A constrained Markov decision process minimizes one cost under inequality constraints on other costs (Altman, 1999), and our reward subtracts a penalty with a fixed weight. Achiam et al. (2017) give a policy search method with near-constraint satisfaction at each iteration, and our loop enforces no bound on the violation rate at any step. Tessler et al. (2019) treat the penalty coefficient as a Lagrange multiplier that rises until the constraint holds, and our weight w stays fixed throughout training. Ng et al. (1999) show that a potential-based shaping term preserves every optimal policy, and our verifier penalty is meant to change which completions rank highest. Under the definition of Skalse et al. (2022), a feasibility-only reward is a hackable proxy for useful feasible value, because the empty selection maximizes feasibility with no value.

Evaluation practice: Henderson et al. (2018) averaged one algorithm's learning curves over two sets of five random seeds and found that the two averages came from statistically different distributions. Our reported seed counts and Wilson intervals in Appendix C.4 address the same uncertainty, and they do not establish a stable remedy ranking.