LEARNING BEYOND WHAT YOU SAMPLE: OFF-POLICY-AWARE CROSS-MODEL TRAJECTORY EXCHANGE FOR RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) methods such as GRPO rely on successful self-generated trajectories, but finite rollout budgets can produce all-fail groups with no reward-based policy-gradient signal. While additional rollouts improve the chance of success at higher cost, successful trajectories missing from one model's rollouts may already have been discovered by another. Indeed, we observe that heterogeneous models often succeed on complementary prompts, creating opportunities for mutual learning without a designated stronger teacher. To exploit this complementarity, we propose **GRAFT** (**G**ated **R**eplacement of **A**nswer-**F**ailed groups with peer **T**rajectories), an off-policy-aware framework that replaces all-fail groups with informative peer groups. GRAFT transfers both successful and unsuccessful peer responses with peer-computed advantages, while controlling cross-model mismatch through sequence-level compatibility weighting and token-level importance ratio clipping. Across three heterogeneous model pairs and five mathematical reasoning benchmarks, GRAFT consistently improves both models over GRPO with the same per-model rollout budget, gaining 2.1 points on average and up to 4.5 points in model-level average performance. Stored peer trajectories preserve most of the gains, improving over GRPO by 1.8 points on average without simultaneous co-training.

1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) has become a key post-training paradigm for improving the reasoning ability of Large Language Models (LLMs), with substantial gains on many reasoning tasks (Shao et al., 2024; Lambert et al., 2025; Guo et al., 2025). Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and its variants (Yu et al., 2025; Liu et al., 2025b; Zheng et al., 2025; Kim et al., 2026) optimize the policy using relative rewards among sampled responses, raising the likelihood of high-reward trajectories and suppressing low-reward ones. In this on-policy setting, learning reinforces successful reasoning strategies from self-generated trajectories, without fine-grained supervision (Guo et al., 2025; Wen et al., 2026).

The effectiveness of this learning process depends on whether the model discovers a successful trajectory within its rollout budget (Liu et al., 2026a; Yue et al., 2025; Dong et al., 2026). This limitation arises when every sampled response fails, leaving all group-relative advantages at zero. Such groups can be dropped, their prompts replaced, or their rewards reshaped to recover a non-zero learning signal (Yu et al., 2025; Le et al., 2026; He et al., 2026), while increasing the group size improves the odds of sampling a correct trajectory at higher rollout cost. These approaches, however, remain dependent on the learner's own exploration.

The growing diversity of open-source LLMs creates an opportunity to move beyond learning solely from self-generated trajectories: a successful response missing from one model's rollouts may already be present in another's. Yet standard single-model RLVR trains each policy in isolation, leaving these successes unused. We observe this complementarity in independent GRPO runs of SmolLM3-3B-Base (Bakouch et al., 2025) and Qwen3-1.7B-Base (Yang et al., 2025), two models with distinct pretraining histories (Figure 1). SmolLM3-3B-Base solves 47.9% of the prompts on which Qwen3-1.7B-Base fails across all eight rollouts, while Qwen3-1.7B-Base solves 18.7% of

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.

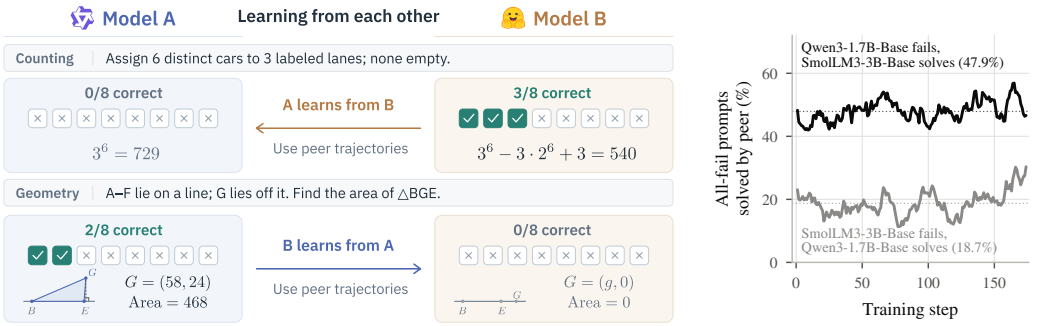


Figure 1: **Complementary successes persist throughout training. (Left)** Illustrative examples where one model solves a prompt its peer fails entirely, so each could learn from the other's trajectories. **(Right)** Fraction of all-fail prompts solved by the peer, with means shown as dotted lines.

SmolLM3-3B-Base's all-fail prompts. These complementary successes suggest that both models could benefit from exchanging verified solutions, without requiring a designated stronger teacher.

Turning complementary successes into learning gains requires deciding *which* prompts warrant peer supervision and *how* to learn from peer responses. These choices interact: all-fail prompts lack a reward-based policy-gradient signal, but successful peer responses may still be poorly matched to the receiver. Conversely, transferring peer responses on prompts with an informative self-generated group changes an update that already has an on-policy learning signal. HACPO (Zhang et al., 2026) accounts for cross-model mismatch but shares peer rollouts beyond receiver-failure prompts, while SGT (Liu et al., 2026b) targets receiver failures through a fixed-weight supervised loss without compatibility-based weighting. Applying either update rule to our selected prompts still underperforms GRAFT (Section 6.3). This motivates pairing complementary prompt selection with compatibility-aware learning from full peer groups.

We propose **GRAFT** (**G**ated **R**eplacement of **A**nswer-**F**ailed groups with peer **T**rajectories), an off-policy-aware framework for cross-model trajectory sharing in RLVR that addresses both decisions. For *which*, GRAFT selects prompts on which the receiver fails entirely and the peer produces both successful and unsuccessful responses, while balancing exchange volume across directions. For *how*, it replaces the selected receiver groups with the corresponding peer groups and retains their source-computed advantages, preserving within-peer reward contrast without pooling rewards across models. It controls peer influence through bounded sequence-level compatibility gating and token-level importance ratio clipping, and keeps the receiver's own data primary by processing peer-containing minibatches after its on-policy minibatches.

Across three heterogeneous pairs of open-source base models and five mathematical reasoning benchmarks, GRAFT improves both models in every pair over GRPO (n=8), by 2.1 points on average and up to 4.5 points in model-level average score. In two of the three pairs, both models match or exceed GRPO trained with four times the rollout budget (n=32). GRAFT also outperforms HACPO and SGT by 4.0 and 1.5 points on average, respectively. The gains largely persist when reusing peer trajectories from completed independent GRPO runs (+1.8 points on average), without simultaneous co-training or additional peer rollouts.

2 Related Work

Exploration limitations in RLVR. GRPO and related RLVR methods learn only from reward variation within self-generated rollout groups (Shao et al., 2024; Yu et al., 2025). Dynamic sampling discards zero-variance groups and resamples (Yu et al., 2025), while larger groups improve success coverage; both cost extra rollouts without guaranteeing success. Entropy-guided advantage shaping recovers a signal without additional rollouts (Le et al., 2026), but on an all-incorrect group it can only suppress the sampled failures, not supply a correct response. Even at scale, RLVR improves sampling efficiency without expanding the base model's solvable prompt set (Yue et al., 2025), and

declining entropy further limits exploration (Cui et al., 2025). Hints, partial solutions, and expert guidance ease exploration but require an external solution source or a stronger model (Li et al., 2026; Huang et al., 2026; Jiang et al., 2026). We instead use trajectories from heterogeneous peers when the learner's own rollouts all fail.

Off-policy guidance and mismatch control. External demonstrations, teacher solutions, and historical trajectories augment RLVR rollouts but introduce policy mismatch (Yan et al., 2025; Dong et al., 2026; Mao et al., 2026). Prior work addresses rollout–training mismatch through importance weighting and truncation (Yao et al., 2025; Ling Team et al., 2025), and studies sequence-level optimization and off-policy correction (Zheng et al., 2025; Chen et al., 2025). We consider distinct peer models with potentially different tokenizers. GRAFT separates within-receiver policy change from cross-model mismatch through token-level importance ratio clipping and sequence-level compatibility filtering with bounded weighting. The compatibility score is an empirical proxy from average token log-likelihoods, not an exact cross-tokenizer importance ratio.

Cross-model learning and trajectory sharing. Recent work enables multiple models to learn from one another during RL. HACPO (Zhang et al., 2026) exchanges peer rollouts with off-policy correction, while Mutual RL (Liu et al., 2026b) introduces SGT to transfer verified peer successes on prompts where the receiver fails. F-TIS (Blagoev et al., 2026) studies collaborative GRPO among models from the same family with a shared vocabulary, using truncated importance sampling and off-policy filtering. Unlike teacher-guided distillation (Agarwal et al., 2024), these approaches motivate learning across peer models without relying exclusively on a designated stronger teacher. We build on this direction by jointly addressing where peer trajectories provide missing supervision and how their influence should be controlled under cross-model mismatch.

3 PRELIMINARIES

Group Relative Policy Optimization. Given a prompt q sampled from a prompt set $\mathcal{D}$, GRPO (Shao et al., 2024) samples a group of n responses $\mathcal{G}(q) = (o_1, \ldots, o_n)$ from an old policy $\pi_{\theta_{\text{old}}}$ and assigns each response a verifiable reward $r_i = r(q, o_i) \in \{0, 1\}$. In this section we identify each response with its token sequence and write $o_i = (o_{i,1}, \ldots, o_{i,|o_i|})$; Section 4 makes tokenizers explicit. Let $\mathcal{R}(q) = (r_1, \ldots, r_n)$ denote the corresponding rewards. The group-relative advantage of response o_i is computed as

$$\hat{a}_i = \frac{r_i - \text{mean}(\mathcal{R}(q))}{\text{std}(\mathcal{R}(q)) + \epsilon_{\text{adv}}}. \tag{1}$$

When $\text{std}(\mathcal{R}(q)) = 0$, *i.e.*, the group is entirely correct or entirely incorrect, all advantages become zero, so the group contributes no policy-gradient signal. Following DAPO (Yu et al., 2025), a GRPO variant, we use token-level loss aggregation with asymmetric clipping:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\mathcal{B}}\left[\frac{1}{\sum_{j\in\mathcal{B}}|o_j|}\sum_{j\in\mathcal{B}}\sum_{t=1}^{|o_j|}\min\left(\rho_{j,t}(\theta)\hat{a}_j,\ \text{clip}(\rho_{j,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}})\,\hat{a}_j\right)\right], \tag{2}$$

where $\mathcal{B}$ denotes a batch of responses sampled from $\pi_{\theta_{\text{old}}}$, with their corresponding prompts drawn from $\mathcal{D}$, and

$$\rho_{j,t}(\theta) = \frac{\pi_\theta(o_{j,t} \mid q_j, o_{j,<t})}{\pi_{\theta_{\text{old}}}(o_{j,t} \mid q_j, o_{j,<t})} \tag{3}$$

is the token-level importance ratio, where q_j denotes the prompt corresponding to response o_j.

Cross-model trajectory sharing. Cross-model RLVR allows heterogeneous models to learn from trajectories generated by their peers. HACPO (Zhang et al., 2026) broadly reuses peer rollouts during policy optimization, using capability-aware advantage estimation, sequence-level importance sampling, and clipping to account for cross-model mismatch. Its sharing is not restricted to prompts where the receiver's rollout group fails. SGT (Liu et al., 2026b) instead transfers a verified peer success only when the receiver's entire rollout group fails and a peer succeeds, and learns from it through an auxiliary negative log-likelihood objective alongside on-policy GRPO. These approaches raise two complementary design questions: *which* peer trajectories should supplement the receiver's own rollouts, and *how* the receiver should optimize on them under cross-model policy mismatch.

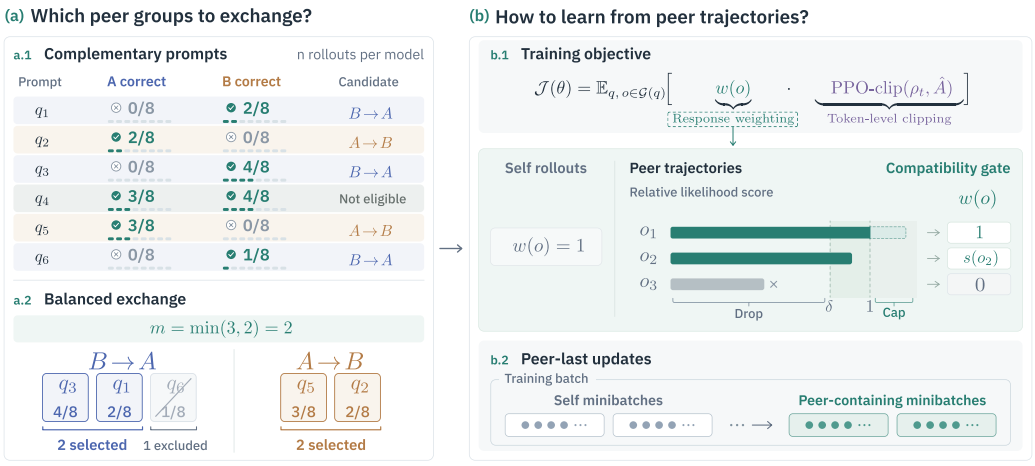


Figure 2: **Overall Framework of GRAFT.** GRAFT selects and balances complementary peer groups, then controls their off-policy influence through compatibility weighting, token-level clipping, and peer-last updates.

4 GRAFT: Gated Replacement of Answer-Failed Groups with Peer Trajectories

Overview. GRAFT addresses the two design questions introduced in Section 3: *which* peer trajectories to transfer, and *how* the receiver should learn from them under cross-model mismatch (Figure 2). GRAFT identifies complementary peer groups, replaces receiver groups that provide no successful trajectory, and balances transfer across the two directions (Section 4.1). The receiver then learns from the transferred trajectories using source-computed advantages, sequence-level compatibility weighting, token-level importance ratio clipping, and peer-last updates (Section 4.2).

Setting. We consider two policies π_{θ^A} and π_{θ^B} trained simultaneously on the same prompt distribution $\mathcal{D}$ with a verifiable binary reward. The models maintain separate parameters and gradients, and exchange only sampled responses, their generation log-probabilities, and rewards. A response o is a string; $o^M = (o_1^M, \ldots, o_{|o^M|}^M)$ denotes its tokenization under M's tokenizer, and for a policy π on M's vocabulary we set $\pi(o \mid q) = \prod_{t=1}^{|o^M|} \pi(o_t^M \mid q, o_{<t}^M)$.[1] For each prompt $q \sim \mathcal{D}$, each model $M \in \{A, B\}$ samples $\mathcal{G}_M(q) = (o_1, \ldots, o_n)$ from its behavior policy $\pi_{\theta^M_{\text{old}}}$. We denote the corresponding GRPO advantages by $\{\hat{a}_i^M\}_{i=1}^n$, and define the number of successful responses as $k_M(q) = \sum_{o \in \mathcal{G}_M(q)} r(q, o)$. Throughout, we describe transfer from model A to model B, treating A as the source and B as the receiver; the reverse direction is symmetric.

4.1 Complementary and Balanced Group Replacement

Complementary group replacement. Model B receives peer trajectories only when its own rollout group fails entirely and the peer group contains both successful and unsuccessful responses:

$$\mathcal{C}_{A \to B} = \{ q \in \mathcal{Q} : k_B(q) = 0 \ \wedge \ 1 \le k_A(q) < n \}, \tag{4}$$

where $\mathcal{Q} \subset \mathcal{D}$. The condition $k_B(q) = 0$ restricts transfer to prompts with no reward-based policy-gradient signal from the receiver's own group. The condition $1 \le k_A(q) < n$ ensures that the peer group contains a verified success and nonzero reward variance.

For each prompt selected from $\mathcal{C}_{A \to B}$ by the balancing procedure below, we replace B's failed group with A's entire rollout group $\mathcal{G}_A(q)$; each transferred response enters B's update with its source-computed advantage $\hat{a}_i^A$ rather than a re-normalized one.

[1] A superscript on a token sequence denotes the tokenizer; on an advantage, the model that computed it. String-level quantities (r, s, w) carry no superscript.

Transferring both successful and unsuccessful responses preserves the reward contrast within the peer group, supplying positive and negative advantages without pooling rewards across models. The receiver then applies the compatibility weights of Section 4.2 while keeping the source advantages fixed.

Balanced exchange. Complementary candidate sets can differ substantially in size across directions, exposing one receiver to many more peer groups than the other. We use the smaller candidate count, $m = \min(|\mathcal{C}_{A\to B}|, |\mathcal{C}_{B\to A}|)$, as a common selection target. In each direction, candidates are ranked by the source model's success count in descending order, and the first m prompts are retained together with all ties at the boundary. This reduces directional imbalance while preserving equal-ranked candidates.

4.2 Off-Policy-Aware Peer Updates

A grafted response is generated by $\pi_\phi \equiv \pi_{\theta^A_{\text{old}}}$ and used to update π_{θ^B}. At the string level, the likelihood ratio factorizes as

$$\frac{\pi_{\theta^B}(o \mid q)}{\pi_\phi(o \mid q)} = \underbrace{\frac{\pi_{\theta^B}(o^B \mid q)}{\pi_{\theta^B_{\text{old}}}(o^B \mid q)}}_{\text{within-receiver change}} \cdot \underbrace{\frac{\pi_{\theta^B_{\text{old}}}(o^B \mid q)}{\pi_\phi(o^A \mid q)}}_{\text{cross-model mismatch}}, \tag{5}$$

which separates the receiver's change during optimization from its initial mismatch with the peer. We use this factorization to motivate treating the two discrepancies separately, with a token-level PPO surrogate for the former and a sequence-level compatibility weight for the latter; we do not use it to derive an exact importance-weighted objective.

To operationalize the cross-model mismatch under differing tokenizers, we define the average token log-likelihood

$$\bar{\ell}\big(\pi, o^M \mid q\big) = \frac{1}{|o^M|} \sum_{t=1}^{|o^M|} \log \pi\big(o_t^M \mid q, o_{<t}^M\big). \tag{6}$$

Compatibility gate. We use sequence-level likelihood only to decide whether and how strongly a peer trajectory is admitted, and token-level importance ratio clipping to control the receiver's update on it. We evaluate each tokenization under its corresponding model and define

$$s(o \mid q) = \exp\Big(\bar{\ell}\big(\pi_{\theta^B_{\text{old}}}, o^B \mid q\big) - \bar{\ell}\big(\pi_\phi, o^A \mid q\big)\Big). \tag{7}$$

This score compares average token log-likelihoods rather than accumulating log-probabilities over the entire response. When the tokenizations coincide, $s(o \mid q)$ reduces to the length-normalized sequence likelihood ratio; with different tokenizers, we instead interpret it as a compatibility score.

The score defines a bounded weight for each peer sequence,

$$w(o \mid q) = \mathbf{1}\big[s(o \mid q) > \delta\big] \cdot \min\{s(o \mid q),\, 1\}. \tag{8}$$

When the prompt is clear from context, we abbreviate $w(o_j) \equiv w(o_j \mid q_j)$. The threshold δ excludes low-scoring peer trajectories regardless of correctness: any response with $s(o \mid q) \le \delta$ is dropped from the transferred group before optimization. The cap then limits the weight of each admitted response to at most one. Correctness identifies a successful response, but on its own it does not say how well the receiver can learn from that response under this update rule.

Token-level clipping. After group replacement, an optimization minibatch $\mathcal{B}$ may contain two kinds of responses: self-generated responses $o_j \in \mathcal{G}_B(q_j)$ and grafted peer responses $o_j \in \mathcal{G}_A(q_j)$. Regardless of which model generated o_j, the receiver updates on its own tokenization $o_j^B = (o_{j,1}^B, \ldots, o_{j,|o_j^B|}^B)$; for grafted responses this amounts to re-tokenizing the peer's response string with the receiver's tokenizer. The token-level importance ratio is defined on this common receiver-side representation,

$$\rho_{j,t}(\theta^B) = \frac{\pi_{\theta^B}\big(o_{j,t}^B \mid q_j,\, o_{j,<t}^B\big)}{\pi_{\theta^B_{\text{old}}}\big(o_{j,t}^B \mid q_j,\, o_{j,<t}^B\big)}, \tag{9}$$

whose denominator is the receiver's behavior policy for both kinds of responses. In particular, for a grafted response the denominator is *not* the generating policy π_ϕ: following Equation (5), the token-level ratio tracks only the within-receiver change, while the cross-model mismatch is carried entirely by the sequence-level weight. The two kinds of responses therefore enter the objective with identically defined ratios and differ only in their weights and advantages: self-generated responses use $w(o_j) = 1$ and $\hat{a}_j = \hat{a}_j^B$, while grafted responses use $w(o_j)$ from Equation (8) and their source-computed $\hat{a}_j = \hat{a}_j^A$.

The receiver maximizes

$$\mathcal{J}(\theta^B) = \mathbb{E}_{\mathcal{B}}\left[\frac{1}{\sum_{j\in\mathcal{B}}|o_j^B|}\sum_{j\in\mathcal{B}} w(o_j)\sum_{t=1}^{|o_j^B|}\min\Big(\rho_{j,t}(\theta^B)\hat{a}_j,\ \mathrm{clip}\big(\rho_{j,t}(\theta^B), 1-\varepsilon_{\mathrm{low}}, 1+\varepsilon_{\mathrm{high}}\big)\hat{a}_j\Big)\right], \quad (10)$$

where both the advantages and the compatibility weights are held fixed during receiver optimization. Without transfer, every $w(o_j) = 1$ and the objective reduces to the GRPO surrogate with the same clipping settings.

Peer-last updates. Token-level clipping moderates peer contributions only once the importance ratios deviate from one. Before the first optimization step on a newly collected rollout batch, $\theta^B = \theta^B_{\mathrm{old}}$ and hence $\rho_{j,t} = 1$, so a grafted minibatch processed first would enter the update unclipped, and the sequence-level weight would be the only control on cross-model mismatch. We therefore place minibatches containing grafted groups after the receiver's own on-policy minibatches. By the time grafted responses are processed, clipping attenuates contributions whose ratios have moved outside $[1-\varepsilon_{\mathrm{low}},\ 1+\varepsilon_{\mathrm{high}}]$ on the side determined by the sign of the advantage. This ordering gives the receiver's own data priority and turns clipping into a second mechanism for moderating peer influence; we assess its empirical effect in Section 6.3 and trace the resulting clipping dynamics in Appendix H.

5 Experiments

5.1 Setup

Models and pairs. We evaluate three heterogeneous model pairs: SmolLM3-3B-Base (Bakouch et al., 2025)↔Qwen3-1.7B-Base (Yang et al., 2025) (Pair 1), OctoThinker-3B-Hybrid-Base (Wang et al., 2025)↔Qwen3-1.7B-Base (Pair 2), and SmolLM3-3B-Base↔OctoThinker-3B-Hybrid-Base (Pair 3). They differ in scale, tokenizer, pretraining corpus, and model architecture.

Training. All runs use `verl` with Ray, FSDP, and vLLM rollout. Each model samples n=8 responses per prompt. Training data, learning rate, and training steps are held fixed across methods. Unless stated otherwise, we use a compatibility gate threshold $\delta = 0.8$ for all pairs. Full hyperparameters are provided in Appendix B.

Evaluation. We report pass@1 on five mathematical benchmarks: MATH500 (Hendrycks et al., 2021), AIME2024, AIME2025, AMC23, and Minerva (Lewkowycz et al., 2022), together with their average. Each checkpoint is evaluated over five runs with 8 samples per prompt, and we report the mean and standard deviation. Δ denotes the change in the five-benchmark average relative to GRPO (n=8).

Baselines. We compare against independent GRPO with n=8, 16, and 32 rollouts, as well as two cross-model training baselines, HACPO (Zhang et al., 2026) and SGT (Liu et al., 2026b). GRAFT, HACPO, and SGT all use n=8 rollouts per model, while GRPO with n=16 and n=32 gives a larger-rollout reference for assessing the benefit of additional independent exploration. Across methods, we keep the training data, learning rate, and number of training steps fixed. Appendix B lists the full baseline configurations.

5.2 Main Results

Table 1 reports the main comparison. GRAFT improves over single-model GRPO in all six model blocks, with average-score gains between $+0.66$ and $+4.46$. Rollout budget alone does not explain

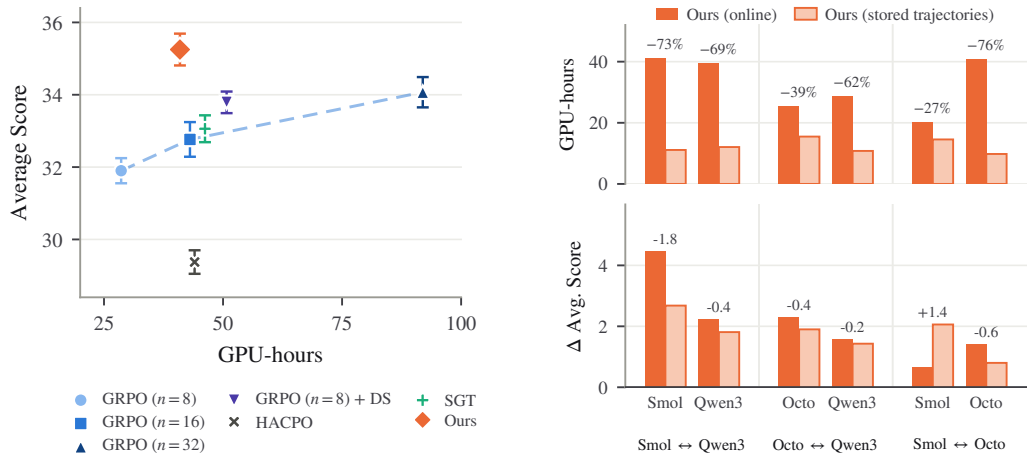
Table 1: **Results of cross-model rollout exchange during co-training.** Each updated policy uses the same GRPO baselines for both of its peer policies. **Bold**: best among the n=8 methods [GRPO (n=8), HACPO, SGT, GRAFT] within each peer block; no bold means GRPO (n=8) is best. †/‡: GRAFT exceeds all baselines up to GRPO (n=16)/(n=32), respectively. Δ: change in average score relative to GRPO (n=8).

Method	MATH500	AIME2024	AIME2025	AMC23	Minerva	**Avg.**	Δ_{Avg}
Updated policy: SmolLM3-3B-Base							
GRPO (n=8)	72.08 ±0.52	8.58 ±1.13	8.50 ±0.96	46.62 ±1.49	27.22 ±0.63	32.60 ±0.49	–
GRPO (n=16)	72.88 ±0.40	9.08 ±0.90	9.83 ±1.34	47.19 ±1.01	27.90 ±0.75	33.38 ±0.25	↑0.78
GRPO (n=32)	76.68 ±0.56	11.42 ±0.81	12.83 ±1.23	49.56 ±1.80	28.08 ±0.57	35.71 ±0.18	↑3.11
Peer policy: Qwen3-1.7B-Base (Pair 1)							
HACPO	69.58 ±0.53	7.08 ±1.79	10.83 ±1.69	42.88 ±1.61	27.17 ±0.61	31.51 ±0.43	↓1.09
SGT	75.53 ±0.51	10.67 ±1.52	11.50 ±0.37	49.06 ±1.51	27.63 ±0.68	34.88 ±0.21	↑2.28
GRAFT	**76.86**‡ **±0.32**	**14.42**‡ **±1.49**	**14.50**‡ **±0.99**	**50.87**‡ **±1.60**	**28.68**‡ **±0.57**	**37.06**‡ **±0.67**	↑4.46
Peer policy: OctoThinker-3B-Hybrid-Base (Pair 3)							
HACPO	66.66 ±0.66	5.92 ±1.68	5.42 ±1.21	38.94 ±1.91	25.22 ±0.77	28.43 ±0.70	↓4.17
SGT	72.36 ±0.43	8.67 ±1.19	8.67 ±1.57	43.62 ±1.07	27.01 ±0.32	32.07 ±0.48	↓0.53
GRAFT	**73.28**† **±0.14**	**10.42**† **±1.02**	**10.83**† **±0.78**	46.44 ±1.49	25.35 ±0.24	**33.26 ±0.32**	↑0.66
Updated policy: Qwen3-1.7B-Base							
GRPO (n=8)	70.69 ±0.44	9.17 ±1.28	4.75 ±0.91	43.50 ±1.57	27.89 ±0.29	31.20 ±0.27	–
GRPO (n=16)	71.64 ±0.60	10.67 ±1.73	8.25 ±1.65	41.44 ±1.12	28.75 ±0.35	32.15 ±0.73	↑0.95
GRPO (n=32)	71.90 ±0.10	10.17 ±0.56	5.75 ±0.75	45.62 ±2.31	28.71 ±0.56	32.43 ±0.65	↑1.23
Peer policy: SmolLM3-3B-Base (Pair 1)							
HACPO	63.74 ±0.60	7.17 ±0.90	4.17 ±1.18	36.06 ±1.20	25.04 ±0.44	27.24 ±0.30	↓3.96
SGT	70.97 ±0.77	10.17 ±1.34	5.92 ±0.85	40.75 ±0.87	28.39 ±0.43	31.24 ±0.56	↑0.04
GRAFT	**72.22**‡ **±0.25**	**12.50**‡ **±0.42**	**7.67 ±1.34**	**45.56**† **±1.09**	**29.23**‡ **±0.32**	**33.44**‡ **±0.23**	↑2.24
Peer policy: OctoThinker-3B-Hybrid-Base (Pair 2)							
HACPO	66.93 ±0.30	6.83 ±0.96	4.00 ±0.48	36.69 ±1.78	27.64 ±0.38	28.42 ±0.38	↓2.78
SGT	69.47 ±0.25	9.58 ±0.29	5.83 ±1.28	41.69 ±1.90	28.42 ±0.81	31.00 ±0.40	↓0.20
GRAFT	**71.75**† **±0.36**	**11.33**‡ **±0.99**	**7.25 ±1.20**	**44.56**† **±3.88**	**29.02**‡ **±0.77**	**32.78**‡ **±0.49**	↑1.58
Updated policy: OctoThinker-3B-Hybrid-Base							
GRPO (n=8)	56.32 ±0.60	2.50 ±0.93	1.08 ±0.86	27.94 ±1.75	18.06 ±0.81	21.18 ±0.40	–
GRPO (n=16)	58.39 ±0.69	3.25 ±0.54	1.83 ±0.48	28.69 ±0.87	19.86 ±0.50	22.40 ±0.43	↑1.22
GRPO (n=32)	60.89 ±0.33	3.42 ±1.19	1.75 ±0.46	31.19 ±2.33	19.70 ±0.73	23.39 ±0.56	↑2.21
Peer policy: Qwen3-1.7B-Base (Pair 2)							
HACPO	55.44 ±0.51	**3.58 ±1.49**	0.92 ±0.19	26.38 ±1.62	18.12 ±0.44	20.89 ±0.38	↓0.29
SGT	**58.11 ±0.41**	2.17 ±1.26	**2.08 ±0.59**	28.69 ±1.89	18.63 ±0.62	21.93 ±0.45	↑0.75
GRAFT	58.01 ±0.36	3.42 ±0.80	1.25 ±0.78	**34.75**‡ **±2.98**	**19.90**‡ **±0.71**	**23.47**‡ **±0.68**	↑2.29
Peer policy: SmolLM3-3B-Base (Pair 3)							
HACPO	56.70 ±0.40	2.92 ±1.02	1.33 ±0.35	29.12 ±0.78	19.37 ±0.68	21.89 ±0.25	↑0.71
SGT	**57.82 ±0.31**	3.50 ±0.48	1.83 ±0.63	29.12 ±0.97	**20.26 ±0.89**	22.51 ±0.32	↑1.33
GRAFT	57.59 ±0.50	**4.25**‡ **±0.75**	**2.25**‡ **±0.70**	**30.19**† **±1.30**	18.74 ±0.76	**22.60**† **±0.22**	↑1.42

these gains. On Pair 1, both models outperform GRPO with 4× the rollouts (n=32), and on Pair 2 both are comparable to it. Over three independent training runs, GRAFT also has a higher mean aggregate score than budget-matched GRPO (Appendix C).

The magnitude of improvement varies across pairs, with the largest gains on Pair 1 and the smallest on Pair 3. Both co-training baselines are weaker. HACPO falls below budget-matched GRPO in five of six blocks, by as much as -4.17 in average score. SGT does better but is inconsistent, ranging from -0.53 to $+2.28$. GRAFT's margin over the budget-matched baselines emerges early and persists throughout training rather than at an isolated checkpoint (Figure 4).

Qualitative case studies show shared solution steps between GRAFT and peer responses, including root shifting and inclusion–exclusion, alongside elements of the receiver's GRPO solution structure (Appendix I).



(a) (b)

Figure 3: **(a)** Average score of Pair 1 vs. total GPU-hours. Error bars: 95% CIs over five evaluation runs; the dashed line traces GRPO with increasing rollout budget, and DS denotes dynamic sampling. GRAFT exceeds GRPO (n=32) by 1.18 at 0.45× the cost. **(b)** Replacing the co-trained partner with its stored trajectories across all six blocks. Top: GPU-hours to the reported checkpoint; bottom: gain over GRPO (n=8). Stored trajectories cut compute by 27–76% while keeping a positive gain in every block (84% of the online gain on average).

6 Analysis

We next examine three questions about the gains of GRAFT: whether they come at a better compute trade-off than simply increasing the rollout budget (Section 6.1), whether they strictly require synchronous co-training or persist with stored peer trajectories (Section 6.2), and which components drive them, including whether the *which* and *how* decisions must be addressed together (Section 6.3).

6.1 Compute-Efficient Gains from Peer Exchange

We compare the GPU-hours required to obtain both models of Pair 1. Because GRAFT jointly trains two policies, we compare the total cost of obtaining both resulting models rather than the cost of either model in isolation. In Figure 3a, GRAFT reaches a pair-mean score of 35.25 at 40.9 GPU-hours. This exceeds GRPO (n=32) by 1.18 points at 0.45× its compute, and GRPO (n=16) by 2.48 points at a comparable compute budget. Thus, increasing independent rollout budgets does not match the performance-compute trade-off of peer exchange in this comparison. Appendix D defines checkpoint-cost accounting and reports full-run costs and per-model results.

6.2 Using Stored Peer Trajectories during Training

Online co-training keeps both models and their optimizer states resident. We therefore test whether GRAFT can instead use peer trajectories stored from the partner's independent GRPO (n=8) run: at each receiver step we load the partner's recorded responses and log-probabilities for the same prompt batch, re-verify their rewards, and apply the same selection, weighting, and update rule (Appendix B). Transfer is unidirectional, so balanced exchange is inactive.

Stored trajectories improve over GRPO (n=8) in all six model blocks, by 1.78 points on average compared with 2.11 for online exchange (Figure 3b; full results in Appendix E). On Pair 1, obtaining both selected receiver checkpoints requires 23.2 GPU-hours, excluding the prior GRPO runs used to collect the peer logs. Most of the performance benefit therefore persists when existing peer trajectories are reused, without simultaneous co-training or keeping the peer model in memory.

Table 2: **Ablations and alternative designs on Pair 1** (S: SmolLM3-3B, Q: Qwen3-1.7B; Δ: change vs. full GRAFT). (a) removes or randomizes one component at a time; (b) replaces the transfer rule, or fixes only one of *which* and *how* while borrowing the other. Variant definitions and per-benchmark scores: Appendices G and F.

(a) Component ablation

	S		Q	
	Avg.	Δ	Avg.	Δ
GRAFT (full)	37.06 ±0.67	–	33.44 ±0.23	–
Component removal				
compat. gate	28.70 ±0.78	↓8.36	30.93 ±0.52	↓2.51
compat. floor	35.17 ±0.59	↓1.89	29.88 ±0.62	↓3.56
balancing	35.06 ±0.27	↓2.00	32.96 ±0.56	↓0.48
token ratio	35.21 ±0.65	↓1.85	32.99 ±0.33	↓0.45
Random selection				
prompts	34.66 ±0.47	↓2.40	31.86 ±0.48	↓1.58
admission	35.20 ±0.85	↓1.86	32.35 ±0.32	↓1.09
Peer position				
first	34.61 ±0.43	↓2.45	31.73 ±0.58	↓1.71
uniform	33.49 ±0.47	↓3.57	32.41 ±0.55	↓1.03
GRPO	32.60 ±0.49	↓4.46	31.20 ±0.27	↓2.24

(b) Alternative designs

	S		Q	
	Avg.	Δ	Avg.	Δ
GRAFT (full)	37.06 ±0.67	–	33.44 ±0.23	–
Transfer rule				
pooled groups	33.07 ±0.47	↓3.99	30.78 ±0.48	↓2.66
success-only	29.22 ±0.26	↓7.84	29.51 ±0.65	↓3.93
+ w/o floor	32.14 ±0.38	↓4.92	29.85 ±0.44	↓3.59
Our which, prior how				
HACPO update	28.97 ±0.52	↓8.09	27.79 ±0.38	↓5.65
SFT update (SGT)	34.08 ±0.15	↓2.98	31.89 ±0.37	↓1.55
LUFFY update	32.51 ±0.89	↓4.55	31.41 ±0.50	↓2.03
Original methods				
HACPO	31.51 ±0.43	↓5.55	27.24 ±0.30	↓6.20
SGT	34.88 ±0.21	↓2.18	31.24 ±0.56	↓2.20
GRPO	32.60 ±0.49	↓4.46	31.20 ±0.27	↓2.24

6.3 Ablations and Alternative Designs

Table 2a evaluates individual design choices on Pair 1. Removing compatibility weighting causes the largest degradation for SmolLM3 (-8.36 points), while removing the floor causes the largest degradation for Qwen3 (-3.56). Removing balancing or replacing the token-level ratio with a sequence-level ratio also lowers performance. Count-matched random prompts and random admission also underperform. Peer-last ordering beats peer-first and uniform; Appendix H additionally shows that it yields the highest clipping rate on peer tokens. We use $\delta = 0.8$ throughout, with a threshold sweep in Appendix F.

Table 2b tests alternative designs. Pooling peer and self groups or transferring only successes underperforms full GRAFT, and the floor improves aggregate scores only under full-group transfer. With our prompt selection fixed, HACPO, SGT, and LUFFY-style (Yan et al., 2025) updates all trail GRAFT. Using GRAFT's update with SGT's unbalanced selection also leaves a gap of 2.00/0.48 points. Together, these comparisons support combining complementary and balanced prompt selection with compatibility-aware peer updates.

7 Conclusion

We have introduced **GRAFT**, an off-policy-aware framework for cross-model trajectory exchange in RLVR. GRAFT exploits complementary successes across heterogeneous models by replacing all-fail rollout groups with informative peer groups, while controlling cross-model mismatch through compatibility-aware and clipped updates. Across three heterogeneous model pairs, GRAFT consistently improves both models over standard GRPO with the same rollout budget and can match or exceed GRPO with substantially more rollouts. Moreover, most of the gains persist when using stored peer trajectories, showing that the benefit of cross-model exploration does not require synchronous co-training.

Limitations. GRAFT's gain depends on how complementary the two models are and is smallest on Pair 3. Across tokenizers, the compatibility score is a proxy, not a density ratio. We also study only two-model pairs, only on math, and only with base models of at most 3B parameters. We leave exchange among more than two peers, and in domains without verifiable rewards, to future work.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology and experiments, implement methods, assist with translation, support qualitative and thematic data analysis, and interpret results. We have not used generative AI tools to formulate mathematical claims, generate synthetic datasets, develop theoretical models or conceptual frameworks, provide critical ingredients for mathematical proofs, assist in writing proofs, or propose or refine hypotheses. Dataset cleaning and reformatting are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures and images, create or edit software code, brainstorm, search for information, identify relevant literature, and propose a title or keywords for the paper. We have reviewed all AI-assisted work. The authors verified and tested LLM-generated code for correctness. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe GRAFT in Section 4 and provide the full algorithm in Appendix A. Appendix B documents the training data, reward verification, model configurations, hyperparameters, baseline implementations, and evaluation and checkpoint selection procedures. We report results across independent training runs in Appendix C and provide GPU-hour measurement and accounting details in Appendix D. Our code is included in the supplementary material.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. 3

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. `https://huggingface.co/blog/smollm3`, 2025. 1, 6

Nikolay Blagoev, Oguzhan Ersoy, Wendelin Boehmer, and Lydia Chen. F-TIS: Harnessing diverse models in collaborative GRPO. In *ICML 2026 Workshop on Multimodal AI Agents*, 2026. 3

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025. 3

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025. 3

Yihong Dong, Xue Jiang, Yongding Tao, Huanyu Liu, Kechi Zhang, Lili Mou, Rongyu Cao, Yingwei Ma, Jue Chen, Binhua Li, et al. Rl-plus: Countering capability boundary collapse of llms in reinforcement learning with hybrid-policy optimization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026. 1, 3

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

Xixiang He, Qiyao Sun, Ao Cheng, Xingming Li, Xuanyu Ji, Hailun Lu, Runke Huang, and Qingyong Hu. Advantage collapse in group relative policy optimization: Diagnosis and mitigation. In *Forty-third International Conference on Machine Learning*, 2026. 1

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. 6

Zeyu Huang, Tianhao Cheng, Zihan Qiu, Zili Wang, Xu Yinghui, Edoardo Ponti, and Ivan Titov. Blending supervised and reinforcement fine-tuning with prefix sampling. In *Forty-third International Conference on Machine Learning*, 2026. 3

Yuxian Jiang, Yafu Li, Guanxu Chen, Dongrui Liu, Yu Cheng, and Jing Shao. Rethinking entropy regularization in large reasoning models. *arXiv preprint arXiv:2509.25133*, 2025. 15

Zishang Jiang, Jinyi Han, tingyun li, Xinyi Wang, Sihang Jiang, Zhaoqian Dai, Ma Shuguang, Fei Yu, Jiaqing Liang, and Yanghua Xiao. Selective expert guidance for effective and diverse exploration in reinforcement learning of LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. 3

Haechan Kim, Soohyun Ryu, Gyouk Chu, Doohyuk Jang, and Eunho Yang. Discounted beta–bernoulli reward estimation for sample-efficient reinforcement learning with verifiable rewards. In *Forty-third International Conference on Machine Learning*, 2026. 1, 15

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. 1

Thanh-Long V. Le, Myeongho Jeon, Kim Vu, Viet Dac Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in LLM reinforcement learning via entropy-guided advantage shaping. In *The Fourteenth International Conference on Learning Representations*, 2026. 1, 2, 15

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. 6

Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in LLMs via question augmentation. In *The Fourteenth International Conference on Learning Representations*, 2026. 3

Ling Team, Anqi Shen, Baihui Li, Bin Hu, Bin Jing, Cai Chen, Chao Huang, Chao Zhang, Chaokun Yang, Cheng Lin, Chengyao Wen, Congqi Li, Deng Zhao, Dingbo Yuan, Donghai You, Fagui Mao, Fanzhuang Meng, Feng Xu, Guojie Li, Guowei Wang, Hao Dai, Haonan Zheng, et al. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model. *arXiv preprint arXiv:2510.18855*, 2025. 3

Huanyu Liu, Jia Li, Yihong Dong, Chang Yu, Taozhi Chen, Lecheng Wang, Yongding Tao, Bin Gu, and Ge Li. Evocot: Overcoming the exploration bottleneck in reinforcement learning for llms. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026a. 1

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 2025a. 15

Xiaoze Liu, Dhananjay Ram, Yuting Zhang, Zhaoyang Zhang, Wei Xia, and Stefano Soatto. Experience sharing in mutual reinforcement learning for heterogeneous language models. *arXiv preprint arXiv:2605.07244*, 2026b. 2, 3, 6, 14

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 *Second Conference on Language Modeling*, 2025b. 1

Yixiu Mao, Yun Qu, Qi Wang, Heming Zou, and Xiangyang Ji. Rlvr without ineffective samples: Group prioritized off-policy optimization for llm reasoning. *arXiv preprint arXiv:2606.01281*, 2026. 3

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 1, 2, 3

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025. 6

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. 1

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 2025. 3, 9, 22

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 1, 6

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. On the rollout-training mismatch in modern RL systems. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025. 3

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru 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, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 2025. 1, 2, 3, 15

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 2025. 1, 2

Zhixia Zhang, Zixuan Huang, Gongxun Li, Huaiyang Wang, Chengyi Yuan, Xin Xia, Deqing Wang, Fuzhen Zhuang, Shuai Ma, Ning Ding, et al. Heterogeneous agent collaborative reinforcement learning. *arXiv preprint arXiv:2603.02604*, 2026. 2, 3, 6, 14

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025. 1, 3

A FULL ALGORITHM OF GRAFT

Algorithm 1 GRAFT: Cross-Model Trajectory Exchange

Require: Policies $\pi_{\theta^A}, \pi_{\theta^B}$; prompt set $\mathcal{D}$; rollout count n; compatibility threshold δ; clipping parameters $\varepsilon_{\text{low}}, \varepsilon_{\text{high}}$

for each training step **do**
  Sample a shared prompt batch $\mathcal{Q} \subset \mathcal{D}$
  for $M \in \{A, B\}$ **do**
    $\theta^M_{\text{old}} \leftarrow \theta^M$
    Sample n responses $\mathcal{G}_M(q)$ from $\pi_{\theta^M_{\text{old}}}$ for each $q \in \mathcal{Q}$
    Store source tokens and generation log-probabilities
    Compute rewards, success counts $k_M(q)$, and advantages $\hat{a}_M(q)$
    Set advantages to zero for zero-variance groups
  end for
  ▷ Select complementary groups with nonzero reward variance
  for $(S, R) \in \{(A, B), (B, A)\}$ **do**
    $\mathcal{C}_{S \to R} \leftarrow \{q \in \mathcal{Q} : k_R(q) = 0,\ 1 \le k_S(q) < n\}$
  end for
  $m \leftarrow \min(|\mathcal{C}_{A \to B}|, |\mathcal{C}_{B \to A}|)$
  for $(S, R) \in \{(A, B), (B, A)\}$ **do**
    $\mathcal{E}_{S \to R} \leftarrow \text{SELECTWITHTIES}(\mathcal{C}_{S \to R}, k_S, m)$
  end for
  ▷ Replace selected groups using the original source rollouts
  for $(S, R) \in \{(A, B), (B, A)\}$ **do**
    Initialize receiver training buffer $\mathcal{B}_R \leftarrow \varnothing$
    for $q \in \mathcal{Q}$ **do**
      if $q \in \mathcal{E}_{S \to R}$ **then**
        Re-tokenize $\mathcal{G}_S(q)$ for receiver R
        Compute $s(o \mid q)$ using $\pi_{\theta^R_{\text{old}}}$ and stored source log-probabilities
        $w(o) \leftarrow \mathbf{1}[s(o \mid q) > \delta] \min\{s(o \mid q), 1\}$
        Add the admitted peer responses ($s(o \mid q) > \delta$) with source advantages $\hat{a}_S(q)$ and weights $w(o)$ to $\mathcal{B}_R$
      else
        Add $\mathcal{G}_R(q)$ with advantages $\hat{a}_R(q)$ and unit weights to $\mathcal{B}_R$
      end if
    end for
  end for
  ▷ Optimize with fixed advantages and compatibility weights
  for $R \in \{A, B\}$ **do**
    for each optimization epoch **do**
      Arrange $\mathcal{B}_R$ into minibatches, placing peer-containing minibatches last
      for each minibatch in this order **do**
        Compute token ratios relative to $\pi_{\theta^R_{\text{old}}}$
        Update θ^R by gradient ascent on Equation (10)
      end for
    end for
  end for
end for
return $\pi_{\theta^A}, \pi_{\theta^B}$

SelectWithTies$(\mathcal{C}, k_S, m)$ ranks candidate prompts by the source model's success count $k_S(q)$ in descending order and retains the first m prompts, including all ties at the boundary. It returns $\varnothing$ when $m = 0$. The selected counts may exceed m and need not be identical across directions. Compatibility weights and group advantages remain fixed during optimization. Peer responses failing the compatibility floor are removed before optimization; group advantages are computed on the full source group prior to this removal and remain fixed.

B Training and Evaluation Details

Data and reward. We train on the 7,500 problems in the MATH training split, using all difficulty levels. Each problem is formatted as a single user message with the suffix *"Let's think step by step and output the final answer within \boxed{}."* The reward is binary: `math_verify` checks the final boxed answer against the reference answer, with verification timeouts assigned zero reward. We use no additional format or length reward, and score responses that reach the generation limit as generated.

Optimization and implementation. We use `verl` with FSDP for policy optimization and vLLM 0.8.5 for rollout generation. Each pair is trained on four NVIDIA H200 GPUs, with both models colocated on the same node and rollout tensor parallelism set to two. Table 3 summarizes the common configuration; baseline-specific exceptions are described below. Both models receive the same prompt batch at each training step. We perform one optimization epoch per rollout batch, partitioned into four minibatches of 32 prompts. The implementation averages the policy loss over response tokens (`token-mean` aggregation). Training uses dynamic microbatching and gradient checkpointing.

Table 3: **Training hyperparameters.** These settings apply to GRAFT and independent GRPO unless otherwise specified. The larger-budget GRPO baselines change only the rollout count.

Hyperparameter	Value
Training epochs	3
Prompt batch size	128
Rollouts per prompt per model	8
Optimization minibatch size	32 prompts
Optimization epochs per rollout batch	1
Maximum prompt / response length	2,048 / 4,096 tokens
Rollout temperature / top-p	1.0 / 1.0
Optimizer	AdamW
Learning rate	10^{-6}, constant, no warmup
Adam coefficients (β_1, β_2)	$(0.9, 0.999)$
Weight decay	0.01
Maximum gradient norm	1.0
PPO clipping $(\varepsilon_{\text{low}}, \varepsilon_{\text{high}})$	$(0.2, 0.28)$
GRPO normalization stabilizer	10^{-6}
KL regularization / entropy bonus	none / none
Precision	bfloat16

GRAFT configuration. We use the same exchange and optimization settings for all three pairs. The compatibility threshold $\delta = 0.8$ was selected on Pair 1 and fixed before running Pairs 2 and 3. Exchange is performed at every training step using the selection and balancing rules in Section 4.1, with no additional exchange-volume cap. Peer responses are re-tokenized for the receiver, while source-computed advantages and compatibility weights remain fixed throughout optimization. Minibatches containing peer groups are placed last. The method uses no additional rollouts; its additional computation is receiver-side likelihood evaluation of transferred responses.

Baseline configurations. Independent GRPO uses the same training settings with $n \in \{8, 16, 32\}$ and no trajectory exchange. The minibatch size remains fixed at 32 prompts, so larger rollout groups increase the number of responses per update while preserving the number of optimizer updates. For HACPO (Zhang et al., 2026), we retain the released method configuration: sequence-level clipping with lower and upper clip deltas of 3×10^{-4} and 4×10^{-4}, a peer-rollout clipping lower bound initialized at 0.8 and increased by 0.025 per step, a peer loss coefficient of 1.0, a minibatch size of 64 prompts, and a KL loss coefficient of 10^{-3}, following the default configuration in their official implementation. HACPO pools peer rollouts for every prompt and uses n=8, the same learning rate, and the same training duration. We also tested HACPO without KL regularization and with 32-prompt minibatches; the reported configuration performed better. For SGT (Liu et al., 2026b), we augment the GRPO loss with a supervised negative log-likelihood term weighted by $\lambda = 0.1$,

following the paper's original setting. Whenever a receiver has no correct rollout and its peer has at least one, we uniformly sample one correct peer response for this term. This rule is applied in both directions. Variants that combine a baseline with a component of GRAFT are described in Appendix G.

Saved peer logs. For experiments using saved peer logs, the trajectories come from the partner's independent GRPO (n=8) training run. At each receiver training step, we load the corresponding recorded peer responses and generation log-probabilities, re-evaluate their rewards, and apply the same selection, compatibility weighting, and receiver update. The partner model is not loaded or trained. Directional balancing is inactive because transfer is unidirectional.

Evaluation. We evaluate on MATH500 (500 problems), AIME2024 (30), AIME2025 (30), AMC23 (40), and Minerva Math (272), using the same prompt formatting, answer verifier, and length limits as in training, with temperature 0.6, top-p 0.95, and eight responses per problem. We estimate pass@1 by averaging correctness over responses and then over problems; the aggregate score is the unweighted mean of the five benchmarks. Since RLVR training curves are non-monotonic across nearby checkpoints, it is common to monitor training on validation sets drawn from the evaluation benchmarks (Yu et al., 2025; Liu et al., 2025a) and report the best checkpoint so selected (Le et al., 2026; Kim et al., 2026; Jiang et al., 2025).

We follow this practice with a single fixed rule: every model is validated every five steps on the aggregate score and the highest-scoring checkpoint is reported, under the identical rule for every method and rollout budget. Figure 4 shows that GRAFT leads the budget-matched baselines throughout most of training. Each selected checkpoint is evaluated over five independent runs, and we report their mean and standard deviation. For Appendix C, we average the five evaluation scores within each training run and report the mean and sample standard deviation over three independent runs; main-table results use the first run.

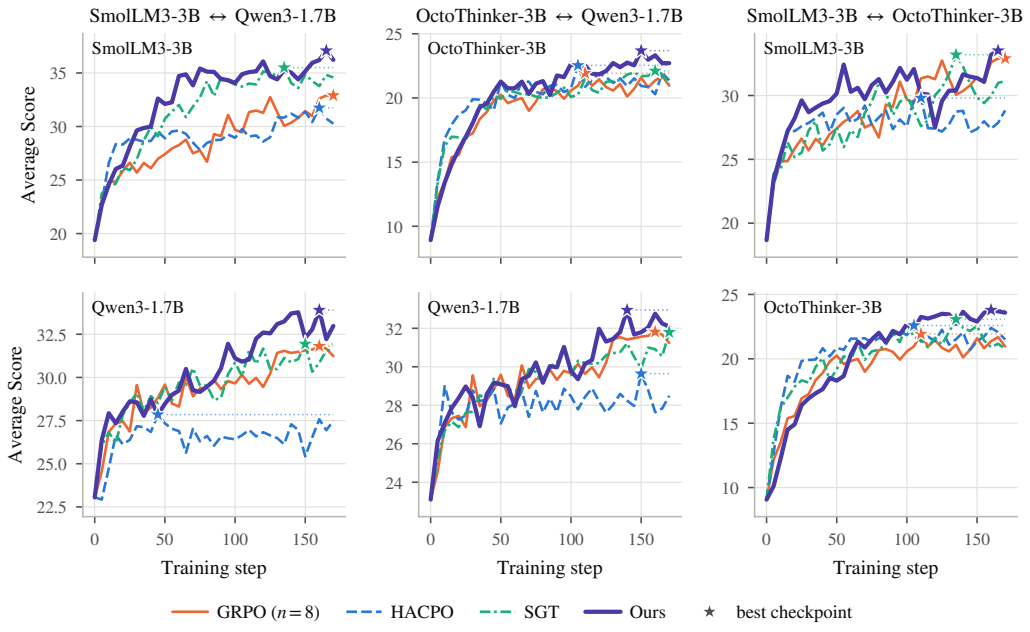


Figure 4: **Validation trajectories used for checkpoint selection.** Five-benchmark average vs. training step for each updated model, evaluated under the identical protocol for all methods. Stars mark the selected checkpoints in Table 1.

C Training Stability Across Independent Runs

To assess sensitivity to training variability, we independently repeat both GRPO (n=8) and GRAFT three times for all three model pairs. For each training run, we evaluate the resulting checkpoint over five evaluation runs with 8 samples per prompt and report their mean performance. Table 4 reports the mean and sample standard deviation across the three independent training runs. The main results in Table 1 use the first training run for each setting. Across all six model blocks, GRAFT maintains higher mean aggregate performance than the single-model GRPO baseline, indicating that the improvements persist across independent training runs.

Table 4: **Training stability across three independent runs.** We repeat GRPO (n=8) and GRAFT three times for all three model pairs and report the mean and sample standard deviation across runs. Δ denotes the difference in the mean average score relative to GRPO (n=8) within each model block. **Bold** indicates the better mean performance between GRPO (n=8) and GRAFT.

Method	MATH500	AIME2024	AIME2025	AMC23	Minerva	**Avg.**	Δ_{Avg}
Pair 1 SmolLM3-3B-Base ↔ Qwen3-1.7B-Base							
SmolLM3-3B-Base							
GRPO (n=8)	72.21 ±2.78	7.91 ±0.58	8.72 ±2.01	45.08 ±2.46	26.84 ±0.45	32.15 ±1.48	–
GRAFT	**75.37** ±1.30	**13.28** ±1.32	**13.22** ±1.29	**50.37** ±0.98	**28.34** ±1.36	**36.12** ±0.85	↑3.97
Qwen3-1.7B-Base							
GRPO (n=8)	70.69 ±0.30	9.66 ±0.43	5.86 ±0.97	42.60 ±0.78	28.10 ±0.35	31.38 ±0.16	–
GRAFT	**71.78** ±0.38	**12.06** ±0.57	**7.33** ±0.96	**45.81** ±0.29	**29.22** ±0.09	**33.24** ±0.33	↑1.86
Pair 2 OctoThinker-3B-Hybrid-Base ↔ Qwen3-1.7B-Base							
OctoThinker-3B-Hybrid-Base							
GRPO (n=8)	55.86 ±0.42	2.94 ±1.15	0.72 ±0.38	28.35 ±0.36	17.82 ±0.40	21.14 ±0.22	–
GRAFT	**57.36** ±0.57	**3.67** ±0.90	**2.05** ±0.94	**33.56** ±2.11	**19.30** ±0.72	**23.19** ±0.24	↑2.05
Qwen3-1.7B-Base							
GRPO (n=8)	70.69 ±0.30	9.66 ±0.43	5.86 ±0.97	42.60 ±0.78	28.10 ±0.35	31.38 ±0.16	–
GRAFT	**71.17** ±0.53	**11.19** ±0.64	**7.72** ±0.97	**43.19** ±1.44	**28.57** ±0.48	**32.37** ±0.41	↑0.99
Pair 3 SmolLM3-3B-Base ↔ OctoThinker-3B-Hybrid-Base							
SmolLM3-3B-Base							
GRPO (n=8)	72.21 ±2.78	7.91 ±0.58	8.72 ±2.01	45.08 ±2.46	**26.84** ±0.45	32.15 ±1.48	–
GRAFT	**72.27** ±1.13	**9.92** ±1.96	**10.19** ±0.90	**46.92** ±0.94	26.03 ±0.63	**33.06** ±0.31	↑0.91
OctoThinker-3B-Hybrid-Base							
GRPO (n=8)	55.86 ±0.42	2.94 ±1.15	0.72 ±0.38	28.35 ±0.36	17.82 ±0.40	21.14 ±0.22	–
GRAFT	**56.47** ±2.47	**5.03** ±1.35	**1.75** ±0.60	**30.25** ±0.06	**19.29** ±1.57	**22.56** ±1.07	↑1.42

D COMPUTE ACCOUNTING

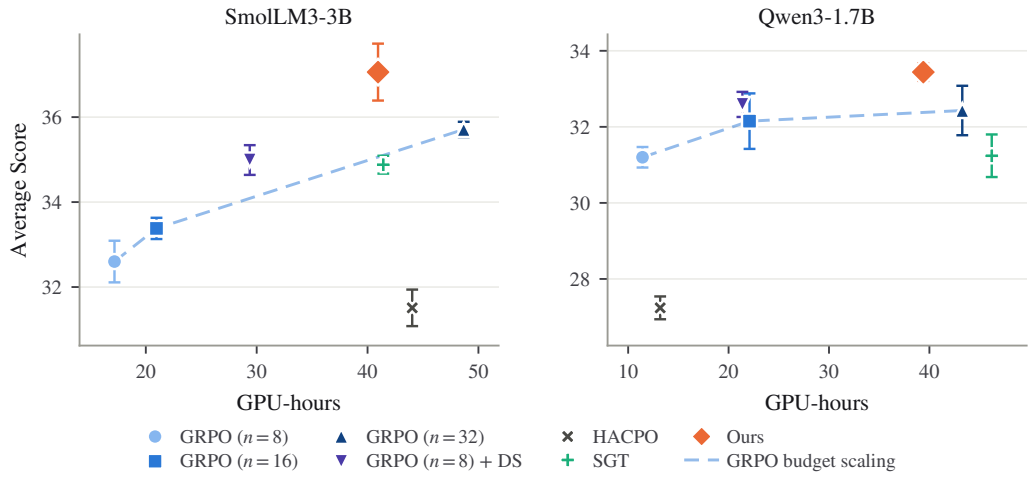


Figure 5: Per-model score against the GPU-hours charged to that model under the most conservative rule: GRPO pays for its own model only; each model of a cross-model method is charged the full joint-run cost (both models' training) up to that model's own selected checkpoint. Error bars are ± 1 s.d. over inference seeds.

Measurement. GPU-hours are computed as N_{GPU} times the summed per-step wall-clock training time, excluding validation and checkpoint writing. All runs in Table 5 use four GPUs. We report both the cost of obtaining the selected checkpoints and the cost of completing all three training epochs (174 steps).

For independent GRPO, the cost of obtaining both models is the sum of the costs of their separate runs up to their respective selected checkpoints. For online cross-model methods, the two selected checkpoints can occur at different steps. The joint checkpoint cost is therefore the cost of running the joint training process through the later of these two steps. It is not generally equal to the sum of the two per-model checkpoint entries. The full-run cost reports the complete training budget without assuming that the selected checkpoint is known in advance.

Conservative per-model accounting. Figure 5 charges independent GRPO only for the model it produces, but charges each model of an online cross-model method the full cost of the joint run, covering both models' training, up to that model's own selected checkpoint (e.g., 40.9 GPU-hours for SmolLM3 at step 165 and 39.4 for Qwen3 at step 160 under GRAFT). This differs from the joint checkpoint cost in Table 5, which runs through the later of the two checkpoints. Under this accounting, GRAFT exceeds GRPO (n=32) by 1.35 points for SmolLM3 and 1.01 points for Qwen3, at approximately $0.84\times$ and $0.91\times$ their respective checkpoint costs. Compared with GRPO (n=16), it improves scores by 3.68 and 1.29 points at approximately $1.95\times$ and $1.78\times$ the cost. Thus, the advantage over the largest tested rollout budget persists even when the full joint cost is charged to each model separately.

Table 5: **Score and training cost on Pair 1.** Avg. is the five-benchmark average at the selected checkpoint (Table 1). GPU-hour columns report the cost up to the selected checkpoint and for the complete three-epoch run. For online cross-model methods, the "Both models / Best ckpt" entry measures the joint run through the later of the two selected checkpoints; for independent GRPO, it sums the two separate checkpoint costs. Stored-trajectory costs exclude the prior runs used to collect peer logs.

	SmolLM3-3B-Base			**Qwen3-1.7B-Base**			**Both models**	
	Avg. (%)	Best ckpt (GPU-h)	3 epochs (GPU-h)	Avg. (%)	Best ckpt (GPU-h)	3 epochs (GPU-h)	Best ckpt (GPU-h)	3 epochs (GPU-h)
GRPO (n=8)	32.60 $\pm$0.49	17.2	17.4	31.20 $\pm$0.27	11.4	12.5	28.6	29.9
GRPO (n=16)	33.38 $\pm$0.25	21.0	34.8	32.15 $\pm$0.73	22.1	22.5	43.1	57.3
GRPO (n=32)	35.71 $\pm$0.18	48.7	99.5	32.43 $\pm$0.65	43.3	45.6	92.0	145.1
GRPO (n=8) + Dynamic Sampling	34.99 $\pm$0.35	29.4	40.5	32.59 $\pm$0.33	21.4	27.4	50.8	67.9
HACPO	31.51 $\pm$0.43	23.5	25.1	27.24 $\pm$0.30	5.9	22.0	44.0	47.1
SGT	34.88 $\pm$0.21	24.0	30.9	31.24 $\pm$0.56	19.4	22.2	46.2	53.1
GRAFT	37.06 $\pm$0.67	25.3	26.7	33.44 $\pm$0.23	15.0	16.6	40.9	43.3
GRAFT w/ stored traj.	35.28 $\pm$0.33	11.1	20.0	33.01 $\pm$0.53	12.1	13.3	23.2	33.3

E Full Results for Stored Trajectory Experiments

Table 6 reports per-benchmark results for the stored trajectory experiments in Section 6.2. GRAFT with stored trajectories improves the average score over GRPO (n=8) in all six model blocks, by 0.80–2.68 points (1.78 on average, 84% of the 2.11 point online gain), and exceeds online GRAFT for SmolLM3-3B-Base in Pair 3.

Table 6: **Effect of learning from peer-model training trajectories.** We compare standard GRPO (n=8), online cross-model rollout exchange (GRAFT), and GRAFT with peer replay, where each model learns from peer trajectories collected from a previous training run rather than from a simultaneously co-trained peer. All methods use n=8 self-rollouts per model. Δ reports the change in average score relative to GRPO (n=8) within each model block.

Method	MATH500	AIME2024	AIME2025	AMC23	Minerva	**Avg.**	Δ_{Avg}
Pair 1 SmolLM3-3B-Base ↔ Qwen3-1.7B-Base							
SmolLM3-3B-Base							
GRPO (n=8)	72.08 ±0.52	8.58 ±1.13	8.50 ±0.96	46.62 ±1.49	27.22 ±0.63	32.60 ±0.49	–
GRAFT	76.86 ±0.32	14.42 ±1.49	14.50 ±0.99	50.87 ±1.60	28.68 ±0.57	37.06 ±0.67	↑4.46
GRAFT w/ stored traj.	74.53 ±0.41	12.00 ±1.62	12.50 ±0.29	49.94 ±2.36	27.44 ±0.69	35.28 ±0.33	↑2.68
Qwen3-1.7B-Base							
GRPO (n=8)	70.69 ±0.44	9.17 ±1.28	4.75 ±0.91	43.50 ±1.57	27.89 ±0.29	31.20 ±0.27	–
GRAFT	72.22 ±0.25	12.50 ±0.42	7.67 ±1.34	45.56 ±1.09	29.23 ±0.32	33.44 ±0.23	↑2.24
GRAFT w/ stored traj.	72.81 ±0.63	10.92 ±0.90	5.83 ±1.21	46.00 ±1.42	29.49 ±0.70	33.01 ±0.53	↑1.81
Pair 2 OctoThinker-3B-Hybrid-Base ↔ Qwen3-1.7B-Base							
OctoThinker-3B-Hybrid-Base							
GRPO (n=8)	56.32 ±0.60	2.50 ±0.93	1.08 ±0.86	27.94 ±1.75	18.06 ±0.81	21.18 ±0.40	–
GRAFT	58.01 ±0.36	3.42 ±0.80	1.25 ±0.78	34.75 ±2.98	19.90 ±0.71	23.47 ±0.68	↑2.29
GRAFT w/ stored traj.	57.91 ±0.83	4.00 ±1.76	2.42 ±0.85	31.06 ±2.28	20.03 ±0.51	23.08 ±0.35	↑1.90
Qwen3-1.7B-Base							
GRPO (n=8)	70.69 ±0.44	9.17 ±1.28	4.75 ±0.91	43.50 ±1.57	27.89 ±0.29	31.20 ±0.27	–
GRAFT	71.75 ±0.36	11.33 ±0.99	7.25 ±1.20	44.56 ±3.88	29.02 ±0.77	32.78 ±0.49	↑1.58
GRAFT w/ stored traj.	71.86 ±0.24	11.08 ±1.46	7.08 ±0.98	44.06 ±2.17	29.03 ±0.33	32.63 ±0.63	↑1.43
Pair 3 SmolLM3-3B-Base ↔ OctoThinker-3B-Hybrid-Base							
SmolLM3-3B-Base							
GRPO (n=8)	72.08 ±0.52	8.58 ±1.13	8.50 ±0.96	46.62 ±1.49	27.22 ±0.63	32.60 ±0.49	–
GRAFT	73.28 ±0.14	10.42 ±1.02	10.83 ±0.78	46.44 ±1.49	25.35 ±0.24	33.26 ±0.32	↑0.66
GRAFT w/ stored traj.	73.83 ±0.36	10.42 ±1.14	12.67 ±1.05	48.56 ±2.51	27.83 ±0.76	34.66 ±0.62	↑2.06
OctoThinker-3B-Hybrid-Base							
GRPO (n=8)	56.32 ±0.60	2.50 ±0.93	1.08 ±0.86	27.94 ±1.75	18.06 ±0.81	21.18 ±0.40	–
GRAFT	57.59 ±0.50	4.25 ±0.75	2.25 ±0.70	30.19 ±1.30	18.74 ±0.76	22.60 ±0.22	↑1.42
GRAFT w/ stored traj.	55.21 ±0.56	3.50 ±1.05	2.25 ±0.76	30.63 ±2.80	18.31 ±0.91	21.98 ±0.33	↑0.80

F Full Results for Ablations and Alternative Designs

Table 7 reports per-benchmark results for Tables 2a and 2b, and the compatibility-threshold sweep. Table 8 reports per-benchmark results for combinations of prior methods with GRAFT's prompt selection or peer update. Implementation details are provided in Appendix G.

All listed variants lower the aggregate score relative to full GRAFT for both models, although individual benchmark scores can improve. Removing compatibility weighting causes the largest aggregate degradation for SmolLM3, whereas removing only the floor causes the largest degradation for Qwen3 among the variants in Table 7. Count-matched random selection and alternative peer-minibatch orderings also reduce aggregate performance. Among the tested thresholds, $\delta = 0.8$ achieves the highest aggregate score for both receivers.

Table 7: **Full ablation of GRAFT on Pair 1.** Per-benchmark scores behind Table 2. Each variant changes a single component of the full method while keeping the compute budget fixed at n=8 rollouts per model. *Count-matched random* variants match the prompt-selection or response-admission count obtained by applying GRAFT to the current rollout batch, separately in each direction, but select uniformly at random. Random admission replaces the compatibility floor while retaining weights $\min\{s, 1\}$. Δ reports the change in average score relative to full GRAFT within each model block.

Variant	MATH500	AIME2024	AIME2025	AMC23	Minerva	**Avg.**	Δ_{Avg}
SmolLM3-3B-Base							
GRAFT (full, δ=0.8)	76.86 ±0.32	14.42 ±1.49	14.50 ±0.99	50.87 ±1.60	28.68 ±0.57	37.06 ±0.67	–
Removing one component							
w/o compatibility gate	68.66 ±0.40	5.42 ±1.47	4.92 ±1.12	39.19 ±1.66	25.33 ±0.57	28.70 ±0.78	↓8.36
w/o balanced exchange	75.05 ±0.25	10.08 ±1.68	12.50 ±0.78	49.44 ±2.20	28.24 ±0.48	35.06 ±0.27	↓2.00
w/o token-level ratio	75.83 ±0.58	10.50 ±1.04	12.50 ±1.56	48.94 ±1.80	28.26 ±0.49	35.21 ±0.65	↓1.85
Count-matched random selection							
random prompts	75.81 ±0.31	10.08 ±1.12	11.75 ±0.68	47.94 ±1.69	27.70 ±0.71	34.66 ±0.47	↓2.40
random admission	76.00 ±0.45	11.25 ±1.53	11.25 ±1.67	50.25 ±3.02	27.24 ±0.53	35.20 ±0.85	↓1.86
Position of peer minibatches							
first	75.74 ±0.32	9.83 ±1.49	11.25 ±0.51	49.06 ±1.98	27.18 ±0.53	34.61 ±0.43	↓2.45
uniform	72.47 ±0.68	12.58 ±1.30	9.92 ±0.90	45.06 ±2.53	27.42 ±0.63	33.49 ±0.47	↓3.57
Advantage computation							
pooled with receiver group	74.21 ±0.44	9.58 ±1.32	9.17 ±1.06	45.88 ±1.39	26.53 ±0.61	33.07 ±0.47	↓3.99
Varying the compatibility gate threshold							
δ=0 (no floor)	75.66 ±0.49	11.75 ±1.23	12.17 ±1.16	49.38 ±1.86	26.89 ±0.66	35.17 ±0.59	↓1.89
δ=0.7	74.16 ±0.48	11.17 ±1.54	9.58 ±1.06	48.69 ±1.16	27.26 ±0.88	34.17 ±0.49	↓2.89
δ=0.9	77.38 ±0.26	9.83 ±1.43	14.00 ±0.48	51.75 ±1.65	27.75 ±0.65	36.14 ±0.18	↓0.92
GRPO (n=8)	72.08 ±0.52	8.58 ±1.13	8.50 ±0.96	46.62 ±1.49	27.22 ±0.63	32.60 ±0.49	↓4.46
Qwen3-1.7B-Base							
GRAFT (full, δ=0.8)	72.22 ±0.25	12.50 ±0.42	7.67 ±1.34	45.56 ±1.09	29.23 ±0.32	33.44 ±0.23	–
Removing one component							
w/o compatibility gate	69.20 ±0.25	8.00 ±0.80	5.75 ±1.12	42.81 ±1.89	28.88 ±0.37	30.93 ±0.52	↓2.51
w/o balanced exchange	71.77 ±0.59	11.25 ±1.14	7.67 ±0.48	44.94 ±2.01	29.17 ±0.33	32.96 ±0.56	↓0.48
w/o token-level ratio	72.18 ±0.46	11.67 ±1.69	6.50 ±1.09	44.81 ±1.44	29.77 ±0.33	32.99 ±0.33	↓0.45
Count-matched random selection							
random prompts	71.51 ±0.29	9.50 ±1.39	6.50 ±1.09	43.44 ±1.29	28.35 ±0.56	31.86 ±0.48	↓1.58
random admission	71.29 ±0.29	11.08 ±1.09	6.50 ±1.52	44.56 ±1.03	28.29 ±0.39	32.35 ±0.32	↓1.09
Position of peer minibatches							
first	71.29 ±0.53	9.75 ±0.56	5.42 ±0.88	43.69 ±2.33	28.48 ±0.36	31.73 ±0.58	↓1.71
uniform	71.25 ±0.28	11.08 ±1.09	6.58 ±0.75	44.31 ±1.88	28.80 ±0.43	32.41 ±0.55	↓1.03
Advantage computation							
pooled with receiver group	69.77 ±0.70	8.92 ±1.00	5.50 ±0.90	41.88 ±1.45	27.87 ±0.58	30.78 ±0.48	↓2.66
Varying the compatibility gate threshold							
δ=0 (no floor)	66.97 ±0.60	8.50 ±1.05	4.58 ±1.18	41.69 ±1.87	27.66 ±0.73	29.88 ±0.62	↓3.56
δ=0.7	71.94 ±0.21	10.83 ±0.78	7.00 ±1.30	46.31 ±1.87	28.41 ±0.79	32.90 ±0.50	↓0.54
δ=0.9	71.33 ±0.66	9.58 ±1.95	6.08 ±1.83	43.38 ±1.91	29.14 ±0.50	31.90 ±0.63	↓1.54
GRPO (n=8)	70.69 ±0.44	9.17 ±1.28	4.75 ±0.91	43.50 ±1.57	27.89 ±0.29	31.20 ±0.27	↓2.24

Table 8: **Full results for prior methods with one side replaced, on Pair 1.** Per-benchmark results for combinations of prior methods with GRAFT's prompt selection or peer update. HACPO and SGT rows are from Table 1; "SGT + GRAFT how" is identical to "w/o balanced exchange" in Table 7. Δ reports the change in average score relative to full GRAFT within each model block.

Method	MATH500	AIME2024	AIME2025	AMC23	Minerva	**Avg.**	Δ_{Avg}
SmolLM3-3B-Base							
GRAFT	76.86 ±0.32	14.42 ±1.49	14.50 ±0.99	50.87 ±1.60	28.68 ±0.57	37.06 ±0.67	–
HACPO	69.58 ±0.53	7.08 ±1.79	10.83 ±1.69	42.88 ±1.61	27.17 ±0.61	31.51 ±0.43	↓5.55
HACPO + GRAFT which	67.73 ±0.43	6.42 ±0.37	6.42 ±1.16	40.06 ±1.64	24.23 ±0.58	28.97 ±0.52	↓8.09
SGT	75.53 ±0.51	10.67 ±1.52	11.50 ±0.37	49.06 ±1.51	27.63 ±0.68	34.88 ±0.21	↓2.18
SGT + GRAFT how	75.05 ±0.25	10.08 ±1.68	12.50 ±0.78	49.44 ±2.20	28.24 ±0.48	35.06 ±0.27	↓2.00
SGT + GRAFT which	75.57 ±0.39	9.08 ±1.94	12.08 ±1.59	46.63 ±1.49	27.02 ±0.66	34.08 ±0.15	↓2.98
GRAFT + LUFFY update	73.62 ±0.45	9.50 ±0.90	7.75 ±1.40	45.19 ±2.28	26.50 ±0.39	32.51 ±0.89	↓4.55
GRPO (n=8)	72.08 ±0.52	8.58 ±1.13	8.50 ±0.96	46.62 ±1.49	27.22 ±0.63	32.60 ±0.49	↓4.46
Qwen3-1.7B-Base							
GRAFT	72.22 ±0.25	12.50 ±0.42	7.67 ±1.34	45.56 ±1.09	29.23 ±0.32	33.44 ±0.23	–
HACPO	63.74 ±0.60	7.17 ±0.90	4.17 ±1.18	36.06 ±1.20	25.04 ±0.44	27.24 ±0.30	↓6.20
HACPO + GRAFT which	65.59 ±0.38	6.25 ±1.28	4.92 ±1.60	35.75 ±2.06	26.45 ±0.45	27.79 ±0.38	↓5.65
SGT	70.97 ±0.77	10.17 ±1.34	5.92 ±0.85	40.75 ±0.87	28.39 ±0.43	31.24 ±0.56	↓2.20
SGT + GRAFT how	71.77 ±0.59	11.25 ±1.14	7.67 ±0.48	44.94 ±2.01	29.17 ±0.33	32.96 ±0.56	↓0.48
SGT + GRAFT which	71.00 ±0.50	10.58 ±1.37	5.83 ±0.78	43.12 ±1.50	28.90 ±1.36	31.89 ±0.37	↓1.55
GRAFT + LUFFY update	71.16 ±0.37	9.42 ±1.52	5.08 ±0.95	43.44 ±1.47	27.94 ±0.66	31.41 ±0.50	↓2.03
GRPO (n=8)	70.69 ±0.44	9.17 ±1.28	4.75 ±0.91	43.50 ±1.57	27.89 ±0.29	31.20 ±0.27	↓2.24

G Implementation of Alternative Designs

All variants in Tables 2 and 8 use Pair 1 and the configuration of Table 3, and differ from full GRAFT or from the corresponding baseline only as described below.

HACPO + GRAFT which. We keep the HACPO configuration of Appendix B but apply its peer-rollout loss only on prompts where the receiver fails on all n rollouts ($k_B(q)=0$) and the peer succeeds at least once ($1 \le k_A(q) < n$), subject to the balanced exchange of Section 4.1. Peer responses on all other prompts receive zero advantage and contribute no gradient.

SGT + GRAFT how. This variant retains SGT's unbalanced prompt-selection rule: the receiver has no successful rollout and the peer has at least one. For the selected prompts, it uses GRAFT's peer-data learning rule, replacing the receiver group with the full peer group and applying source-computed advantages, compatibility weighting, token-level clipping, and peer-last updates. Thus, it preserves SGT's prompt eligibility rather than its single-success response selection. Peer groups with $k_A(q) = n$ have zero source-computed advantages and provide no reward-based policy gradient. The reported variant corresponds to GRAFT without balanced exchange.

SGT + GRAFT which. We retain SGT's auxiliary supervised objective and loss coefficient, but restrict transfer to prompts selected by GRAFT's complementary and balanced selection rule. For each selected prompt, we uniformly sample one successful peer response for the supervised loss.

LUFFY-style peer updates. Following LUFFY (Yan et al., 2025), peer tokens are optimized with the regularized importance-sampling objective $f\big(\pi_\theta(o_t \mid q, o_{<t})\big)\hat{a}$ with the shaping function $f(x) = x/(x+\gamma)$ and $\gamma = 0.1$, in place of the compatibility gate and the token-level clipped ratio. As in LUFFY, the behavior probability of peer tokens is set to one and no clipping is applied to peer tokens, while receiver tokens use the standard clipped surrogate. Selection, balanced exchange, source-computed advantages, and peer-last ordering follow GRAFT.

Count-matched random selection. At each step, we apply GRAFT's selection rules to the current rollout batch to determine the selection count separately for each transfer direction. For *random prompts*, we sample the same number of prompts uniformly from the prompt batch, irrespective of either model's rewards, and transfer their peer groups. The standard compatibility weighting is then applied to these responses.

For *random admission*, we retain GRAFT's prompt selection and count the peer responses satisfying $s(o \mid q) > \delta$. We then sample exactly this many responses uniformly from the transferred peer groups. If $\mathcal{A}_{\text{rand}}$ denotes the randomly selected responses, their weights are

$$w_{\text{random}}(o) = \mathbf{1}[o \in \mathcal{A}_{\text{rand}}] \min\{s(o \mid q), 1\}. \tag{11}$$

The compatibility floor is used only to determine the admission count; it is not applied to the randomly selected responses. Thus, a selected response with $s(o \mid q) \le \delta$ retains its bounded compatibility weight. All other settings follow GRAFT.

Peer minibatch position. GRAFT places minibatches containing peer groups last. *First* places them before the receiver's own minibatches, and *uniform* shuffles the minibatch order uniformly at random at every step.

Pooled groups. Instead of replacing the receiver's all-fail group, this variant retains both the receiver and peer responses and computes advantages over their concatenation:

$$\mathcal{G}_{\text{pool}}(q) = \mathcal{G}_B(q) \,\|\, \mathcal{G}_A(q), \qquad \hat{a}_i = \frac{r_i - \text{mean}(\mathcal{G}_{\text{pool}})}{\text{std}(\mathcal{G}_{\text{pool}}) + \epsilon_{\text{adv}}}. \tag{12}$$

Here, $\|$ denotes concatenation and $\epsilon_{\text{adv}} = 10^{-6}$. Since $k_B(q) = 0$, the pooled reward mean is $k_A(q)/(2n)$, and the receiver's failed responses receive negative advantages. Peer responses retain compatibility weights $w(o)$, while receiver responses have unit weight. All retained response tokens are included in the token-mean loss denominator. This variant changes the advantage normalization, the responses included in optimization, and the loss normalization; it therefore evaluates pooled group construction as a whole.

Success-only transfer. This variant uses the same prompt selection as full GRAFT and computes advantages from the original peer group before filtering. Only successful peer responses contribute to the peer policy-gradient term, with their source-computed advantages retained. Unsuccessful peer responses are excluded from both the policy-gradient numerator and the token-mean loss denominator.

Compatibility gating and floor. The *w/o compatibility gate* variant sets $w(o) = 1$ for every transferred peer response, removing both the floor and compatibility-dependent weighting. The *no-floor* variant removes only the admission threshold and uses $w(o) = \min\{s(o \mid q), 1\}$. It therefore retains bounded compatibility weighting while admitting every transferred response.

H Clipping Dynamics under Peer Minibatch Position

We examine whether peer-minibatch ordering changes the activation of token-level clipping on Pair 1. Over training steps 1–48, we measure the fraction of tokens for which the clipped surrogate is strictly smaller than the unclipped surrogate: $\hat{a}_j > 0$ with $\rho_{j,t} > 1 + \varepsilon_{\text{high}}$, or $\hat{a}_j < 0$ with $\rho_{j,t} < 1 - \varepsilon_{\text{low}}$. We report these fractions separately for self-generated and grafted responses, using one run per ordering (Table 9).

Table 9: Token-level clipping under different peer-minibatch orderings on Pair 1, measured over training steps 1–48.

		Peer minibatch position		
Clipped-token fraction (%)	Update	First	Uniform	Last (ours)
Self-generated	SmolLM3-3B	0.046	0.043	0.045
	Qwen3-1.7B	0.068	0.068	0.072
Grafted (peer)	SmolLM3-3B ← Qwen3-1.7B	0.004	0.130	0.277
	Qwen3-1.7B ← SmolLM3-3B	0.002	0.248	0.440
Steps with zero clipping on peer tokens (%)		97.6	13.0	0.0

Peer-last ordering yields the highest clipped-token fraction on grafted responses (0.28–0.44%), compared with 0.13–0.25% under uniform ordering and less than 0.01% under peer-first. This pattern is consistent with earlier receiver updates moving peer-token ratios away from one before peer optimization.

I Qualitative Analysis

We present three examples for each target model in Pair 1, for a total of six examples. Figures 6–8 show Qwen3-1.7B examples and Figures 9–11 show SmolLM3-3B examples. Each figure presents a GRPO response from the target model, a GRPO response from its peer, and a GRAFT response from the target model. We show excerpts of the responses and condense equations for readability. Vertical ellipses mark omitted steps. The shaded *Step analysis* boxes summarize our analysis of the approach taken in each response.

I.1 Qwen3-1.7B

Case 1. Using the root shift

Problem. Let $f(x) = x^3 - 49x^2 + 623x - 2015$ and $g(x) = f(x+5)$. Find the sum of the roots of g.

Target model · GRPO
Qwen3-1.7B

$$(x+5)^3 = x^3 + 15x^2 + 75x + 125$$
$$\vdots$$
$$g(x) = x^3 - 34x^2 + 208x$$
$$\vdots$$
$$-\frac{-34}{1} = 34$$
$$\vdots$$
$$\boxed{34}$$

Step analysis
Expands the shifted polynomial before applying Vieta's formula.

Answer: Correct

Peer model · GRPO
SmolLM3-3B

"the roots of g are $-5 + x_1, -5 + x_2, -5 + x_3$."
$$\vdots$$
$$-15 + (x_1 + x_2 + x_3)$$
$$= -15 + 49$$
$$= 34$$
$$\vdots$$
$$\boxed{34}$$

Step analysis
Shifts each root by -5, so their sum decreases by 15.

Answer: Correct

Target model · GRAFT
Qwen3-1.7B

"But expanding this might be time-consuming."
$$\vdots$$
$$r_1 - 5, \quad r_2 - 5, \quad r_3 - 5$$
$$\vdots$$
$$r_1 + r_2 + r_3 = -\frac{-49}{1} = 49$$
$$49 - 15 = 34$$
$$\vdots$$
$$\boxed{34}$$

Step analysis
Uses the same root shift, while retaining an explicit coefficient calculation.

Answer: Correct

Figure 6: **Root shift in Qwen3-1.7B (MATH500).** Target GRPO expands $g(x) = f(x+5)$ and uses the expanded polynomial to compute the root sum with Vieta's formula. The peer instead shifts each root of f by -5, so their sum decreases by 15. GRAFT uses the same root shift as the peer, computes the original root sum with Vieta's formula, and obtains $49 - 15 = 34$.

Case 2. Keeping every lane occupied

Problem. Six distinct cars arrive in a fixed order at a red light and choose one of three lanes: left-turn, straight-going, or right-turn. Count the assignments in which every lane is occupied.

Target model · GRPO
Qwen3-1.7B

"All three lanes must be occupied by the six cars."
⋮
$3^6 = 729$

"There are 729 ways for the six cars to stack up so that all three lanes are occupied."
⋮
$\boxed{729}$

Step analysis
States the occupancy constraint, but stops at the unrestricted count 3^6.

Answer: Incorrect

Peer model · GRPO
SmolLM3-3B

"finding the number of onto functions from the set of cars to the set of lanes."
⋮
$3^6 - 3 \times 2^6 + 3 \times 1^6$
⋮
$\boxed{540}$

Step analysis
Counts onto assignments by subtracting empty-lane cases and restoring their overlaps.

Answer: Correct

Target model · GRAFT
Qwen3-1.7B

"Number of ways where the left-turn lane is empty: 2^6 (since each car can go to straight or right-turn)"
⋮
$3^6 - 3 \times 2^6 + 3 \times 1^6$
$= 729 - 192 + 3$
$= 540$
⋮
$\boxed{540}$

Step analysis
Implements the same correction lane by lane, with an explicit overlap add-back.

Answer: Correct

Figure 7: **Enforcing the occupancy constraint in Qwen3-1.7B (MATH500).** Target GRPO counts all $3^6 = 729$ lane assignments without enforcing that every lane is occupied. The peer treats an assignment as an onto mapping from the six cars to the three lanes and applies inclusion–exclusion. GRAFT applies the same inclusion–exclusion correction through empty-lane events: it subtracts 3×2^6 assignments, adds back the three assignments in which only one lane is occupied, and obtains $729 - 192 + 3 = 540$.

Case 3. Keeping the logarithmic factor once

Problem. Write $\prod_{k=4}^{63} \frac{\log_k(5^{k^2-1})}{\log_{k+1}(5^{k^2-4})} = m/n$ in lowest terms and find $m+n$.

Target model · GRPO
Qwen3-1.7B

$$\frac{k^2-1}{k^2-4}\,\frac{\ln(k+1)}{\ln k}$$
$$\vdots$$
$$\frac{(k-1)\ln(k+1)}{(k-2)\ln k} \times \frac{(k+1)\ln(k+1)}{(k+2)\ln k}$$
$$\vdots$$
$$\frac{63\ln 64}{2\ln 4} = \frac{189}{2}$$
$$m+n = 189+2 = \boxed{191}$$

Step analysis
Duplicates the logarithmic ratio when splitting the rational factor.

Answer: Incorrect

Peer model · GRPO
SmolLM3-3B

$$\left(\prod_{k=4}^{63} \frac{k^2-1}{k^2-4}\right) \times \left(\prod_{k=4}^{63} \frac{\log_k 5}{\log_{k+1} 5}\right)$$
$$\vdots$$
$$\frac{\log_4 5}{\log_{64} 5} = 3$$
$$\vdots$$
$$\frac{31}{13} \times 3 = \frac{93}{13}$$
$$m+n = 93+13 = \boxed{106}$$

Step analysis
Separates the rational product from one telescoping logarithmic product.

Answer: Correct

Target model · GRAFT
Qwen3-1.7B

$$\prod_{k=4}^{63} \frac{k-1}{k-2} = 31$$
$$\prod_{k=4}^{63} \frac{k+1}{k+2} = \frac{1}{13}$$
$$\prod_{k=4}^{63} \frac{\log_5(k+1)}{\log_5 k} = \frac{\log_5 64}{\log_5 4} = 3$$
$$\vdots$$
$$31 \times \frac{1}{13} \times 3 = \frac{93}{13}$$
$$m+n = 93+13 = \boxed{106}$$

Step analysis
Keeps one logarithmic product and evaluates the two rational products separately.

Answer: Correct

Figure 8: **Separating the rational and logarithmic factors in Qwen3-1.7B (AIME2025).** Target GRPO separates the rational and logarithmic factors, then duplicates the logarithmic ratio when factoring the rational term. The peer separates the rational product from a single logarithmic product that telescopes to 3. GRAFT uses the same separation with base-5 logarithms, evaluates the two rational products as 31 and $1/13$, and obtains $31 \times (1/13) \times 3 = 93/13$, so $m+n = 106$.

I.2 SMOLLM3-3B

Case 1. Switching to constrained optimization

Problem. For $|z| = 4$, find the maximum of $\mathrm{Re}\big((75+117i)z + (96+144i)/z\big)$.

Target model · GRPO
SmolLM3-3B

$z = 4(\cos\theta + i\sin\theta)$

$\vdots$

$324\cos\theta - 432\sin\theta$

$\vdots$

$\sqrt{324^2 + (-432)^2} = 540$

$\vdots$

$\boxed{540}$

Step analysis
Keeps the polar representation and finds the amplitude of a sinusoid.

Answer: Correct

Peer model · GRPO
Qwen3-1.7B

$\begin{aligned}&\text{maximize} \quad 81x - 108y\\ &\text{subject to} \quad x^2 + y^2 = 16\end{aligned}$

$\vdots$

$\begin{cases} 81 = 2\lambda x \\ -108 = 2\lambda y \end{cases}$

$\vdots$

$y = -\dfrac{4}{3}x$

$\vdots$

$x = \dfrac{12}{5}, \quad y = -\dfrac{16}{5}$

$81x - 108y = \dfrac{2700}{5} = \boxed{540}$

Step analysis
Recasts the problem on a circle and solves the Lagrange conditions.

Answer: Correct

Target model · GRAFT
SmolLM3-3B

$z = 4(\cos\theta + i\sin\theta)$

$\vdots$

"Alternatively, use the method of letting $z = x + iy$"

$\vdots$

$\begin{aligned}&75x - 117y + 6x + 9y\\ &= 81x - 108y\end{aligned}$

$\begin{aligned}&\text{maximize} \quad 81x - 108y\\ &\text{subject to} \quad x^2 + y^2 = 16\end{aligned}$

$\vdots$

$\begin{cases} 81 = 2\lambda x \\ -108 = 2\lambda y \end{cases}$

$\vdots$

$x = \dfrac{12}{5}, \quad y = -\dfrac{16}{5}$

$81x - 108y = \dfrac{2700}{5} = \boxed{540}$

Step analysis
Combines the real terms explicitly, then uses the same constrained objective as the peer.

Answer: Correct

Figure 9: **Cartesian optimization in SmolLM3-3B (AIME2024).** Target GRPO uses the polar parameterization to reduce the objective to $324\cos\theta - 432\sin\theta$, whose maximum is 540. The peer writes $z = x + iy$, rewrites the objective as $81x - 108y$ under $x^2 + y^2 = 16$, and solves the constrained problem with Lagrange multipliers. GRAFT starts with the polar parameterization, then switches to $z = x + iy$, derives the same constrained objective as the peer, and obtains $x = 12/5$, $y = -16/5$, and the maximum 540 from the Lagrange equations.

Case 2. Preserving the off-line height

Problem. Points $A, \ldots, F$ lie in order on a line; G lies off it. Given $AC = 26$, $BD = 22$, $CE = 31$, $DF = 33$, $AF = 73$, $CG = 40$, and $DG = 30$, find the area of BGE.

Target model · GRPO
SmolLM3-3B

"Point G: (x6, 0)"

⋮

"G = (g, 0)"

⋮

"since all points are collinear, the area of triangle BGE is zero if we consider the line as a base. This indicates a misunderstanding in the approach."

Step analysis
Detects its collinear setup error, but does not recover a valid height.

Answer: Incorrect

Peer model · GRPO
Qwen3-1.7B

$(x_G - 26)^2 + y_G^2 = 1600$
$(x_G - 40)^2 + y_G^2 = 900$
$28x_G - 924 = 700$
$x_G = 58, \quad y_G = \pm 24$

⋮

$$\text{Area} = \frac{|432 - 1368|}{2} = \boxed{468}$$

Step analysis
Keeps a height variable in both distance equations, then eliminates it.

Answer: Correct

Target model · GRAFT
SmolLM3-3B

$G = (g, h), \quad h \neq 0$

⋮

$(g - 26)^2 + h^2 = 1600$
$(g - 40)^2 + h^2 = 900$

⋮

$g = 58, \quad h = \pm 24$

⋮

$BE = 57 - 18 = 39$
$$\text{Area} = \frac{39 \times 24}{2} = \boxed{468}$$

Step analysis
Recovers the same height as the peer, then computes the area from base and height.

Answer: Correct

Figure 10: **Recovering the height in SmolLM3-3B (AIME2025).** Target GRPO places G on the line containing $A, \ldots, F$, recognizes the resulting collinearity as an error, but does not recover a nonzero height. The peer keeps a vertical coordinate for G in the distance constraints $CG = 40$ and $DG = 30$, obtains a height of 24, and computes the area as 468 with the shoelace formula. GRAFT places G off the line, uses the two distance constraints to recover a height of 24, and computes the area from $BE = 39$ as $\frac{39 \times 24}{2} = 468$.

Case 3. Taking the square root of the maximum

Problem. Let $A = \left(\begin{smallmatrix} 2 & 3 \\ 0 & -2 \end{smallmatrix}\right)$. Find the smallest positive C such that $\|Av\|_2 \le C\|v\|_2$ for every vector v.

Target model · GRPO
SmolLM3-3B

$t = y/x, \quad x \neq 0$
$\vdots$
$f(t) = \dfrac{13t^2 + 12t + 4}{t^2 + 1}$
$\vdots$
$f(2) = 16$
$\vdots$
"Therefore, the smallest positive real number C for which the inequality holds is 16."

Step analysis
Computes the squared ratio, then mistakes its maximum for C.

Answer: Incorrect

Peer model · GRPO
Qwen3-1.7B

$A^T A = \begin{pmatrix} 4 & 6 \\ 6 & 13 \end{pmatrix}$
$\vdots$
$\mu_1 = 16, \quad \mu_2 = 1$
$\sigma_1 = \sqrt{16} = 4$
$\|A\| = 4$
$\vdots$
$\boxed{4}$

Step analysis
Takes the square root of the largest eigenvalue of $A^T A$.

Answer: Correct

Target model · GRAFT
SmolLM3-3B

$t = y/x, \quad x \neq 0$
$\vdots$
$f(t) = \dfrac{13t^2 + 12t + 4}{t^2 + 1}$
$\vdots$
$f(2) = 16$
$\vdots$
$C = \sqrt{\max_t f(t)}$
$= \sqrt{16} = 4$
$\vdots$
$\boxed{4}$

Step analysis
Keeps scalar optimization and applies the same square-root step as the peer.

Answer: Correct

Figure 11: **Recovering the norm from the squared ratio in SmolLM3-3B (MATH500).** Target GRPO maximizes the squared norm ratio $f(t)$ with $t = y/x$, obtains a maximum of 16, and sets $C = 16$ without taking the square root. The peer computes the largest eigenvalue of $A^\top A$ as 16 and takes its square root to obtain the operator norm $C = 4$. GRAFT keeps the scalar optimization used by target GRPO and also takes the square root of the resulting maximum to obtain $C = \sqrt{16} = 4$.