LEARNING WHERE TO CONSTRAIN: META-LEARNED TRUST ALLOCATION FOR LANGUAGE-MODEL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy updates in auto-regressive reasoning change both token probabilities and the prefixes on which later decisions depend. Recent trust-region methods for LLM reinforcement learning account for token position and accumulated prefix divergence to improve training stability. However, a fixed or prespecified allocation may not suit different response lengths or changing task distributions. To study whether these constraints should instead be adaptive, we introduce Meta-TA (Meta-learned Trust Allocation), which learns an admissible position-weight floor and prefix-divergence budget by differentiating through the policy update. We show why a one-step update starting from the rollout policy provides no allocator gradient through an inactive constraint. A detached warm-up followed by a differentiable soft-constrained update enables allocator learning when the constraints become active. We then freeze the learned allocation and evaluate its transfer to fresh policy training. Our main evaluation uses the Qwen model family on mathematical reasoning, and we extend the study to abrupt task changes, gradual distribution shifts, and recurring tasks to examine adaptation and retention. To mitigate allocator overfitting, we select among saved allocator checkpoints using a separate subset and examine whether repeated updates on the query subset lead to overfitting. Decoupling allocator-regularization from policy-entropy helps distinguish restrictions on the learned update rule from changes in policy exploration. Together, these comparisons examine when learned constraints improve LLM reinforcement learning reasoning and if those gains justify additional cost.

1 INTRODUCTION

A policy update in autoregressive reasoning typically changes both token probabilities at a given prefix and the prefixes the policy visits afterward. For a response $y = (a_1, \ldots, a_T)$ to a given prompt x, this dependence generally follows from

$$p_\theta(y \mid x) = \prod_{t=1}^{T} \pi_\theta(a_t \mid x, a_{<t}). \tag{1}$$

Changing an early token alters the context for more subsequent decisions than changing a late token. Position therefore directly determines the extent of this influence, but token content also matters. Even a final sign or answer token contributes directly to correctness. PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) control local policy movement through clipping or regularization. CPPO (Mao et al., 2026) extends these controls to account for token position and accumulated prefix divergence. Position weights determine the local divergence allowance, while a prefix budget constrains divergence accumulated along the response. These controls provide a structured allocation of policy movement, but their parameters are chosen by hand. The same allocation may not suit short algebra problems and long proof-like solutions, or different stages of a changing training stream.

Using meta-gradient (Xu et al., 2018) methodology, we study whether performance on held-out prompts after a policy update provides a useful signal for learning this allocation.

We introduce Meta-TA, which meta learns the trust region allocation. Learning requires addressing the issue of no allocator signal at rollout start. Immediately after rollout collection, the current and behavior policies coincide. The divergence gate is then inactive, so a one-step update provides no

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.

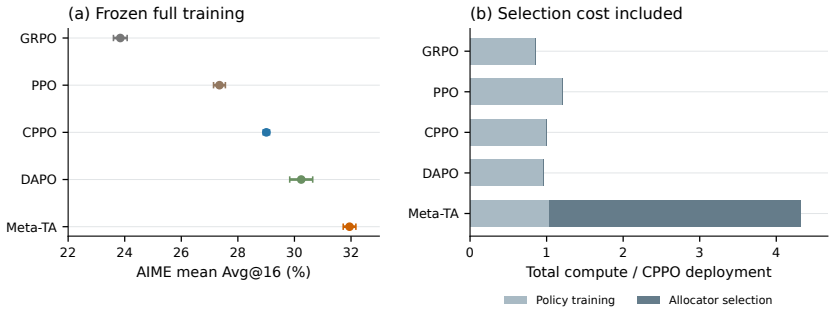

Figure 1: Frozen-allocation performance and total computational cost for Qwen3-8B-Base. Left: final AIME Avg@16, averaged across three yearly benchmarks and five training seeds. Right: normalized compute, including allocator selection and policy training. Although Meta-TA incurs more one-time selection cost, it improves AIME over CPPO under the same deployment-training budget; when the frozen allocator is reused across subsequent training runs, this selection overhead is amortized, making the accuracy-compute trade-off increasingly favorable.

query-mediated allocator gradient through the gate. Meta-TA first takes a detached warm-up step, then differentiates through a soft-constrained support update. After selection, we freeze the allocation and train a fresh policy using a hard mask. Our evaluation separates allocator selection from its value in subsequent training. We compare frozen allocations against predetermined allocations and include selection in the total computational cost.

Our paper investigates three questions: (1) how to adapt trust-region budgets during training, (2) whether learned budget allocations improve training over prespecified allocation parameters, and (3) how this adaptation affects performance as the task mixture changes in continual learning. We establish theoretical results identifying conditions that govern the meta-gradient signal and demonstrate empirical benefits for LLM RL post-training on mathematical tasks requiring long-horizon reasoning, across multiple generation-length budgets. Our continual-learning evaluations further highlight the value of adaptation under changing task mixtures, where overfitting to the current mixture can impair performance on other tasks. These findings contribute to the limited literature on LLM RL post-training methods designed explicitly to address continual-learning challenges.

2 Related Work

Trust regions and adaptive constraints. Relative Entropy Policy Search (REPS) (Peters et al., 2010) and Trust Region Policy Optimization (TRPO) (Schulman et al., 2015a) control policy improvement through relative-entropy constraints. Maximum a Posteriori Policy Optimization (Abdolmaleki et al., 2018) alternates policy improvement and policy fitting under KL constraints, while Adaptive TRPO (Shani et al., 2020) analyzes occupancy-weighted mirror-descent updates. Several methods adapt the strength or form of these constraints: PPO (Schulman et al., 2017) introduces an adaptive KL-penalty variant in addition to its clipped surrogate, TRGPPO (Wang et al., 2019) derives adaptive clipping ranges, PPO-DICE (Touati et al., 2020) regularizes PPO based policy optimization, it learns an auxiliary function to estimate divergence between the state-action occupancy distributions of the current and rollout policies using transitions collected under the rollout policy, uncertainty-Aware TRPO (Queeney et al., 2021) accounts for estimation uncertainty when determining policy updates. These methods establish precedents for adaptive policy constraints. Our method instead meta-learns a token-level soft masking function through the performance of the resulting policy update, allowing the update rule to adapt according to its effect on subsequent rollout performance.

Trust-region methods for language-model post-training. GRPO (Shao et al., 2024) replaces the learned critic with group-relative advantage estimates. DAPO (Yu et al., 2025) investigates asymmetric clipping, dynamic sampling, token-level optimization, and reward handling. QUATRO (Lee et al., 2026) derives prompt-wise trust-region updates using query-specific dual variables, while Trust Region Masking (TRM) (Li et al., 2026) studies sequence-level trust-region control for training

stability. Our position-dependent local bounds and cumulative-prefix mask build on CPPO (Mao et al., 2026). CPPO adjusts effective thresholds according to token position and accumulated prefix divergence, using predefined schedules and budget-selection rules. Meta-TA instead learns the parameters governing this allocation through meta-gradients, allowing the budgets to evolve according to their effect on subsequent policy optimization.

Meta-learning and online hyperparameter adaptation. MAML (Finn et al., 2017), meta-gradient reinforcement learning (Xu et al., 2018), and bilevel hyperparameter optimization (Franceschi et al., 2018) optimize initializations or learning hyperparameters through their effects on subsequent performance. Self-Tuning Actor-Critic (STAC) (Zahavy et al., 2020) applies meta-gradients to actor-critic hyperparameters, including entropy regularization. Other approaches include Hyperparameter Optimization on the Fly (HOOF) (Paul et al., 2019), which reuses trajectories for gradient-free hyperparameter selection, and Adaptive Interest (Klissarov et al., 2022), which learns state weights for emphatic reinforcement learning. Meta-TA follows the online meta-gradient approach, applying it to the parameters governing position-dependent and cumulative-prefix trust-region budgets. Closely related work by Occorso et al. (2022) learns a controller for selecting TRPO budgets from previously collected training experience. Meta-TA adapts budgets online within PPO's rollout-and-update loop: it samples from the current policy and differentiates through the policy update to optimize the budgets against a subsequent policy optimization objective.

Adaptive RL post-training. Recent LLM post-training methods increasingly adapt the optimization rule to the current policy or training state rather than using a fixed update throughout training. CPPO (Mao et al., 2026) uses position-dependent local constraints together with a sequence-adaptive prefix-divergence budget, improving reasoning accuracy and training stability across model scales. QUATRO (Lee et al., 2026) derives query-adaptive trust regions and reports more stable optimization under policy staleness and aggressive learning rates. DPPO (Qi et al., 2026) replaces sampled-token ratio clipping with direct divergence control, motivated by the instability of uniform PPO-style clipping in large-vocabulary language models. TRAPO (Su et al., 2025) similarly adapts the amount of supervised guidance during reasoning post-training through utility-dependent prefix selection while using trust-region control to stabilize the update. DAPO (Yu et al., 2025) provides a complementary example in which sampling and reward handling adapt to the evolving policy during RL training. These results support the broader view that the appropriate post-training constraint can depend on the current policy, sample, or stage of learning. Appendix C.9 contrasts them with Meta-TA.

Reasoning supervision and dense regularization. Let's Verify Step by Step (Lightman et al., 2023) studies process supervision, while OmegaPRM (Luo et al., 2024) automates intermediate-step labeling. Meta-TA instead uses final-answer rewards on query data to learn position- and prefix-dependent constraints, without a process-reward model. These constraints provide a form of intermediate guidance by regulating which token-level updates are permitted. This is analogous to token-level KL penalties in RL fine-tuning (Ouyang et al., 2022), which provide dense regularization by discouraging deviations from a reference policy. Meta-TA regulates deviations from the rollout policy through learned trust-region allocations. Unlike process supervision, neither mechanism directly evaluates the correctness of intermediate reasoning steps.

3 Meta-learned trust allocation: Methodology

Our proposed method, Meta-TA (Meta-learned Trust Allocation), extends PPO's uniform token level trust region control by learning position-dependent weights and prefix-divergence budgets to improve training stability. These parameters are learned through bilevel meta-gradient optimization. After selecting the allocator, we freeze its parameters and restart policy training from the initial checkpoint to assess whether the learned constraints transfer to a full training run. See Figure 2.

3.1 Position and prefix constraints

Let $s_t = (x, a_{<t})$ be the prefix at token t, and let μ denote the fixed policy used to collect rollouts. We define the importance ratio $\rho_t(\theta) = \pi_\theta(a_t \mid s_t)/\mu(a_t \mid s_t)$ and hold the support advantage $\widehat{A}_t$ fixed during each update. CPPO (Mao et al., 2026) limits local divergence through $w_t D_t \le \delta$ and

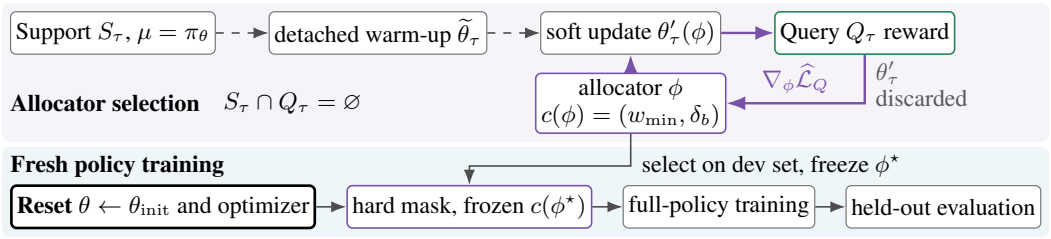

Figure 2: Meta-TA overview: allocator selection (top; thick arrows are differentiated and dashed arrows are detached) and fresh training with a reset policy and optimizer and the frozen $\phi^\star$ (bottom).

accumulated prefix divergence through $\sum_{j\le t} w_j D_j \le \delta + \delta_b W_{t-1}$, where $W_{t-1} = \sum_{j<t} w_j$ and $W_0 = 0$. Here total variation (TV) D_t measures current policy to behavior token divergence, $\delta > 0$ sets the base allowance, and δ_b controls how quickly the prefix budget grows.

For $T > 1$, the global allocator uses the linear schedule $w_t = 1 - (1 - w_{\min})(t-1)/(T-1)$. Lowering $w_{\min}$ increases the local allowance δ/w_t toward the end of a response. The prefix constraint couples this choice to accumulated divergence and the budget increment δ_b. We learn unconstrained parameters $\phi = (u, v) \in \mathbb{R}^2$ through the admissible parameterization

$$w_{\min} = \epsilon_w + (1 - \epsilon_w)\sigma(u), \quad r_b = \epsilon_b + (1 - 2\epsilon_b)\sigma(v), \quad \delta_b = \delta w_{\min} r_b. \tag{2}$$

Here σ is the logistic function, with $0 < \epsilon_w < 1$ and $0 < \epsilon_b < 1/2$. We set $w_1 = 1$ for $T = 1$. The ratio r_b expresses the budget increment as a fraction of $\delta w_{\min}$. The resulting map $c(\phi) = (w_{\min}, \delta_b)$ takes values in $\Phi = \{(w_{\min}, \delta_b) : 0 < w_{\min} < 1,\ 0 < \delta_b < \delta w_{\min}\}$.

For the differentiated support step, we detach the measured divergence, $\bar{D}_t = \mathrm{sg}(D_t)$, where sg denotes stop-gradient. We combine the two constraints into their largest normalized value:

$$x_t(\phi) = \max\left\{ \frac{w_t \bar{D}_t}{\delta}, \frac{\sum_{j\le t} w_j \bar{D}_j}{\delta + \delta_b W_{t-1}} \right\}. \tag{3}$$

Both constraints hold when $x_t \le 1$. The soft mask assigns full weight to locally inward directions and to outward directions satisfying both constraints

$$M_t^{\text{soft}}(\phi) = \begin{cases} 1, & \widehat{A}_t(\rho_t - 1) \le 0 \text{ or } x_t \le 1, \\ x_t^{-1}, & \widehat{A}_t(\rho_t - 1) > 0 \text{ and } x_t > 1. \end{cases} \tag{4}$$

The sign of $\widehat{A}_t(\rho_t - 1)$ identifies movement toward or away from the rollout policy under the token's local surrogate. Shared parameters couple token updates, so this local test does not determine each token's movement after a full policy step. We hold divergence measurements and Boolean branches fixed for policy differentiation, while allocator gradients pass through the weights and budgets. The outer objective thus learns to adjust and control gradient scaling.

With G responses per support prompt, the support loss averages over responses and sums over their valid tokens:

$$\mathcal{L}_S(\theta, \phi, S_\tau) = -\frac{1}{|S_\tau| G} \sum_{i=1}^{|S_\tau|} \sum_{j=1}^{G} \sum_{t=1}^{T_{ij}} M_{ijt}^{\text{soft}}(\phi) \rho_{ijt}(\theta) \widehat{A}_{ij}. \tag{5}$$

Here T_{ij} is the length of response j to prompt i. After allocator selection, policy training uses the hard mask obtained by replacing the second branch of Equation 4 with zero. The soft gate scales update gradients and leaves the policy distribution defined by π_θ.

3.2 Learning from disjoint prompts

A meta-task τ specifies a prompt distribution $p_\tau(x)$, from which we draw disjoint support and query sets S_τ and Q_τ. The allocator is trained through bilevel meta-gradient optimization, using support rollouts for the inner policy update and query rollouts for the outer objective. To mitigate overfitting from repeated query reuse, we select allocator checkpoints using a separate development set and reserve the test set for final evaluation.

The global study draws both sets from one pooled mathematical distribution and tests generalization across prompts. Subject-conditioned and continual learning studies also change the task distribution, extending the evaluation to transfer across these distributions.

3.3 A WARM-STARTED BILEVEL UPDATE

Immediately after rollout collection, the current policy matches μ, so $D_t = 0$ and all constraint masks equal one. The first policy update is therefore independent of the allocator parameters, yielding a zero query-mediated allocator gradient (Proposition 1). A related issue arises in MDPO (Tomar et al., 2020), which uses multiple update steps because its KL penalty has zero policy gradient at the reference policy. We first take a detached warm-up step while keeping μ fixed to move the policy away from the rollout policy so that constraints become active. We then differentiate through the subsequent support update, enabling query feedback to train the allocator when the masks depend on its parameters.

Let $F_Q(\vartheta, Q_\tau) = -|Q_\tau|^{-1} \sum_{x \in Q_\tau} \mathbb{E}_{y \sim \pi_\vartheta(\cdot|x)} R(x, y)$ denote negative expected query reward. We optimize the allocator using meta-gradients through the policy update

$$\begin{aligned} \min_{\phi \in \mathbb{R}^2} \quad & \mathbb{E}_{\tau, S_\tau, Q_\tau} \left[F_Q(\theta'_\tau(\phi), Q_\tau) \right], \\ \text{subject to} \quad & \widetilde{\theta}_\tau = \mathrm{sg}\left[\theta - \alpha_0 \nabla_\theta \mathcal{L}_0(\theta, S_\tau) \right], \\ & \theta'_\tau(\phi) = \widetilde{\theta}_\tau - \alpha_1 \nabla_{\widetilde{\theta}} \mathcal{L}_S(\widetilde{\theta}_\tau, \phi, S_\tau), \\ & S_\tau \cap Q_\tau = \varnothing, \qquad c(\phi) \in \Phi. \end{aligned} \tag{6}$$

Both support losses are policy gradient clipped GRPO objective (Shao et al., 2024). In the warm-up loss $\mathcal{L}_0$, the mask equals one, so the update is independent of the allocator. In $\mathcal{L}_S$, the learned soft mask scales each token's policy-gradient contribution as defined in Equation 5. Both steps reuse the same support responses and keep μ fixed. We detach the warm-up parameters $\widetilde{\theta}_\tau$ and differentiate only through the second support update.

We estimate the query gradient using REINFORCE (Williams, 1992). For $G > 1$, let b_{ij} be the mean reward of the other $G - 1$ responses to prompt i. This yields the following surrogate meta-loss:

$$\widehat{\mathcal{L}}_Q = -\frac{1}{|Q_\tau| G} \sum_{i,j} \mathrm{sg}(R_{ij} - b_{ij}) \log \pi_{\theta'_\tau}(y_{ij} \mid x_i). \tag{7}$$

Conditioned on the prompt and the adapted policy, the leave-one-out baseline is independent of response j. Under the score-function assumptions in Appendix A, the expected surrogate gradient equals ∇F_Q. Algorithm 1 summarizes the allocation training and its deployment within policy optimization loop, and Figure 7 shows the update graph.

Algorithm 1 Meta-TA: Meta-learned trust allocation rule for policy optimization

Inputs: Initial policy θ_{init}, allocator ϕ, and disjoint policy-training, support, query, and development prompt pools.

1. Initialize the persistent policy $\theta \leftarrow \theta_{\text{init}}$ and allocator ϕ.
2. Repeat for each allocator-selection episode:
   (a) Sample a task τ: disjoint prompt sets S_τ, Q_τ.
   (b) Collect support responses with $\mu = \pi_\theta$.
   (c) Take the detached warm-up step followed by a update step to obtain $\theta'_\tau(\phi)$ (Eq. 6).
   (d) Sample fresh query responses from $\pi_{\theta'_\tau}$ and update only ϕ using Eq. 7.
   (e) Discard the temporary parameters and advance the persistent policy θ to the next shared selection checkpoint.
3. Select the allocator checkpoint with the highest reward under development subset. Freeze its parameters ϕ. Reset the policy to θ_{init} and reset the policy optimizer.
4. Train the full policy on the policy-training pool with the frozen allocation and hard mask.
5. Evaluate the final policy without updating policy or allocator parameters.

3.4 STRUCTURED AND CONTINUAL ALLOCATION

To extend allocation beyond a linear schedule, we use a mixture of position-dependent curves. For $T > 1$, let $q_t = (t-1)/(T-1)$ and define

$$w_t = w_{\min} + (1 - w_{\min}) \textstyle\sum_{k=1}^{3} a_k (1 - q_t)^{p_k}, \quad a_k \geq 0, \quad \sum_k a_k = 1, \quad 0 < p_k \leq 1. \tag{8}$$

Softmax coefficients and bounded exponents enforce these constraints. A conditioned allocator receives task identity and support statistics. Proposition 2 establishes positivity, monotonicity of w_t and $(T-t)/w_t$, and admissible prefix budgets for each update. The guarantee concerns per-update schedule admissibility rather than convergence.

We evaluate three deployment modes. A frozen global allocator keeps its schedule parameters fixed throughout full training. A frozen conditioned controller also keeps its parameters fixed, while its outputs vary with task identity and support statistics. In continual training, online Meta-TA refreshes the allocator at declared checkpoints using only current and previously available query pools. The policy continues across task boundaries.

Our overfitting study regularizes the physical parameters toward a reference allocation and adds an entropy-related penalty for nonlinear mixtures:

$$\Omega(\phi) = \lambda_c \|(w_{\min}, r_b) - (w^0_{\min}, r^0_b)\|_2^2 + \lambda_a D_{\mathrm{KL}}(a \| \mathrm{Unif}(3)). \tag{9}$$

The first term favors allocations near $(w^0_{\min}, r^0_b)$. Since $D_{\mathrm{KL}}(a\|\mathrm{Unif}(3)) = \log 3 - H(a)$, the second discourages concentrated mixture weights. The two-parameter linear allocator uses only the physical-parameter prior. We test policy entropy separately to distinguish regularization of the allocator from changes in policy exploration.

4 Experiments

4.1 Experimental setup

We evaluate whether learned allocation transfers to fresh policy training, how its benefit depends on response length, and whether refreshing the allocator helps under distribution shift. Our primary model is Qwen3-8B-Base (Yang et al., 2025), trained on a DAPO-Math-17k (Yu et al., 2025) prompt split. We divide prompts into policy-training, support, query, and development pools, keeping each duplicate group within a single pool. Evaluation covers AIME24/25/26, MATH-500 (Hendrycks et al., 2021; Lightman et al., 2023), and the English text-only mathematics subset of OlympiadBench (He et al., 2024). Each terminal checkpoint is evaluated with $R_{\mathrm{eval}} = 5$ independent reporting repetitions that use the same benchmark prompts and decoding configuration but independent generation randomness. Within each repetition, AIME Avg@16 averages correctness over 16 independently sampled responses per prompt and weights AIME24/25/26 equally, while MATH-500 and OlympiadBench sample one response per problem and report pass@1. The per-training-seed benchmark score is the arithmetic mean of these repetition-level scores.

Each stationary run uses 120M generated response tokens over 480 updates, with an 8,192-token response cap. We use five paired seeds and share the initial checkpoint, data pools, verifier, and evaluation schedule across methods. Reporting repetitions are averaged within each training seed before any cross-seed aggregation; the five independent training seeds, rather than the individual reporting repetitions, are the units used to compute the reported confidence intervals. Meta-TA, CPPO, and GRPO also share the policy-optimizer configuration.

We note that CPPO (Mao et al., 2026) uses $w_{\min} = 0.80$ throughout, but treats the prefix-budget parameter differently for post-trained and Base models. For the post-trained reasoning model, CPPO uses the fixed value $\delta_b = 0.015$. For Base models, including Qwen3-8B-Base, it uses the sequence-adaptive calibration $\delta_b^{\mathrm{seq}} = \min\{2\delta_b^{\min}, \max[\delta_b^{\min}, P_{90}(D_{1:T})]\}$, with $\delta_b^{\min} = 0.020$ for Qwen3-8B-Base. Our CPPO baseline reproduces the same $w_{\min} = 0.80$ and lower endpoint $\delta_b^{\min} = 0.020$ of the same published Base-model calibration. For every Base-model CPPO training sequence, we recompute $P_{90}(D_{1:T})$ and apply the clamp. PPO and DAPO retain their defining objectives under same token and evaluation budgets, and PPO's compute cost includes its critic. Appendix B gives the shared and method-specific settings.

4.2 Transfer to fresh policy training

The frozen global allocator reaches 31.95 AIME points after fresh full-policy training, compared with 29.01 for CPPO, a gain of 2.94 points. Gains on MATH-500 and OlympiadBench are smaller. These comparisons measure transfer to a restarted policy under the same deployment budgets. During selection, $w_{\min}$ decreases from 0.80 to 0.689, and the deployed allocation uses $\delta_b = 0.01329$; see

Table 1: Frozen Qwen3-8B-Base retraining. AIME averages yearly Avg@16 scores; MATH-500 and OlympiadBench report pass@1. Each final checkpoint is evaluated with five independent reporting repetitions. Within each training seed, we first average the repetition-level benchmark scores; reported means and confidence intervals are then computed across five independent training seeds. Each training run receives 120M deployment tokens and 480 updates. KL reports nats per token. Total compute includes allocator selection and uses one deployment run as the unit.

Method	AIME	MATH-500	OlympiadBench	KL	Compute
GRPO	23.84±0.24	73.06±0.32	46.56±0.64	0.01984	0.86
PPO	27.33±0.21	73.52±0.52	47.00±0.52	0.01721	1.21
DAPO	30.25±0.42	74.76±0.48	48.56±0.40	0.01628	0.96
CPPO	29.01±0.11	74.23±0.36	48.08±0.28	0.01514	1.00
Meta-TA	31.95±0.21	75.36±0.20	49.10±0.40	0.01478	4.33

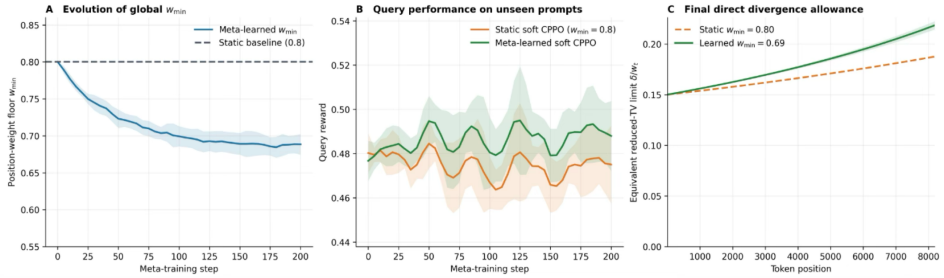

Figure 3: Global allocator learning. Left: the position floor decreases from 0.80 toward 0.689. Middle: reward on disjoint query prompts under the temporarily updated policy. Right: the corresponding local divergence allowance δ/w_t at $\delta = 0.15$. Curves summarize five seeds, with 95% seed intervals. Prefix feasibility also depends on accumulated divergence and the prefix budget, so the larger late local allowance does not relax every prefix constraint.

Figure 3. We use query reward to measure the performance of the temporarily updated policy. The main benchmark scores are obtained by averaging the independent terminal-checkpoint reporting repetitions within each training seed, after training has reset both the policy and optimizer. This separates allocator transfer from policy progress during selection. Appendix C.1 provides the full training curves.

4.3 Dependence on response horizon

Longer responses increase the potential reach of an early policy change. We examine this dependence with two response-cap sweeps. The evaluation-only sweep changes the inference cap while holding the 8k-trained policy fixed. The matched sweep retrains separately at each cap. Figure 4 compares Meta-TA with CPPO under both settings, separating changes in training from inference truncation.

For response i, define the accepted outward ratio-deviation mass as $q_{it} = \mathbf{1}\{\widehat{A}_i(\rho_{it} - 1) > 0\} M_{it} |\widehat{A}_i(\rho_{it} - 1)|$. Its normalized remaining horizon is

$$h_{\text{eff}} = \frac{\sum_{i:T_i>1} \sum_t q_{it}(T_i - t)/(T_i - 1)}{\sum_{i:T_i>1} \sum_t q_{it}}. \tag{10}$$

We exclude singleton responses and treat a zero denominator as undefined. Lower h_{eff} places more accepted mass at later positions. Response length and policy-gradient variance are separate quantities. Since masking also changes the denominator, we interpret this statistic alongside acceptance rates and response lengths. At 16k, CPPO and Meta-TA have h_{eff} values of 0.475 and 0.430, respectively, with mean response lengths of 9,566 and 9,530 tokens. Their overall ratio variances are 0.0534 and 0.0475. Meta-TA thus places accepted update mass later in responses of similar length.

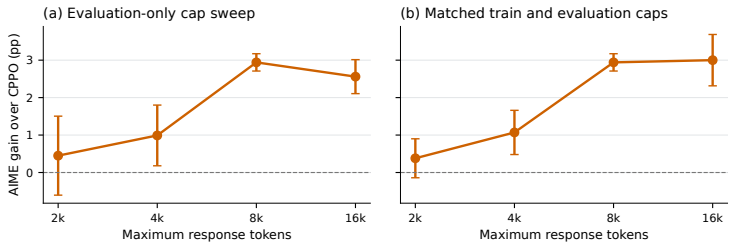

Figure 4: Response-cap controls for Qwen3-8B-Base. Both panels show Meta-TA's paired AIME advantage over CPPO in percentage points. Left: vary the inference cap of policies trained at 8k. Right: retrain each policy at the indicated cap. Intervals summarize five paired seeds, and the 8k condition is shared between panels. Gains are smaller at short caps. The evaluation-only advantage peaks at 8k, while the matched-training advantage persists at 16k.

4.4 Continual Adaptation

We study three forms of distribution shift: abrupt subject changes, gradual mixture drift, and a recurring stream. Frozen Meta-TA tests transfer across the stream without changing the allocator. Online Meta-TA refreshes the allocator using current and historical query pools. Each method receives either no replay or a fixed 10% replay share. Equal access to past data separates the contribution of allocation from the benefit of replay.

Let $A_{j,k}$ denote task-k accuracy after stage j. Final average accuracy summarizes performance on both earlier and newly acquired tasks. Forgetting compares each earlier task's best pre-final score with its final score, while backward transfer compares its final score with the score after first acquisition. Together, these metrics distinguish new-task gains from retention. Revisited tasks retain their first-acquisition reference. Appendix B.4 specifies the task matrices and denominators.

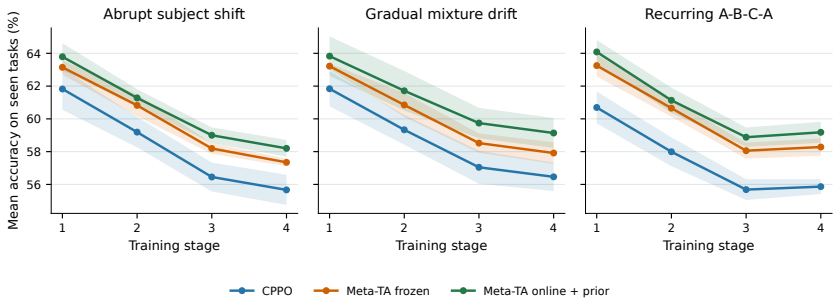

Figure 5: Continual mathematical training with 10% replay. Each panel shows stage-end mean accuracy across subjects seen so far. The averaging set grows as new subjects arrive, and the final recurrent stage revisits algebra. Online Meta-TA includes the allocator prior. The policy continues across task boundaries, and online selection uses only available query pools, excluding future and test prompts. Table 12 reports final all-task accuracy and forgetting.

With 10% replay, online Meta-TA reaches 59.22 mean accuracy under gradual drift, compared with 56.46 for CPPO and 57.92 for frozen Meta-TA. Forgetting is 0.96 points for online Meta-TA and 1.69 points for CPPO. The gain from refreshing rather than freezing the allocator is smaller under abrupt shifts, and the prior does not improve the gradual-drift result. These results support allocator refresh as a complement to replay, with benefits varying across streams and regularization choices. Table 12 reports all final metrics.

4.5 Allocator Overfitting

Repeated outer updates risk overfitting the query pool even with disjoint support and query prompts. We vary query-pool size, allocator capacity, and regularization, then compare selection reward with

reward on unseen reporting prompts. The gap between these rewards measures overfitting from query reuse.

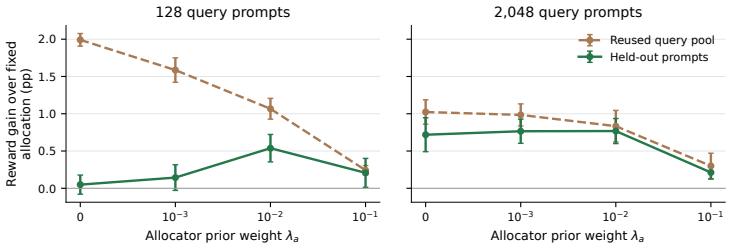

Figure 6: Allocator overfitting under repeated query reuse. Panels vary query-pool size and the strength of the nonlinear mixture penalty. Selection and held-out reporting gains use disjoint prompts, with a separate pool for choosing the regularization coefficient.

Small query pools produce the largest gap between selection and held-out reward. Moderate mixture regularization improves held-out reward, whereas the largest coefficient suppresses both selection and held-out gains. The benefit decreases as the query pool grows and is absent in some settings. Capacity and policy-entropy controls distinguish query reuse from other sources of adaptation. Appendix B.5 describes coefficient selection.

4.6 Ablations and Computational Cost

The ablations examine how the learning signal and deployment choices affect performance. Removing warm-up tests gate inactivity, and reusing prompts tests support memorization. Removing divergence detachment changes the differentiated update. Comparing hard and soft deployment tests sensitivity to the learning surrogate. Appendix C.4 reports these interventions separately.

Including candidate evaluation and reference-checkpoint construction, Meta-TA uses 4.33 times the compute of one CPPO deployment. This end-to-end cost reflects a single deployment with the full selection cost charged upfront; because the selected allocator is reusable, its amortized overhead decreases with the number of subsequent training runs. Reusing a frozen allocator spreads selection cost across later training runs, while online refresh adds recurring cost. Table 13 gives the cost breakdown. Appendix C.8 gives a worked example of the local and prefix constraints.

5 Conclusion

In this work, we propose Meta-TA, a novel LLM RL policy optimization method learns position and prefix constraints from the performance of a temporarily updated policy on disjoint query prompts. A detached warm-up followed by a differentiated soft-constrained update trains the allocator, while an admissible parameterization preserves the schedule conditions. We additionally observe that frozen retraining improves downstream accuracy, with larger gains at longer matched training caps. Our continual-learning and query-pool experiments further show how allocator refresh, replay, and regularization affect transfer. We assess these gains through accuracy and retention alongside the full cost of allocator selection and policy training. In summary, we now have a concrete and more informed understanding about when learned constraints and allocations improve reasoning and whether those gains indeed justify their additional cost and trade-offs.

AI USE STATEMENT

OpenAI ChatGPT assisted with manuscript writting, mathematical checks, and reference checking.

ETHICS STATEMENT

The study concerns mathematical problems with verifiable final answers. Accuracy, ratio variance, and retention do not establish safety outside this setting.

REPRODUCIBILITY STATEMENT

We describe allocator selection, gradient flow, and frozen policy training in Section 3 and Algorithm 1. Section 4 and Appendix B and document the data partitions, training settings, token budgets, and evaluation metrics. The main comparisons use five paired seeds with a shared initial checkpoint, data pools, verifier, and evaluation schedule. Appendix C furthermore includes details about allocator selection, policy training, reference-checkpoint construction, and online refresh costs.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1568–1577, 2018. URL https://proceedings.mlr.press/v80/franceschi18a.html.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Mikael Henaff, Scott Fujimoto, Michael Matthews, and Michael Rabbat. Scalable option learning in high-throughput environments. *arXiv preprint arXiv:2509.00338*, 2025. doi: 10.48550/arXiv.2509.00338. URL https://arxiv.org/abs/2509.00338.

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. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. doi: 10.48550/arXiv.2106.09685. URL https://arxiv.org/abs/2106.09685.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Martin Klissarov, Rasool Fakoor, Jonas W. Mueller, Kavosh Asadi, Taesup Kim, and Alexander Smola. Adaptive interest for emphatic reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/008079ec00eec9760ee93af5434ee932-Abstract-Conference.html.

Doyeon Lee, Eunyi Lyou, Hyunsoo Cho, Sookyung Kim, Joonseok Lee, and Jaemoo Choi. QUATRO: Query-adaptive trust region policy optimization for LLM fine-tuning. *arXiv preprint arXiv:2602.04620*, 2026. doi: 10.48550/arXiv.2602.04620. URL https://arxiv.org/abs/2602.04620.

Yingru Li, Jiacai Liu, Jiawei Xu, Yuxuan Tong, Ziniu Li, Qian Liu, and Baoxiang Wang. Trust region masking for long-horizon llm reinforcement learning. *ICML*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. doi: 10.48550/arXiv.2305.20050. URL https://arxiv.org/abs/2305.20050.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024. doi: 10.48550/arXiv.2406.06592. URL https://arxiv.org/abs/2406.06592.

Renjie Mao, Xiangxin Zhou, Lvfang Tao, Yixin Ding, Yu Shi, Yongguang Lin, Yuheng Wu, Honglin Zhu, Qian Qiu, and Wenxi Zhu. Beyond uniform token-level trust region in LLM reinforcement learning. *arXiv preprint arXiv:2606.10968*, 2026. doi: 10.48550/arXiv.2606.10968. URL https://arxiv.org/abs/2606.10968.

Manuel Occorso, Luca Sabbioni, Alberto Maria Metelli, and Marcello Restelli. Trust region meta learning for policy optimization. In *ECMLPKDD Workshop on Meta-Knowledge Transfer*, pp. 62–74. PMLR, 2022.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Supratik Paul, Vitaly Kurin, and Shimon Whiteson. Fast efficient hyperparameter tuning for policy gradient methods. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/743c41a921516b04afde48bb48e28ce6-Abstract.html.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010. doi: 10.1609/aaai.v24i1.7727. URL https://ojs.aaai.org/index.php/AAAI/article/view/7727.

Penghui Qi, Xiangxin Zhou, Zichen Liu, Tianyu Pang, Chao Du, Min Lin, and Wee Sun Lee. Rethinking the trust region in LLM reinforcement learning. *arXiv preprint arXiv:2602.04879*, 2026. URL https://arxiv.org/abs/2602.04879.

James Queeney, Ioannis Ch. Paschalidis, and Christos G. Cassandras. Uncertainty-aware policy optimization: A robust, adaptive trust region approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9377–9385, 2021. doi: 10.1609/aaai.v35i11.17130. URL https://ojs.aaai.org/index.php/AAAI/article/view/17130.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, 2015a. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b. doi: 10.48550/arXiv.1506.02438. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi: 10.48550/arXiv.1707.06347. URL https://arxiv.org/abs/1707.06347.

Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5668–5675, 2020. doi: 10.1609/aaai.v34i04.6021. URL https://ojs.aaai.org/index.php/AAAI/article/view/6021.

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. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Mingyu Su, Jian Guan, Yuxian Gu, Minlie Huang, and Hongning Wang. Trust-region adaptive policy optimization. *arXiv preprint arXiv:2512.17636*, 2025. URL https://arxiv.org/abs/2512.17636.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*, 2020. doi: 10.48550/arXiv.2005.09814. URL https://arxiv.org/abs/2005.09814.

Ahmed Touati, Amy Zhang, Joelle Pineau, and Pascal Vincent. Stable policy optimization via off-policy divergence regularization. In *Conference on Uncertainty in Artificial Intelligence*, pp. 1328–1337. PMLR, 2020.

Yuhui Wang, Hao He, Xiaoyang Tan, and Yaozhong Gan. Trust region-guided proximal policy optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/a666587afda6e89aec274a3657558a27-Abstract.html.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://link.springer.com/article/10.1007/BF00992696.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2715518c875999308842e3455eda2fe3-Abstract.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

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. doi: 10.48550/arXiv.2503.14476. URL https://arxiv.org/abs/2503.14476.

Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. A self-tuning actor-critic algorithm. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f02208a057804ee16ac72ff4d3cec53b-Abstract.html.

A FORMAL ANALYSIS AND GRADIENT SEMANTICS

The analysis addresses two questions: when the temporary update carries an allocator gradient, and which schedules satisfy the per-update constraints. Both results concern the specified update.

A.1 POPULATION QUERY OBJECTIVE AND ITS ESTIMATOR

Fix a meta-episode, its behavior policy, its support rollouts, and its detached warm-up parameters. Let $\theta' = \theta'_\tau(\phi)$ denote the adapted parameters. For a query prompt x, a response $y = (a_1, \dots, a_T)$ has probability

$$p_{\theta'}(y \mid x) = \prod_{t=1}^{T} \pi_{\theta'}(a_t \mid x, a_{<t}). \tag{11}$$

The response includes its termination decision under the chosen generation cap. We assume a bounded verifier reward $R(x, y)$ with no explicit dependence on the policy parameters. The population outer loss is

$$F_Q(\theta') = -\mathbb{E}_{x\sim p_\tau}\mathbb{E}_{y\sim\pi_{\theta'}(\cdot|x)}[R(x, y)]. \tag{12}$$

The REINFORCE (Williams, 1992) score-function identity gives, under the usual differentiation-under-expectation conditions,

$$\nabla_{\theta'} F_Q = -\mathbb{E}\left[R(x, y)\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t \mid x, a_{<t})\right]. \tag{13}$$

The derivative acts on response probabilities, with sampled tokens and verifier outputs held fixed.

For each query prompt, sample $G \geq 2$ responses independently conditional on the adapted policy. Define the leave-one-out reward baseline

$$b^{\text{loo}}_{i,j} = \frac{1}{G-1}\sum_{\ell\neq j} R(x_i, y_{i,\ell}), \qquad A^Q_{i,j} = R(x_i, y_{i,j}) - b^{\text{loo}}_{i,j}. \tag{14}$$

The query score loss is

$$\widehat{\mathcal{L}}_Q = -\frac{1}{|Q_\tau|G}\sum_{i,j} \text{sg}[A^Q_{i,j}] \sum_t \log \pi_{\theta'}(a_{i,j,t} \mid s_{i,j,t}). \tag{15}$$

Conditional on the prompt and adapted policy, the baseline is independent of response j. Since the expected policy score is zero, subtracting the baseline preserves Equation 13. Fresh responses therefore give a Monte Carlo estimator of the population query gradient.

The scalar score loss is a surrogate whose gradient has the stated expectation. Support-side reward standardization defines the inner update and leaves this query argument unchanged. Query-side standardization differs: a standard deviation computed from the same response group depends on response j, so the independence argument no longer applies.

A.2 THE DIFFERENTIATED COMPUTATION

Write $\theta' \in \mathbb{R}^d$ and $\phi \in \mathbb{R}^m$. For the global allocator, $m = 2$. During one episode, the persistent checkpoint and behavior policy act as fixed inputs. The stop-gradient on the warm-up gives $\partial\widetilde{\theta}/\partial\phi = 0$. For one differentiated support step,

$$J_\phi := \frac{\partial\theta'}{\partial\phi^\top} = -\alpha_1 H_{\theta\phi}, \qquad H_{\theta\phi} := \frac{\partial\nabla_{\widetilde{\theta}}\mathcal{L}_S}{\partial\phi^\top} \in \mathbb{R}^{d\times m}. \tag{16}$$

The query-mediated hypergradient is

$$\widehat{g}_\phi = J_\phi^\top \nabla_{\theta'} \widehat{\mathcal{L}}_Q. \tag{17}$$

Automatic differentiation computes this expression for the cached temporary update. Averaging over fresh queries recovers its conditional population hypergradient. Earlier persistent training and support sampling remain fixed inputs, so their dependence on past allocator parameters lies outside this derivative.

For precision, let c_t denote fixed nonnegative sample-normalization weights. Freeze the support advantages, token-divergence measurements, and Boolean inward/outward decisions when taking a policy derivative. Away from gate boundaries, the support loss has the local form

$$\mathcal{L}_S = -\sum_t c_t \widehat{A}_t M_t(\phi)\rho_t(\widetilde{\theta}). \tag{18}$$

The corresponding Jacobian is

$$J_\phi = \alpha_1 \sum_t c_t \widehat{A}_t \nabla_{\widetilde{\theta}} \rho_t (\nabla_\phi M_t)^\top. \tag{19}$$

The mixed derivative holds divergence measurements fixed and differentiates the weights, budgets, and prefix sums through ϕ. Detaching the whole mask would remove this signal. A support regularizer independent of ϕ contributes no additional mixed-derivative term.

The reciprocal gate is continuous at its feasibility threshold and piecewise differentiable. Its threshold, the maximum in x_t, and the direction test create branch boundaries. Thus "soft" refers to gradient attenuation. Equations 18–19 describe the local surrogate away from those boundaries.

The sign $\widehat{A}_t(\rho_t - 1) \le 0$ identifies an inward local surrogate direction. The full update combines tokens through shared parameters and optimizer state. Its effect on individual ratios or full-policy divergence need not follow every local direction.

A.3 Behavior-matched one-step degeneracy

MDPO (Tomar et al., 2020) identifies a zero policy gradient of the KL proximity term at the reference policy, motivating multiple inner steps. The proposition below concerns a different derivative: allocator sensitivity of a masked support update. Local mask constancy removes this sensitivity throughout the support gradient. The result specializes the inactivity issue under explicit mask assumptions.

Proposition 1 (Inactive-gate degeneracy)**.** *Fix support data, a behavior policy $\mu = \pi_{\theta^0}$, and an allocator setting ϕ^0. Assume the allocator enters the support objective only through masks $M_t(\theta, \phi)$. Assume a neighborhood of (θ^0, ϕ^0) in the differentiated support computation satisfies $M_t(\theta, \phi) = 1$ for every token. Let*

$$\theta^1(\phi) = \theta^0 - \alpha \nabla_\theta \mathcal{L}_S(\theta^0, \phi), \tag{20}$$

where θ^0 and α have no allocator dependence. For any differentiable query objective F_Q without direct allocator dependence,

$$\left.\frac{\partial \theta^1}{\partial \phi^\top}\right|_{\phi^0} = 0, \qquad \nabla_\phi F_Q(\theta^1(\phi))\big|_{\phi^0} = 0. \tag{21}$$

Proof. In the stated neighborhood, all masks equal one and have no allocator dependence. Every remaining term in the support objective is independent of ϕ. Thus $\partial \nabla_\theta \mathcal{L}_S / \partial \phi^\top = 0$. Differentiating Equation 20 proves the first equality. The chain rule gives $\nabla_\phi F_Q = (\partial \theta^1 / \partial \phi^\top)^\top \nabla_{\theta^1} F_Q = 0$. □

At exact behavior matching, a divergence estimator which vanishes on identical distributions gives $D_t = 0$, $\rho_t = 1$, and $x_t = 0$. Positive budgets and continuity make the gate locally inactive. The proposition then gives zero allocator gradient even if the policy gradient is nonzero.

Warm-up changes the point at which the constrained update acts. A nonzero hypergradient requires an allocator-sensitive outward gate and a resulting policy direction aligned with the query gradient. Zero advantages, cancellation, or orthogonality still produce zero signal. Multiple differentiated steps require a separate argument because each step changes the input to the next gate.

A.4 Admissible global and nonlinear schedules

For $T \ge 2$, let $q_t = (t-1)/(T-1)$. Consider

$$w_t = w_{\min} + (1 - w_{\min}) \sum_{k=1}^{K} a_k (1 - q_t)^{p_k}, \qquad \sum_k a_k = 1. \tag{22}$$

The linear family is the case $p_k = 1$ for all active components. A constrained nonlinear allocator emits simplex weights a_k and exponents $0 < p_k \leq 1$. The terminal weight and normalized prefix budget follow Equation 2.

Proposition 2 (Per-update schedule admissibility)**.** *Let $\delta > 0$, $0 < w_{\min} < 1$, $a_k \geq 0$, $\sum_k a_k = 1$, $0 < p_k \leq 1$, and $\delta_b = \delta w_{\min} r_b$ with $0 < r_b < 1$. For the schedule in Equation 22,*

1. *$w_1 = 1$, $w_T = w_{\min}$, and $w_t \geq w_{\min} > 0$.*
2. *w_t is non-increasing in t.*
3. *$(T - t)/w_t$ is non-increasing in t.*
4. *$0 < \delta_b < \delta w_{\min}$.*

Proof. The endpoint identities and positivity follow from the simplex constraint. Define $w(q) = w_{\min} + (1 - w_{\min}) \sum_k a_k (1 - q)^{p_k}$. For $0 \leq q < 1$,

$$w'(q) = -(1 - w_{\min}) \sum_k a_k p_k (1 - q)^{p_k - 1} \leq 0. \tag{23}$$

The schedule is continuous at $q = 1$, so monotonicity extends to the final position. Also,

$$w(q) + (1 - q) w'(q) = w_{\min} + (1 - w_{\min}) \sum_k a_k (1 - p_k)(1 - q)^{p_k} > 0. \tag{24}$$

For $q < 1$, this identity gives

$$\frac{d}{dq} \frac{1 - q}{w(q)} = -\frac{w(q) + (1 - q) w'(q)}{w(q)^2} < 0. \tag{25}$$

Continuity extends the result to $q = 1$. Since $(T - t)/w_t = (T - 1)(1 - q_t)/w(q_t)$, the discrete horizon-weight ratio is non-increasing. Finally, multiplying $0 < r_b < 1$ by $\delta w_{\min} > 0$ gives the strict budget inequality. □

For a single-token response, set $w_1 = 1$ and omit the normalized remaining-horizon statistic. A conditioned allocator satisfies the proposition whenever each emitted parameter vector obeys its constraints. Freezing the controller fixes its parameters, while its outputs still vary with support context.

The proposition establishes properties of the schedule passed to one optimizer update. Exact feasibility after a shared-parameter step and monotonic return require further assumptions. Reciprocal gradient scaling also does not define a mixture policy.

A.5 Finite-difference checks and multi-step extensions

The gradient check targets a deterministic finite-batch surrogate. Cache the support rollouts, behavior log probabilities, support advantages, detached warm-up state, divergence measurements, query responses, and stopped query advantages. For coordinate e_j, compare autograd with

$$d_j(h) = \frac{\widehat{\mathcal{L}}_Q(\theta'(\phi + h e_j)) - \widehat{\mathcal{L}}_Q(\theta'(\phi - h e_j))}{2h}. \tag{26}$$

At each perturbation, recompute the differentiable weights, ratios, support update, and query log probabilities while fixing cached stopped quantities. Sweep h to check convergence away from branch boundaries. Record distances to $x_t = 1$, ties in the maximum, and sign changes in the direction test. At a kink, central differences combine two directions and need not match the derivative selected by automatic differentiation.

For $K > 1$ differentiated steps, cache the detached quantities at every unperturbed step and reuse them throughout the check. Recomputing a stopped statistic after perturbing ϕ changes its value without adding its derivative to the automatic-differentiation graph. The two procedures would then evaluate different functions.

Fresh-query repetitions measure sampling variance and meta-update quality. Fixed-batch finite differences instead test derivative correctness. Sampling noise does not explain disagreement when both calculations use identical cached inputs.

A.6 What the effective-horizon diagnostic measures

For an eligible response with $T_i \geq 2$, define

$$M_{i,t}^{\text{out}} = M_{i,t}\mathbf{1}\{\widehat{A}_i(\rho_{i,t}-1) > 0\}, \qquad q_{i,t} = M_{i,t}^{\text{out}}|\widehat{A}_i(\rho_{i,t}-1)|. \tag{27}$$

Equation 10 weights normalized remaining length by this nonnegative mass. The token-valued companion is

$$h_{\text{eff}}^{\text{tok}} = \frac{\sum_{i,t} q_{i,t}(T_i - t)}{\sum_{i,t} q_{i,t}}. \tag{28}$$

Report both statistics as undefined when their denominator is zero. Report this event rate separately instead of replacing undefined values with zero.

The mass $q_{i,t}$ measures masked, advantage-weighted ratio deviation. The policy-gradient contribution instead includes $M_{i,t}\widehat{A}_i\rho_{i,t}\nabla_\theta \log \pi_\theta$. Its norm depends on the score vector and on cancellation across tokens. Thus h_{eff} measures update-location tendency under the stated proxy, not a gradient-norm-weighted horizon or the number of reasoning tokens. The normalized statistic and the token-valued statistic use different length weights. In general, $h_{\text{eff}}^{\text{tok}} \neq h_{\text{eff}}(\overline{T}-1)$.

Concentrating accepted mass at the final token makes the horizon statistic zero regardless of answer correctness. We therefore interpret the statistic alongside reward, termination, accepted mass, and response length. Similar mean KL also leaves update directions and causal mechanisms unresolved.

Token-ratio variance at position t conditions on responses reaching that position. Survivor counts, length strata, and unmasked ratios are needed to interpret late-position changes. Token-ratio variance, full-response importance-weight variance, and policy-gradient variance measure different quantities.

B Experimental protocol

We separate allocator selection, full-policy training, and final evaluation. Table 2 lists the primary settings. Full-policy training uses AdamW (Loshchilov & Hutter, 2019), and allocator updates use Adam (Kingma & Ba, 2015). Token budgets control rollout work, while query counts and total compute account for selection.

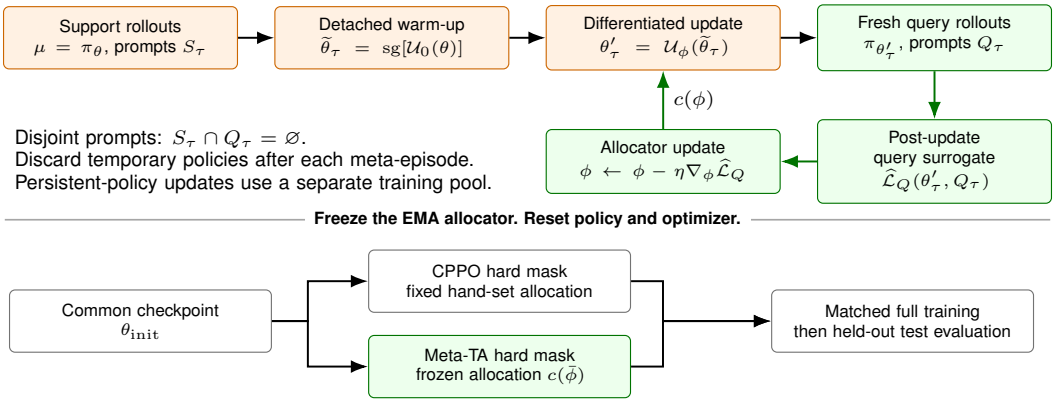

Figure 7: Allocator selection and frozen deployment. Support rollouts produce a detached warm-up, then a differentiated constrained update. Disjoint query prompts evaluate the temporary policy and update only the allocator. Temporary policy states are discarded. Fresh full training resets the policy and optimizer, freezes the allocator, and uses the hard mask.

B.1 Prompt identities and permitted access

The pool specification totals 17,398 prompts: 12,000 for policy training, 2,000 for meta-support, 2,000 for meta-query, and 1,398 for development. Global allocation ignores subject labels. Conditioned analyses use algebra, geometry, combinatorics, and number theory labels available before adaptation.

Table 2: Primary stationary protocol. A response-token budget determines each update batch, so prompt counts vary with response length.

Item	Setting
Policy	Qwen3-8B-Base
Full-training updates / rollout tokens	480 / 120M
Response cap / generations per prompt	8,192 / 8
Support / query prompts per meta-episode	16 / 16
Selection episodes	200
Warm-up / differentiated steps	1 / 1
Base allowance / divergence approximation	$\delta = 0.15$ / Top-20 reduced TV
CPPO default	$(w_{\min}, \delta_b^{\min}) = (0.80, 0.020)$
Frozen global allocation	$(0.689, 0.01329)$
Policy optimizer	AdamW, peak learning rate 10^{-6}
AdamW coefficients / weight decay	$(0.9, 0.95)$ / 0.1
Policy schedule	30-update warm-up, cosine decay
Allocator optimizer	Adam, learning rate 10^{-4}
Allocator EMA / coordinate floors	0.99 / $\epsilon_w = \epsilon_b = 10^{-3}$
Global gradient norm clip	1.0
Arithmetic / higher-order differentiation	bfloat16 model, float32 reductions
Primary final-report seeds	5

Reference-solution token-count quartiles define length strata. Allocator inputs exclude query rewards, test answers, and future-stage performance. Query rewards enter only through outer updates and therefore constitute training feedback.

The query-reuse study has a separate split: 8,192 policy prompts, 2,062 support prompts, 2,048 query prompts, 4,096 held-out reporting prompts, and 1,000 coefficient-selection prompts. These pools total 17,398. The 128-query condition is nested within the 2,048-query pool. The 4,096-prompt reporting pool never selects coefficients.

B.2 Support update and numerical differentiation

Each support prompt produces eight responses and verifier rewards. Policy, support, and query generation use temperature 1 and the full vocabulary. Stored behavior probabilities therefore match the sampling distribution. Support advantages center rewards within each group and divide by the group standard deviation plus 10^{-8}. Equal-reward groups have zero reward advantage. We fix advantages, sampled actions, and lengths within the differentiated update. Warm-up reuses these responses and behavior probabilities without learned allocation.

Recompute token probabilities at the warmed-up parameters. Top-20 reduced TV compares probability masses on the selected token set and one residual bucket. For a fixed set $\mathcal{K}_t$, use

$$D_t^{\mathrm{red}} = \tfrac{1}{2}\left(\sum_{a\in\mathcal{K}_t} |\pi(a \mid s_t) - \mu(a \mid s_t)| + \left|\sum_{a\notin\mathcal{K}_t} (\pi(a \mid s_t) - \mu(a \mid s_t))\right|\right). \tag{29}$$

The selected set follows behavior probabilities and stays fixed during the update. Aggregating all remaining tokens into one bucket makes reduced TV a lower bound on full-vocabulary TV. The schedule proof concerns allocation geometry under the chosen divergence.

The temporary probe uses SGD with $\alpha_0 = \alpha_1 = 10^{-6}$ and no optimizer-state derivative. Warm-up retains the GRPO clipping thresholds and group advantages, without a reference-KL penalty. At 8B, the probe updates rank-16 LoRA (Hu et al., 2021) adapters on attention and MLP projections, with scale 32 and zero dropout. Base weights stay fixed, so θ in the temporary-update equations denotes adapter coordinates. Persistent and frozen policy training update the full model with AdamW.

Stationary selection uses a checkpoint bank built by default hard CPPO on the policy pool. The reference policy advances once per episode with a prespecified allocation. Each meta-episode treats the selected checkpoint as a fixed input. After 200 episodes, the allocator freezes and full training restarts with a fresh policy and optimizer. Continual training instead retains each method's policy history across refreshes.

Query evaluation uses the leave-one-out estimator in Equation 7. Group-standardized query rewards define a different surrogate because their normalization depends on the sampled group. Comparisons between these estimators require identical support episodes and evaluation rules.

B.3 Baselines and allocator selection

GRPO (Shao et al., 2024) specifies symmetric clipping with one ϵ. CPPO (Mao et al., 2026) replaces the acceptance mask under the shared reward and policy-training settings. DAPO (Yu et al., 2025) retains asymmetric clipping, token-level normalization, dynamic sampling, and reward handling. PPO (Schulman et al., 2017) includes a value model and value loss. These objectives have different scales, so the comparison records learning rates and realized divergence separately.

Allocator selection operates in admissible coordinates $(w_{\min}, r_b)$. Development evaluation chooses among allocator EMA snapshots from episodes 160, 170, 180, 190, and 200. Each snapshot receives 20 support-adaptation/development-evaluation episodes on the shared checkpoint bank. These 100 episodes supplement the 200 meta-training episodes and enter selection cost. Final benchmarks never select allocator parameters or policy checkpoints.

The selected allocator alone transfers to fresh full training. Temporary policy states are discarded, and the full training run uses only the policy pool.

PPO initializes an 8B value backbone from the same model family and adds a scalar value head. Its value-loss coefficient is 0.5, both clipping thresholds are 0.2, $\gamma = 1$, and generalized advantage estimation (GAE) (Schulman et al., 2015b) uses decay 0.95. The critic learning rate is 5×10^{-6}. DAPO resamples zero-variance groups and applies its configured terminal penalty to truncated completions. Every method records rejected and resampled tokens. Baseline policy learning rates use the grid $\{5 \times 10^{-7}, 10^{-6}, 2 \times 10^{-6}\}$ under common development access.

KL reporting uses stored behavior-policy states. Under behavior actions and unclipped $\rho = \pi_\theta / \mu$, the nonnegative estimator $\rho - 1 - \log \rho$ has expectation $D_{\mathrm{KL}}(\mu \| \pi_\theta)$. We aggregate over valid tokens in the final reporting window and record behavior-policy age. This quantity differs from divergence to the initial reference model and from reduced-TV gating.

B.4 Continual streams and replay

Each stream contains four stages of 120 persistent updates and 30M response tokens, totaling the same 480 updates and 120M tokens as stationary training. Abrupt drift uses algebra, geometry, combinatorics, and number theory in sequence. Gradual drift starts with algebra, then linearly transfers mixture weight from the previous subject to geometry, combinatorics, and number theory within successive stages. Recurrence uses algebra, geometry, combinatorics, and algebra again. The recurring stream has three unique evaluation tasks, not four.

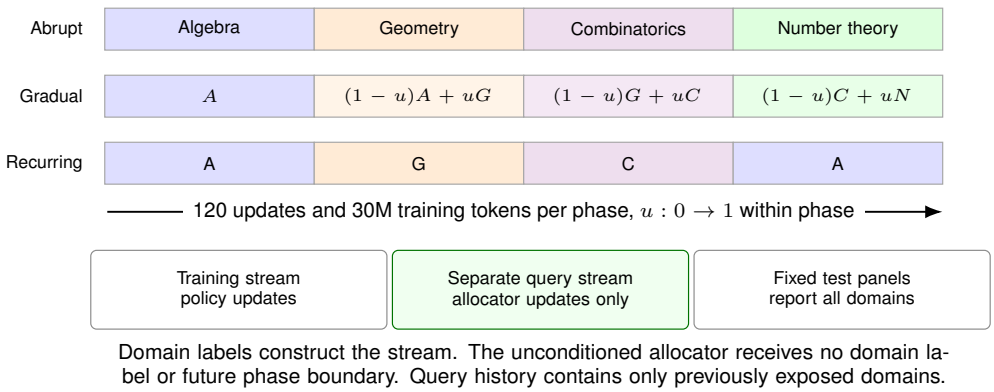

Figure 8: Training distributions and permitted feedback in the continual protocols. Abrupt and gradual streams expose subjects in the same order but differ in transition speed. The recurring stream revisits the first subject. The evaluator scores all domains after each stage, but future-domain evaluation never feeds the policy, allocator, or model-selection procedure.

The policy and optimizer persist across stages. Replay either occupies 0% or 10% of the fixed policy-token budget. The replay buffer holds up to 1,024 unique prompts with subject-balanced reservoir sampling from past policy-training pools. Answers do not cross from held-out pools into replay. Every compared method uses the same buffer membership for a paired seed. Replay changes training-data composition, not the total deployment-token budget.

Pre-stream selection uses 200 episodes. Online Meta-TA refreshes at the three later stage boundaries, with 20 episodes per boundary. Half of each refresh query batch comes from the current query pool and half from previously exposed pools, balanced across seen subjects. First-stage selection uses only the current subject. Frozen methods keep their pre-stream allocation. Total online cost includes all 60 refresh episodes.

Evaluation matrices contain $A_{j,k}$ for every checkpoint j and domain k, including sealed future-domain evaluations. Let s_k be the first stage exposing task k and let $\mathcal{P} = \{k : s_k < J\}$, where $J = 4$. The reported metrics are

$$\text{AvgAcc} = |\mathcal{K}|^{-1} \sum_{k \in \mathcal{K}} A_{J,k}, \tag{30}$$

$$\text{F} = |\mathcal{P}|^{-1} \sum_{k \in \mathcal{P}} \left(\max_{s_k \le j < J} A_{j,k} - A_{J,k} \right), \tag{31}$$

$$\text{BWT} = |\mathcal{P}|^{-1} \sum_{k \in \mathcal{P}} (A_{J,k} - A_{s_k,k}). \tag{32}$$

Average accuracy uses unique domains. Forgetting and backward transfer use the same previously introduced domains. We do not clip negative forgetting: improvement beyond a previous maximum gives a negative term. If a domain is revisited, the first acquisition stage remains the BWT reference. High positive BWT is favorable. Lower forgetting is favorable, subject to final accuracy.

B.5 Allocator capacity and regularization

The global linear allocator has two raw parameters. The structured controller emits a position floor, normalized prefix budget, three mixture logits, and three bounded exponents. Its features comprise available subject identity, support reward and divergence, reference-length stratum, and training progress. Lookup-table and two-layer controllers compare capacity under identical features and query access.

The mixture sweep uses $\lambda_a \in \{0, 10^{-3}, 10^{-2}, 10^{-1}\}$. A separate λ_c sweep tests the physical-parameter prior. The reserved selection pool chooses coefficients before the 4,096-prompt held-out evaluation. Full coefficient curves retain both beneficial and underfitting settings. Global linear allocation has no mixture penalty.

A policy-entropy control adds entropy to the policy update under the same coefficient-selection access. The target of weight decay is recorded separately for policy and controller parameters. These penalties act on different objects from allocator-mixture entropy. The generalization gap is the query-minus-held-out reward change in percentage points.

B.6 Final evaluation and reporting repetitions

Repeated final-checkpoint evaluation. For each training seed s, we evaluate the terminal checkpoint using $R_{\text{eval}} = 5$ independent reporting repetitions. All repetitions use the same benchmark prompts, verifier, generation hyperparameters, and terminal checkpoint, but use independent generation randomness. For benchmark B, let $Z_{s,r}^{(B)}$ denote the score from reporting repetition $1 \le r \le 5$. We report the within-seed estimate $\bar{Z}_s^{(B)} = \frac{1}{R_{\text{eval}}} \sum_{r=1}^{R_{\text{eval}}} Z_{s,r}^{(B)}$. The benchmark mean and confidence interval in the main tables are computed across the five independent training-seed estimates; evaluation repetitions are not treated as independent training seeds.

Each AIME reporting repetition independently samples 16 responses for each problem. If $c_{s,r,i,j} \in \{0, 1\}$ denotes correctness of response j to problem i, the repetition-level yearly score is

$$Z_{s,r}^{(\text{AIME}_y)} = 100 \, \frac{1}{N_y \, 16} \sum_{i=1}^{N_y} \sum_{j=1}^{16} c_{s,r,i,j}, \quad N_y = 30.$$

The per-seed yearly score averages these values over $R_{\text{eval}} = 5$ independent reporting repetitions. The AIME value reported in the main table then averages the three yearly per-seed scores with equal

weight. For MATH-500 and OlympiadBench, each reporting repetition independently samples one response for every benchmark problem. For a benchmark with N_B problems,

$$Z_{s,r}^{(B)} = 100 \, \frac{1}{N_B} \sum_{i=1}^{N_B} c_{s,r,i}.$$

We report the arithmetic mean of these repetition-level pass@1 scores within each training seed. This is a repeated estimate of pass@1: there is still exactly one sampled response per problem in each reporting repetition; we do not use best-of-R_{eval}, majority vote, or pass@R_{eval}.

C Additional results and controls

The additional analyses distinguish full-policy training, derivative correctness, allocator selection, and update diagnostics. Model scale is explicit throughout: the Qwen3-1.7B studies examine selection behavior, while the main comparison evaluates Qwen3-8B.

C.1 Full training trajectories and seed-level scores

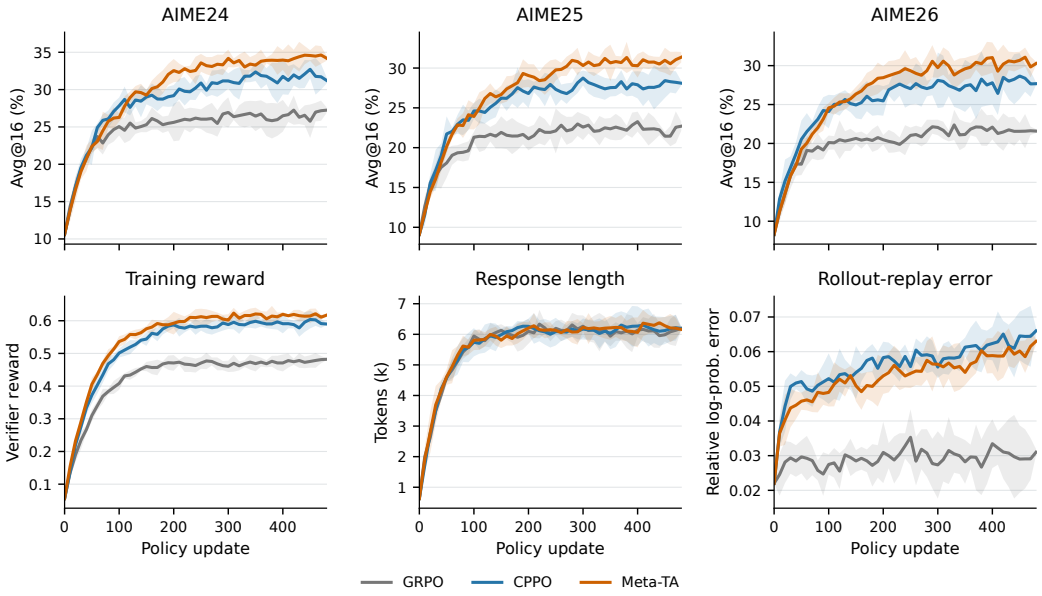

Figure 9: Complete Qwen3-8B-Base training trajectory. Top: AIME24, AIME25, and AIME26 Avg@16. Bottom: verifier reward, mean response length in thousands of tokens, and relative rollout-replay log-probability error. Each line aggregates five seed trajectories without smoothing. Response length and replay error describe different effects from final accuracy.

Training reward rises before AIME accuracy stabilizes, and the three yearly AIME curves fluctuate separately. We therefore use the terminal checkpoint rather than select a peak from these curves. Rollout-replay log-probability error measures consistency between generation and training computations. Accuracy and KL address different questions.

The main table recomputes means and Student intervals from these rounded aggregates. Paired comparisons match the listed seeds.

Table 3: Per-seed Qwen3-8B-Base AIME results. Yearly columns report Avg@16. The mean column reports the arithmetic mean of the three displayed yearly scores, rounded to two decimal places. Each run receives 120M rollout tokens.

Method	Seed	AIME24	AIME25	AIME26	Mean
GRPO	1	27.08	22.75	21.58	23.80
GRPO	2	26.92	23.13	21.33	23.79
GRPO	3	27.67	22.00	21.67	23.78
GRPO	4	27.00	22.46	21.50	23.65
GRPO	5	27.50	23.13	21.88	24.17
CPPO	1	30.96	27.96	27.88	28.93
CPPO	2	31.17	28.25	27.67	29.03
CPPO	3	31.67	27.79	27.71	29.06
CPPO	4	30.83	28.17	27.71	28.90
CPPO	5	31.58	28.33	27.46	29.12
Meta-TA	1	34.00	31.92	30.42	32.11
Meta-TA	2	34.04	31.25	30.29	31.86
Meta-TA	3	34.63	30.67	30.00	31.77
Meta-TA	4	33.96	31.13	30.46	31.85
Meta-TA	5	34.42	31.83	30.21	32.15

Table 4: Per-seed Qwen3-8B-Base held-out results on MATH-500 and OlympiadBench. Both benchmarks report pass@1. Seed IDs match the AIME table.

Method	Seed	MATH-500	OlympiadBench
GRPO	1	72.92	47.16
GRPO	2	73.20	45.96
GRPO	3	72.76	46.20
GRPO	4	73.40	46.52
GRPO	5	73.04	46.96
CPPO	1	74.04	48.32
CPPO	2	74.08	47.76
CPPO	3	74.36	47.96
CPPO	4	74.00	48.24
CPPO	5	74.68	48.20
Meta-TA	1	75.36	49.52
Meta-TA	2	75.48	48.84
Meta-TA	3	75.08	48.76
Meta-TA	4	75.40	49.32
Meta-TA	5	75.48	49.08

C.2 Local numerical correctness and empirical gradient diagnostics

The categorical check compares forward-mode automatic differentiation with central finite differences on the same cached surrogate. Behavior matching gives zero allocator gradient. A detached warm-up activates the gates and gives a nonzero derivative. Tokens, rewards, divergence measurements, and branch decisions stay fixed, isolating numerical correctness of this finite computation.

Table 5: Hypergradient checks on 180 paired episodes.

Phase	Pearson correlation	Sign agreement (%)
Pooled	0.85	80.6
Early	0.67	76.7
Middle	0.91	83.3
Late	0.90	81.7

Warm-up (K_0)	Active outward tokens	Gradient norm	Query change
0	0.26%	(1.5\times10^{-6})	-0.02 pp
1	7.06%	0.0138	+0.17 pp
2	10.58%	0.0186	+0.29 pp

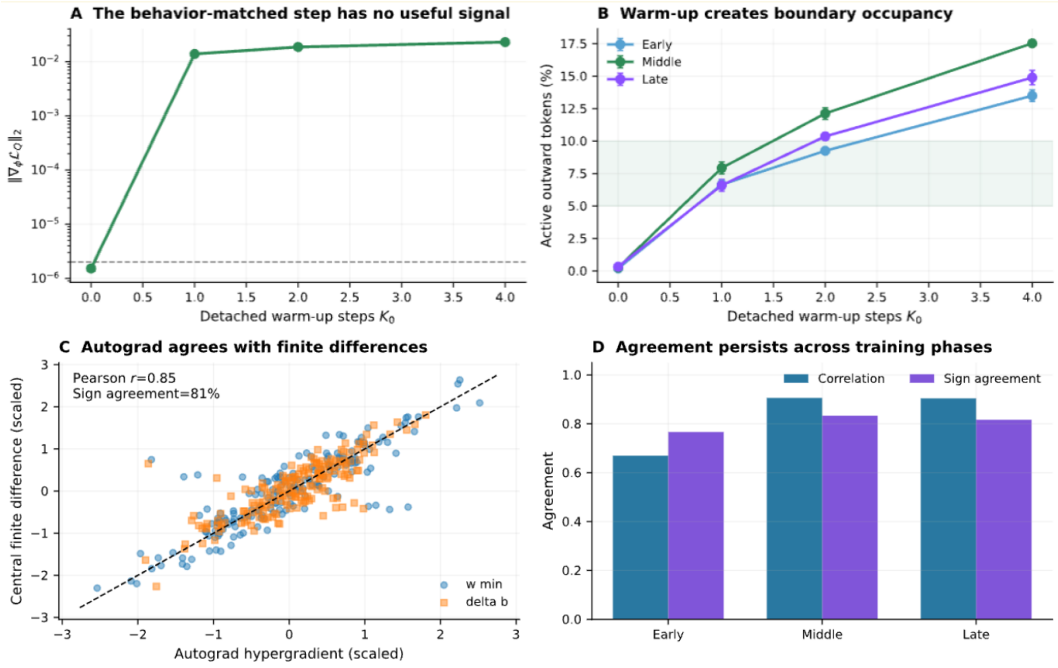

Figure 10: Qwen3-1.7B-Base gradient diagnostics. Panels compare warm-up count, active outward constraints, and finite-difference agreement across checkpoint phases.

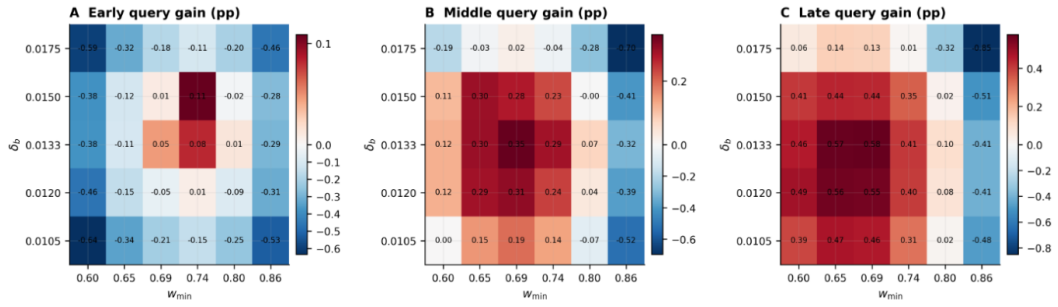

Figure 11: Post-update query-reward surfaces over the position floor and prefix increment at early, middle, and late Qwen3-1.7B-Base checkpoints. Each cell reports a change in percentage points. The best displayed region varies by phase. The three panels measure reward changes and reveal phase-dependent preferences.

Table 6: Allocator selection on Qwen3-1.7B-Base with 200 query evaluations. ΔJ_Q uses held-out query prompts. Frozen Δ uses a fresh development retraining run and reports change relative to published CPPO.

Selector	Query evaluations	Held-out ΔJ_Q	Frozen Δ
Meta-gradient	200	0.81 ± 0.01	0.36 ± 0.01

C.3 Response surfaces and allocator selection

The response surface is shallow, with a preferred region near $(0.69, 0.0133)$ whose location varies by training phase.

C.4 Bilevel, masking, and capacity controls

Table 7: Frozen Qwen3-8B-Base ablations relative to CPPO. The selector budget is 200 query episodes. Entries give AIME percentage-point differences and confidence intervals across five matched seeds. The no-warm-up condition tests gate inactivity.

Variant	AIME gain	Realized KL
Full Meta-TA	2.94 ± 0.23	0.01478
Position only	2.42 ± 0.29	0.01470
Prefix only	2.30 ± 0.38	0.01474
No warm-up	0.00 ± 0.00	0.01514
Same prompt reuse	2.11 ± 0.36	0.01482
No divergence detach	2.21 ± 0.49	0.01477
Shuffled positions	2.44 ± 0.47	0.01483
Local budget only	2.36 ± 0.38	0.01477
Soft deployment	2.85 ± 0.55	0.01482
Two differentiated steps	2.99 ± 0.62	0.01489

Table 8: Qwen3-1.7B-Base allocator and bilevel-design ablations. Query and frozen changes report percentage points relative to published CPPO.

Variant	Query ΔJ	Frozen Δ	KL	Gate (%)
Position only	0.76	0.20	0.01505	11.5%
Prefix only	0.63	0.16	0.01478	10.6%
Joint global	1.08	0.36	0.01520	12.5%
No warm-up	0.01	0.00	0.01445	0.1%
Two warm-ups	0.29	0.39	0.01524	12.1%
Non-detached divergence	1.16	0.09	0.01625	13.8%
Hard frozen deployment	1.08	0.36	0.01520	12.5%
Soft frozen deployment	1.08	0.28	0.01532	12.5%
Same-prompt leakage control	1.62	0.06	0.01518	12.8%

We discover that reusing support prompts improves the online score but weakens frozen transfer, supporting disjoint evaluation within the episode. Removing divergence detachment also weakens transfer while changing the update being differentiated. Specifically, hard and soft deployment differ by 0.08 points in the smaller-model study. Hard deployment preserves comparability with CPPO without relying on a large hard-versus-soft advantage.

We notice that batch size, rollout count, inner depth, and parameter subspace change both adaptation and cost. Additionally, we apply two differentiated steps to deepen the graph. We use LoRA (Hu et al., 2021) to restrict the probe to adapter coordinates, so its derivative check has a narrower scope than full-policy differentiation. Within-response shuffling preserves the weight distribution while removing its positional order.

C.5 Structured allocation

The nonlinear family changes schedule shape while preserving the per-update conditions. Curves close to linear indicate little additional headroom in this study. Subject-specific endpoints characterize fitted allocations, while held-out-family transfer requires evaluation on subjects absent from selection. Scalable Option Learning (SOL) (Henaff et al., 2025) instead learns temporally extended control. Meta-TA's update-location diagnostic measures a different object.

Table 9: Task-specific terminal weights and prefix-budget increments from the Qwen3-1.7B-Base study.

Task	$w_{\min}$	δ_b
Algebra	0.646	0.01415
Combinatorics	0.738	0.01235
Geometry	0.697	0.01341

Table 10: Cap-controlled effective horizon, averaged over five matched seeds. The normalized statistic weights remaining response fractions. The token statistic weights remaining token counts. Mean length measures generated response length.

Cap	Method	h_{eff}	$h_{\mathrm{eff}}^{\mathrm{tok}}$	Mean length
2k	GRPO	0.507	915	1,803
2k	CPPO	0.469	851	1,815
2k	Meta-TA	0.457	820	1,796
4k	GRPO	0.522	1,828	3,501
4k	CPPO	0.464	1,621	3,489
4k	Meta-TA	0.445	1,546	3,471
8k	GRPO	0.539	3,340	6,193
8k	CPPO	0.473	2,928	6,194
8k	Meta-TA	0.440	2,711	6,161
16k	GRPO	0.557	5,370	9,637
16k	CPPO	0.475	4,547	9,566
16k	Meta-TA	0.430	4,102	9,530

Table 11: Importance-ratio variance and realized KL in the cap-controlled runs. Late variance uses the final response quartile. Entries average five matched seeds.

Cap	Method	Ratio variance	Late variance	KL
2k	GRPO	0.0260	0.0435	0.01645
2k	CPPO	0.0229	0.0337	0.01550
2k	Meta-TA	0.0221	0.0304	0.01507
4k	GRPO	0.0387	0.0677	0.01679
4k	CPPO	0.0299	0.0451	0.01568
4k	Meta-TA	0.0274	0.0391	0.01524
8k	GRPO	0.0555	0.0933	0.01687
8k	CPPO	0.0416	0.0596	0.01580
8k	Meta-TA	0.0365	0.0501	0.01569
16k	GRPO	0.0798	0.1326	0.01724
16k	CPPO	0.0534	0.0779	0.01599
16k	Meta-TA	0.0475	0.0682	0.01597

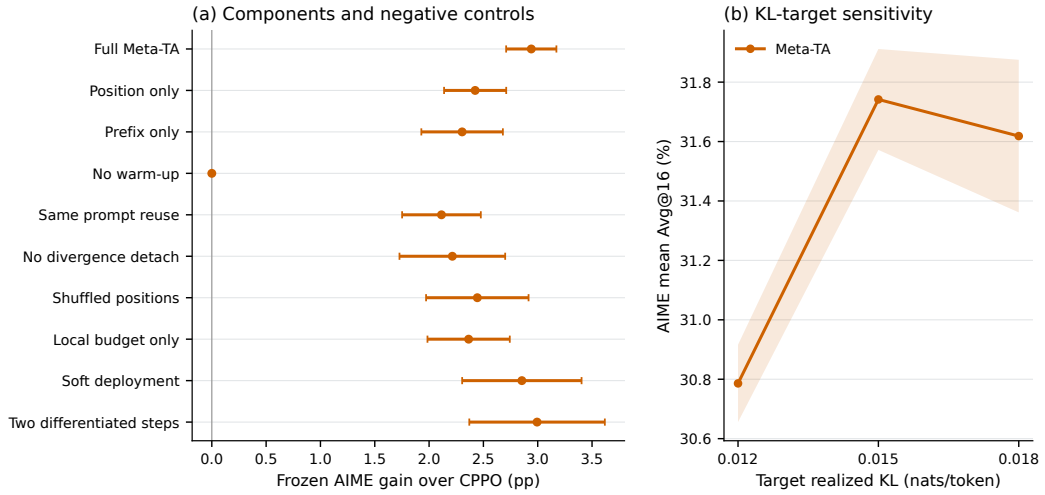

Figure 12: Ablations of frozen performance. Left: AIME gains over default CPPO after changing one allocator-learning component. Without warm-up, the allocator retains its default CPPO initialization. Right: Meta-TA accuracy under three KL targets. Error bars summarize five training seeds. The right panel measures sensitivity within Meta-TA and does not compare equal-KL baselines.

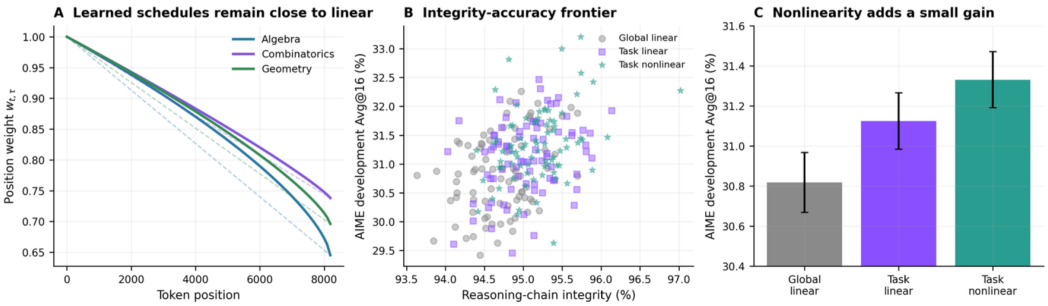

Figure 13: Qwen3-1.7B-Base structured allocation. Left: subject-conditioned weight schedules. Middle: reasoning-chain integrity and development accuracy. Right: global linear, task-conditioned linear, and task-conditioned nonlinear development scores.

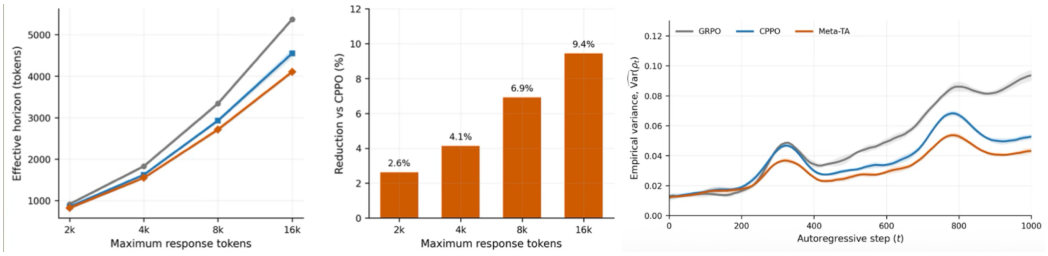

Figure 14: Cap and position diagnostics. Left: token-valued effective horizon and normalized-horizon reduction percentages. Right: unclipped importance-ratio variance across absolute positions.

C.6 Horizon and variance summaries

The cap table reports similar mean response lengths for CPPO and Meta-TA alongside later accepted update mass.

C.7 Continual, overfitting, and compute comparisons

Table 12: Continual mathematical reasoning with 10% replay and a fixed 120M deployment-token budget. Final accuracy averages all unique task test sets. Forgetting averages the maximum post-acquisition, pre-final score minus the final score. Negative forgetting records improvement beyond the earlier maximum. Backward transfer (BWT) averages the final score minus the first-acquisition score. Both retention metrics use tasks acquired before the final stage. Accuracy entries give Student-t 95% intervals across five seeds. All metrics use percentage points.

Stream	Allocator	Final accuracy	Forgetting	BWT
Abrupt	CPPO	55.67 ± 0.92	2.46	-2.46
Abrupt	Meta-TA frozen	57.35 ± 0.23	2.45	-2.45
Abrupt	Meta-TA online	57.46 ± 0.72	1.89	-1.89
Abrupt	Meta-TA online + prior	58.20 ± 0.51	1.80	-1.80
Gradual	CPPO	56.46 ± 0.87	1.69	-1.69
Gradual	Meta-TA frozen	57.92 ± 0.66	1.62	-1.62
Gradual	Meta-TA online	59.22 ± 0.51	0.96	-0.96
Gradual	Meta-TA online + prior	59.14 ± 0.91	1.00	-0.98
Recurring	CPPO	55.87 ± 0.45	0.97	-0.97
Recurring	Meta-TA frozen	58.28 ± 0.53	0.88	-0.88
Recurring	Meta-TA online	59.00 ± 0.72	0.63	-0.63
Recurring	Meta-TA online + prior	59.18 ± 0.64	0.65	-0.65

Table 13: End-to-end compute cost. Columns give normalized forward-equivalent costs relative to one default CPPO deployment run on the same hardware configuration. Selection includes 200 allocator episodes, 100 finalist-evaluation episodes, and checkpoint-bank construction. Online refresh adds three boundary updates. Peak memory gives the fraction of accelerator memory in use. Values average five seeds.

Method	Training	Selection	Refresh	Total	Peak memory
GRPO	0.861	0.000	0.000	0.861	61.7%
PPO	1.209	0.000	0.000	1.209	91.4%
DAPO	0.955	0.000	0.000	0.955	68.0%
CPPO	1.000	0.000	0.000	1.000	72.0%
Meta-TA	1.037	3.291	0.000	4.328	88.8%
Meta-TA online	1.011	3.291	0.663	4.965	89.3%
Meta-TA online + prior	1.031	3.291	0.663	4.985	89.1%

Table 14: Allocator overfitting on a disjoint 4,096-prompt development set. A three-component nonlinear allocator receives a prior penalty $\lambda_a D_{\mathrm{KL}}(a \| \mathrm{Unif}(3))$. The 128-prompt query pool is nested within the 2,048-prompt pool. Reward gains use percentage points relative to prespecified allocation. Intervals are Student-t 95% intervals across five seeds. Allocator entropy, in nats, concerns schedule-mixture weights rather than policy-token entropy.

Query pool	λ_a	Reused-query gain	Held-out gain	Entropy
128	0	1.99 ± 0.08	0.05 ± 0.13	0.29
128	0.001	1.59 ± 0.16	0.14 ± 0.17	0.47
128	0.01	1.07 ± 0.14	0.54 ± 0.18	0.75
128	0.1	0.24 ± 0.06	0.21 ± 0.19	1.03
2048	0	1.02 ± 0.16	0.72 ± 0.23	0.29
2048	0.001	0.98 ± 0.15	0.77 ± 0.16	0.47
2048	0.01	0.83 ± 0.21	0.77 ± 0.17	0.73
2048	0.1	0.30 ± 0.17	0.21 ± 0.09	1.03

Table 15: Allocator capacity and regularization controls. Query and held-out gains score the temporary update on separate prompt pools. The global, lookup, and two-layer allocators test capacity. Coefficient-prior, mixture-prior, and policy-token entropy rows isolate distinct penalties. Entries give means and Student-t 95% intervals across five seeds.

Pool	Variant	Query gain	Held-out gain
128	Global two-parameter	0.79 ± 0.27	0.39 ± 0.12
2048	Global two-parameter	0.99 ± 0.16	0.77 ± 0.26
128	Conditioned lookup	1.70 ± 0.14	0.10 ± 0.11
2048	Conditioned lookup	1.13 ± 0.06	0.78 ± 0.15
128	Conditioned two-layer	2.56 ± 0.10	-0.16 ± 0.11
2048	Conditioned two-layer	1.37 ± 0.04	0.79 ± 0.14
128	Nonlinear unregularized	1.99 ± 0.08	0.05 ± 0.13
2048	Nonlinear unregularized	1.02 ± 0.16	0.72 ± 0.23
128	Nonlinear mixture prior 0.01	1.07 ± 0.14	0.54 ± 0.18
2048	Nonlinear mixture prior 0.01	0.83 ± 0.21	0.77 ± 0.17
128	Coefficient prior only 0.01	1.25 ± 0.11	0.39 ± 0.20
2048	Coefficient prior only 0.01	0.89 ± 0.11	0.76 ± 0.13
128	Policy entropy only 0.001	1.93 ± 0.10	0.09 ± 0.15
2048	Policy entropy only 0.001	1.05 ± 0.02	0.75 ± 0.14
128	Nonlinear mixture prior 0.1	0.24 ± 0.06	0.21 ± 0.19
2048	Nonlinear mixture prior 0.1	0.30 ± 0.17	0.21 ± 0.09

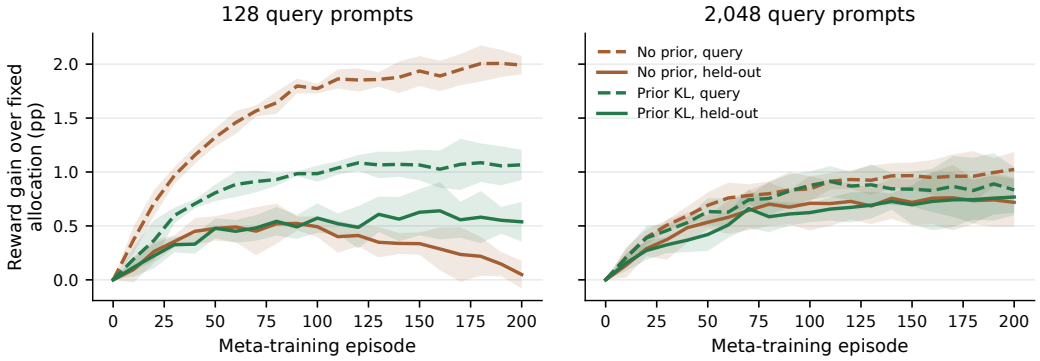

Figure 15: Repeated query optimization and held-out generalization. The query pool supplies repeated allocator updates. The disjoint study-development pool supplies reporting only after coefficient selection on its separate pool. The gap between the two curves measures reuse-related overfitting. Narrowing this gap is useful only when held-out reward or frozen policy performance also improves.

C.8 WORKED REASONING AND GATE EXAMPLE

Figure 16 places a reasoning dependency alongside an exact gate calculation to explain how local and prefix constraints interact.

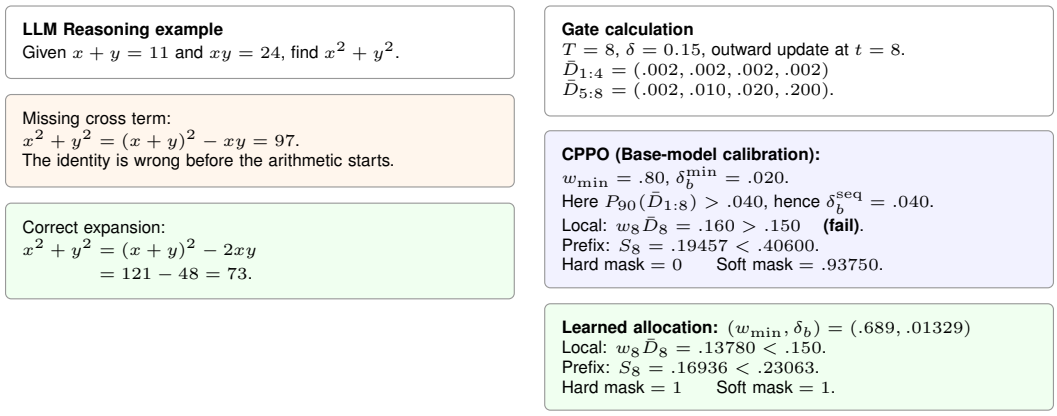

Figure 16: Local and prefix constraints in a reasoning example. An incorrect early transformation changes the subsequent reasoning. The adjacent gate calculation compares the fixed or prespecified vs. the learned allocations using the stated token divergences.

C.9 COMPARISON WITH ADAPTIVE RL POST-TRAINING METHODS

Table 16 compares recent RL post-training methods by what they adapt and which signal drives the adaptation. Apart from DPPO, which keeps a fixed threshold, each adjusts a different part of training with a rule or dual solution computed from the current rollouts. Meta-TA instead learns the position floor and prefix budget from the reward of a temporarily updated policy on prompts disjoint from its support set, then freezes them and evaluates their transfer to fresh policy training.

Table 16: Update control in recent LLM RL post-training methods. *Adapted* lists the quantity a method adjusts from training-time signals (–: none; the policy is still updated), and *Signal* lists what drives that adjustment. *Pos.* and *Prefix* refer to divergence bounds that depend on token position or on divergence accumulated along the response prefix (×: no such bound). TRAPO's expert prefixes provide supervision rather than a divergence bound. The Meta-TA row describes the global allocator; online Meta-TA also refreshes ϕ at declared checkpoints.

Method	Control	Adapted	Signal	Pos.	Prefix
DAPO	ratio clip, fixed $\varepsilon_{\text{low}}, \varepsilon_{\text{high}}$	prompt batch (dynamic sampling)	group accuracy of current rollouts	×	×
TRAPO	Dr.GRPO on completions; SFT floor α on expert prefixes	expert-prefix length per prompt	current return vs. preset thresholds	×	×
QUATRO	sequence-level KL radius δ	per-query dual variables	current rewards of each query	×	×
DPPO	token TV/KL threshold δ	–	–	×	×
CPPO	token TV; local and prefix bounds	per-sequence δ_b (Base models)	clamped P_{90} of the sequence's D_t	fixed $w_{\min}$	✓
Meta-TA	CPPO bounds; soft mask in selection, hard mask in training	$(w_{\min}, \delta_b)$ during selection, then frozen	post-update reward on disjoint query prompts	learned $w_{\min}$	learned δ_b