LEARNING TO ASK FOR HELP: BUDGET-CONSTRAINED DECISION ROUTING VIA DUAL-ADVANTAGE GRPO

Anonymous authors
Paper under double-blind review

ABSTRACT

Small language models (SLMs) on devices answer queries locally and request cloud assistance when needed. Each cloud request consumes resources, motivating a prescribed allowance for collaboration. Existing collaborative post-training methods use a fixed assistance reward, but the resulting cloud usage can drift as local competence changes. DA-GRPO treats expected cloud usage as a constraint: task reward and cloud cost yield two group-relative advantages, combined by a multiplier updated from measured usage relative to the target. The method retains the clipped GRPO surrogate and adds no measurable training overhead in our configuration. On Qwen2.5-1.5B-Instruct and Llama-3.2-3B-Instruct across mathematical reasoning, code generation, and knowledge benchmarks, DA-GRPO achieves the highest mean joint accuracy in all eight settings under common task-specific training budget targets, improving on the strongest fixed-reward baseline by 1.8 to 7.0 percentage points. In a separate budget-control experiment, mean usage is 0.313 at the end of training and 0.314 at in-distribution test time against a target of 0.30. Cloud usage increases from 0.10 on the easiest MATH level to 0.49 on the hardest. The same framework supports changing targets during continued training and cloud-token costs, while test-time usage under distribution shift can exceed the training target.

1 INTRODUCTION

Language model services increasingly run where the user is, on phones, in vehicles, and at base stations, where memory, energy, and connectivity limit them to small language models (SLMs) (Zhu et al., 2024; Zhang et al., 2024; Liu et al., 2024; Xu et al., 2024b; Qu et al., 2025; Zhao et al., 2024; Lu et al., 2025). An SLM handles routine queries well and fails on harder reasoning, coding, and knowledge questions. A cloud LLM handles those, but each call costs latency, bandwidth, and money. Local-cloud collaboration answers locally by default and forwards a query only when needed (Xu et al., 2024a), as in Figure 1. Because every forwarded query consumes backend capacity and user budget, a deployment specifies how much help it can afford, as a fraction of queries or a token allowance, and seeks to maximize answer quality within that allowance (Belcak et al., 2025; Magister et al., 2023). In many deployments this share is fixed before accuracy is considered, by a metered API contract, a link budget, or a policy on what may leave the device. We call the fraction of queries forwarded to the cloud the collaboration ratio and the allowed value the budget τ.

Who decides when to ask? A router outside the model can decide, using confidence rules or a classifier over frozen models (Ong et al., 2024; Ding et al., 2024; Chen et al., 2023). Alternatively, the model itself can decide, if post-training adds a help request to its action space and rewards it appropriately (Fang et al., 2026). The second option is attractive. The SLM learns which queries it can and cannot handle while it learns to handle them, and no second model is needed at inference. Its difficulty is how to express the budget in the training signal. Existing methods add a fixed assistance term for cloud use to the task reward and tune it so that the training-time collaboration ratio lands near the target.

A fixed assistance term is a price, not a budget. How much help a policy buys at a given price depends on how much it can do on its own, and that changes throughout post-training. Figure 2 shows the consequence on a fixed task with a fixed target. Under three policy optimizers, each with an assistance term tuned for the target, the realized ratio rises above 0.5 in the first 50 iterations and then trends

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.

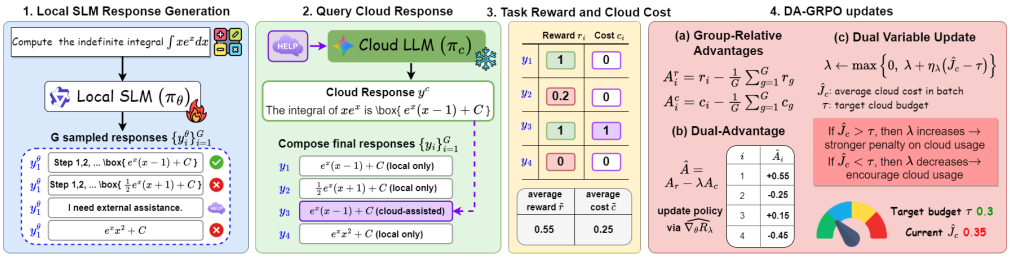


Figure 1: DA-GRPO. For each prompt the local SLM samples a group of responses, and responses that end in a help request are completed by the cloud LLM once per prompt. Task reward and cloud cost give two group-relative advantages that are combined by the dual variable λ. The policy takes a GRPO step, and λ moves with the measured collaboration ratio.

downward for the remaining 550, ending at a value that depends on the optimizer. The usual remedy is to retune the term for every task, model, and budget. In our experiments the value that lands the target ranges from 0.05 to 0.4 across tasks (Table 7), and retuning does not stop the ratio from moving within a run.

The budget should set the price. Maximizing task reward subject to a collaboration budget is a constrained policy optimization problem. Its multiplier rises when help is overused and falls when help is underused. DA-GRPO builds this feedback into group relative policy optimization (GRPO). For each prompt, the local model samples a group of responses; task reward and cloud cost each yield a group-relative advantage, which are combined by the multiplier before the usual GRPO update. A projected dual update then uses the batch collaboration ratio to adjust the price. Centering the cost within a group makes the direct cost signal zero when all sampled responses make the same collaboration decision and nonzero when they disagree. Across our tasks and models, the same dual learning rate supports approximate target tracking without a per-task search over a fixed assistance reward; changing the target during continued training changes the operating point. Our contributions are as follows.

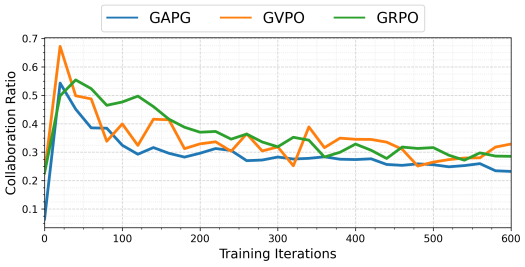


Figure 2: A fixed assistance term does not hold the budget. On a fixed task with a fixed target ($\tau = 0.3$), three policy optimizers trained with a fixed assistance term all push the collaboration ratio above 0.5 within the first 50 iterations, after which it trends downward for the remaining 550 and ends between 0.23 and 0.33, depending on the optimizer.

- We identify drift in cloud usage under fixed assistance rewards during collaborative post-training and explain how changing local competence can move the operating point even when the desired budget remains fixed (Section 3).
- We propose DA-GRPO, which combines task and cost group-relative advantages with a multiplier updated from measured usage. We establish the unbiasedness of the underlying unclipped, unnormalized score-function estimator and characterize interior stationary points of the mean dual dynamics. The practical update retains the clipped GRPO surrogate (Section 4).
- On two SLMs and four benchmarks, DA-GRPO achieves the highest mean joint accuracy in all eight settings under common training budget targets. A fixed-set control supports a contribution from improved allocation, while budget-control experiments examine in-distribution usage, difficulty-dependent decisions, changing targets, and token costs, with no measurable training overhead over GRPO (Section 5).

2 Related Work

Routing between models. Cascades and routers select a model per query with a rule or a learned selector over frozen models (Chen et al., 2023; Ding et al., 2024; Ong et al., 2024). The selector

carries the budget and the local model is trained as if it had to answer every query, so the decision and the model's competence are optimized separately. DA-GRPO trains the local model to make the decision itself and needs no selector at inference.

Post-training with a help action. Fang et al. (2026) introduce a help request into the local model's action space. Their GAPG method combines a group-level policy-gradient estimator, a hierarchical reward with a fixed assistance term, and adaptive prompt filtering to preserve learning signals for both local reasoning and assistance. DA-GRPO instead updates the assistance price from the discrepancy between realized usage and a specified target. GRPO and related critic-free post-training methods use group-relative learning signals without a learned value function (Guo et al., 2025; Liu et al., 2025; Zhang et al., 2025). We retain this machinery and incorporate resource cost into the advantage.

Constrained policy optimization. Constrained RL uses Lagrange multipliers to trade task reward against resource constraints (Achiam et al., 2017; Tessler et al., 2019; Bhatia et al., 2019; Wang et al., 2023). For concise reasoning, PALU minimizes a length budget subject to a performance constraint and implements a regime-based controller with quantile-driven length updates (Gao et al., 2025). DA-GRPO applies primal-dual feedback to a cloud-help action generated by the local model. Section 5.3 compares the advantage-level implementation with a separately clipped objective-level dual and a length-control-inspired baseline; their adaptation to cloud cost is described in Appendix E.1.

3 Budgeted Local-Cloud Post-Training

Setup. A local policy π_θ with parameters $\boldsymbol{\theta}$ runs on the device, and a fixed cloud model π_c is available on request. Given a prompt $\mathbf{x}$, the local model generates $\mathbf{y}^\theta$. If $\mathbf{y}^\theta$ ends with the help request, the cloud produces $\mathbf{y}^c$ and the final response is $\mathcal{C}(\mathbf{y}^\theta, \mathbf{y}^c)$, which replaces the request with the cloud completion; otherwise the response is $\mathbf{y}^\theta$. We write $p_{\text{coll}}(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})$ for the induced response distribution. A task reward $r(\mathbf{x}, \mathbf{y})$ scores correctness and format, and a cost $c(\mathbf{x}, \mathbf{y}) \in \{0, 1\}$ records whether the cloud was used (Appendix G). With $\mathcal{D}$ the training distribution, the expected reward and the expected collaboration ratio are

$$J_r(\boldsymbol{\theta}) := \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\, \mathbb{E}_{\mathbf{y}\sim p_{\text{coll}}(\cdot|\mathbf{x};\boldsymbol{\theta})}\big[r(\mathbf{x}, \mathbf{y})\big], \qquad J_c(\boldsymbol{\theta}) := \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\, \mathbb{E}_{\mathbf{y}\sim p_{\text{coll}}(\cdot|\mathbf{x};\boldsymbol{\theta})}\big[c(\mathbf{x}, \mathbf{y})\big].$$

Only $\boldsymbol{\theta}$ is trained. The cloud is a shared service that the device neither controls nor updates. A deployment specifies the budget τ, and the post-training objective is

$$\max_{\boldsymbol{\theta}} \; J_r(\boldsymbol{\theta}) \quad \text{s.t.} \quad J_c(\boldsymbol{\theta}) \le \tau. \tag{1}$$

Why a fixed assistance term drifts. Existing collaborative post-training replaces (1) by an unconstrained hierarchical reward in which a cloud-assisted correct answer earns a fixed assistance term in place of the full correctness reward (Fang et al., 2026). In a simplified decision between local execution and assistance, the fixed term sets a threshold on the expected benefit of asking, which also depends on cloud success and format compliance. As post-training raises local success on more prompts, prompts cross that threshold and the collaboration ratio falls, even though τ has not changed. Figure 2 shows this on a fixed task: the ratio overshoots early, when the local model is weak, and then trends downward for the rest of training. Holding the ratio at τ requires the assistance term to track the model's changing competence. Current practice does this by hand, once per task, model, and budget; in our experiments the assistance term, paid whenever the cloud returns a correct answer, is 0.2 for mathematics, 0.05 for code, and 0.4 for knowledge questions (Table 7). A constant chosen by search cannot follow competence within a run.

What the update needs. Problem (1) has the Lagrangian $\mathcal{L}(\boldsymbol{\theta}, \lambda) = J_r(\boldsymbol{\theta}) - \lambda\,(J_c(\boldsymbol{\theta}) - \tau)$ with multiplier $\lambda \ge 0$, and primal-dual methods ascend in $\boldsymbol{\theta}$ and descend in λ under this sign convention. Two features of LLM post-training shape how this can be done. GRPO-style updates are critic-free: the learning signal is a group-relative advantage, centered per prompt, so a group-centered cost advantage provides a direct way to incorporate a constraint without introducing a separate critic. And the cost is a property of a discrete decision inside generated text, so the only estimate of the constraint, and of its gradient, comes from the sampled responses themselves. DA-GRPO meets both requirements with one change to the advantage and one scalar update.

Algorithm 1: DA-GRPO: Dual-Advantage Group-Relative Policy Optimization

Input: training distribution $\mathcal{D}$, iterations T, group size G, budget τ, policy LR η_θ, dual LR η_λ
Output: policy parameters $\boldsymbol{\theta}$

initialize policy parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{\text{pretrained}}$ and dual variable $\lambda \leftarrow \lambda_{\text{init}}$ ▷ price of cloud help
for $t = 1$ *to* T **do**
  sample batch of prompts $\mathcal{B} \sim \mathcal{D}$;
  for *each prompt* $\mathbf{x} \in \mathcal{B}$ **do**
    sample G local responses $\{\mathbf{y}_i^\theta\}_{i=1}^G \sim \pi_\theta(\cdot \mid \mathbf{x})$;
    if *any response in* $\{\mathbf{y}_i^\theta\}_{i=1}^G$ *requests cloud* **then**
      query cloud LLM π_c once to obtain $\mathbf{y}^c \sim \pi_c(\cdot \mid \mathbf{x})$;
      set $\mathbf{y}_i \leftarrow \mathcal{C}\big(\mathbf{y}_i^\theta, \mathbf{y}^c\big)$ for each help response;
      set $\mathbf{y}_i \leftarrow \mathbf{y}_i^\theta$ for the remaining responses;
    else
      set $\mathbf{y}_i \leftarrow \mathbf{y}_i^\theta$ for all $i \in [1, G]$;
    evaluate task reward r_i and cloud cost c_i for responses $i \in [1, G]$;
    compute group-relative reward and cost advantages A_i^r, A_i^c using (2);
    form dual-weighted advantages $\hat{A}_i = A_i^r - \lambda A_i^c$ ▷ budget enters the learning signal
  update $\boldsymbol{\theta}$ using the clipped GRPO surrogate with advantages $\hat{A}_i$ on the sampled local responses; ▷ primal step
  estimate the batch collaboration ratio $\hat{J}_c \leftarrow \mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\Big[\frac{1}{G}\sum_{i=1}^G c_i\Big]$;
  dual update $\lambda \leftarrow \Big[\lambda + \eta_\lambda(\hat{J}_c - \tau)\Big]_+$ ▷ budget sets the price

4 DA-GRPO: Dual-Advantage Group-Relative Policy Optimization

Algorithm 1 gives the procedure. The local model samples G responses per prompt. If any response asks for help, one cloud completion is shared across the help-seeking responses. Each finalized response receives a task reward and a cost. The policy is updated using the dual-weighted advantages, and the multiplier is updated from the batch collaboration ratio. The cloud remains fixed, and policy likelihoods refer to the sampled local responses.

Dual-advantage construction. For a group of G responses with rewards r_i and costs c_i, define

$$A_i^r = r_i - \tfrac{1}{G}\textstyle\sum_{g=1}^G r_g, \qquad A_i^c = c_i - \tfrac{1}{G}\textstyle\sum_{g=1}^G c_g, \tag{2}$$

and let the advantage used in the GRPO update be

$$\hat{A}_i = A_i^r - \lambda A_i^c, \qquad \lambda \geq 0. \tag{3}$$

Because group mean-centering is linear, $\hat{A}_i = (r_i - \lambda c_i) - \overline{(r - \lambda c)}$. Thus, for fixed λ, the combined advantage is the centered advantage of the Lagrangian reward. Proposition A.2 in Appendix A establishes an unbiased score-function estimator before normalization and clipping. The practical policy update uses the corresponding clipped GRPO surrogate; unbiasedness of the underlying estimator does not imply unbiasedness after sample-dependent normalization, token-length averaging, or clipping. No critic or learned cost model is introduced.

Where the budget pressure acts. For binary cost, $A_i^c = 0$ whenever all responses in the group make the same collaboration decision. On a split group with a fraction $\bar{c}$ asking, the cost term contributes $-\lambda(1 - \bar{c})$ to help-seeking responses and $+\lambda\bar{c}$ to local responses. The direct sampled cost gradient therefore acts on groups that disagree about asking. This also limits the feedback: increasing λ cannot directly change a prompt through a group whose cost advantages are all zero. Changes on other prompts and subsequent sampling can still alter its behavior.

What makes cloud help worthwhile? In a two-action abstraction, let $R_{\text{local}}(x)$ and $R_{\text{cloud}}(x)$ denote expected task rewards from local execution and assistance. With unit cloud cost, assistance has the larger Lagrangian value when

$$R_{\text{cloud}}(x) - R_{\text{local}}(x) > \lambda. \tag{4}$$

The multiplier prices the additional reward that assistance provides, rather than difficulty alone. As local competence changes, these gains change; the dual update adjusts the price using aggregate

usage. This is an interpretation of the objective, not a claim that the trained policy implements an optimal threshold.

Shared cloud query. Within each sampled group, the cloud is queried at most once and its completion is shared by all help-seeking responses. The cloud receives the original prompt x, independently of the sampled local reasoning traces. This reduces actual training API calls relative to querying once per help-seeking rollout. The cost c_i nevertheless records whether rollout i uses assistance: its group average estimates per-response collaboration, not the number of physical API calls after sharing. Task reward is evaluated on the composed response $\mathcal{C}(y_i^\theta, y^c)$.

Dual update. After the policy step, let $\hat{J}_c$ be the fraction of sampled responses in the batch that used the cloud. Then

$$\lambda \leftarrow \left[\lambda + \eta_\lambda(\hat{J}_c - \tau)\right]_+, \tag{5}$$

where $[\cdot]_+ := \max(\cdot, 0)$. This is projected gradient descent on the Lagrangian in λ and needs only the scalar $\hat{J}_c$: when usage exceeds the budget, λ rises and asking becomes more expensive in (3); when usage is below budget, λ falls. Proposition B.1 (Appendix B) records the stationary-point interpretation: if the primal step approximately optimizes the policy for the current λ, any interior stationary point $\lambda^* > 0$ of the mean dual dynamics satisfies $J_c(\boldsymbol{\theta}^*(\lambda^*)) = \tau$. At a boundary stationary point $\lambda^* = 0$, the mean dynamics permit $J_c \leq \tau$: the budget is a ceiling and need not be exhausted. The proposition does not establish convergence or finite-sample feasibility; Sections 5.3 and 5.4 examine the dynamics empirically.

Relation to Lagrangian methods. DA-GRPO instantiates a primal-dual scheme for collaborative post-training. Its distinction from fixed assistance shaping is feedback from measured usage to the price. Compared with applying λ to separate reward and cost surrogates, the combined advantage yields the same linear surrogate before clipping when the normalization and reduction conventions agree. After clipping, a combined surrogate and separately clipped surrogates need not coincide. Our formulation retains one GRPO surrogate and extends to several costs as $A_i^r - \sum_k \lambda_k A_i^{c_k}$, or to token cost by replacing c_i (Appendix D.3). These are implementation properties, rather than an intrinsic optimality advantage over other Lagrangian formulations.

5 Experiments

Section 5.2 compares accuracy under common training budget targets and examines allocation. Section 5.3 measures realized usage, alternative dual formulations, and training overhead. Section 5.4 examines initialization, changing targets during training, token costs, and input-dependent usage.

5.1 Setup

Tasks and models. The experiments cover MATH-lighteval (all subsets), MATH-500, ARC (Easy and Challenge combined), MMLU, and TACO-Verified, whose problems carry executable test cases (Hendrycks et al., 2021; Cobbe et al., 2021; Clark et al., 2018; Hendrycks et al., 2020; Li et al., 2023); GSM8K (Cobbe et al., 2021) and AIME 2024 and 2025 serve as shifted test distributions. Correctness is checked automatically per task, by the final boxed answer for mathematics and multiple choice and by execution against all test cases for code, with no learned reward model (Appendix G). The local models are Qwen2.5-1.5B-Instruct and Llama-3.2-3B-Instruct (Bai et al., 2023; Touvron et al., 2023). The cloud model is DeepSeek-R1 (Guo et al., 2025) throughout, used as a fixed service.

Budgets and hyperparameters. The budget is $\tau = 0.3$ for mathematical reasoning, 0.5 for code generation, and 0.2 for knowledge questions, reflecting how much each task type relies on the cloud. DA-GRPO uses $\lambda_{\text{init}} = 0.5$ and $\eta_\lambda = 10^{-2}$ on every task. The remaining hyperparameters are listed in Table 7, and Appendix C reports sensitivity to the policy learning rate and to η_λ. Each accuracy configuration is trained with three seeds; accuracy tables report mean and standard deviation, and curves show the mean with a band over seeds. The two mathematics usage columns report means only. Appendix C shows that small dual steps respond slowly, while aggressive steps can oscillate; $\eta_\lambda = 10^{-2}$ is reused across the reported tasks and models.

Budget-control setting. The comparisons of realized usage, alternative dual formulations, and training cost (Table 2 and Appendix E) train Qwen2.5-1.5B-Instruct on the MATH-lighteval and ARC

Table 1: Accuracy (%) under common training budget targets: $\tau = 0.30$ for mathematics, 0.50 for TACO-Verified, and 0.20 for MMLU. Accuracies are means and standard deviations over three seeds. Mean realized cloud ratios are reported separately for MATH-lighteval and MATH-500; usage measurements are unavailable for TACO-Verified and MMLU. Ratios describe the full evaluation set and are repeated across response blocks. Joint accuracy evaluates delivered responses; conditional local accuracy uses each method's retained subset. The cloud-alone row is a full-usage reference.

Evaluated response	Model	Method	Mean cloud ratio		Accuracy (↑)			
			MATH-lighteval	MATH-500	MATH-lighteval	MATH-500	TACO-Verified	MMLU
		Cloud LLM alone (DeepSeek-R1)	1.00	1.00	98.4	97.3	96.3	90.8
Joint local-cloud response $[\mathcal{C}(\mathbf{y}_i^\theta, \mathbf{y}^c)]$	Qwen2.5-1.5B-Instruct	Local tuning only	0.00	0.00	55.3±1.3	54.2±1.5	44.0±2.7	55.4±0.7
		Local tuning + naive router	0.30	0.30	68.3±1.2	66.4±1.6	69.8±2.2	58.9±0.8
		Local tuning + trained router	0.32	0.31	73.1±5.5	74.8±4.2	75.8±4.3	59.8±2.2
		Collaborative training, GRPO	0.35	0.36	73.5±4.3	70.2±2.6	70.4±2.5	61.5±3.1
		GVPO	0.34	0.34	72.7±4.8	74.0±3.2	76.9±3.7	63.1±3.5
		GAPG	0.31	0.31	78.1±5.0	81.0±3.2	82.2±4.3	64.8±4.1
		Collaborative training, DA-GRPO (ours)	0.31	0.30	84.5±5.0	87.2±2.3	84.8±2.1	66.7±4.1
	Llama-3.2-3B-Instruct	Local tuning only	0.00	0.00	51.6±2.2	46.6±1.2	42.7±3.4	57.9±0.8
		Local tuning + naive router	0.30	0.30	64.7±2.2	62.0±1.2	66.8±3.5	61.2±0.8
		Local tuning + trained router	0.32	0.32	70.3±3.4	68.2±3.7	69.7±6.6	64.2±1.1
		Collaborative training, GRPO	0.33	0.34	61.9±1.9	65.6±1.8	60.8±5.4	63.0±1.0
		GVPO	0.33	0.33	72.2±3.3	75.0±3.5	66.7±4.2	65.4±1.2
		GAPG	0.30	0.30	77.4±2.1	79.6±3.6	71.6±3.1	67.7±1.3
		Collaborative training, DA-GRPO (ours)	0.31	0.30	80.7±1.0	86.6±3.1	74.4±3.3	69.5±1.4
Locally answered queries only $[\mathbf{y}_i^\theta]$	Qwen2.5-1.5B-Instruct	Local tuning only	0.00	0.00	55.3±1.3	54.2±1.5	44.0±2.7	55.4±0.7
		Local tuning + naive router	0.30	0.30	55.2±1.3	54.6±1.4	43.7±2.4	55.2±0.6
		Local tuning + trained router	0.32	0.31	62.2±4.8	65.6±3.2	55.2±6.1	56.3±1.0
		Collaborative training, GRPO	0.35	0.36	56.0±3.5	58.5±3.4	51.8 ±4.1	57.3±1.8
		GVPO	0.34	0.34	59.0±3.8	60.1±3.8	57.4±4.0	58.2±1.2
		GAPG	0.31	0.31	67.8±4.3	75.2±4.1	61.6±3.8	58.9±1.4
		Collaborative training, DA-GRPO (ours)	0.31	0.30	77.2±5.5	77.7±4.1	65.8±4.5	59.5±1.7
	Llama-3.2-3B-Instruct	Local tuning only	0.00	0.00	51.6±2.2	46.6±1.2	42.7±3.4	57.9±0.8
		Local tuning + naive router	0.30	0.30	50.3±2.2	45.7±0.9	42.5±2.8	57.5±0.6
		Local tuning + trained router	0.32	0.32	58.7±3.2	55.7±3.2	48.1±6.1	61.5±0.9
		Collaborative training, GRPO	0.33	0.34	53.1±3.2	52.0±3.1	46.6±8.2	59.2±1.4
		GVPO	0.33	0.33	62.0±3.5	65.4±2.9	50.9±3.4	61.1±0.9
		GAPG	0.30	0.30	63.8±4.0	71.1±2.5	51.1±2.9	63.1±0.9
		Collaborative training, DA-GRPO (ours)	0.31	0.30	70.3±4.7	73.3±2.6	56.5±2.8	64.8±0.8

mixture at $\tau = 0.3$ for 300 updates with five seeds, and measure the collaboration ratio and the cloud input tokens per query at test time with greedy decoding (temperature 0), the decoding configuration of the accuracy results. Two further baselines target the same budget: an objective-level dual, which applies λ to separately clipped reward and cost surrogates in the loss, and a length-control-inspired controller adapted from Gao et al. (2025) to cloud cost (Appendix E.1).

Help action and reward. The local prompt includes the explicit request "I need external assistance." Local reasoning is organized into at most six steps for mathematics and code and four for knowledge questions. The fixed cloud model is queried with the original prompt. The finalized response receives reward 1 when correct and correctly formatted, -0.1 for a format violation, and 0 otherwise; the resource cost records whether assistance was used. Training samples at temperature 1, while evaluation uses greedy decoding. Full prompts and task-specific correctness checks appear in Appendix G.

Metrics. We report four quantities. *Joint accuracy* is the accuracy of the response the user receives, local or composed with the cloud, under each method's own collaboration decisions and a common training budget target. This is the deployment metric. *Accuracy on locally answered queries* is computed on the queries a method chose to answer locally. Since that subset differs across methods, this number measures allocation quality, whether a method keeps the queries it can solve, and not the intrinsic ability of the local model. *Fixed-set local accuracy* evaluates every method on one fixed problem set with the cloud disabled, which isolates the local model (Figure 3). *Realized usage* is measured by the collaboration ratio and cloud input tokens per query averaged over all queries. Table 1 reports mean collaboration ratios for the two mathematics benchmarks; Table 2 reports additional usage statistics for the separate budget-control setting.

Baselines. Two inference-time routing baselines tune the local model without a help action and add a router afterwards: a naive router that offloads a random subset of queries to a given target rate τ, and a trained router, a binary classifier that predicts whether a query should be forwarded. Local-only tuning without any router provides a reference without cloud access. Three collaborative-training baselines give the local model the help action and train it with a hierarchical reward containing a fixed assistance term, using GRPO (Guo et al., 2025), GVPO (Zhang et al., 2025), or GAPG (Fang et al., 2026). For every baseline, task, and budget we grid-search the assistance term so that the training-time ratio lands on the target; the selected values are 0.2 for mathematics, 0.05 for code, and 0.4 for knowledge questions (Table 7). DA-GRPO receives no such search over an assistance reward.

Table 2: Budget satisfaction and input-dependent cloud usage in the $\tau = 0.3$ setting (MATH-lighteval with ARC, 300 updates, five seeds). (a) Collaboration ratio at the end of training and at test time (mean and standard deviation over seeds), cloud input tokens per query at test time averaged over all queries, and joint accuracy where measured; the first three methods target 0.30, and GRPO with fixed shaping has no explicit target. (b) Cloud usage of the DA-GRPO policy on the MATH-lighteval test set stratified by its native difficulty labels (437 to 1324 problems per level), with the local accuracy on each level.

(a) Realized budget by method

Method	Train ratio	Test ratio	Tokens/query	Joint acc.
DA-GRPO (ours)	$0.313_{\pm 0.083}$	$0.314_{\pm 0.070}$	93.0	80.4
Objective-level dual	$0.345_{\pm 0.073}$	$0.346_{\pm 0.069}$	99.1	82.4
Length-control-inspired	$0.322_{\pm 0.277}$	$0.308_{\pm 0.268}$	88.5	–
GRPO, fixed shaping	$0.345_{\pm 0.076}$	$0.355_{\pm 0.070}$	102.1	–

(b) DA-GRPO usage by difficulty

Level	Local acc.	Cloud ratio
1 (easiest)	0.778	$0.104_{\pm 0.021}$
2	0.698	$0.205_{\pm 0.026}$
3	0.565	$0.278_{\pm 0.008}$
4	0.437	$0.366_{\pm 0.028}$
5 (hardest)	0.212	$0.494_{\pm 0.046}$

5.2 Accuracy under common training budget targets

DA-GRPO reaches the highest mean joint accuracy in every setting. Table 1 compares two models on four benchmarks. DA-GRPO improves on GAPG, the strongest fixed-reward baseline, by 1.8 to 7.0 percentage points, and on the trained router by 4.7 to 18.4 points. These are differences in three-seed means; the reported standard deviations do not establish statistical significance. The baselines were tuned toward the same task-specific training targets. Table 1 reports realized mean cloud ratios of 0.31 on MATH-lighteval and 0.30 on MATH-500 for DA-GRPO with both models. The corresponding GAPG means are 0.31 on both datasets for Qwen and 0.30 on both for Llama. For TACO-Verified and MMLU, realized usage is unavailable; those comparisons share training targets but do not establish matched test-time cost. Figure 11 reports the training trajectories; Section 5.3 separately evaluates controllers in a five-seed, 300-update budget-control setting.

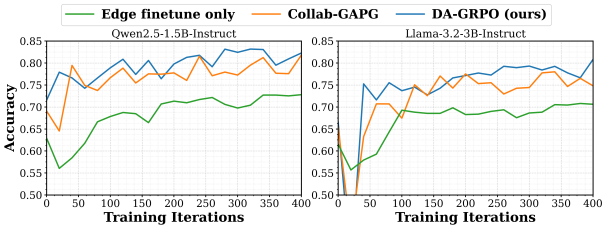


Figure 3: Fixed-set local accuracy with the cloud disabled. All methods answer the same fixed set of MATH-lighteval problems selected using a Qwen2.5-7B model, which removes the allocation effect and isolates the local model.

The results support a contribution from allocation. DA-GRPO has the highest accuracy on locally retained queries in all eight cells, consistent with retaining queries it can answer reliably. Since each method selects a different subset, this metric alone does not isolate local competence. Figure 3 evaluates all methods on the same selected set with the cloud disabled. DA-GRPO and GAPG both improve on local-only tuning, while their difference is smaller than the joint-accuracy differences in Table 1. Together, these observations support an allocation benefit, but the selected-set control does not quantitatively decompose the gain into local competence, routing, and realized cloud usage.

Accuracy and realized resource use. Relative to the gap between local-only tuning and the cloud-alone reference, DA-GRPO closes 62 to 79% on the mathematics benchmarks, 59 to 78% on TACO-Verified, and 32 to 35% on MMLU. These configurations use training targets of 0.30, 0.50, and 0.20, respectively; those targets should not be substituted for measured test-time usage. For an evaluation run with realized ratio ρ, joint accuracy decomposes as $a_{\text{joint}} = (1-\rho)a_{\text{local}} + \rho a_{\text{assisted}}$, where the two conditional accuracies are measured on the retained and assisted subsets. Appendix D.1 examines further training targets.

5.3 Budget satisfaction at training and test time

Realized usage tracks the target approximately. In the separate five-seed budget-control setting of Table 2(a), DA-GRPO has a mean collaboration ratio of 0.313 at the end of training and 0.314 at in-distribution test time, consuming 93.0 cloud input tokens per query. The test mean exceeds the 0.30 target by 1.4 percentage points, and the across-seed standard deviation is 0.070. In the reported

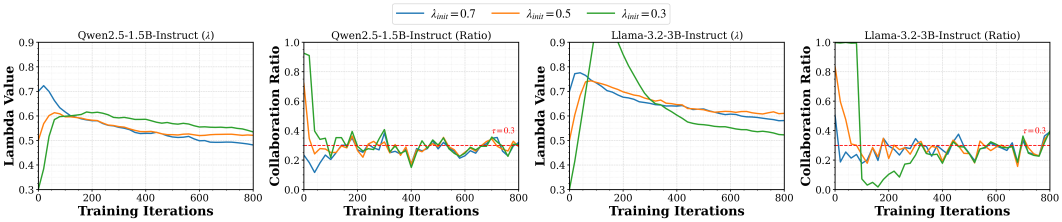


Figure 4: Dual variable (first and third panels) and realized collaboration ratio (second and fourth panels) on MATH-lighteval with ARC at $\tau = 0.3$, from $\lambda_{\text{init}} \in \{0.3, 0.5, 0.7\}$. The trajectories approach one another for both models while the ratio holds near the target (dashed line).

within-run analysis, the 50-update moving average lies between 0.310 and 0.319. GRPO with fixed shaping evaluates at 0.355 and 102.1 tokens per query. The length-control-inspired controller has a similar mean ratio but greater variation across seeds (training standard deviation 0.277 versus 0.083); this statistic measures repeatability, not switching within a run.

Advantage level versus objective level. The two formulations share the linear surrogate before clipping under the same normalization and reduction conventions. In Table 2(a), the objective-level dual reaches a test ratio of 0.346 and 99.1 input tokens per query, compared with 0.314 and 93.0 for DA-GRPO. Its joint accuracy is higher (82.4 versus 80.4), but so is usage; these operating points do not establish either method's accuracy advantage at matched cost. The advantage-level form retains a single clipped GRPO surrogate, while changing or adding a resource affects the cost advantages and scalar dual updates.

No measurable training overhead. Over 300 updates on the same H100 setup and seeds, DA-GRPO takes 2.32 hours and 9.28 GPU-hours, the same as vanilla GRPO to the reported precision, since it adds one group-relative advantage and one scalar update and no model, rollout, or forward or backward pass (Appendix E). At inference it needs no router.

5.4 The price the budget sets

Different initializations approach similar operating points. Figure 4 trains on MATH-lighteval mixed with ARC at $\tau = 0.3$ and starts λ at 0.3, 0.5, and 0.7. For both models the three trajectories approach one another over 800 iterations while the realized ratio fluctuates around 0.3. The transient is informative. For the Llama run started at $\lambda = 0.3$, the model first forwards nearly every query; λ climbs out of the plotted range, the ratio drops to nearly zero, and both then settle near the target. The dual variable rises while the model is weak and over-uses the cloud, then falls as competence grows. The price therefore responds to the training trajectory. In these experiments, the same τ and η_λ serve both models without a model-specific assistance-reward search.

Budget specification versus controller tuning. The target τ specifies the resource allowance, whereas η_λ controls how quickly the assistance price responds to observed budget violations. A fixed assistance reward must be calibrated to induce a desired collaboration ratio; the dual learning rate instead governs the dynamics of that calibration. Appendix C illustrates this distinction: $\eta_\lambda \leq 10^{-3}$ responds slowly, while $\eta_\lambda \in \{10^{-1}, 3 \times 10^{-1}\}$ produces oscillatory behavior. We use $\eta_\lambda = 10^{-2}$ across the reported binary-cost tasks and models. The practical benefit is therefore reuse of a controller setting across these configurations, reducing repeated searches for task-specific assistance rewards. This does not eliminate optimization sensitivity: changes in cost units or the training regime may require controller recalibration.

The policy responds to changing budgets during continued training. Figure 5 holds the training data fixed and changes τ through four phases in four orders. Tightening the target increases the price when usage exceeds the new allowance; relaxing it reduces that pressure when usage is below target. The trajectories show adjustment toward successive targets without fixing the policy to its initial operating point. Both policy and multiplier continue to update throughout this experiment. It therefore demonstrates adaptation during post-training, not an inference-time budget knob for a frozen policy. Appendix D.2 lists the schedules.

The same framework supports token budgets. Binary cost measures whether a response uses the cloud. To account for prompt length, we can instead set c_i to the number of prompt tokens sent to the

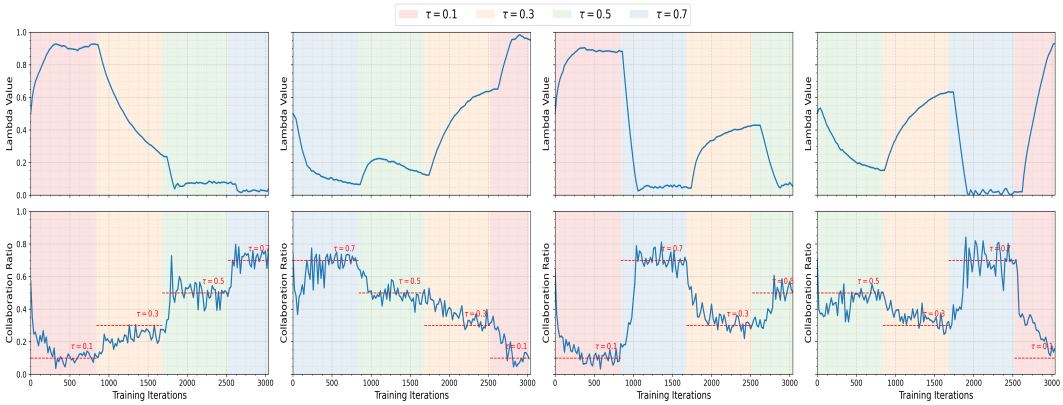


Figure 5: Tracking changing collaboration targets during continued training on fixed data. The top row shows λ and the bottom row realized collaboration. Shaded regions and dashed lines indicate the successive targets. All four schedules are retained; both policy parameters and the multiplier continue to update.

cloud, zero for a local response, and τ to a token allowance, leaving (5) unchanged and still following Algorithm 1. With an allowance of 30% of the mean prompt length, $0.30 \times 305.5 = 91.6$ tokens per query, realized usage in the final 50-update window is 88.2 and 89.7 tokens for two seeds while λ decreases from 0.500 to 0.375 and 0.388. At test time the same policy uses 90.6 tokens per query in distribution, 82.3 on the easier GSM8K, and 136.0 on the harder MATH-500 (Appendix E). A variant that weights input and output tokens against a 120-token target approaches the allowance from three initial settings (Figure 10, Appendix D.3).

Usage follows difficulty at test time. The training objective constrains an average, while the learned decision remains input-dependent. On the MATH-lighteval test set stratified by its native difficulty labels, the $\tau = 0.3$ policy forwards 0.104 of level-1 problems and 0.494 of level-5 problems, rising monotonically with level as local accuracy falls from 0.778 to 0.212 (Table 2(b)). Across datasets the same policy forwards 0.219 of GSM8K, 0.390 of MATH-500, and 0.733 of AIME 2024 and 2025 problems, against 0.314 in distribution; these sets differ in more than difficulty, and AIME has 30 problems per year, so the stratified comparison is the controlled one (Appendix E). The target regulates the training-distribution expectation. Usage may exceed that target after a distribution shift; difficulty-dependent behavior alone does not establish budget feasibility or optimal allocation on the shifted distribution. Appendix D.1 repeats the comparison at $\tau = 0.5$ and 0.7, where DA-GRPO again has the highest reported conditional local and joint accuracy, with the corresponding usage trajectories shown explicitly.

6 Conclusion

DA-GRPO uses feedback from realized cloud usage to adjust the assistance price during collaborative post-training. Group-relative reward and cost advantages enter a single clipped GRPO surrogate, and one scalar update compares usage with the training target. Across two SLMs and four benchmarks, DA-GRPO achieves the highest mean joint accuracy under common task-specific training budget targets, with fixed-set results supporting a contribution from allocation. Additional experiments demonstrate near-target average usage in distribution, adjustment to changing targets during continued training, and an extension to token costs, with no measurable training overhead in the reported configuration.

The objective constrains expected cost under the training distribution and rollout policy. The observed 0.314 in-distribution test ratio against a 0.30 target is approximate tracking, not a hard cap; distribution shift can increase usage further. Strict deployment allowances or a budget adjustable at inference without retraining require additional mechanisms. The cloud model is fixed, the budget-control comparisons use one training mixture and cloud model, and the cloud is conditioned on the original prompt rather than the local reasoning trace. The fixed-set control does not fully separate improvements in local competence from allocation, and unanimous sampled groups provide no direct cost-advantage signal.

AI USE STATEMENT

In this work, we used generative AI tools to help develop theoretical models or conceptual frameworks, design or provide feedback on research methodology or experiments, and interpret results. We have not used generative AI tools to generate synthetic data sets, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis. Additionally, we used generative AI tools to edit a research paper to improve readability. We have reviewed all AI-assisted work. We have verified the final text, derivations, and reported measurements against the implementation and experimental records. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses public benchmarks and involves no human subjects and no personal data. The method decides when a query leaves the device for a cloud service. Deployments that handle sensitive inputs should account for the fact that the learned policy neither inspects sensitivity nor guarantees a hard test-time cap; its training budget is not a privacy guarantee. We are not aware of other ethical concerns specific to this work.

REPRODUCIBILITY STATEMENT

Algorithm 1 specifies the full procedure, and Appendices A and B state and prove the two propositions. Table 7 lists the reported training hyperparameters, including the per-task assistance terms used by the fixed-reward baselines, and Appendix G gives the exact system and user prompts, the reward and cost functions, and the correctness-checking procedure for each benchmark. Appendix E describes the budget-control setting, the alternative dual formulations, and the token-budget and distribution-shift protocols, and Appendix C reports the sensitivity to the policy learning rate and the dual step size. All datasets are public.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. PMLR, 2017.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*, 2025.

Abhinav Bhatia, Pradeep Varakantham, and Akshat Kumar. Resource constrained deep reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pp. 610–620, 2019.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024.

Wenzhi Fang, Dong-Jun Han, Liangqi Yuan, Evan Chen, and Christopher Brinton. Bridging on-device and cloud llms for collaborative reasoning: A unified methodology for local routing and post-training. *arXiv preprint arXiv:2509.24050*, 2026.

Chengqian Gao, Haonan Li, Taylor W Killian, Jianshu She, Renxi Wang, Liqun Ma, Zhoujun Cheng, Shibo Hao, and Zhiqiang Xu. Concise reasoning in the lens of lagrangian optimization. *arXiv preprint arXiv:2510.10168*, 2025.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning*, 2024.

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. *arXiv preprint arXiv:2503.20783*, 2025.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Wei Liu, Jian Luan, Xiwen Zhang, Nicholas D Lane, and Mengwei Xu. Demystifying small language models for edge deployment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14747–14764, 2025.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 2: short papers)*, pp. 1773–1781, 2023.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

Guanqiao Qu, Qiyuan Chen, Wei Wei, Zheng Lin, Xianhao Chen, and Kaibin Huang. Mobile edge intelligence for large language models: A contemporary survey. *IEEE Communications Surveys & Tutorials*, 27(6):3820–3860, 2025.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Yixuan Wang, Simon Sinong Zhan, Ruochen Jiao, Zhilu Wang, Wanxin Jin, Zhuoran Yang, Zhaoran Wang, Chao Huang, and Qi Zhu. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In *International Conference on Machine Learning*, pp. 36593–36604. PMLR, 2023.

Daliang Xu, Wangsong Yin, Hao Zhang, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. Edgellm: Fast on-device llm inference with speculative decoding. *IEEE Transactions on Mobile Computing*, 2024a.

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*, 2024b.

Kaichen Zhang, Yuzhong Hong, Junwei Bao, Hongfei Jiang, Yang Song, Dingqian Hong, and Hui Xiong. Gvpo: Group variance policy optimization for large language model post-training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. To appear; arXiv:2504.19599.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Junchen Zhao, Yurun Song, Simeng Liu, Ian G Harris, and Sangeetha Abdu Jyothi. Lingualinked: Distributed large language model inference on mobile devices. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 160–171, 2024.

Yichen Zhu, Minjie Zhu, Ning Liu, Zhiyuan Xu, and Yaxin Peng. Llava-phi: Efficient multi-modal assistant with small language model. In *Proceedings of the 1st International Workshop on Efficient Multimedia Computing under Limited*, pp. 18–22, 2024.

Appendix

A Training Objective of DA-GRPO **14**

B Interpretation of the Dual Variable Update **14**

C Sensitivity Analysis **15**

C.1 Analysis on primal step size η 15

C.2 Analysis on dual variable step size η_λ 16

D Additional Experiments **17**

D.1 Different Collaboration Ratios 17

D.2 Results on Time-varying Collaboration Ratio τ_t 18

D.3 Extension to token-aware cloud cost 19

D.4 Training curves on MATH-lighteval 19

E Budget Satisfaction, Alternative Dual Formulations, and Test-Time Cloud Usage **20**

E.1 Baselines that target the same budget 20

E.2 Realized usage and temporal stability 20

E.3 Cloud input-token budget 20

E.4 Cloud usage under test-time distribution shift 21

E.5 Training cost 22

F Details of Hyperparameters **22**

G Prompt Template and Reward Function **22**

G.1 Prompt Template 22

G.2 Reward Function 23

H Case Study: Local Responses and Unnecessary Assistance **25**

A TRAINING OBJECTIVE OF DA-GRPO

We distinguish an unbiased score-function estimator from the clipped surrogate used in practical training. Fix a prompt x, policy parameters θ, and a multiplier λ independent of the current sampled group. Draw local responses y_i^θ independently from $\pi_\theta(\cdot \mid x)$. A shared cloud completion can be represented by an auxiliary draw $z \sim \pi_c(\cdot \mid x)$, independent of the local samples and of θ; it need only be evaluated when a response requests assistance. Conditional on z, each finalized response and its reward depend on its own local sample. Assume the usual regularity conditions for differentiating the expectation and finite relevant moments.

Result A.1 (Group-gradient estimator; cf. Proposition 3.1 in Fang et al. (2026))**.** *Let $r_i = r(x, y_i)$ and $\bar{r} = G^{-1} \sum_{i=1}^{G} r_i$. For $G \geq 2$,*

$$\widehat{\nabla_\theta R}(\theta, x) = \frac{1}{G-1} \sum_{i=1}^{G} \nabla_\theta \log \pi_\theta(y_i^\theta \mid x)(r_i - \bar{r}) \tag{6}$$

is unbiased for $\nabla_\theta \mathbb{E}_{y^\theta, z}[r(x, y)]$ under the sampling assumptions above. Equivalently, the estimator is the group average multiplied by $G/(G-1)$.

Define

$$A_i^r = r_i - \frac{1}{G} \sum_{g=1}^{G} r_g, \qquad A_i^c = c_i - \frac{1}{G} \sum_{g=1}^{G} c_g, \tag{7}$$

and

$$\hat{A}_i = A_i^r - \lambda A_i^c. \tag{8}$$

Proposition A.2 (Unbiased dual-weighted score-function estimator)**.** *Under the same assumptions, define $v_i = r_i - \lambda c_i$ and $\bar{v} = G^{-1} \sum_i v_i$. Then*

$$\widehat{\nabla_\theta R}_\lambda(\theta, x) = \frac{1}{G-1} \sum_{i=1}^{G} \nabla_\theta \log \pi_\theta(y_i^\theta \mid x) \hat{A}_i \tag{9}$$

is unbiased for $\nabla_\theta \mathbb{E}_{y^\theta, z}[r(x, y) - \lambda c(x, y)]$.

Proof. By linearity of centering, $\hat{A}_i = v_i - \bar{v}$. Write $s_i = \nabla_\theta \log \pi_\theta(y_i^\theta \mid x)$. Conditional on the auxiliary cloud draw z, $\mathbb{E}[s_i \mid z] = 0$, and for $i \neq j$, conditional independence gives $\mathbb{E}[s_i v_j \mid z] = 0$. If $g(z) = \mathbb{E}[s_i v_i \mid z]$, then

$$\mathbb{E}[s_i(v_i - \bar{v}) \mid z] = \left(1 - \frac{1}{G}\right) g(z).$$

Summing over i and dividing by $G - 1$ yields $g(z)$. Averaging over the θ-independent draw z gives the stated policy gradient by the score-function identity. The reward-only result follows by taking $\lambda = 0$. □

These results concern on-policy, sequence-level score-function estimators without sample-dependent normalization or clipping. They do not establish unbiasedness for a clipped GRPO surrogate, for response-length-weighted reductions, or after dividing advantages by a random group standard deviation. Nor do they establish convergence of the coupled policy and multiplier updates. Sharing a cloud completion is compatible with the conditional argument above because the cloud receives only x; conditioning cloud generation on the sampled local group would require revisiting that argument.

B INTERPRETATION OF THE DUAL VARIABLE UPDATE

The following result characterizes stationary points of an idealized mean update. It does not assert that stochastic training converges or that every realized batch or test set satisfies the budget.

Proposition B.1 (Stationary points of the mean dual update)**.** *For the projected stochastic update*

$$\lambda_{t+1} = [\lambda_t + \eta_\lambda(\hat{J}_c(\theta_t) - \tau)]_+, \tag{10}$$

assume $\hat{J}_c$ is an unbiased cost estimate and let $\theta^*(\lambda)$ denote an idealized primal response to λ. Consider the associated deterministic mean map

$$F(\lambda) = [\lambda + \eta_\lambda(J_c(\theta^*(\lambda)) - \tau)]_+, \qquad \eta_\lambda > 0.$$

Any fixed point $\lambda^* > 0$ of F satisfies $J_c(\theta^*(\lambda^*)) = \tau$. At a fixed point $\lambda^* = 0$, $J_c(\theta^*(0)) \leq \tau$.

Proof. If $F(\lambda^*) = \lambda^* > 0$, the projected expression must itself be positive, so the projection is inactive at that point. Subtracting λ^* from the fixed-point equation gives $\eta_\lambda(J_c(\theta^*(\lambda^*)) - \tau) = 0$, proving equality. If $\lambda^* = 0$, the fixed-point equation is $[\eta_\lambda(J_c(\theta^*(0)) - \tau)]_+ = 0$, which is equivalent to $J_c(\theta^*(0)) \leq \tau$. □

The mean map is formed by replacing the sampled residual with its expectation before projection. It is generally not identical to the expectation of the projected stochastic update. Consequently, the proposition provides a stationary-point interpretation, not a finite-step or stochastic feasibility guarantee. It also explains why the inequality constraint need not be active: when $\lambda = 0$, unused budget is compatible with stationarity.

For fixed λ, the unnormalized group advantage is that of the shaped reward $\tilde{r} = r - \lambda c$. With the binary actual-use cost, a correctly formatted correct local answer receives 1, a correctly formatted correct assisted answer receives $1 - \lambda$, and an incorrect but correctly formatted assisted answer receives $-\lambda$. Format violations receive $-0.1 - \lambda c$; resource cost is charged whenever assistance was actually used, independently of correctness or final formatting. This decomposition distinguishes task quality from resource consumption.

C Sensitivity Analysis

C.1 Analysis on primal step size η

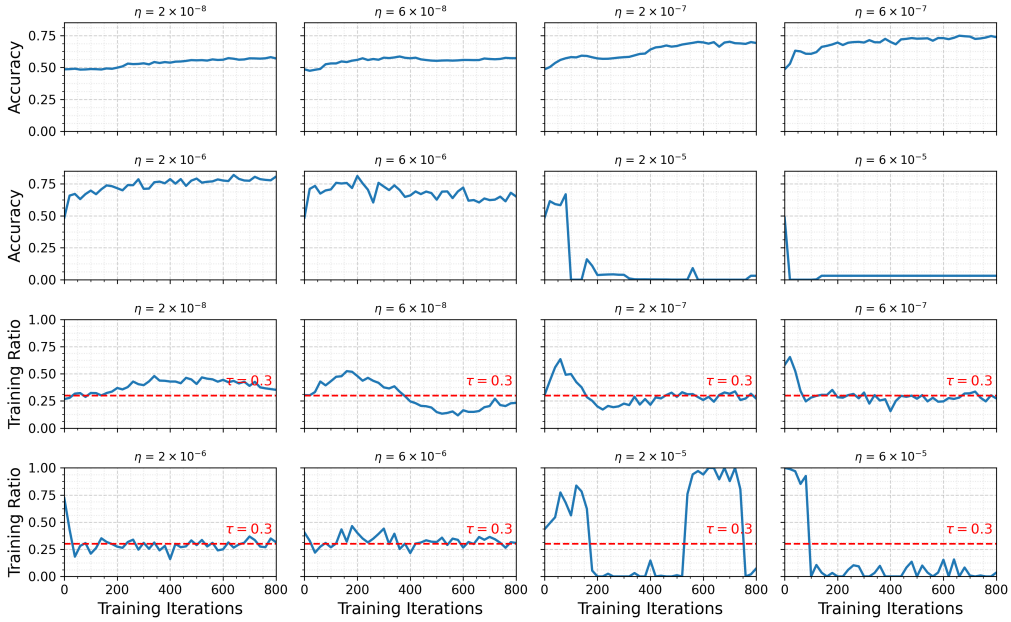


Figure 6: Accuracy (top two rows) and collaboration ratio (bottom two rows) trajectories over training iterations for different learning rates η. Moderate learning rates ($\eta = 2 \times 10^{-6}$ and 6×10^{-6}) yield stable convergence with sustained accuracy and collaboration ratios fluctuating around the target threshold $\tau = 0.3$, while smaller η leads to slow or negligible progress and larger η causes instability or collapse in both accuracy and collaboration ratio.

Fig. 6 illustrates the sensitivity of both model accuracy and the collaboration ratio to the learning rate η. We observe that extremely small learning rates ($\eta \leq 6 \times 10^{-8}$) result in negligible progress,

where the collaboration ratio remains largely inactive and accuracy stagnates near its initialization level. As η increases to a moderate range ($\eta = 2 \times 10^{-6}$ and 6×10^{-6}), the system exhibits stable convergence behavior, achieving consistently high accuracy while maintaining the collaboration ratio close to the target threshold $\tau = 0.3$. This regime represents a favorable balance between effective learning and controlled training participation.

In contrast, overly large learning rates ($\eta \geq 2 \times 10^{-5}$) lead to pronounced instability. In these cases, the collaboration ratio oscillates or collapses to degenerate values, which in turn causes sharp drops in accuracy or complete training failure. These results highlight the importance of proper learning rate selection in our framework, as it directly impacts both optimization stability and the ability to regulate training dynamics according to the desired ratio constraint.

C.2 Analysis on dual variable step size η_λ

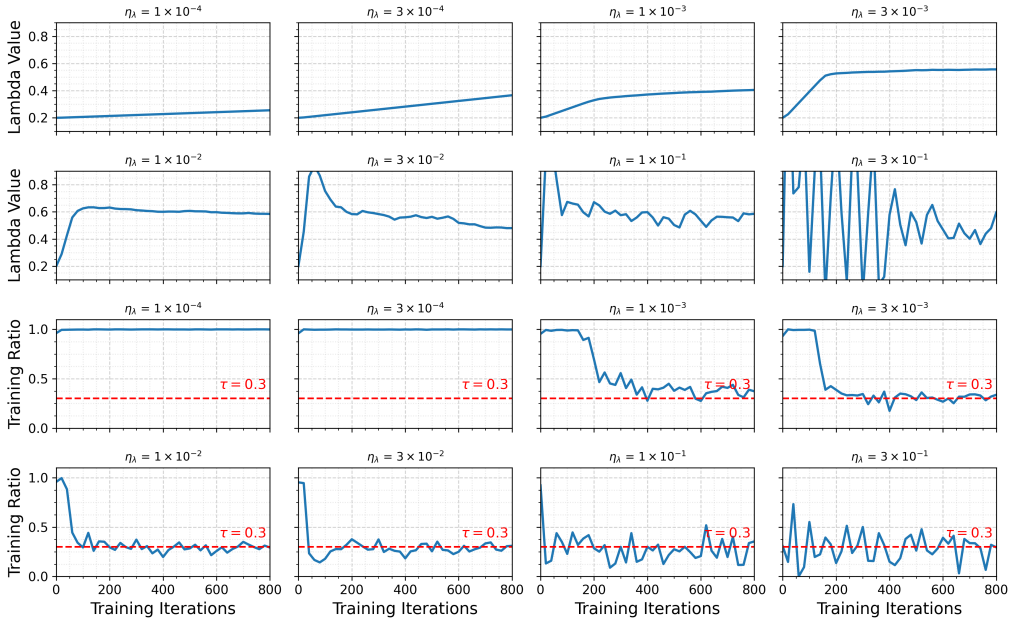


Figure 7: Evolution of the dual variable λ (top two rows) and the resulting collaboration ratio (bottom two rows) over training iterations for different step sizes η_λ. Smaller step sizes lead to smooth and stable updates of λ, while larger values of η_λ induce oscillatory or unstable behavior. The dashed red line indicates the target collaboration ratio threshold $\tau = 0.3$, which is approximately maintained for moderate step sizes but violated under overly aggressive updates.

Fig. 7 studies the impact of the λ-update stepsize η_λ on the evolution of the adaptive control variable λ and the resulting collaboration ratio on Qwen2.5-1.5B-Instruct. The first two rows show the trajectory of λ over training iterations for different values of η_λ, while the last two rows report the corresponding collaboration ratio relative to the target threshold $\tau = 0.3$.

For very small stepsizes ($\eta_\lambda \leq 10^{-3}$), λ evolves slowly and remains nearly constant throughout training. Although this regime is stable, it limits the responsiveness of the adaptive control mechanism, effectively reducing it to a static weighting strategy. Consequently, the collaboration ratio converges slowly and even fails to achieve the desired ratio.

When η_λ is increased to a moderate value ($\eta_\lambda = 1 \times 10^{-2}$), λ adapts smoothly and converges rapidly to a stable range without oscillations. In this regime, the collaboration ratio closely tracks the target threshold τ with low variance across iterations. This behavior reflects a favorable balance between responsiveness and stability, enabling effective correction of communication imbalance while maintaining stable convergence.

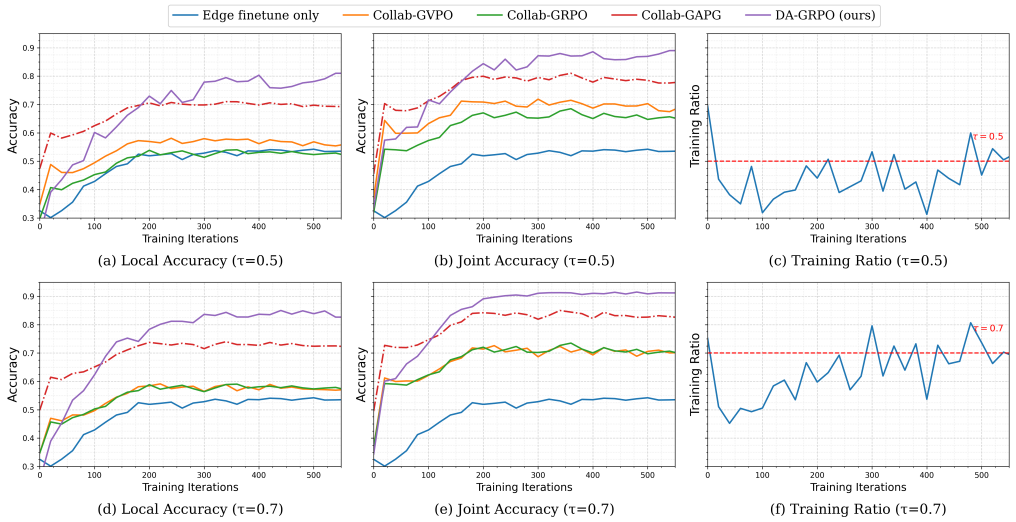
For larger stepsizes ($\eta_\lambda \geq 3 \times 10^{-2}$), the dynamics of λ become increasingly unstable. Significant oscillations and overshooting are observed, particularly for $\eta_\lambda = 10^{-1}$ and 3×10^{-1}. These instabilities propagate directly to the collaboration ratio, leading to large fluctuations around the

target and, in some cases, extreme deviations. Such behavior undermines convergence stability and degrades the practical effectiveness of the adaptive control mechanism.

Overall, $\eta_\lambda = 1 \times 10^{-2}$ provides the most reliable performance, achieving fast convergence of λ, stable long-term behavior, and accurate tracking of the desired collaboration ratio. For this reason, this setting is adopted throughout the main experiments unless stated otherwise.

D Additional Experiments

D.1 Different Collaboration Ratios



(a) Local Accuracy (τ=0.5) (b) Joint Accuracy (τ=0.5) (c) Training Ratio (τ=0.5)

(d) Local Accuracy (τ=0.7) (e) Joint Accuracy (τ=0.7) (f) Training Ratio (τ=0.7)

Figure 8: Training dynamics under different collaboration ratios. We report local accuracy, joint accuracy, and the effective collaboration ratio over training iterations for $\tau = 0.5$ (top row) and $\tau = 0.7$(bottom row). DA-GRPO consistently achieves higher local and joint accuracy compared to edge-only fine-tuning and other collaborative baselines, while adaptively adjusting the collaboration frequency to meet the target ratio.

Fig. 8 shows the training dynamics of different methods for Qwen2.5-1.5B-Instruct under target collaboration ratios $\tau = 0.5$ and $\tau = 0.7$, including local accuracy, joint accuracy, and the effective collaboration ratio. Across both settings, DA-GRPO consistently achieves higher local and joint accuracy than edge-only fine-tuning and other collaborative baselines.

The fixed-set local-only evaluation (Figure 3) supports improvements over local-only tuning on that selected set. The larger joint-accuracy differences are consistent with a contribution from collaboration decisions, but do not independently quantify that contribution or eliminate differences in realized cloud usage.

Overall, these experiments extend the accuracy comparison to additional training targets. Conditional local accuracy, joint accuracy, and realized collaboration should be interpreted together, because changing the target also changes which queries remain local.

We then directly compare the final achieved mathematical reasoning performance on the MATH-lighteval benchmark under different collaboration ratios τ. For each model, we report two accuracy metrics: 1) We measure the joint response accuracy, which accounts for both locally generated answers and those delegated to the cloud model. This reflects the overall task performance of the local-cloud system. 2) We compute conditional local accuracy on questions retained on the device. This subset changes with the policy and budget, so improvements in this metric alone do not demonstrate improved intrinsic local reasoning ability.

Fig. 9 shows increasing joint and conditional local accuracy across the reported targets. Higher conditional local accuracy is compatible with offloading more difficult queries and retaining an easier subset. The figure therefore characterizes system behavior as the target changes, rather than isolating changes in standalone local competence.

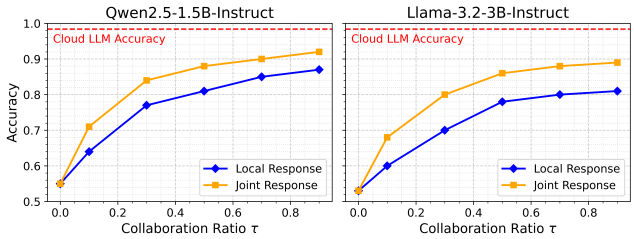


Figure 9: Joint and conditional local accuracy on MATH-lighteval across training collaboration targets. Conditional local accuracy uses a policy-dependent retained subset.

Algorithm 2: DA−GRPO with token-aware cloud cost

Input: training distribution $\mathcal{D}$, iterations T, group size G, target token budget τ, policy LR η_θ, dual LR η_λ, token weights α, β

Output: policy parameters $\boldsymbol{\theta}$

initialize policy parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{\text{pretrained}}$ and dual variable $\lambda \leftarrow \lambda_{\text{init}}$;
for $t = 1$ *to* T **do**
  sample batch of prompts $\mathcal{B} \sim \mathcal{D}$;
  for *each prompt* $\mathbf{x} \in \mathcal{B}$ **do**
    sample G local responses $\{\mathbf{y}_i^\theta\}_{i=1}^G \sim \pi_\theta(\cdot \mid \mathbf{x})$;
    for *each response* $i \in [1, G]$ **do**
      if $\mathbf{y}_i^\theta$ *requests cloud* **then**
        query cloud LLM π_c to obtain $\mathbf{y}_i^c \sim \pi_c(\cdot \mid \mathbf{x})$ and set $\mathbf{y}_i \leftarrow \mathcal{C}(\mathbf{y}_i^\theta, \mathbf{y}_i^c)$;
        record cloud input and output token counts L_i^{in} and L_i^{out};
      else
        set $\mathbf{y}_i \leftarrow \mathbf{y}_i^\theta$, $L_i^{\text{in}} \leftarrow 0$, and $L_i^{\text{out}} \leftarrow 0$;
    evaluate task reward r_i and token-aware cost $c_i \leftarrow \alpha L_i^{\text{in}} + \beta L_i^{\text{out}}$ for $i \in [1, G]$;
    compute reward and cost advantages using (2) and form $\hat{A}_i = A_i^r - \lambda A_i^c$;
  update $\boldsymbol{\theta}$ using the clipped GRPO surrogate with advantages $\hat{A}_i$ on the sampled local responses;
  estimate average cloud-token usage per question: $\hat{J}_c \leftarrow \mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\left[\frac{1}{G}\sum_{i=1}^G c_i\right]$;
  dual update: $\lambda \leftarrow \left[\lambda + \eta_\lambda(\hat{J}_c - \tau)\right]_+$;

D.2 Results on Time-varying Collaboration Ratio τ_t

Figure 5 demonstrates the robustness of our method to changes in the target collaboration ratio τ_t during post-training. Rather than fixing the collaboration level through a static penalty or reward formulation, our algorithm continuously updates the dual variable λ in response to observed constraint violations. To establish a thorough evaluation through all different scenarios, we observe four different time-varying τ settings: 1) $\tau_t : 0.1 \to 0.3 \to 0.5 \to 0.7$, 2) $\tau_t : 0.7 \to 0.5 \to 0.3 \to 0.1$, 3) $\tau_t : 0.1 \to 0.7 \to 0.3 \to 0.5$, and 4) $\tau_t : 0.5 \to 0.3 \to 0.7 \to 0.1$. As τ varies across different phases, the λ adapts accordingly, increasing when the achieved collaboration ratio exceeds the target and decreasing otherwise. This mechanism enables stable tracking of the desired collaboration level, as reflected in the bottom row.

The observed policy can move away from its initial operating point when both policy optimization and dual updates continue. This experiment does not establish inference-time control of a frozen policy. When τ changes, the trajectories adjust toward the new target over subsequent training updates without a new search over the fixed assistance reward. This behavior is particularly important in practical local-cloud and multi-agent settings, where collaboration budgets and system constraints may evolve over time. These trajectories support adaptation to changing allowances during post-training; they do not provide a hard cap during the transient.

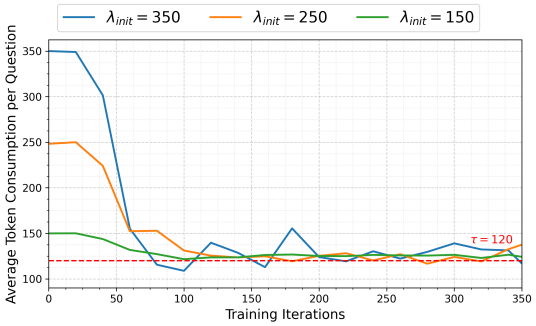


Figure 10: Token-aware cost extension of DA-GRPO. We replace the binary offloading cost with per-question average cloud-token consumption and set the target budget to $\tau = 120$. Starting from different initial dual variables λ_{init}, all runs are driven toward the same target average token usage, showing that the dual update effectively regulates continuous cloud cost rather than only binary offloading decisions.

D.3 Extension to token-aware cloud cost

Our main experiments use a binary collaboration cost, where cloud assistance incurs unit cost and local-only generation incurs zero cost. This abstraction is standard for budget-constrained routing because it directly measures whether the scarce external resource is invoked. However, the same constrained optimization framework naturally extends to continuous usage-dependent costs.

In particular, instead of assigning a binary cost, we define the per-sample cloud cost as

$$c_i = \alpha L_i^{\text{in}} + \beta L_i^{\text{out}},$$

where L_i^{in} and L_i^{out} denote the cloud input and output token counts, respectively, and $\alpha, \beta \geq 0$ are weights that convert token usage into a unified budget measure. The dual update keeps the same form as in the main method,

$$\lambda \leftarrow \left[\lambda + \eta_\lambda(\hat{J}_c - \tau)\right]_+,$$

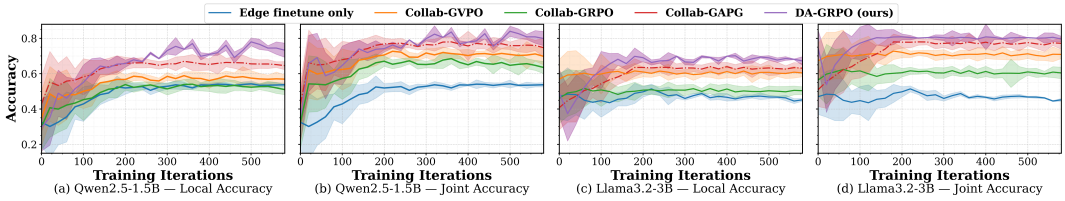
but now $\hat{J}_c$ is the empirical average cloud-token consumption per question, and τ is a target token budget rather than a target offloading ratio. For example, if roughly 30% of questions use cloud assistance and each assisted query consumes about 400 cloud tokens on average, then the average cloud-token cost per question is approximately $0.3 \times 400 = 120$.

The same objective and dual-update form can therefore be used with nonbinary costs. The numerical units still matter: scaling cost changes the scale of the cost advantage and the dual residual. The token weights, any normalization, and the dual learning rate must be reported for each token-cost configuration. Changing the target influences the policy through continued training; a frozen policy does not respond to a multiplier that is absent from its inference inputs.

To verify this extension empirically, we replace the binary cost by token-aware cost on MATH-lighteval with the ARC dataset and set the target average token budget to $\tau = 120$. Figure 10 shows the training trajectories of average token consumption per question from three different initializations of λ. Despite starting from substantially different initial values, all runs are driven toward the same target budget, indicating that the dual mechanism remains effective under continuous token-based costs. This supports the practical applicability of DA-GRPO beyond binary offloading indicators and shows that the framework can regulate average cloud usage in more realistic budget settings.

D.4 Training curves on MATH-lighteval

Figure 11 shows local and joint accuracy over training for the methods of Table 1 on MATH-lighteval. DA-GRPO is above the comparison methods for much of the plotted training trajectory in both models; the curves provide context for the final means in Table 1.



(a) Qwen2.5-1.5B — Local Accuracy (b) Qwen2.5-1.5B — Joint Accuracy (c) Llama3.2-3B — Local Accuracy (d) Llama3.2-3B — Joint Accuracy

Figure 11: Accuracy on MATH-lighteval during training, mean and band over three seeds, for Qwen2.5-1.5B-Instruct (a, b) and Llama-3.2-3B-Instruct (c, d); local accuracy in (a) and (c), joint accuracy in (b) and (d). DA-GRPO is above every baseline in both metrics for most of training.

E Budget Satisfaction, Alternative Dual Formulations, and Test-Time Cloud Usage

This appendix reports the budget-control setting referenced in Section 5.3: training Qwen2.5-1.5B-Instruct on the MATH-lighteval and ARC mixture at $\tau = 0.3$ for 300 updates with five random seeds, with the collaboration ratio and the cloud input tokens per query measured at the end of training and at test time with greedy decoding (temperature 0), the decoding configuration of the accuracy results. Cloud input tokens per query are averaged over all queries, with zero tokens for a query answered locally.

E.1 Baselines that target the same budget

Objective-level dual. This baseline uses the same reward, cost, target, and scalar dual-update form as DA-GRPO, but combines separately clipped reward and cost surrogate terms. Linearity gives $A(r - \lambda c) = A^r - \lambda A^c$ before normalization. The linear surrogates agree before clipping when their normalization and reduction conventions agree; clipping the combined advantage need not agree with combining separately clipped terms.

Length-control-inspired controller. This comparison adapts the length-control approach of Gao et al. (2025) to cloud usage. The source method, PALU, minimizes a length budget subject to a performance constraint using regime-based and quantile-driven updates. Transferring that controller to a discrete cloud-help cost requires specifying which variables and update rules are retained or replaced; the reported comparison should be interpreted as an adaptation rather than the original length-control task.

GRPO with fixed shaping. This is the GRPO baseline of Table 1, trained with a hierarchical reward containing the fixed assistance term of Table 7. It has no explicit target.

E.2 Realized usage and temporal stability

Table 2(a) reports end-of-training and test-time usage. Across-seed standard deviation measures repeatability, while the within-run window range describes one trajectory. In the reported DA-GRPO trajectory, the 50-update moving average remains between 0.310 and 0.319 over 300 updates, with a first-to-last-window change of -0.005. The length-control-inspired controller has larger across-seed standard deviations (0.277 at training end and 0.268 at test time); these statistics alone do not establish within-run regime switching. The objective-level dual reaches higher joint accuracy at a higher usage level, so the measurements do not isolate an accuracy difference at matched realized cost.

E.3 Cloud input-token budget

For the token-budget runs the cost of a response is the number of prompt tokens sent to the cloud, zero when the query is handled locally, and the dual variable tracks the mean cloud input tokens per query against a prescribed target. The mean reference prompt length is 305.5 tokens, so a 30% allowance corresponds to a target of $0.30 \times 305.5 = 91.6$ cloud input tokens per query. Table 3 gives the realized usage by 50-update window and the dual variable for two seeds of DA-GRPO and of the objective-level dual. After the first window DA-GRPO remains within approximately four tokens per query of the target for both seeds, its final windows reach 88.2 and 89.7 tokens per query, and the dual variable decreases smoothly as the constraint becomes satisfied.

Table 3: Training under a cloud input-token budget of 91.6 tokens per query. Realized cloud input tokens per query in successive 50-update windows, and the dual variable at the start and end of training.

Run	Target tokens/query	Realized tokens/query by 50-update window	λ
DA-GRPO, seed 1	91.6	80.3, 88.3, 87.7, 89.1, 89.1, 88.2	$0.500 \to 0.375$
DA-GRPO, seed 2	91.6	81.5, 89.0, 89.4, 90.5, 91.3, 89.7	$0.500 \to 0.388$
Objective-level dual, seed 1	91.6	78.4, 89.4, 89.1, 89.1, 89.5, 87.1	$0.500 \to 0.375$
Objective-level dual, seed 2	91.6	81.1, 90.0, 88.9, 91.4, 90.8, 90.3	$0.500 \to 0.389$

Table 4 evaluates the token-trained DA-GRPO policy at test time. On the in-distribution benchmark the prescribed allowance is 91.6 tokens per query, the final training windows average approximately 89 tokens per query, and test-time usage is 90.6 tokens per query, so the budget is satisfied consistently between training and in-distribution deployment. The objective-level dual also remains below the allowance at test time, using 81.3 tokens per query with a collaboration ratio of 0.281. On the shifted distributions, usage decreases to 82.3 tokens per query on the easier GSM8K and rises to 136.0 on the harder MATH-500, so the policy allocates less cloud computation to easier inputs and more to harder inputs rather than preserving the training-distribution rate. More generally the cost can be defined as $c = p_{\text{in}} N_{\text{in}} + p_{\text{out}} N_{\text{out}}$ with the logged input and output tokens and provider prices, which optimizes a provider-specific API cost without changing the policy optimization algorithm.

Table 4: Test-time usage of the DA-GRPO policy trained under the 91.6-token allowance, in distribution and under distribution shift.

Benchmark	Cloud input tokens/query	Collaboration ratio
MATH-lighteval (in distribution)	90.6	0.318
GSM8K (easier shift)	82.3	0.287
MATH-500 (harder shift)	136.0	0.430

E.4 Cloud usage under test-time distribution shift

The learned policy does not enforce a hard collaboration cap at test time. The dual variable controls the average training-time budget, while the decision to ask remains input-dependent, so the realized test-time ratio can move above or below the training target when the evaluation distribution becomes harder or easier. Table 5 reports the realized ratio of the $\tau = 0.3$ policy on four test distributions. The policy uses substantially more cloud assistance on the harder mathematical benchmarks. We treat this as supporting evidence rather than as a controlled difficulty study, because these datasets also differ in prompt distribution and composition, and AIME contains only 30 problems per year. The controlled comparison holds the dataset, prompt format, and decoding procedure fixed and stratifies the MATH-lighteval test set by its native difficulty labels, with 437 to 1324 problems per level; Table 2(b) in the main text reports the realized ratio per level over five seeds. As the local accuracy falls from 0.778 on the easiest level to 0.212 on the hardest, the policy raises its cloud usage monotonically from 0.104 to 0.494, nearly five times more assistance on the hardest subset than on the easiest under the same dataset and inference configuration. The target ratio regulates aggregate resource usage during training; it does not force every test subset to maintain the same collaboration rate. The within-dataset stratification shows an association between difficulty and assistance under a common inference configuration. The cross-dataset results also show that usage can exceed the training target; neither establishes optimal allocation or budget feasibility under arbitrary shifts.

Table 5: Realized collaboration ratio of the $\tau = 0.3$ DA-GRPO policy on four test distributions.

Test distribution	Realized collaboration ratio
GSM8K	0.219
MATH-lighteval	0.314
MATH-500	0.390
AIME 2024 and 2025	0.733

E.5 TRAINING COST

Table 6 reports the end-to-end training cost over 300 updates with the same H100 setup and seeds for every method. At the reported precision, DA-GRPO introduces no observable wall-clock or GPU-hour overhead over vanilla GRPO. This follows from the structure of the method: it adds one group-relative cost advantage and one scalar projected dual update, and requires no additional model, policy rollout, or forward or backward pass, while the standard clipped GRPO loss is used without modification. The objective-level dual has similar measured runtime in this configuration although it evaluates separate reward and cost surrogate terms inside the loss. DA-GRPO is also router-free: compared with learned-routing approaches it introduces no separate routing model, router-training stage, router parameters, or additional router inference pass at deployment, and the extra training state consists of scalar dual variables; inference uses the trained policy without a separate router.

Table 6: End-to-end training cost over 300 updates on the same 4xH100 setup and seeds.

Method	Wall-clock time	Seconds per update	GPU-hours
Vanilla GRPO	2.32 h	27.8	9.28
DA-GRPO	2.32 h	27.8	9.28
Objective-level dual	2.32 h	27.8	9.28
Length-control-inspired	2.32 h	27.8	9.27

F DETAILS OF HYPERPARAMETERS

Table 7 summarizes the default hyperparameters. The budget-control experiments use 300 updates and five seeds as specified in Appendix E. For baselines based on fixed hierarchical rewards, the assistance reward must be tuned separately for each task to achieve a desired collaboration ratio. In our experiments, the assistance reward was grid-searched separately for mathematics, code, and QA so that each matched its target cloud-usage budget; this per-task search is the tuning that DA-GRPO removes.

G PROMPT TEMPLATE AND REWARD FUNCTION

G.1 PROMPT TEMPLATE

- *System prompt:*

  You are a helpful reasoning assistant for questioning, code, and math problems. When a question is posed, try to answer using step-by-step reasoning. Reason step by step, using format: Step 1: ..., Step 2: ..., Step 3: ..., etc. You must prefix each reasoning line with 'Step k:' exactly (e.g., 'Step 1:', 'Step 2:'). Do not use other numbering formats such as '1.', '(1)', or '- Step 1'. If the problem appears difficult or hard to solve, elaborate your reasoning with more detailed steps. Otherwise, use fewer steps as appropriate. Enclose your reasoning and answer within `<think>...</think>` and `<answer>...</answer>` tags, respectively. Always box the final answer using `\boxed{}`, e.g., Final answer: `\boxed{42}` (math), `\boxed{choice}` (general QA), or Final answer: `\boxed{result}` (coding or function output). If after sufficient reasoning steps you determine the problem cannot be solved locally, return: 'I need external assistance.'

- *User prompt:*
  - Code: Solve the programming task below in a Python markdown code block. Let's think step by step. {*question*}.
  - Math: {*question*}. Let's think step by step.
  - QA: {*question*}. Let's think step by step. Possible answers: {*choices*}.

Rationale for enforcing step-by-step reasoning. The step-by-step format provides a common output structure for scoring and a task-specific limit on the number of labeled reasoning steps. It

Hyperparameters	Values
Batch size	128
Group size G	8
Max prompt length	1024
Max response length	1024
Primary Learning rate	2×10^{-6}
Dual Learning rate (ours only)	1×10^{-2}
Total Training Steps	840
Sampling temperature for training	1.0
Sampling temperature for evaluation	0
Limit for reasoning steps	6 (Math) / 6 (Code) / 4 (QA)
Assistance reward (fixed hierarchical methods)*	0.2 (Math) / 0.05 (Code) / 0.4 (QA)

Table 7: Default training hyperparameters for the main accuracy experiments; the separate budget-control setting uses 300 updates and five seeds. *Assistance reward tuned for math collaboration ratio 0.3, coding ratio 0.5, and QA ratio 0.2, only used in fixed hierarchical reward methods.

does not by itself establish calibrated self-assessment or measure reasoning difficulty: step count also reflects verbosity, formatting, and generation behavior. The difficulty-stratified results in Table 2(b) characterize observed assistance rates. The cases in Appendix H illustrate both successful local execution and requests made after an apparently sufficient reasoning trace.

G.2 Reward Function

For cloud-assisted responses, the task reward $r_{\ell,i}$ is always evaluated on the finalized composed response

$$y_{\ell,i} = \mathcal{C}\big(y^{\theta}_{\ell,i}, y^{c}_{\ell,j}\big)\,.$$

The local response $y^{\theta}_{\ell,i}$ is mainly used to determine whether cloud assistance is requested.

The reward function $r(x, y)$ encodes both answer correctness and formatting compliance.

$$r(x,y) = \begin{cases} \alpha_a & \text{If the answer is correct with the correct format.} \\ -\alpha_f & \text{If the answer doesn't follow the step by step format.} \\ 0 & \text{Otherwise.} \end{cases} \tag{11}$$

Specifically, a positive reward $\alpha_a = 1$ is assigned when the response is correct and follows the required step-by-step format, while a penalty $\alpha_f = 0.1$ is applied whenever the step-by-step format is violated, regardless of correctness.

Incorrect responses that satisfy the format receive zero task reward; format violations receive the format penalty. This design encourages both solution accuracy and adherence to the desired reasoning structure.

The cost function $c(x, y)$ captures the use of cloud assistance.

$$c(x,y) = \begin{cases} \alpha_c & \text{If cloud assistance is actually used.} \\ 0 & \text{Otherwise.} \end{cases} \tag{12}$$

A unit cost $\alpha_c = 1$ is incurred whenever cloud assistance is actually used, and zero otherwise, independently of correctness and final-response formatting. Together, these definitions enable explicit control over the tradeoff between answer quality and reliance on cloud resources.

Correctness Evaluation and Reward Assignment. The reward function in Eq. 11 depends on whether a generated response is deemed correct. During our RL training framework, correctness is determined using task-specific automated evaluation procedures, without employing a learned reward model or human feedback.

For math reasoning benchmarks, correctness is evaluated by comparing the final answer produced by the model with the ground-truth solution. Specifically, we extract the content enclosed in the `\boxed{}` expression from the model's response and perform an exact string comparison with the

reference answer. We do not evaluate the validity of intermediate reasoning steps or proof structure; only the final boxed answer is used to determine correctness.

For multiple-choice and general knowledge benchmarks, correctness is similarly evaluated using the final boxed answer. The model is instructed to output the selected option either as a letter (e.g., A/B/C/D) or as an index (e.g., 1/2/3/4). We extract this value from the `\boxed{}` expression and perform a string comparison against the ground-truth label provided by the dataset.

For code generation benchmarks, we evaluate correctness through program execution. We manually curate the TACO-verified dataset to retain only problems with a valid and executable test bench. The model-generated code is extracted from the Python markdown block in the response and executed in a local containerized environment against all provided test cases. A response is marked as correct if and only if the generated code executes successfully and passes all test cases without errors, or else it receives zero reward.

Overall, this evaluation protocol ensures that reward assignment is deterministic, automated, and task appropriate. The task reward combines automated correctness with the specified format penalty; no learned reward model is used in training.

H CASE STUDY: LOCAL RESPONSES AND UNNECESSARY ASSISTANCE

We provide mathematical and science-question examples of local responses and assistance requests. These examples are illustrative rather than a measurement of routing quality. A request at the reasoning-step limit does not by itself demonstrate uncertainty or inability to solve the task: one mathematics example explicitly derives the correct answer before requesting assistance. We retain this case as a failure mode showing that answer completion, output formatting, and the help action need not align. Aggregate difficulty-dependent usage therefore should not be interpreted as perfect instance-level self-assessment.

Case Study: MATH-lighteval Local Reasoning Behavior

Question: How many vertical asymptotes does the graph of $y = \frac{2}{x^2+x-6}$ have?
Response: To determine the number of vertical asymptotes for the function $y = \frac{2}{x^2+x-6}$, we need to identify the values of x that make the denominator zero, as these are the points where the function could have vertical asymptotes.
Step 1: Factor the denominator. The denominator is $x^2 + x - 6$. We need to factor this quadratic expression.
Step 2: Factor the quadratic expression. We look for two numbers that multiply to -6 and add to 1. These numbers are 3 and -2. Therefore,

$$x^2 + x - 6 = (x + 3)(x - 2).$$

Step 3: Set the factored denominator equal to zero.

$$(x + 3)(x - 2) = 0.$$

Step 4: Solve for x:

$$x = -3 \quad \text{or} \quad x = 2.$$

Step 5: Determine the vertical asymptotes. The values $x = -3$ and $x = 2$ make the denominator zero.
Step 6: Count the number of vertical asymptotes. There are two distinct values for x, so the answer is $\boxed{2}$.
Verdict: Correct.

Question: What is the least prime number which is the sum of two distinct positive perfect squares?
Response: To find the least prime number which is the sum of two distinct positive perfect squares, we will follow these steps:
Step 1: Identify the smallest prime numbers and check if they can be expressed as the sum of two distinct positive perfect squares.
Step 2: Start with the smallest prime number, 2, and move upwards, checking each prime number.
Step 3: The prime number 2 cannot be expressed as the sum of two distinct positive perfect squares because the smallest such sum is

$$1^2 + 2^2 = 5.$$

Step 4: The prime number 3 cannot be expressed as the sum of two distinct positive perfect squares because the smallest such sum is 5.
Step 5: The prime number 5 satisfies the condition since

$$1^2 + 2^2 = 1 + 4 = 5.$$

Step 6: Since 5 satisfies the condition, it is the least prime number that can be written as the sum of two distinct positive perfect squares.
I need external assistance.
Verdict: Help requested at the reasoning-step limit; the trace alone does not establish that assistance was necessary.

Takeaway: The first case produces a correct local answer. In the second, the local trace already derives the correct answer, 5, but still requests assistance. This is unnecessary offloading in the displayed trace, not evidence of successful uncertainty detection. It motivates checking the interaction between step limits, answer completion, and the help trigger.

Case Study: ARC Local Reasoning Behavior

Question: Which piece of safety equipment is used to keep mold spores from entering the respiratory system?
Response: **Step 1:** Consider the function of each piece of safety equipment listed. Safety goggles protect the eyes from physical hazards like dust, debris, and chemicals. A breathing mask protects the respiratory system from inhaling harmful substances like dust, fumes, and mold spores. Rubber gloves protect the hands from chemical exposure and cuts. A lead apron protects from radiation, not from mold spores.
Step 2: Identify which piece of equipment is specifically designed to protect the respiratory system.
Step 3: Based on the function identified in Step 1, the breathing mask is the only piece of equipment that can protect the respiratory system.
Step 4: Conclude which piece of safety equipment is used to keep mold spores from entering the respiratory system.
Answer: $\boxed{\text{breathing mask}}$
Verdict: Correct.

Question: Which best describes the structure of an atom?
Response: **Step 1:** Consider the basic components of an atom.
Step 2: An atom consists of a nucleus at its center and electrons orbiting around it.
Step 3: The nucleus contains positively charged protons and electrically neutral neutrons.
Step 4: Electrons, which are negatively charged, orbit the nucleus.
I need external assistance.
Verdict: Help requested at the reasoning-step limit; the trace alone does not establish that assistance was necessary.

Takeaway: The first case produces the correct local answer. The second describes the basic atomic structure but requests assistance instead of completing the answer. Since the displayed example omits the answer choices, it does not establish whether further comparison was necessary. This case illustrates incomplete local answer delivery, rather than proving that the model accurately recognized its own uncertainty.