PACO-PO: POLICY-ADAPTIVE COST-CONSTRAINED POLICY OPTIMIZATION FOR JOINT REASONING EFFORT AND RETRIEVAL DEPTH IN BUDGETED SINGLE-TURN RAG

Anonymous authors
Paper under double-blind review

ABSTRACT

Single-turn retrieval-augmented question answering must decide, for every query, how hard to reason and how deeply to retrieve, and providers charge both the generated reasoning and the retrieved passages to one token budget. However, existing controllers tune the two levers apart, length control for reasoning effort and retrieval control for depth, so they miss that strong retrieval substitutes for reasoning while weak retrieval demands more of it, and cost-aware training bills reasoning alone, through a reward weight that must be swept or a per-query target length. We propose PACO-PO, a controller of roughly 0.2M parameters that, with the base LLM and retriever frozen, treats reasoning effort and retrieval depth as one coupled budgeted decision and learns it as a factorized joint effort–depth policy whose depth head is conditioned on the selected effort. The policy reads a zero-cost retrieval preview, the retriever's score profile and the passage lengths, to judge whether retrieval already suffices before any passage enters the context, so the choice itself spends no LLM tokens. Training casts the choice as a one-step constrained Markov decision process optimized by group-relative policy optimization, where a single Lagrangian dual variable prices the token budget, so one training run replaces a reward-weight sweep. On HotpotQA, Natural Questions, and MATH-500 with retrieval, PACO-PO comes within one point of the best fixed configuration (46.9% versus 47.6% macro accuracy) at 64% of its token cost, exceeds the best budget-matched single-axis controller by 2.9 points, and keeps the macro training-time expected cost within 3% of the budget.

1 INTRODUCTION

Retrieval-augmented question answering with large language models (LLMs) is billed per token, and the two levers that most improve an answer, the reasoning the model generates and the passages it retrieves, both draw on that bill (Chen et al., 2024; Ding et al., 2024; Gao et al., 2023). How much reasoning effort and retrieved context each query receives therefore sets both the accuracy a deployment reaches and the tokens it spends (Besta et al., 2025). The right amount varies by query: a decisive passage lets a short reasoning pass answer a factual question, while a multi-step problem gains more from thinking than from more passages. A fixed allocation thus overspends on easy queries and starves hard ones.

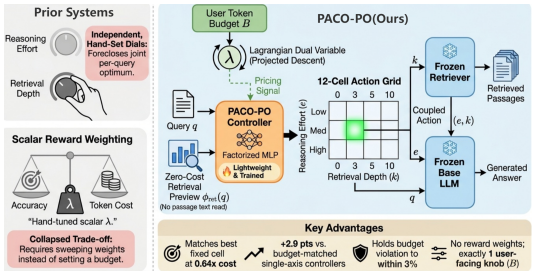

Figure 1: **Left:** prior systems tune the two dials separately, or trade accuracy against cost by a hand-tuned weight. **Right:** PACO-PO picks both as one joint action under one budget B, priced by a dual variable λ.

Three lines of work have made per-query control of this spend practical. Surveys of test-time reasoning and of retrieval augmentation lay out how hard a model thinks and how much context it reads as two controllable axes (Besta et al., 2025; Gao et al., 2023). Adaptive procedures decide whether

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.

to retrieve at all (Jiang et al., 2023), restructure the question before answering (Zheng et al., 2024), or route each query to a retrieval strategy by its predicted complexity (Jeong et al., 2024), recovering accuracy that a uniform pipeline leaves behind. Reinforcement-learning methods train the model to spend fewer reasoning tokens, through a length penalty weighted in the reward (Arora & Zanette, 2025) or a target reasoning length stated in the prompt (Aggarwal & Welleck, 2025). Together these methods show that spending fewer tokens need not cost accuracy, yet each fixes part of the allocation before the query arrives.

However, the three lines split the allocation in the way Figure 1 (left) depicts. Tuning one axis at a time misses the interaction between them: the depth a query can exploit depends on the reasoning it is given, and the reasoning that pays off depends on the passages it reads. Hand-set switches choose among coarse options and decide by running the model, spending the very tokens the budget is meant to save. Reward-based cost control prices reasoning alone, through a swept weight or a per-query target length, and leaves retrieved context unbilled, although, as Figure 2 shows, the provider charges both to one per-query budget. Reasoning effort and retrieval depth are therefore not two dials to tune apart but two coordinates of one budgeted decision, and the quantity a deployer can state is the token budget itself, not an exchange rate between accuracy and cost.

To address this problem, we propose PACO-PO (Policy-Adaptive Cost-Constrained Policy Optimization), a lightweight controller that, with the base LLM and retriever frozen, allocates reasoning effort and retrieval depth per query as one decision under a single expected token budget, as Figure 1 (right) shows. PACO-PO has three components: a factorized joint policy that emits effort and depth as one coupled action; a retrieval preview, read from retriever scores and index metadata, that informs the choice before any passage enters the context and spends no LLM tokens; and one Lagrangian dual variable that turns the user's budget into a token price during training. The key insight is that charging reasoning and retrieved context to one priced budget turns the substitution between them into a per-query choice that a small controller can learn from signals available before generation. Our contributions are as follows:

- We identify per-query effort–depth allocation as one coupled decision under one token budget, which existing methods split across separate axes, hand-set rules, and penalty weights.
- We propose PACO-PO, which casts this decision as a one-step constrained Markov decision process, learns a factorized effort–depth policy from zero-cost retrieval previews, and prices the user's budget with one dual variable, so one training run replaces a weight sweep.
- On HotpotQA, Natural Questions, and MATH-500, PACO-PO nearly matches the best fixed cell (46.9% versus 47.6% macro) at 64% of its cost, beats the best budget-matched single-axis controller by 2.9 points, and holds macro training-time cost within 3% of budget.

2 MOTIVATION

To study how reasoning effort and retrieval depth should be allocated per query under a token budget, we conduct a series of measurements and identify three challenges.

Challenge #1: Reasoning effort and retrieval depth form one coupled decision. Surveys of test-time reasoning and of retrieval augmentation lay the two out as separable dials (Besta et al., 2025; Gao et al., 2023). This separability is doubtful: on knowledge-heavy queries a correct passage at a shallow depth lets a low-effort pass answer directly, so retrieval substitutes for reasoning; on reasoning-heavy queries retrieved text helps less and thinking helps more, and depth cannot be maximized freely: accuracy depends on where a passage sits in the context (Liu et al., 2024a) and falls as input grows (Hsieh et al., 2024). Figure 2 (left) measures the ceiling this imposes: the static effort–depth cells trace a diminishing curve whose best cell reaches 48.2% on HotpotQA, yet a per-query oracle reaches 56.0%, a 7.8-point gap (8.5 on NQ, 3.5 on MATH-500) that no fixed cell can close. *How can one policy choose effort and depth jointly, each in light of the other?*

Challenge #2: A cheap query-side signal, not the passage text, must drive the choice. A second assumption is that a fixed rule, whether to retrieve (Jiang et al., 2023) or how to reframe the question (Zheng et al., 2024), captures the headroom. But a coarse switch cannot express the graded choices different queries demand, and reading passages to decide would spend the tokens the budget aims to save. What a policy can afford is a cheap preview: the retriever's score profile, available before any passage is consumed. Figure 2 (middle) quantifies both halves: the retrieve-or-not binary gate and the

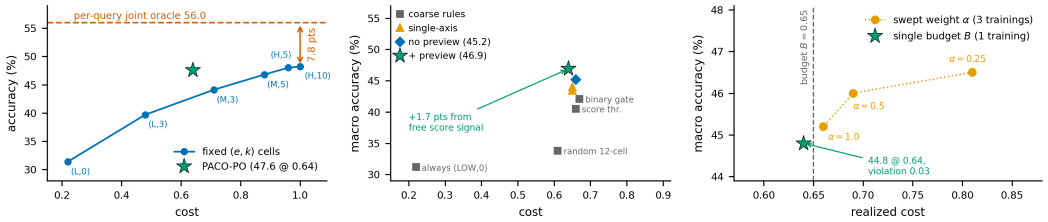

Figure 2: Operating points for the three challenges. **Left:** static cells versus a per-query oracle. **Middle:** coarse and single-axis control versus the joint policy. **Right:** swept weights versus a budget.

score-threshold heuristic stall at 42.1% and 40.5% macro accuracy, single-axis learned controllers reach 43.4–44.0%, while the graded joint policy reading the zero-cost preview reaches 46.9% at a lower cost, and removing the preview forfeits 1.7 points at the same budget. A sharp score profile signals a decisive passage (retrieval substitutes for effort), a flat profile signals ambiguity (added passages inject noise), and a graded policy acts on that distinction at a finer grain than the coarse switches tested here. *From what token-free signal should a graded joint choice be driven?*

Challenge #3: Practitioners set a budget, not an exchange rate. Cost-aware policy optimization either subtracts a weighted token penalty, asking the deployer for an exchange rate (Arora & Zanette, 2025), or trains the model to hit a per-query target reasoning length (Aggarwal & Welleck, 2025). The latter is a budget interface, but it caps reasoning alone and leaves retrieved context uncharged. A retrieval-augmented deployment pays for both from one allowance, in expectation per query. Figure 2 (right) shows the consequence: sweeping the weight α over $\{0.25, 0.5, 1.0\}$ scatters realized costs from 0.81 to 0.66 (macro over HotpotQA and NQ), so the weight has to be swept before one setting lands near the intended spend, whereas setting the budget $B = 0.65$ of the best-cell cost yields 0.64 cost in a single training run, with the three-benchmark macro budget violation held at 0.03, the price found by one dual variable. *How can training price a given token budget automatically instead of asking for a weight?*

3 Problem Formulation

To address these challenges, we formulate per-query allocation as a constrained optimization problem: a controller policy π_θ maximizes expected accuracy,

$$\max_\theta \; J(\theta) = \mathbb{E}_{q\sim\mathcal{D},\, a\sim\pi_\theta(\cdot|q)}\big[r(q,a)\big], \tag{1}$$

subject to an expected per-query token budget,

$$J_C(\theta) = \mathbb{E}_{q\sim\mathcal{D},\, a\sim\pi_\theta(\cdot|q)}\big[c(q,a)\big] \le B, \tag{2}$$

where q is a query drawn from a distribution $\mathcal{D}$ and answered with a frozen retriever and a frozen base LLM; $a = (e, k) \in \mathcal{E} \times \mathcal{K}$, with $\mathcal{E} = \{\text{LOW}, \text{MED}, \text{HIGH}\}$ and $\mathcal{K} = \{0, 3, 5, 10\}$, is one joint action on the resulting 12-cell grid, with the effort level e capping the reasoning tokens and the depth k setting how many passages are retrieved; $r(q, a) \in \{0, 1\}$ is the correctness of the answer under the official metric; $c(q, a)$ is the total tokens the query consumes across prompt, retrieved passages, reasoning, and answer; and B is the token budget. The program is a one-step constrained Markov decision process: the query is the context, the 12-cell grid the action set, and B the single constraint.

To address Challenge #1, the policy π_θ selects effort and depth as one joint action over the effort–depth grid. To address Challenge #2, π_θ reads only query-side signals and token-free retrieval metadata, never the passage text. To address Challenge #3, the program carries a single constraint, the budget B, with no reward weight.

4 Solution

4.1 Overall Framework

To solve the constrained program above, we propose PACO-PO. Figure 3 shows its framework, which comprises three parts: (i) a factorized joint effort–depth policy that emits the two dials as one coupled action; (ii) a zero-cost retrieval-preview featurization that drives the choice before any

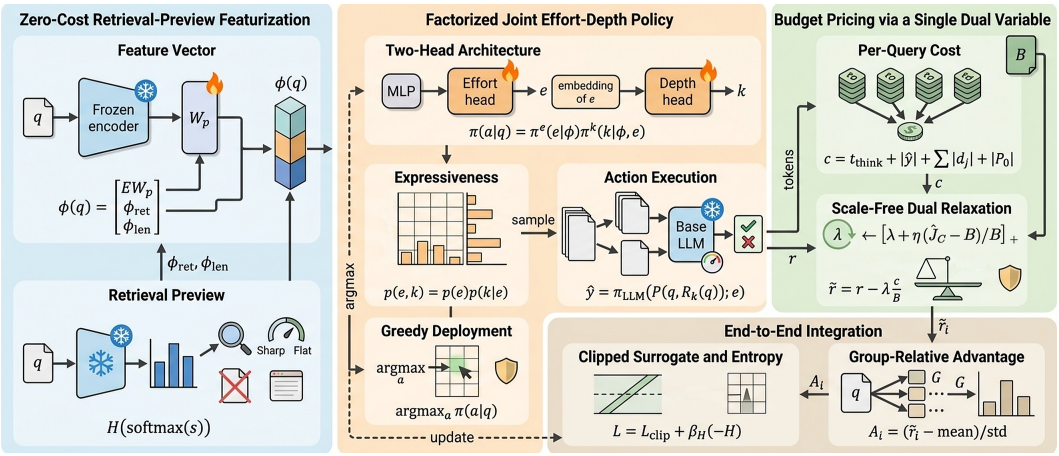

Figure 3: Overview of PACO-PO. **Left:** zero-cost retrieval preview. **Middle:** factorized joint effort–depth policy. **Right:** budget pricing via a single dual variable. **Bottom:** end-to-end integration.

passage text is read; and (iii) budget pricing that converts the budget B into a scale-free price through one Lagrangian dual variable. A final step assembles them into one training and deployment loop.

The three parts form one system: the preview supplies the signals the joint decision needs, the factorized head keeps the two axes coupled, and the dual variable charges every sampled action the same token price, so the policy learns where on the grid each query belongs *at the current price* of tokens. The base LLM and retriever are frozen; only the small controller is learned. Training is preceded by a probe on a held-out split, where all 12 fixed cells are evaluated once: it selects the best fixed cell, sets the token value of B as the chosen fraction of that cell's mean probe cost, fits the score normalization of the retrieval preview, and estimates the floor cost C_{floor} used for budget admissibility. Details in Appendix A.6.

4.2 Factorized Joint Effort–Depth Policy

To address Challenge #1, effort and depth must be chosen as one coupled decision, since any controller that fixes one axis without regard to the other generally settles below the joint optimum. Figure 3 (middle) shows how PACO-PO emits both dials as one factorized action on the grid.

Two-head architecture. The controller has two heads, each a small MLP, reading the feature vector $\phi(q)$ defined below: an effort head over $\mathcal{E}$, and a depth head over $\mathcal{K}$ that reads $\phi(q)$ concatenated with a learned embedding of the sampled effort level e:

$$\pi_\theta(a \mid q) = \pi^e_\theta(e \mid \phi(q)) \cdot \pi^k_\theta(k \mid \phi(q), e). \tag{3}$$

Conditioning depth on the sampled effort keeps the joint decision coupled.

Expressiveness. The factorization is fully expressive over the grid,

$$\forall p \in \Delta(\mathcal{E} \times \mathcal{K}): \quad p(e,k) = p(e)\, p(k \mid e), \tag{4}$$

where $\Delta(\mathcal{E} \times \mathcal{K})$ is the set of distributions over the 12 cells, so no joint distribution is lost relative to a flat 12-way softmax, while the effort-first order follows the substitution pattern of Challenge #1.

Action execution. An action is executed by the frozen retriever and base LLM π_{LLM}:

$$\hat{y} = \pi_{\text{LLM}}\big(P(q, R_k(q));\, e\big), \tag{5}$$

where $R_k(q)$ are the top-k passages of the retriever call that also produces the retrieval preview, $R_0(q)$ being empty, and P assembles the prompt. The effort level e sets the cap $u(e)$ on reasoning tokens, stated in the prompt and enforced as a decoder stop; the answer $\hat{y}$ then follows under a fixed answer cap $Y_{\max}$.

Greedy deployment. The trained controller is deployed unchanged and without the price, selecting the highest-probability cell of Eq. 3 over the 12 joint cells, which Eq. 5 then executes:

$$a^{\text{deploy}}(q) = \arg\max_{a} \pi_\theta(a \mid q). \tag{6}$$

4.3 Zero-Cost Retrieval-Preview Featurization

To address Challenge #2, a cheap query-side signal, not the passage text, must drive the choice, since reading passages to decide would spend the very tokens the budget aims to save. Figure 3 (left) shows the compact feature vector PACO-PO builds from the query and a cheap retrieval preview.

Feature vector. The controller reads

$$\phi(q) = \big[\, E_{\text{enc}}(q)\, W_p \,;\; \phi_{\text{ret}}(q) \,;\; \phi_{\text{len}}(q) \,\big] \in \mathbb{R}^{d_\phi}, \tag{7}$$

with three blocks. A frozen sentence embedding $E_{\text{enc}}(q)$, projected by a trained linear map W_p, encodes the query; a retrieval-preview block summarizes the retriever's score profile; and a length block exposes the exact retrieved-token count of each depth, the retrieval term of the cost in Eq. 10.

Retrieval preview. The preview features are computed from the retriever's score vector without reading any passage text, so they cost one retriever call and no LLM tokens:

$$\phi_{\text{ret}}(q) = \big[\, s_1(q),\; s_1(q) - s_{10}(q),\; \text{mean}_{j\le 10}\, s_j(q),\; \text{H}\big(\sigma(s_{1:10}(q))\big) \,\big], \tag{8}$$

where $s_j(q)$ is the j-th retrieval score, min–max normalized per benchmark with constants fit on the probe split, and $\text{H}(\sigma(\cdot))$ is the entropy of the softmax-normalized score profile. Read before any passage enters the context, the top score measures the strongest match, the top-to-tenth gap its separation from the tenth-ranked match, the mean the overall score strength, and the entropy how sharply the scores concentrate, which operationalizes the sharp-versus-flat distinction of Challenge #2. The length block passes the query length $|q|$ in tokens and the cumulative token lengths $\sum_{j\le k} |d_j(q)|$ of the retrieved passages $d_j(q)$ at each candidate depth $k \in \{3, 5, 10\}$, read from the index metadata,

$$\phi_{\text{len}}(q) = \big[\, |q|,\; \textstyle\sum_{j\le 3} |d_j(q)|,\; \sum_{j\le 5} |d_j(q)|,\; \sum_{j\le 10} |d_j(q)| \,\big]. \tag{9}$$

4.4 Budget Pricing via a Single Dual Variable

To address Challenge #3, the deployer must state a token budget instead of an exchange rate, and one training run must find the price that budget implies, with no weight sweep. Figure 3 (right) shows how PACO-PO converts the budget B into a scale-free price through one dual variable.

Per-query cost. The cost $c(q, a)$ of Eq. 2 charges the generated tokens and the prompt tokens alike:

$$c(q, a) = t^{\text{think}} + |\hat{y}| + \sum_{j\le k} |d_j(q)| + |P_0(q)|, \tag{10}$$

where t^{think} is the number of reasoning tokens actually generated, capped as $t^{\text{think}} \le u(e)$, $|\hat{y}|$ the answer length, and $|P_0(q)|$ the serialized prompt scaffold, the query and template tokens excluding retrieved passages. Passage lengths are known before the LLM call, so the retrieval component of the cost is deterministic given (q, k). Per-query cost is bounded by construction,

$$c(q, a) \le c_{\max} = u(\text{HIGH}) + Y_{\max} + D_{10} + P_{\max}, \tag{11}$$

where D_{10} is the maximum total length of ten passages and $P_{\max} = \max_q |P_0(q)|$ the largest scaffold; the bound keeps J_C finite for every policy, which the dual analysis of Appendix A.3 uses. The budget constrains the training-time expected cost; the deployed controller's spend is measured separately at evaluation.

Scale-free dual relaxation. We relax the constraint of Eq. 2 with a single Lagrangian dual variable, writing the constraint in scale-free relative form so the price does not depend on the token magnitude of B:

$$L(\theta, \lambda) = J(\theta) - \lambda\, \frac{J_C(\theta) - B}{B}, \qquad \lambda \ge 0. \tag{12}$$

Penalized reward and price step. PACO-PO approaches the saddle problem $\min_{\lambda \geq 0} \max_\theta L$ by scoring each sampled action at the current price λ_t of outer iteration t:

$$\tilde{r}(q, a) = r(q, a) - \lambda \frac{c(q, a)}{B}. \tag{13}$$

At a fixed price the expected penalized reward is $L(\theta, \lambda) - \lambda$, which differs from L only by a term constant in θ, so maximizing it is the inner problem of Eq. 12, which the group-relative clipped surrogate below approximates. The dual variable acts as a price: an action must earn λ units of accuracy to justify spending one budget's worth of tokens. After the policy epochs, the price moves once by a projected step on the batch estimate $\hat{J}_C = (N_q G)^{-1} \sum_i c_i$, the mean cost of the rollout batch of N_q queries with G actions each, drawn from the frozen $\pi_{\theta_{\text{old}}}$:

$$\lambda_{t+1} = \Big[\lambda_t + \eta_\lambda \frac{\hat{J}_C - B}{B} \Big]_+, \tag{14}$$

where η_λ is the dual step size and $[\cdot]_+$ projects onto $\lambda \geq 0$. Starting from zero, the price rises while the batch overspends and falls while it underspends; when the behaviour policy is an exact inner maximizer at λ_t, Eq. 14 is a stochastic projected subgradient step on the convex dual function $g(\lambda) = \max_\theta L(\theta, \lambda)$. The budget is required to be admissible, $B \geq 1.1\, C_{\text{floor}}$ with C_{floor} the probed expected cost of always playing $(\text{LOW}, 0)$, which on the probe sample gives the reference policy positive slack and bounds the optimal price. Details in Appendix A.3.

4.5 End-to-End Integration

Training loop. Figure 3 (bottom) shows the loop that trains the three parts jointly; one outer iteration t proceeds in order: the current controller is frozen as the behaviour policy $\pi_{\theta_{\text{old}}}$, its projection W_p included; N_q training queries are sampled uniformly without replacement; for each, $\phi(q)$ is formed from one retriever call and G actions are drawn i.i.d. from $\pi_{\theta_{\text{old}}}(\cdot \mid q)$ by Eq. 3; each action is executed once by Eq. 5, yielding r and c, and is scored by Eq. 13 at the price λ_t; the penalized rewards become group-relative advantages; E inner epochs of Eqs. 15 and 16 update θ against the frozen $\pi_{\theta_{\text{old}}}$; and one dual step of Eq. 14 closes the iteration on the mean cost of this batch, an estimate of the expected cost of $\pi_{\theta_{\text{old}}}$ on the training split. Details in Appendix A.6.

Group-relative advantage. Following group-relative policy optimization (Shao et al., 2024), the G penalized rewards $\tilde{r}_i$ of one query are centered and scaled within their group, $A_i = (\tilde{r}_i - \bar{r}_q)/(\sigma_q + \epsilon_A)$, with $\bar{r}_q$ and σ_q the group mean and population standard deviation and ϵ_A a small stabilizer. Because the setting is single-turn, each action's advantage reflects its own outcome: there is no trajectory-level credit dilution.

Clipped surrogate and entropy. Pooling the N_q groups into one batch indexed by i, with q_i the query of sample i, and writing $\rho_i(\theta) = \pi_\theta(a_i \mid q_i)/\pi_{\theta_{\text{old}}}(a_i \mid q_i)$ for the importance ratio and ε for the clip range, the policy loss is the clipped surrogate:

$$\mathcal{L}_{\text{clip}}(\theta) = -\,\mathbb{E}_i \Big[\min \big(\rho_i(\theta) A_i,\ \text{clip}(\rho_i(\theta), 1 - \varepsilon, 1 + \varepsilon)\, A_i \big) \Big]. \tag{15}$$

The full loss adds a regularizer on the joint action entropy with coefficient β_H,

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{clip}}(\theta) + \beta_H\, \mathbb{E}_q \big[-\mathcal{H}(\pi_\theta(\cdot \mid q)) \big], \tag{16}$$

and the controller is updated with Adam at learning rate η_θ. With only 12 actions and a small group size, the group can concentrate on a few cells early in training; the entropy term encourages all cells to stay visited while the dual price stabilizes.

5 Experiments

5.1 Implementation Details

Datasets. Three single-turn benchmarks stress the two axes differently: HotpotQA (Yang et al., 2018) (multi-hop, retrieval and reasoning interact), Natural Questions (Kwiatkowski et al., 2019) (single-hop, retrieval substitutes for reasoning), and MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024) with retrieval over MATH training solutions (reasoning-dominant). All methods share one BGE-base-en-v1.5 index (Xiao et al., 2024). Details in Appendix A.4.

Baselines. Table 1 compares PACO-PO with the best fixed grid cell and three adaptive controllers that also vary (e, k) per query: a confidence cascade (Chen et al., 2024), which escalates from the cheapest cell when answer confidence falls below a tuned threshold; a difficulty router (Ding et al., 2024), which maps a predicted difficulty bucket to a cell; and a LinUCB bandit (Li et al., 2010), which selects cells online from reward feedback. Details in Appendix A.4.

Evaluation metrics. Each (method, benchmark) cell reports accuracy (%, ↑), per-query cost relative to the best fixed cell (↓), and relative budget violation (↓), the deviation of expected cost from B relative to B, a training-time estimate for trained controllers and deployed spend otherwise; (n/a) marks the budget-free best fixed cell, and bold marks PACO-PO. Details in Appendix A.4.

Training. Effort caps reasoning at 128/1024/4096 tokens and depth is $k \in \{0, 3, 5, 10\}$; $B = 0.65\times$ the best-cell cost. The 0.2M-parameter controller reads a frozen MiniLM (Wang et al., 2020) embedding, with 3 seeds for the main comparison, 1 elsewhere. Details in Appendix A.6.

Hyperparameters. Each run takes 40 outer iterations of 32 queries with $G = 6$ actions per query and 2 inner epochs, using Adam at a constant learning rate of $3\mathrm{e}{-4}$, clip range 0.2, entropy coefficient 0.01, dual step $\eta_\lambda = 0.1$ from $\lambda_0 = 0$, and gradient-norm clip 1.0. Details in Appendix A.6.

5.2 Main Results

Main comparison. Table 1 compares PACO-PO with the best fixed cell and three adaptive controllers on the three benchmarks, and PACO-PO is the only method that stays close to the best fixed cell's accuracy on all three while cutting cost and holding the budget. At the macro level it reaches 46.9% against 47.6% for the best fixed cell at 0.64 of that cell's cost, and 46.9% against 45.5% for the strongest adaptive controller, the difficulty router, while spending less (0.64 against 0.73). The adaptive controllers change rank from one benchmark to the next because each reads a single signal. The confidence cascade tracks PACO-PO on the two factual QA sets but falls more than six points below the best fixed cell on MATH-500 while spending the most of any method there (0.82): answer confidence separates right from wrong on short factual answers but does not identify which math problems repay more computation, so the cascade escalates often without gaining accuracy. The difficulty router is the strongest baseline on MATH-500 (50.8%), where difficulty is visible in the problem statement, and drops to 39.2% on NQ, where short factual questions look equally easy and the query alone does not reveal whether retrieval will find the answer. The bandit explores online with no budget and records the highest macro violation (0.12) and lowest NQ accuracy. All three overspend, with deployed macro violations of 0.08 to 0.12 against PACO-PO's training-time 0.03: choosing effort and depth jointly under a priced budget keeps it ahead across benchmarks whose difficulty lives in different places.

Table 1: PACO-PO versus the best fixed cell and three adaptive controllers.

Benchmark	Method	Acc. (%) ↑	Cost ↓	Viol. ↓
HotpotQA	Best fixed cell	48.2	1.00	(n/a)
	Confidence cascade	47.4	0.75	0.07
	Difficulty router	46.5	0.72	0.09
	Bandit (LinUCB)	46.8	0.71	0.12
	PACO-PO	**47.6**	**0.64**	**0.03**
NQ	Best fixed cell	42.5	1.00	(n/a)
	Confidence cascade	41.6	0.70	0.05
	Difficulty router	39.2	0.68	0.06
	Bandit (LinUCB)	37.0	0.74	0.13
	PACO-PO	**41.9**	**0.63**	**0.04**
MATH-500	Best fixed cell	52.0	1.00	(n/a)
	Confidence cascade	45.9	0.82	0.13
	Difficulty router	50.8	0.79	0.11
	Bandit (LinUCB)	49.4	0.69	0.10
	PACO-PO	**51.3**	**0.66**	**0.02**
Macro	Best fixed cell	47.6	1.00	(n/a)
	Confidence cascade	45.0	0.76	0.08
	Difficulty router	45.5	0.73	0.09
	Bandit (LinUCB)	44.4	0.71	0.12
	PACO-PO	**46.9**	**0.64**	**0.03**

Budget-matched static mixture. Table 2 compares PACO-PO on HotpotQA with representative fixed cells and with the best query-independent mix of those cells at PACO-PO's spend. The static cells show sharply diminishing returns: moving from $(\mathrm{LOW}, 0)$ to $(\mathrm{MED}, 3)$ roughly triples the cost and lifts accuracy from 31.4% to 44.1%, whereas the last step from $(\mathrm{HIGH}, 5)$ to $(\mathrm{HIGH}, 10)$ buys only 0.2 points for four percent more cost. Because the curve is concave, a query-independent random mix can only reach the chord below it, and the best mix at a spend of 0.64, $(\mathrm{LOW}, 3)$ with $(\mathrm{MED}, 3)$, reaches 42.8%. PACO-PO reaches 47.6% at the same spend, 4.8 points above that mix and 0.6 points below the most expensive cell. Sitting above the static curve of the displayed cells,

Table 2: Representative fixed (e, k) cells, their best random mix at PACO-PO's spend, and PACO-PO on HotpotQA.

Configuration	Acc. (%) ↑	Cost ↓
Fixed (LOW, 0)	31.4	0.22
Fixed (LOW, 3)	39.7	0.48
Fixed (MED, 3)	44.1	0.71
Fixed (MED, 5)	46.8	0.88
Fixed (HIGH, 5)	48.0	0.96
Best fixed (HIGH, 10)	48.2	1.00
Mix of (LOW, 3), (MED, 3)	42.8	0.64
PACO-PO	**47.6**	**0.64**

Table 3: Ablation ladder, one change per row, macro over the three benchmarks.

Configuration	Acc. (%) ↑	Cost ↓	Viol. ↓
Best fixed cell (reference)	47.6	1.00	(n/a)
Score-threshold heuristic	40.5	0.66	0.14
Retrieve-or-not binary gate	42.1	0.67	0.12
Learned depth only (e = MED)	43.4	0.65	0.06
Learned effort only (k fixed)	44.0	0.65	0.05
Flat 12-way head (unfactorized)	45.6	0.66	0.05
Without retrieval preview	45.2	0.66	0.05
λ frozen at its final value	45.1	0.71	0.18
Tuned scalarization weight	46.3	0.68	0.09
PACO-PO (full)	**46.9**	**0.64**	**0.03**

PACO-PO owes this margin to choosing a cell per query, placing computation where a query repays it, rather than to mixing cheap and expensive cells. Details in Appendix A.5.

5.3 Ablation Studies

Factorized joint effort–depth policy. Table 3 builds the method up from hand-set rules one change per row, every trained row given the same budget, so each accuracy difference is attributable to one change. Learning even one axis beats hand-set control: the score-threshold heuristic and the retrieve-or-not gate stall at 40.5% and 42.1% while overspending the budget (violations 0.14 and 0.12), whereas a learned depth-only or effort-only controller reaches 43.4% and 44.0%. Learning both axes together adds a further 2.9 points over the stronger single axis, more than any earlier step of the ladder, which shows that the two axes cannot be tuned apart: the depth a query can exploit depends on the reasoning it is given. The flat 12-way head, which can express exactly the same joint distribution, reaches only 45.6%, 1.3 points below the factorized policy; since both reach the same actions, this margin comes from how the action is parameterized, with depth conditioned on effort in the way the substitution between retrieval and reasoning requires.

Zero-cost retrieval preview. Table 3 also tests the retrieval preview by removing the score-profile features from the full method while keeping the length block. Without them the controller falls from 46.9% to 45.2% at the same budget and spends slightly more (0.66 against 0.64), so accuracy and efficiency degrade together. The preview is thus what lets the controller judge whether retrieval already suffices: a sharp score profile signals that a shallow depth with little reasoning will do, and a flat one that more passages or more thought are worth their cost. Without that signal the controller has to hedge across queries, spending more for less, and the signal comes from the retriever's scores before any passage enters the context, at no LLM-token cost.

Budget pricing via a single dual variable. Table 3 finally tests budget pricing by training the full policy with λ frozen at the value dual descent converges to. With the same final price the outcome is clearly worse: 45.1% at 0.71 cost with a violation of 0.18, against 46.9% at 0.64 with 0.03 for the full method. What matters is therefore not the level of the price but that it moves with the policy: early in training the policy spends differently from the converged one, so a price that is right at the end is wrong along the way. The dual variable adjusts the price to the policy's measured spend as it changes, and this buys both budget compliance and accuracy.

Dual step size and group size. Table 4 addresses whether removing the reward-weight sweep merely moves the search onto the dual step size. Across step sizes of 0.02, 0.1, and 0.5 accuracy stays between 46.5% and 46.9%, so the accuracy PACO-PO reaches does not depend on the step size. What the step size governs is how closely spending tracks the budget: a small step lets the price lag the policy's changing spend (violation 0.11), and a large one overshoots (0.08). Growing the group from 4 to 6 improves both accuracy and compliance, with little change at 8, consistent with a group-relative advantage whose noise falls as the group grows. One step size therefore serves all three benchmarks without a search for accuracy, unlike a reward weight, which sets spend and accuracy together and must be swept.

Table 4: Sensitivity to the dual step size and group size, macro over three benchmarks.

Setting	Acc. (%) ↑	Viol. ↓
$\eta_\lambda = 0.02$	46.8	0.11
$\eta_\lambda = 0.1$	**46.9**	**0.03**
$\eta_\lambda = 0.5$	46.5	0.08
$G = 4$	46.4	0.05
$G = 6$	**46.9**	**0.03**
$G = 8$	47.0	0.03

Matched scalarization. Table 5 replaces the dual variable by a fixed reward weight α in $r - \alpha\, c/B$, trained at three settings, macro over HotpotQA and NQ. Raising the weight from 0.25 to 1.0 lowers cost from 0.81 to 0.66 and accuracy from 46.5% to 45.2%: accuracy follows spend, but the weight does not fix spend in advance. At matched cost the tuned weight and the dual nearly tie, 45.2% at 0.66 against 44.8% at 0.64, so pricing a budget costs no accuracy. The difference is the interface: the weight needs three runs before one lands near B, whereas setting B lands PACO-PO at 0.64 in one run, as a deployer setting a budget, not an exchange rate, needs. At about 2.1 GPU-hours per run and benchmark, the sweep triples compute. Details in Appendix A.5.

Table 5: Tuned scalarization weights versus the budget B on HotpotQA and NQ.

Controller	Acc. (%) ↑	Cost ↓	Training runs ↓
Scalarization, $\alpha = 0.25$	46.5	0.81	
Scalarization, $\alpha = 0.5$	46.0	0.69	3
Scalarization, $\alpha = 1.0$	45.2	0.66	
PACO-PO (dual, budget B)	**44.8**	**0.64**	**1**

Cross-backbone transfer. Table 6 tests whether the learned allocation is tied to its training backbone, deploying the controller unchanged after swapping Llama-3.1-8B-Instruct for the same-class Qwen2.5-7B-Instruct. Accuracy is largely retained, 46.2% against 46.9% macro, with a drop below one point on every benchmark: the controller reads a query representation and the retrieval preview rather than backbone internals, so which queries need more effort or more passages carries over to the new backbone. Cost rises from 0.64 to 0.69 of the source best-cell cost, so the replacement spends more tokens under the same caps while the price was fit to the source; the allocation transfers, and the cost calibration is what a backbone swap moves. Details in Appendix A.5.

Table 6: Controller transfer from Llama-3.1-8B to Qwen2.5-7B.

	Accuracy (%)		Cost	
Benchmark	Source	Swap	Source	Swap
HotpotQA	47.6	46.9	0.64	0.69
NQ	41.9	41.2	0.63	0.68
MATH-500	51.3	50.4	0.66	0.70
Macro	46.9	46.2	0.64	0.69

6 Related Work

Adaptive Retrieval and Reasoning Effort. Retrieval adapts to the query (Gao et al., 2023): active retrieval fires on a knowledge gap and routing picks a strategy by complexity (Jiang et al., 2023; Jeong et al., 2024), while long-context studies tie accuracy to passage position and length (Liu et al., 2024a; Hsieh et al., 2024). For effort, tree search and step-back abstraction reshape reasoning, and evaluations chart where deliberation stops paying off (Yao et al., 2023; Zheng et al., 2024; Besta et al., 2025; Shojaee et al., 2025). Each adapts one axis alone. PACO-PO emits both as one factorized action, chosen from a zero-cost retrieval preview and priced against one token budget.

Efficient Language-Model Inference. Speculative decoding and its tree and multi-head variants emit the same tokens faster (Leviathan et al., 2023; Chen et al., 2023; Miao et al., 2024; Cai et al., 2024; Wang et al., 2025; Xia et al., 2024), and pruning, quantization, cache eviction, and expert routing cut the cost of each token (Ma et al., 2023; Wang et al., 2024a; Zhang et al., 2023; Shazeer et al., 2017; Yi et al., 2025). Both lower the price of a token, not how many a query consumes. PACO-PO sets that number per query with the LLM and retriever frozen, so it composes with either.

Reinforcement Learning for Language Models. Group-relative policy optimization trains reasoning without a learned critic (Shao et al., 2024), process supervision refines the reward (Lightman et al., 2024), and surveys map this field (Cao et al., 2025; Uc-Cetina et al., 2023). Their scalar reward admits token cost only as a hand-set weight. PACO-PO runs this solver on a one-step constrained problem whose single dual variable prices the token budget in one run, with no weight sweep.

7 Conclusion

Reasoning and retrieved context share one token budget. PACO-PO spends it through one factorized effort–depth action, chosen from a zero-cost retrieval preview and priced by a single dual variable, so one training run replaces a weight sweep. On HotpotQA, Natural Questions, and MATH-500, it is within one point of the best fixed cell (46.9% versus 47.6% macro) at 64% of its token cost, exceeds the best budget-matched single-axis controller by 2.9 points, and holds macro training-time cost within 3% of budget. Effort and depth form one decision, learnable from a token budget.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. In *Advances in Neural Information Processing Systems 38*, 2025. doi: 10.52202/085713-2032.

Maciej Besta, Julia Barth, Eric Schreiber, Ales Kubicek, Afonso Claudino Catarino, Robert Gerstenberger, Piotr Nyczyk, Patrick Iff, Yueling Li, Sam Houliston, Tomasz Sternal, Marcin Copik, Grzegorz Kwasniewski, Jürgen Müller, Lukasz Flis, Hannes Eberhard, Zixuan Chen, Hubert Niewiadomski, and Torsten Hoefler. Reasoning language models: A blueprint, 2025.

Oscar Brown, Zhengjie Wang, Andrea Do, et al. Dynamic depth decoding: Faster speculative decoding for llms, 2024.

Tianle Cai, Yuhong Li, Zhengyang Geng, et al. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Yuji Cao, Huan Zhao, Yuheng Cheng, et al. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Trans. Neural Networks Learn. Syst.*, 36(6): 9737–9757, 2025. doi: 10.1109/TNNLS.2024.3497992. URL https://doi.org/10.1109/TNNLS.2024.3497992.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, et al. Accelerating large language model decoding with speculative sampling, 2023.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=cSimKw5p6R.

Felipe Leno da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *J. Artif. Intell. Res.*, 64:645–703, 2019. doi: 10.1613/JAIR.1.11396. URL https://doi.org/10.1613/jair.1.11396.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=02f3mUtqnM.

Ning Ding, Yujia Qin, Guang Yang, et al. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models, 2022.

Nan Du, Yanping Huang, Andrew M. Dai, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 2022. URL https://proceedings.mlr.press/v162/du22c.html.

Yunfan Gao, Yun Xiong, Xinyu Gao, et al. Retrieval-augmented generation for large language models: A survey, 2023.

Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: A survey. *Artificial Intelligence Review*, 55:895–943, 2022. doi: 10.1007/s10462-021-09996-w.

Zeyu Han, Chao Gao, Jinyang Liu, et al. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=lIsCS8b6zj.

Dan Hendrycks, Collin Burns, Saurav Kadavath, et al. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, et al. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024. doi: 10.18653/v1/2024.naacl-long.389.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, et al. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 7969–7992. Association for Computational Linguistics, 2023.

Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Lambert Mathias, Marzieh Saeidi, Veselin Stoyanov, and Majid Yazdani. Prompt-free and efficient few-shot learning with language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3638–3652, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.254. URL https://aclanthology.org/2022.acl-long.254/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7: 453–466, 2019. doi: 10.1162/tacl_a_00276.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Lihong Li, Wei Chu, John Langford, et al. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pp. 661–670. ACM, 2010. doi: 10.1145/1772690.1772758. URL https://doi.org/10.1145/1772690.1772758.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, et al. Let's verify step by step. In *Proceedings of the Twelfth International Conference on Learning Representations*, 2024.

Nelson F. Liu, Kevin Lin, John Hewitt, et al. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, et al. Online speculative decoding. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 31131–31146. PMLR / OpenReview.net, 2024b. URL https://proceedings.mlr.press/v235/liu24y.html.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/44956951349095f74492a5471128a7e0-Abstract-Conference.html.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, et al. Fine-tuning language models with just forward passes. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a627810151be4d13f907ac898ff7e948-Abstract-Conference.html.

Yuren Mao, Yuhang Ge, Yijiang Fan, et al. A survey on lora of large language models. *Frontiers Comput. Sci.*, 19(7):197605, 2025. doi: 10.1007/S11704-024-40663-9. URL https://doi.org/10.1007/s11704-024-40663-9.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, et al. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024. doi: 10.1145/3620666.3651335.

Thomas M. Moerland, Joost Broekens, Aske Plaat, et al. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023. doi: 10.1561/2200000086.

David Rein, Betty Li Hou, Asa Cooper Stickland, et al. GPQA: A graduate-level google-proof Q&A benchmark, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Luohe Shi, Zuchao Li, Lefei Zhang, et al. Scaling LLM speculative decoding: Non-autoregressive forecasting in large-batch scenarios. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 32947–32955. AAAI Press, 2026. doi: 10.1609/AAAI.V40I39.40576. URL https://doi.org/10.1609/aaai.v40i39.40576.

Noah Shinn, Federico Cassano, Ashwin Gopinath, et al. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh-Vahid, et al. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/9b26ad15462c81548c0689188d2e8018-Abstract-Conference.html.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, et al. Spectr: Fast speculative decoding via optimal transport. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/6034a661584af6c28fd97a6f23e56c0a-Abstract-Conference.html.

Víctor Uc-Cetina, Nicolás Navarro-Guerrero, Anabel Martín-González, Cornelius Weber, and Stefan Wermter. Survey on reinforcement learning for language processing. *Artif. Intell. Rev.*, 56(2): 1543–1575, 2023. doi: 10.1007/S10462-022-10205-5. URL https://doi.org/10.1007/s10462-022-10205-5.

Hongyu Wang, Shuming Ma, Li Dong, et al. Bitnet: Scaling 1-bit transformers for large language models, 2023a.

Jikai Wang, Yi Su, Juntao Li, et al. Opt-tree: Speculative decoding with adaptive draft tree structure. *Trans. Assoc. Comput. Linguistics*, 13:188–199, 2025. doi: 10.1162/TACL_A_00735. URL https://doi.org/10.1162/tacl_a_00735.

Wenhui Wang, Furu Wei, Li Dong, et al. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Wenxiao Wang, Wei Chen, Yicong Luo, et al. Model compression and efficient inference for large language models: A survey, 2024a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, et al. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yubo Wang, Xueguang Ma, Ge Zhang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. URL http://papers.nips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Heming Xia, Zhe Yang, Qingxiu Dong, et al. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 7655–7671. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.456. URL https://doi.org/10.18653/v1/2024.findings-acl.456.

Shitao Xiao, Zheng Liu, Peitian Zhang, et al. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pp. 641–649. ACM, 2024. doi: 10.1145/3626772.3657878. URL https://doi.org/10.1145/3626772.3657878.

Yunfan Xiong, Ruoyu Zhang, Yanzeng Li, et al. Dyspec: Faster speculative decoding with dynamic token tree structure. *World Wide Web (WWW)*, 28(3):36, 2025. doi: 10.1007/S11280-025-01344-0. URL https://doi.org/10.1007/s11280-025-01344-0.

Lingling Xu, Haoran Xie, S. Joe Qin, et al. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *IEEE Trans. Pattern Anal. Mach. Intell.*, 48 (6):6107–6126, 2026. doi: 10.1109/TPAMI.2026.3657354. URL https://doi.org/10.1109/TPAMI.2026.3657354.

Zhilin Yang, Peng Qi, Saizheng Zhang, et al. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1259. URL https://doi.org/10.18653/v1/d18-1259.

Shunyu Yao, Dian Yu, Jeffrey Zhao, et al. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Rongjie Yi, Liwei Guo, Shiyun Wei, et al. Edgemoe: Empowering sparse large language models on mobile devices. *IEEE Trans. Mob. Comput.*, 24(8):7059–7073, 2025. doi: 10.1109/TMC.2025.3546466. URL https://doi.org/10.1109/TMC.2025.3546466.

Qiying Yu, Zheng Zhang, Ruofei Zhu, et al. DAPO: an open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Jun Zhang, Jue Wang, Huan Li, et al. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, 2024. doi: 10.18653/v1/2024.acl-long.607.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract-Conference.html.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, et al. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=3bq3jsvcQ1.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=rsY6J3ZaTF.

Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, et al. Transfer learning in deep reinforcement learning: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(11):13344–13362, 2023. doi: 10.1109/TPAMI.2023.3292075. URL https://doi.org/10.1109/TPAMI.2023.3292075.

AI Use Statement

We used generative AI tools to translate the text (Claude Opus 5.0), to check grammar and spelling and polish the wording (GPT-5.6), to guide the drawing of the figures (Gemini), and to check the content (Grok). We did not use generative AI tools to modify code. We have reviewed all AI-assisted work: we checked every reference for hallucinated citations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility Statement

The details needed to reproduce the results of this paper, including the models, the data splits, the training procedure, the evaluation metrics and the software setup, are disclosed in this appendix.

Appendix

Table S1 lists the contents of each part of the appendix.

Appendix A.0: Limitations

Price control by a hand-set constant. The comparison of budget pricing covers a constant price taken from the end of dual descent, but it lacks a constant price that a practitioner picks by hand for each benchmark. Such a price is the simplest alternative to the dual update, and a per-benchmark comparison of the budget deviation it leaves would show how much of the budget tracking comes from letting the price move during training rather than from the level at which the price is set.

Sampling uncertainty of the main comparison. The main comparison gives point estimates of accuracy on evaluation splits of 1,000 queries for HotpotQA and Natural Questions and on the fixed 500-problem MATH-500 set, and it lacks interval estimates for them. Several of the differences it

Table S1: What each part of the appendix holds.

Section	Contents
Appendix A.0	Limitations
Appendix A.1	Extended related work
Appendix A.2	Figure details
Appendix A.3	Dual update: admissibility, proposition, and proof
Appendix A.4	Experimental details
Appendix A.5	Experiments
Appendix A.6	Reproducibility and implementation details
Appendix A.7	Discussion
Appendix A.8	Future work

reads are below one point, such as the macro gap between PACO-PO and the best fixed cell, and a reader needs standard errors at these sample sizes to tell which differences are resolved and which fall within sampling noise.

Uncertainty of the ablation and transfer margins. The ablation rows and the backbone transfer are single-seed runs read against the three-seed full method, and the margins between them lack standard errors. These margins carry the claims about which component matters, from the factorized head to the retrieval preview and the moving price, so a measure of their uncertainty is needed to judge whether each intervention is separated from noise.

APPENDIX A.1: EXTENDED RELATED WORK

The related work is set out here, one paragraph per line of work plus one on evaluation resources.

Speculative and parallel decoding. The draft-and-verify family accelerates autoregressive generation without changing its output distribution: speculative sampling verifies a cheap draft against the target model (Leviathan et al., 2023; Chen et al., 2023; Sun et al., 2023), and successor systems raise acceptance rates through tree-structured attention, extra decoding heads, and self-drafting layers (Miao et al., 2024; Cai et al., 2024; Zhang et al., 2024; Wang et al., 2025). A complementary strand learns when to extend the speculative horizon or distills dedicated drafters (Brown et al., 2024; Xiong et al., 2025; Zhou et al., 2024; Liu et al., 2024b; Shi et al., 2026), and the design space is cataloged in survey form (Xia et al., 2024). All of these methods hold the semantic workload fixed, producing the same tokens faster, whereas PACO-PO changes the workload itself: it decides per query how many reasoning tokens and retrieved passages to pay for, and is therefore composable with any of these decoders.

Parameter-efficient adaptation and compression. The second line lowers a model's standing cost. Parameter-efficient fine-tuning and low-rank adapter families are reviewed (Han et al., 2024; Mao et al., 2025; Xu et al., 2026; Ding et al., 2022), with prompt-free and forward-only variants reducing tuning overhead further (Karimi Mahabadi et al., 2022; Malladi et al., 2023). Structural pruning and quantization pipelines shrink a trained network (Ma et al., 2023; Wang et al., 2023a; 2024a), cache-eviction schemes drop low-attention key-value entries during generation rather than altering the weights (Zhang et al., 2023), and mixture-of-experts routing makes the parameter subset conditional on the token (Shazeer et al., 2017; Du et al., 2022; Yi et al., 2025). These interventions set the compute envelope at training or deployment time and apply it uniformly across queries; our controller instead keeps the base LLM and retriever frozen and moves the efficiency decision to inference time, where the envelope is re-allocated per query under one budget.

Adaptive retrieval and budgeted inference control. The nearest neighbors condition retrieval on the query. Surveys of retrieval-augmented generation organize when-to-retrieve and what-to-retrieve decisions (Gao et al., 2023); active retrieval schemes trigger lookups during generation when the model signals a knowledge gap (Jiang et al., 2023); learned routing selects among no-retrieval, single-step, and multi-step retrieval from a predicted query complexity (Jeong et al., 2024); and

long-context studies quantify why depth must be chosen rather than maximized: accuracy varies sharply with where a relevant passage sits in the supplied context (Liu et al., 2024a), and it decays as the input is lengthened even on controlled retrieval probes built for the purpose (Hsieh et al., 2024). On the effort side, deliberate search over intermediate thoughts enlarges the reasoning workload a query receives (Yao et al., 2023) and step-back abstraction restructures it (Zheng et al., 2024), while blueprint taxonomies organize this design space (Besta et al., 2025) and evaluation studies chart where added deliberation stops paying off (Shojaee et al., 2025); verbal-reinforcement agents revise their tool use across repeated attempts on a task (Shinn et al., 2023). What separates PACO-PO from this neighborhood is joint, graded control with an explicit constraint: the retrieve-or-not gates and effort switches of this line adapt one axis with no shared budget, while our factorized policy prices both axes against one expected-cost budget through a single dual variable, with the substitution between them read from a token-free retrieval-score preview.

Reasoning and reinforcement learning for language models. Group-relative policy optimization removes the learned value function from proximal policy optimization and is the inner solver we adopt (Shao et al., 2024); process supervision refines the reward side of reasoning training (Lightman et al., 2024). Broader surveys cover reinforcement learning for language processing and language-model-enhanced reinforcement learning (Cao et al., 2025; Uc-Cetina et al., 2023), as well as the multi-agent, model-based, and transfer settings whose machinery our one-step problem deliberately avoids (Gronauer & Diepold, 2022; Moerland et al., 2023; Zhu et al., 2023; da Silva & Costa, 2019). These systems optimize a scalar objective with hand-set efficiency knobs; PACO-PO instead treats these knobs as the action space and the budget as a constraint.

Benchmarks and evaluation resources. Established resources relevant to evaluation include HotpotQA for multi-hop QA (Yang et al., 2018), self-consistency, which raises reasoning accuracy by sampling several chains at nonzero temperature and marginalizing over them (Wang et al., 2023b), graduate-level and contamination-resistant QA benchmarks (Rein et al., 2023), a large-scale reasoning-training system (Yu et al., 2025), and a multi-task evaluation suite (Wang et al., 2024b).

APPENDIX A.2: FIGURE DETAILS

This appendix describes the three main-text figures: the teaser, the motivation figure panel by panel, and the framework overview. Each panel of the motivation figure is reproduced individually below.

Separate dials versus one budgeted decision. Figure 1 contrasts prior systems with PACO-PO. Its left panel, Prior Systems, shows two regimes: reasoning effort and retrieval depth set on independent, hand-set dials, which forecloses the joint per-query choice, and scalar reward weighting, which trades accuracy against token cost through a hand-tuned weight and therefore requires sweeping weights instead of setting a budget. Its right panel, PACO-PO (Ours), shows the user token budget B entering a Lagrangian dual variable λ updated by projected descent, whose price is the pricing signal for the controller; the controller reads the query and the zero-cost retrieval preview $\phi_{\mathrm{ret}}(q)$, with no passage text read, and selects one cell (e, k) of the 12-cell action grid, whose depth k goes to the frozen retriever and whose effort e goes, with the query and the retrieved passages, to the frozen base LLM. The Key Advantages strip summarizes four results of the main text: accuracy within 0.7 points of the best fixed cell at 0.64 of its cost, 2.9 points over the stronger single-axis controller trained at the same budget, a macro budget violation of 0.03, and the budget B as the one user-facing setting in place of reward weights.

Challenge #1: Reasoning effort and retrieval depth form one coupled decision. Figure S1 plots accuracy against cost for the six representative static (e, k) cells on HotpotQA from Table 2, from $(\mathrm{LOW}, 0)$ at 31.4% and 0.22 cost to $(\mathrm{HIGH}, 10)$ at 48.2% and 1.00 cost. The dashed horizontal line marks the per-query joint oracle of Table S3 (56.0%), and the arrow marks the 7.8-point gap between the best static cell and that oracle; the corresponding gaps are 8.5 points on NQ and 3.5 on MATH-500. The star is PACO-PO's operating point, which the PACO-PO row of Table 1 reports at 47.6% and 0.64 cost. The static curve flattens while the oracle sits well above its endpoint, so the headroom belongs to per-query joint choice, which motivates the effort-conditioned depth head.

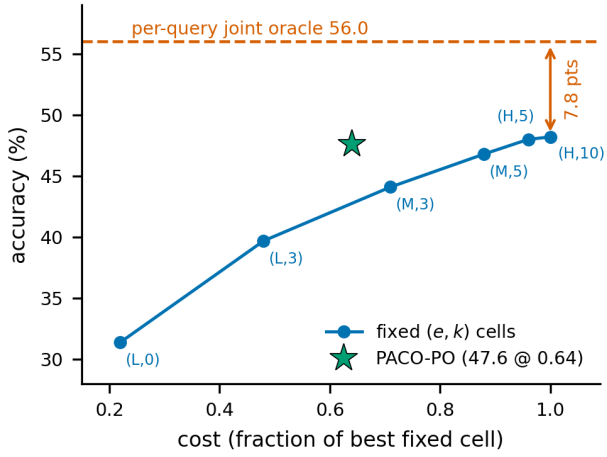

Figure S1: Left panel of the motivation figure: static (e, k) cells versus the per-query oracle on HotpotQA. The plotted oracle and gap are the HotpotQA row of Table S3.

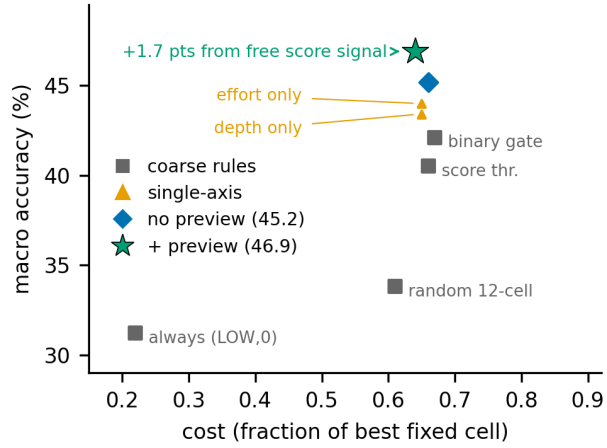

Figure S2: Middle panel of the motivation figure: coarse rules and single-axis controllers versus the graded joint policy, macro over the three benchmarks.

Challenge #2: The choice must be driven by a cheap query-side signal rather than the passage text. Figure S2 plots the macro accuracy–cost operating points of Table 3 and Appendix A.5: the untrained coarse rules (random 12-cell 33.8%, always-$(\mathrm{LOW}, 0)$ 31.2%, score-threshold heuristic 40.5%, retrieve-or-not binary gate 42.1%), the single-axis learned controllers (depth-only 43.4%, effort-only 44.0%), the graded joint policy with the preview features removed (45.2%), and the full graded policy reading the zero-cost score preview (46.9% at 0.64 cost). Graded joint choice beats every coarse or single-axis rule on accuracy and undercuts the cost of all but the two untrained floors, and the 1.7-point drop from removing the preview indicates the value of the token-free score signal, which motivates the preview featurization.

Challenge #3: Practitioners set a budget, not an exchange rate. Figure S3 plots the matched-scalarization comparison of Table 5 (macro over HotpotQA and NQ): the three fixed-weight controllers at $\alpha \in \{0.25, 0.5, 1.0\}$ land at realized costs 0.81, 0.69, and 0.66 with accuracies 46.5%, 46.0%, and 45.2%, the weight having to be swept before one lands near the intended spend, while the single-budget dual controller lands at 44.8% and 0.64 cost in one training run with the three-benchmark macro budget violation held at 0.03. The dashed vertical line sits at cost fraction 0.65, the numerical value of the budget B. Sweeping the weight scatters operating points the deployer

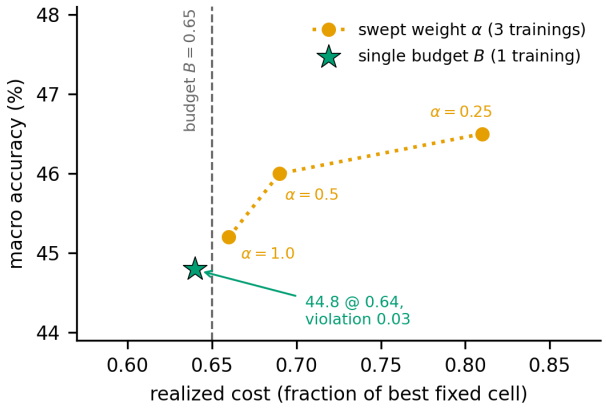

Figure S3: Right panel of the motivation figure: swept scalarization weights versus a single budget B, macro over HotpotQA and NQ.

must still choose among, while setting the budget yields a single point priced automatically by the dual variable, which motivates the budget pricing.

Framework overview. Figure 3 lays out the three parts of PACO-PO and their training loop. The left panel is the zero-cost retrieval-preview featurization: one retriever call yields the top-10 score profile, from which the preview features of Eq. 8 (top score, top-to-tenth score gap, mean score, and the entropy of the softmax-normalized profile) are computed, and the index metadata yields the length block of Eq. 9; together with the projected 64-dimensional query embedding they form the 72-dimensional feature vector $\phi(q)$ of Eq. 7, with no passage text read and no LLM tokens spent. The middle panel is the factorized joint effort–depth policy of Eq. 3: an effort head over $\{\mathrm{LOW}, \mathrm{MED}, \mathrm{HIGH}\}$ and a depth head over $\{0, 3, 5, 10\}$ conditioned on the sampled effort level through a learned embedding, whose joint sample (e, k) is executed by the frozen retriever and base LLM (Eq. 5) and, at deployment, replaced by the highest-probability cell (Eq. 6). The right panel is budget pricing via a single dual variable: the executed action's token cost of Eq. 10 enters the penalized reward $r - \lambda\, c/B$ of Eq. 13, and projected dual descent (Eq. 14) raises λ while the measured expected cost exceeds B and relaxes it when the budget is slack. The bottom strip is the end-to-end integration: for each query in a batch, G actions are sampled from the policy snapshot $\pi_{\theta_{\mathrm{old}}}$ frozen for the outer iteration, each is executed once, the penalized rewards are centered and scaled within the group into advantages, the controller is updated on the clipped surrogate with the entropy regularizer (Eqs. 15 and 16), and one dual step closes the outer iteration; Algorithm S1 gives the procedure line by line.

Appendix A.3: Dual Update: Admissibility, Proposition, and Proof

This appendix states and proves the formal claims about the dual update of Eq. 14.

Assumption 1 (Budget admissibility)**.** The budget satisfies $B \geq 1.1\, C_{\mathrm{floor}}$, where C_{floor} is the probed expected cost of always playing $(\mathrm{LOW}, 0)$.

Admissibility is a working condition on the budget; since C_{floor} is estimated on the probe sample, the test is carried out there. Algorithm S1 evaluates it at the probe step, before any training run, and raises an inadmissible budget to the admissibility threshold. The assumption is used only to bound the optimal price of the budget, and plays no part in the convexity, finiteness, or subgradient statements below.

Proposition S1 (Dual function and its subgradients)**.** Let $g(\lambda) = \sup_\theta L(\theta, \lambda)$. Then g is finite and convex on $\lambda \geq 0$; for any $\theta^\star(\lambda)$ attaining the inner supremum, $-\big(J_C(\theta^\star(\lambda)) - B\big)/B$ is a subgradient of g at λ; and the batch quantity $\hat{J}_C = (N_q G)^{-1} \sum_i c_i$ is an unbiased estimator of $J_C(\theta_{\mathrm{old}})$ on the training split, the cost of the behaviour policy that generated the batch. If, in addition,

the probe-sample budget is admissible in the sense of Assumption 1, then every minimizer of g on $\lambda \geq 0$ satisfies $\lambda^\star \leq 11$ for the program defined on that sample.

The proposition describes the dual function and the subgradient formula at an exact inner maximizer.

Throughout, $r(q,a) \in \{0,1\}$ is the task reward, $J(\theta) = \mathbb{E}_{q,a\sim\pi_\theta}[r(q,a)]$, $J_C(\theta) = \mathbb{E}_{q,a\sim\pi_\theta}[c(q,a)]$, and $L(\theta,\lambda) = J(\theta) - \lambda\,(J_C(\theta) - B)/B$ as in Eq. 12. We write π_0 for the minimal policy that always plays $(\mathrm{LOW}, 0)$, so that $J_C(\pi_0) = C_{\mathrm{floor}}$.

Finiteness. For every θ, the per-query cost obeys $0 \leq c(q,a) \leq c_{\max}$ by Eq. 11, so $J_C(\theta) \in [0, c_{\max}]$, and $J(\theta) \in [0,1]$ because the reward is an accuracy indicator. Hence for any fixed $\lambda \geq 0$,

$$L(\theta,\lambda) \leq 1 + \lambda\,\frac{B}{B} = 1 + \lambda, \tag{S1}$$

uniformly in θ, and $L(\theta,\lambda) \geq -\lambda\,(c_{\max} - B)/B$ for every θ by the same two bounds. The supremum $g(\lambda) = \sup_\theta L(\theta,\lambda)$ is therefore finite for every $\lambda \geq 0$, and we write $\max$ for it below whenever the supremum is attained.

Convexity. For each fixed θ, the map $\lambda \mapsto L(\theta,\lambda) = J(\theta) - \lambda\,(J_C(\theta) - B)/B$ is affine, since $J(\theta)$ and $J_C(\theta)$ do not depend on λ. The function g is the pointwise supremum of this family of affine functions indexed by θ, and a pointwise supremum of affine functions is convex on the set where it is finite. By the previous paragraph that set contains $\lambda \geq 0$, so g is convex there. Convexity holds regardless of how the policy is parameterized, and in particular does not require the inner problem in θ to be concave, which it is not for a neural controller.

Subgradient. Fix $\lambda \geq 0$ and let $\theta^\star(\lambda)$ attain the inner maximum. For any $\lambda' \geq 0$,

$$\begin{aligned} g(\lambda') \;&\geq\; L\big(\theta^\star(\lambda), \lambda'\big) = J\big(\theta^\star(\lambda)\big) - \lambda'\,\frac{J_C\big(\theta^\star(\lambda)\big) - B}{B} \\ &=\; g(\lambda) \;+\; \Big(-\,\frac{J_C\big(\theta^\star(\lambda)\big) - B}{B}\Big)(\lambda' - \lambda), \end{aligned} \tag{S2}$$

where the inequality is the definition of the supremum and the last equality substitutes $g(\lambda) = L(\theta^\star(\lambda),\lambda)$ and collects terms. Inequality S2 is exactly the subgradient inequality, so $-\big(J_C(\theta^\star(\lambda)) - B\big)/B \in \partial g(\lambda)$. No differentiability of g is needed, and no envelope theorem is invoked; when the maximizer is not unique, each maximizer supplies one subgradient by the same computation.

Unbiasedness of the batch estimate. At outer iteration t, N_q queries are sampled uniformly without replacement from the training split; for each query, G actions are sampled conditionally independently from the frozen behaviour policy $\pi_{\theta_{\mathrm{old}}}$, executed once, and assigned costs $c_i = c(q_i, a_i)$. The $N_q G$ pairs are not mutually independent, since the G pairs formed from one query share that query; they are, however, identically distributed, with a uniform query marginal on the training split and action conditional $\pi_{\theta_{\mathrm{old}}}(\cdot \mid q)$, and identical distribution is all that unbiasedness requires. The expectation below is therefore taken over the uniform draw of queries from the training split, the action sampling, and LLM generation, so $J_C(\theta_{\mathrm{old}})$ here is the cost of Eq. 2 with $\mathcal{D}$ taken as the uniform distribution on the training split, the program that training optimizes:

$$\mathbb{E}\big[\hat{J}_C\big] = \frac{1}{N_q G}\sum_i \mathbb{E}\big[c(q_i,a_i)\big] = J_C(\theta_{\mathrm{old}}), \tag{S3}$$

by linearity of expectation, which does not require independence. The estimate is bounded by $c_{\max}$ and hence has finite variance. Note that the expectation on the right is taken at θ_{old}, the policy frozen at the start of the outer iteration and used to roll out the batch, and not at the parameters that emerge from the E inner epochs.

Form of the update. The update of Eq. 14 applies $[\,\cdot\,]_+$, which is the Euclidean projection onto the closed convex set $\{\lambda : \lambda \geq 0\}$ and is nonexpansive. Writing the subgradient formula of Eq. S2 with $\hat{J}_C$ in place of $J_C(\theta^\star(\lambda_t))$ and taking a projected descent step gives

$$\lambda_{t+1} = \Big[\lambda_t - \eta_\lambda\Big(-\,\frac{\hat{J}_C - B}{B}\Big)\Big]_+ = \Big[\lambda_t + \eta_\lambda\,\frac{\hat{J}_C - B}{B}\Big]_+, \tag{S4}$$

which is Eq. 14. This substitution is what makes Eq. 14 the sample analogue of a subgradient step, and it is not an identity: the exact subgradient of Eq. S2 is evaluated at an inner maximizer $\theta^\star(\lambda_t)$, whereas $\hat{J}_C$ is an unbiased estimate of the cost at θ_{old}, which is neither an exact maximizer nor the iterate reached after the inner epochs. This completes the proof of the first three statements.

Role of admissibility. Assumption 1 enters only here, and only through feasibility of the program defined on the probe sample. The quantity C_{floor} is measured on the 200-query probe, so $B \geq 1.1\, C_{\text{floor}}$ certifies feasibility for the empirical program on that sample, and extends to the population program only to the extent that the probe estimate of the minimal policy's cost is accurate. Read on the probe sample, $C_{\text{floor}} \leq B/1.1$ gives $B - J_C(\pi_0) \geq B(1 - 1/1.1) = B/11$. Any minimizer $\lambda^\star$ of g on $\lambda \geq 0$ then obeys

$$\begin{aligned} J(\pi_0) + \frac{\lambda^\star}{11} \;&\leq\; L(\pi_0, \lambda^\star) \;\leq\; g(\lambda^\star) \\ &\leq\; g(0) = \sup_\theta J(\theta) \;\leq\; 1, \end{aligned} \tag{S5}$$

whence $\lambda^\star \leq 11\,(1 - J(\pi_0)) \leq 11$. The slack is bounded away from zero by construction: admissibility leaves B at worst equal to $1.1\, C_{\text{floor}}$. The admissibility check supplies a stated, checkable slack, without which the bound above is unavailable; the probe therefore verifies it before training. At the adopted configuration the check passes with room to spare, so the raise of Algorithm S1 is inactive across the budgets studied here and binds only for a budget set close to the floor; the room that is present tightens the bound. With $B = 0.65$ and the minimal policy's cost fraction $J_C(\pi_0) = 0.22$ (the always-$(\text{LOW}, 0)$ row of Table S4), the admissibility margin is $B/(1.1\, C_{\text{floor}}) = 0.65/0.242 = 2.69$. The realized relative slack is therefore $(B - J_C(\pi_0))/B = (0.65 - 0.22)/0.65 = 0.662$ rather than the worst-case $1/11 = 0.091$ that Eq. S5 assumes. Substituting the measured floor accuracy $J(\pi_0) = 0.312$, which Table S4 reports for the always-$(\text{LOW}, 0)$ policy, into the stated bound already gives $\lambda^\star \leq 11\,(1 - 0.312) = 7.57$; carrying the realized slack in place of $1/11$ gives $\lambda^\star \leq B\,(1 - J(\pi_0))/(B - J_C(\pi_0)) = 0.65 \times 0.688/0.43 = 1.04$, an order of magnitude below the stated 11. Both figures read the floor cost and $J(\pi_0)$ off the evaluation-split cost-fraction tables.

Scope of the proposition. The proposition covers the exact inner maximizer; two points lie outside it. First, the bound of Eq. S5 constrains the minimizers $\lambda^\star$ of the dual function; it says nothing about the trajectory of the iterates λ_t produced by Eq. 14 at a constant step size. Second, nothing above bounds the gap between $\min_{\lambda \geq 0} g(\lambda)$ and the value of the constrained program, nor certifies that the returned greedy controller is feasible.

APPENDIX A.4: EXPERIMENTAL DETAILS

This appendix expands the implementation summary of the main text with the full configuration.

Datasets and scoring. HotpotQA is multi-hop QA; Natural Questions is single-hop open-domain QA; MATH-500 is evaluated with reference retrieval. HotpotQA and NQ are scored by exact match, MATH by boxed-answer match. Full task counts and split details are listed in Appendix A.6.

Base model, retriever, and index. The frozen base LLM, the replacement backbone of the transfer study, and the dataset versions are named in Appendix A.6. The retriever is BGE-base-en-v1.5 over a Wikipedia index for the QA sets and a corpus of MATH training solutions and reference material for MATH-500. Passages are capped at 256 tokens, and the index is identical across all methods and ablations, so no comparison mixes retrieval stacks.

Budget probe. For each benchmark a probe evaluates the accuracy and cost of all 12 fixed cells. The per-query budget is set to $B = 0.65\times$ the mean cost of the best fixed cell and verified admissible against $1.1\, C_{\text{floor}}$, where C_{floor} is the probed cost of always playing $(\text{LOW}, 0)$; accuracy ties among cells break toward lower cost, and an inadmissible budget is raised to the admissibility threshold before training. Cost is reported as a fraction of the best-fixed-cell mean cost. B is fixed once, at the probe step, from the probe split's mean cost for the selected cell, and is held at that value for the whole run. The cost fractions of Eq. S7 instead divide by the mean cost of the same cell measured afresh on the evaluation split.

Controller configuration. The query is embedded by a frozen MiniLM sentence encoder and projected to 64 dimensions by the trained map W_p; retrieval scores are min–max normalized per benchmark, with the normalization constants fit on the probe split and frozen; out-of-range evaluation scores are clipped to $[0, 1]$. The retrieval preview is computed from the top 10 retrieval scores of each query. The feature vector has $d_\phi = 72$ dimensions (64 projected embedding, 4 preview, 4 length); the 16-dimensional effort embedding is concatenated for the depth head, whose input is therefore 88-dimensional. The controller comprises two MLP heads, an effort head and an effort-conditioned depth head, each with two hidden layers of width 256, roughly 0.2M trainable parameters in all, orders of magnitude below the frozen base LLM. Training hyperparameters are listed in Appendix A.6.

Baseline implementations. (i) The best fixed cell is selected on the probe split and is the reference every target is stated against. (ii) Single-axis learned controllers train one head only: learned effort with k fixed to the best per-benchmark depth, and learned depth with e fixed to MED. (iii) Untrained rules comprise a random 12-cell policy, the minimal always-$(\mathrm{LOW}, 0)$ policy, a score-threshold heuristic that retrieves deep only when the top retrieval score is low, and a retrieve-or-not binary gate. (iv) The scalarization sweep trains three controllers with fixed weights $\alpha \in \{0.25, 0.5, 1.0\}$ and no dual update. (v) The adaptive controllers: the confidence cascade runs the cheapest cell first and escalates when answer confidence falls below a tuned threshold; the difficulty router predicts a difficulty bucket from the query encoding and maps it to a cell; the LinUCB bandit selects cells online from reward feedback and enforces no budget; its expected cost is nevertheless measured against B, which yields its violation entries; the cascade and router violations are measured the same way. Because none of the three has outer iterations, all three use the evaluation-pass form of Eq. S8 given below.

Baseline fairness. Every comparison point reported in this paper is a reproduction run under this paper's protocol on this paper's setting; none is transcribed from a published table. All adaptive baselines are tuned on the same probe split, use the same frozen base LLM, retriever, and index, and read the same query-side features where applicable: the cascade threshold is swept on the probe split, the router maps three probe-fit difficulty buckets to cells with the buckets fit by k-means over 10 restarts, and the LinUCB bandit uses the query embedding and the eight retrieval and length features as its context with exploration coefficient 1.0. Token counts for all methods use the base LLM's tokenizer after final prompt serialization, with reasoning and answer tokens read from the serving counters.

Metrics. Reported accuracy and cost are measured on the evaluation split under the deployed greedy controller. For a policy π_θ, task accuracy is the expected reward under the deployed policy,

$$\mathrm{Acc} = \mathbb{E}_q\big[r\big(q,\, a^{\mathrm{deploy}}(q)\big)\big]. \tag{S6}$$

The cost of the same policy is reported relative to the strongest static reference,

$$\mathrm{CostFrac} = \frac{\mathbb{E}_q\big[c\big(q,\, a^{\mathrm{deploy}}(q)\big)\big]}{C_{\mathrm{best}}}, \tag{S7}$$

where C_{best} is the mean per-query cost of the best fixed cell on the evaluation split. The relative budget violation is

$$\mathrm{Viol} = \frac{\big|\hat{J}_C(\theta) - B\big|}{B}, \tag{S8}$$

the absolute relative deviation of the realized expected cost from B, computed per seed over the final outer iterations and then averaged. The three adaptive baselines and the two hand-set rules (the score-threshold heuristic and the retrieve-or-not gate) run no outer iterations, so for them $\hat{J}_C$ is instead the mean per-query cost realized over one pass of the evaluation split, per seed, and Eq. S8 is applied to that quantity; their violation entries are therefore deviations of a deployed spend from B, not of a training-time running estimate, and are not iteration-averaged like the trained rows. In both forms the B of Eq. S8 is the probe-split budget fixed above, whereas the cost fraction of Eq. S7 is normalized by the evaluation-split C_{best}. The two denominators differ by the fixed factor 0.65 and by the gap between two measurements of the same quantity on two samples, so a cost fraction does not convert into a violation by subtracting 0.65. The measure penalizes under- and overspend symmetrically and is therefore not derivable from the mean cost fraction alone.

APPENDIX A.5: EXPERIMENTS

This appendix supplements each table of the main experiments, in the order the tables appear: the protocol behind it, the meaning of its rows and columns, and the observations the main text does not read from it. It then reports the learnability probe and the untrained floor configurations. Table S2 collects the protocols and their main reported values; as in the main text, the macro aggregates are the primary reported quantities. More details about the training configuration, seeds included, are provided in Appendix A.6.

Table S2: The PACO-PO evaluation protocols and their main reported values.

Protocol	Reported quantities	Main value
Main comparison	Accuracy, cost fraction, budget violation vs. best fixed cell and adaptive controllers	46.9% at 0.64 cost
Budget-matched static mixture	Accuracy of the best random mix of fixed cells at PACO-PO's spend	42.8% vs. 47.6% at 0.64
Ablation ladder	Accuracy/cost of each single intervention	full 46.9%
Dual step size and group size	Accuracy and budget violation across the dual step size and group size	within 0.6 pts; 0.03–0.11
Matched scalarization	Accuracy, cost, and training runs to place a weight	1 vs. 3 training runs
Cross-backbone transfer	Accuracy and cost under a same-class backbone swap	46.2% retained
Learnability probe	Oracle-vs-best-fixed-cell accuracy gap per benchmark	gaps 3.5–8.5 pts

Main comparison. Table 1 compares PACO-PO with the best fixed cell and three adaptive controllers, per benchmark and macro over the three. Accuracy is exact match for HotpotQA and NQ and boxed-answer match for MATH-500; cost is the mean per-query cost as a fraction of the best-fixed-cell cost measured on the evaluation split (Eq. S7); budget violation is the seed-averaged relative deviation of expected cost from B (Eq. S8), a training-time running estimate averaged over the final outer iterations for PACO-PO and the deviation of a realized evaluation-split spend from the same B for the three adaptive baselines, which have no outer iterations. The macro rows are unweighted means over the three benchmarks. The best fixed cell has no budget and is not trained against one, so no budget deviation is measured for it and its violation entry reads (n/a). Beyond the macro verdict, the per-benchmark rows show where each adaptive controller loses ground: the confidence cascade tracks PACO-PO closely on the two factual QA sets (within 0.3 points on NQ) and falls furthest on MATH-500, more than six points below the best fixed cell, where its escalation is also costliest (0.82); the difficulty router is the strongest baseline on MATH-500 (50.8%) and falls behind the cascade on NQ (39.2% against 41.6%); the bandit, which enforces no budget, carries the largest violations (0.12 and 0.13 on HotpotQA and NQ) and loses most on NQ (37.0%). Reading accuracy against spend, the macro accuracy per cost (accuracy as a fraction of 1 divided by the cost fraction) is 0.48 for the best fixed cell, 0.59 for the confidence cascade, 0.62 for the difficulty router, 0.63 for the bandit, and 0.73 for PACO-PO.

Budget-matched static mixture. Table 2 lists six static (e, k) cells on HotpotQA, each one fixed choice run on the same evaluation split, together with the best query-independent random mix of them at PACO-PO's spend. The marginal cell $(\mathrm{HIGH}, 10)$ buys only 0.2 points over $(\mathrm{HIGH}, 5)$ for four percent more cost, the tail of the diminishing-return curve, and PACO-PO beats every displayed cell below 0.96 cost on accuracy while undercutting the best cell's cost. The mix row is obtained by linear interpolation on the segment bracketing 0.64, from $(\mathrm{LOW}, 3)$ at 39.7% and 0.48 cost to $(\mathrm{MED}, 3)$ at 44.1% and 0.71: the static curve passes 42.8% there, the expected accuracy of a random mix of the two cells at that spend. This pair is the best of the eight pairs of displayed cells that bracket 0.64 (the next, $(\mathrm{LOW}, 3)$ with $(\mathrm{MED}, 5)$, gives 42.5%), and a mix of any number of cells at a fixed expected cost attains its maximum on a mix of at most two, so 42.8% is the highest accuracy any random mix of the six displayed cells reaches at that spend. Accuracy per cost, accuracy as a fraction of 1 divided by the cost fraction, reads 1.43, 0.83, 0.62, 0.53, 0.50, and 0.48 for the six cells in the order listed and 0.74 for PACO-PO.

Factorized joint effort–depth policy. Table 3 applies one intervention per row, read against the full method or the untrained floor rather than as a leave-one-out subtraction; rows are macro means over the three benchmarks, every trained row given the same budget B while their realized costs differ. The reference row is the best fixed (e, k) cell selected on the probe split, the static reference against which every target is stated. The score-threshold heuristic retrieves deep only when the top retrieval score is low, a hand-set rule on one signal; the retrieve-or-not binary gate is the coarsest hand-set switch; together they show what hand-set rules recover (40.5–42.1%). The learned-depth-only row trains the depth head with effort fixed to MED and the learned-effort-only row trains the effort head with depth fixed per benchmark, each isolating one axis; a single learned axis plateaus at 43.4–44.0%. The flat 12-way head replaces the effort-conditioned factorization by an unfactorized softmax over the twelve cells and isolates the factorization, costing 1.3 points against the full method.

Zero-cost retrieval preview. The row without retrieval preview of Table 3 removes the four preview features of Eq. 8 from the full method while the length block of Eq. 9 is retained, so it isolates the token-free score signal; the removal costs 1.7 points (45.2% against 46.9%) and raises the realized cost from 0.64 to 0.66.

Budget pricing via a single dual variable. The frozen-price row of Table 3 trains the full policy from initialization with λ held fixed at the final trained value of a reference run; it isolates dual descent and shows that a fixed price misses B (violation 0.18 against 0.03). The tuned-scalarization row replaces the dual variable by the best tuned scalarization weight and reuses the selected run of the weight sweep of Table 5, the $\alpha = 1.0$ run, evaluated here as a macro over the three benchmarks (46.3% at 0.68 cost with violation 0.09).

Dual step size and group size. Table 4 sweeps the dual step size η_λ and the group size G, each varied one at a time from the adopted setting, on a single seed; the adopted-setting rows reproduce the three-seed macro of the full method. Accuracy is insensitive to both over the tested range (within 0.6 points). Budget compliance is governed by η_λ: a small step (0.02) leaves an eleven-percent violation, a large step (0.5) an eight-percent one, and the adopted 0.1 sits at the three-percent floor. Group size trades compute for a marginally tighter estimate: $G = 8$ matches $G = 6$ on compliance and the 0.1-point accuracy difference lies inside the spread of this sweep, so the cheaper $G = 6$ is adopted.

Matched scalarization. The three fixed-weight controllers of Table 5 are trained separately with the penalized reward $r - \alpha\, c/B$ and no dual update, for $\alpha \in \{0.25, 0.5, 1.0\}$, three training runs per benchmark; the table aggregates the HotpotQA and NQ results of each weight, which is why its PACO-PO row reads 44.8% at 0.64 cost, the two-benchmark macro of the three-seed values of Table 1, rather than the three-benchmark 46.9%. Each controller is compared at its realized cost on the evaluation split; the weight whose realized cost lands nearest B, $\alpha = 1.0$ at 0.66, is the fair comparison point, and the tuning cost is charged to the baseline: the training-runs column prints one count for the whole sweep, three runs per benchmark to place the weight for this budget alone, against one run for the dual. This protocol tests whether setting the budget removes the weight search.

Cross-backbone transfer. Table 6 evaluates the controller trained against one frozen backbone, unchanged, on a same-class replacement; the controller reads a difficulty representation rather than backbone internals. Accuracy is largely retained (46.2% macro after the swap versus 46.9% at source) while the cost fraction drifts from 0.64 to 0.69: the difficulty-to-allocation mapping transfers, and the cost calibration drifts by 0.05. Cost fractions remain normalized by the source backbone's best-cell cost. The source columns are the three-seed PACO-PO values of Table 1 and the swap columns are single-seed, so the 0.7-point macro accuracy difference and the 0.05 cost drift are read across seed counts and are not seed-matched. A controller moved to a same-class backbone answers about as well without retraining, at a small, measurable budget drift. More details about the replacement backbone and the swap protocol are provided in Appendix A.6.

Learnability probe. Table S3 reports, for each benchmark, the per-query oracle accuracy against the best fixed cell and PACO-PO. All 12 cells are evaluated on 200 probe queries at temperature 0

(deterministic, avoiding sampling-noise inflation of the oracle), giving the oracle column; the best-fixed and PACO-PO columns report the evaluation-split values of Table 1, and the gaps compare the probe-estimated oracle with those values. The same 200 queries supply the probe estimate of C_{floor}, against which admissibility $B \geq 1.1\, C_{\text{floor}}$ is verified. The budget B is set from the probe-split estimate of the best-fixed-cell mean cost, whereas the C_{best} normalizing the reported cost fractions (Eq. S7) is the evaluation-split value of the same quantity. Averaged over the three benchmarks as in the macro rows of Table 1, the oracle column is 54.2%, 6.6 points above the best fixed cell's 47.6. The HotpotQA row is the one plotted in Figure S1.

Untrained floor configurations. Table S4 reports two floor rows kept out of the main ladder: a random policy over the 12 cells and the minimal always-$(\text{LOW}, 0)$ policy. They bound the ladder from below; neither has a budget, so no budget deviation is measured for them and their violation entries read (n/a).

Table S3: Learnability probe: PACO-PO and the best fixed cell versus the per-query oracle and their gap; the oracle column is estimated on the 200-query probe split, the other columns on the evaluation split.

Benchmark	PACO-PO	Best fixed	Oracle	Gap
HotpotQA	47.6	48.2	56.0	7.8
NQ	41.9	42.5	51.0	8.5
MATH-500	51.3	52.0	55.5	3.5

Table S4: Untrained floor configurations complementing Table 3.

Configuration	Acc. (%) ↑	Cost ↓	Viol. ↓
Random 12-cell policy	33.8	0.61	(n/a)
Always-(LOW, 0) minimal	31.2	0.22	(n/a)

APPENDIX A.6: REPRODUCIBILITY AND IMPLEMENTATION DETAILS

This appendix records the implementation and reproducibility details deferred from the main text: the frozen models, the data splits and their sampling, the fixed design constants, the training scheme, the compute environment, and the prompt templates.

Frozen models. The base LLM is `Llama-3.1-8B-Instruct` (release 2024-07, `bfloat16` weights, served with an 8,192-token context window), held frozen throughout: every LLM call in training and evaluation is pure inference and no base-model parameter is updated. The retriever is `BAAI/bge-base-en-v1.5` (110M parameters), also frozen, and the query encoder feeding the controller is `sentence-transformers/all-MiniLM-L6-v2` (384-dimensional output). The same-class replacement backbone used in the transfer study of Table 6 is `Qwen2.5-7B-Instruct` (release 2024-09), matched to the source backbone in parameter class, instruction tuning, and serving stack; only the generator is exchanged, while the retriever, index, prompt templates, cost-accounting protocol, and the controller weights are held fixed. PACO-PO itself comprises two MLP heads, an effort head and an effort-conditioned depth head, each with two hidden layers of width 256, about 0.2M trainable parameters in all; it holds the only weights that are trained anywhere in the pipeline.

Datasets, versions, and sampling. HotpotQA is the distractor-free open-domain setting of the official distribution (`hotpot_qa`, `fullwiki` configuration); Natural Questions is the open-domain variant (open-NQ, the short-answer subset in the DPR-style splits); MATH-500 is the standard 500-problem evaluation subset of the MATH benchmark. All three are obtained from their public releases through the Hugging Face datasets hub, and the Wikipedia index is a late-2023 English snapshot chunked into passages of at most 256 tokens; the math reference corpus consists of MATH training solutions and standard reference material chunked identically. The passage index is a flat inner-product index (FAISS `IndexFlatIP`), and passages are encoded in batches of 256. For HotpotQA and Natural Questions, a pool of 2,000 queries is drawn from the official train splits by uniform sampling without replacement, of which the first 200 are held out as the probe split and the remaining 1,800 are used for training; evaluation uses 1,000 held-out dev queries per benchmark, drawn from the official dev split by the same uniform procedure under the run seed. MATH-500 uses the

standard 500-problem evaluation subset, with the training pool and probe drawn by the same procedure from the MATH training problems. The training, probe, and evaluation pools are disjoint by construction. The answer cap is $Y_{\max} = 256$ tokens and the serialized prompt scaffold is at most $P_{\max} = 192$ tokens.

Design constants and training scheme. Reasoning effort is drawn from the three levels $\{\mathrm{LOW}, \mathrm{MED}, \mathrm{HIGH}\}$ with reasoning-token caps $u(e) = 128/1024/4096$, and retrieval depth from $k \in \{0, 3, 5, 10\}$, giving the 12-cell grid. Group-relative policy optimization samples G actions per query, scores each by the penalized reward $\tilde{r} = r - \lambda\, c/B$, and centers and scales within the group; the single per-query budget is targeted by projected dual descent on the one dual variable λ (Eq. 14). The token value of B for each benchmark is not fixed a priori: it is instantiated at run time from the probe measurement, which also verifies the admissibility condition. Tables S5 and S6 list every constant of the configuration.

Table S5: Architecture and feature constants of PACO-PO.

Symbol	Quantity	Value
	Controller hidden layers / width	2 / 256
	Controller trainable parameters	≈ 0.2M
d_ϕ	Feature dimension (64 + 4 + 4)	72
	Query embedding, raw / projected	384 / 64
	Effort embedding for depth head	16
$u(e)$	Reasoning-token caps (LOW/MED/HIGH)	128 / 1024 / 4096
k	Retrieval depths	0, 3, 5, 10
$Y_{\max}$	Answer-token cap	256
$P_{\max}$	Prompt-scaffold cap	192
	Maximum passage chunk length	256

Table S6: Training scheme of PACO-PO.

Symbol	Quantity	Value
K_{outer}	Outer iterations	40
N_q	Queries per outer iteration	32
G	Group size (actions per query)	6
E	Inner epochs per outer iteration	2
η_θ	Policy learning rate	$3\mathrm{e}{-4}$
η_λ	Dual step size	0.1
ε	Clip range	0.2
β_H	Entropy coefficient	0.01
ϵ_A	Advantage-std stabilizer	$1\mathrm{e}{-8}$
λ_0	Dual variable at initialization	0
	Optimizer	Adam
	Adam (β_1, β_2), ϵ	(0.9, 0.999), $1\mathrm{e}{-8}$
	Learning-rate schedule	constant
	Weight decay	0
	Gradient-norm clip	1.0
	Base-LLM sampling temperature, train / eval	0.7 / 0.0
	Seeds, main method	17, 42, 2026
	Seeds, ablation and secondary rows	17

The controller is trained with Adam at a constant learning rate; no scheduler, warmup, or weight decay is used, and gradients are clipped to unit norm. The dual variable is initialized at $\lambda_0 = 0$, so the first outer iteration is unpriced and later prices follow Eq. 14, rising while the measured spend exceeds B. Training runs under a loss-explosion guard that aborts when the loss exceeds 5.0 times its median over the last 50 steps. Automatic mixed precision is disabled. All entries of the main comparison, PACO-PO and the baselines alike, report means over seeds $\{17, 42, 2026\}$; ablation rows and the secondary studies run the single seed 17.

Algorithm S1: PACO-PO: budget probe, group-relative training with a single dual price, and greedy deployment

Require: Frozen base LLM π_{LLM}, retriever R, and query encoder E_{enc}; queries from $\mathcal{D}$ in disjoint probe, training, and evaluation splits; hyperparameters $N_q, G, K_{\mathrm{outer}}, E, \eta_\theta, \eta_\lambda, \varepsilon, \beta_H, \epsilon_A$ of Table S6

Ensure: Trained controller π_θ and its greedy action rule $a^{\mathrm{deploy}}(\cdot)$ of Eq. 6

Probe: evaluate all 12 fixed cells $(e, k) \in \mathcal{E} \times \mathcal{K}$ on the probe split; select the most accurate cell, breaking ties toward lower cost; measure C_{floor}, the mean cost of always playing $(\mathrm{LOW}, 0)$

Fit the per-benchmark min–max normalization of the retrieval scores on the probe split and freeze its constants

Instantiate the token value of B as the chosen fraction of the selected cell's mean probe cost (0.65 in all experiments)

if $B \le 1.1\, C_{\mathrm{floor}}$ **then** $B \leftarrow 1.1\, C_{\mathrm{floor}}$ **end if** ▷ Assumption 1

Initialize the controller θ: projection W_p, effort head π^e_θ over $\mathcal{E}$, effort-conditioned depth head π^k_θ over $\mathcal{K}$; price $\lambda_0 = 0$

for outer iteration $t = 0$ **to** $K_{\mathrm{outer}} - 1$ **do**

  Freeze a copy $\pi_{\theta_{\mathrm{old}}} \leftarrow \pi_\theta$, its W_p included; sample N_q queries uniformly without replacement from the training split

  for each sampled query q **do**

    Form $\phi(q)$ by Eqs. 7–9 from one retriever call (top-10 scores) and the index metadata (passage lengths)

    for $i = 1$ **to** G **do**

      $e_i \sim \pi^e_{\theta_{\mathrm{old}}}(\cdot \mid \phi(q))$, then $k_i \sim \pi^k_{\theta_{\mathrm{old}}}(\cdot \mid \phi(q), e_i)$; $a_i \leftarrow (e_i, k_i)$ (Eq. 3)

      Assemble $R_{k_i}(q)$ from that retriever call and run one LLM generation episode at effort e_i under the cap $u(e_i)$: $\hat{y}_i \leftarrow \pi_{\mathrm{LLM}}(P(q, R_{k_i}(q)); e_i)$ (Eq. 5)

      $r_i \leftarrow r(q, a_i) \in \{0, 1\}$; $c_i \leftarrow t^{\mathrm{think}} + |\hat{y}_i| + \sum_{j \le k_i} |d_j(q)| + |P_0(q)|$ (Eq. 10)

      $\tilde{r}_i \leftarrow r_i - \lambda_t\, c_i / B$ (Eq. 13)

    end for

    Center the $\tilde{r}_i$ on their within-group mean and scale by the within-group population standard deviation with the stabilizer ϵ_A, giving $A_1, \ldots, A_G$

  end for

  Pool the N_q groups into one rollout batch and reindex its $N_q G$ samples by i

  $\hat{J}_C \leftarrow (N_q G)^{-1} \sum_i c_i$ ▷ cost of $\pi_{\theta_{\mathrm{old}}}$

  for inner epoch $= 1$ **to** E **do**

    $\rho_i(\theta) \leftarrow \pi_\theta(a_i \mid q_i) / \pi_{\theta_{\mathrm{old}}}(a_i \mid q_i)$ over the rollout batch, q_i the query of sample i, each policy factorized by Eq. 3 on the features its own W_p produces

    $\mathcal{L}(\theta) \leftarrow \mathcal{L}_{\mathrm{clip}}(\theta) + \beta_H\, \mathbb{E}_q[-\mathcal{H}(\pi_\theta(\cdot \mid q))]$, with clip range ε (Eqs. 15, 16)

    Update θ on $\mathcal{L}(\theta)$ using Adam at learning rate η_θ, with gradients clipped to unit norm

  end for

  $\lambda_{t+1} \leftarrow \left[\, \lambda_t + \eta_\lambda\, (\hat{J}_C - B)/B \,\right]_+$ ▷ projected dual descent, Eq. 14

end for

Deployment: for a new query q, make one retriever call, apply the frozen normalization with out-of-range scores clipped to $[0, 1]$, and form $\phi(q)$

Play $a^{\mathrm{deploy}}(q) = \arg\max_a \pi_\theta(a \mid q)$ over the 12 joint cells (Eq. 6) and execute it by Eq. 5; λ is not used at inference

The procedure end to end. Algorithm S1 collects the three parts of PACO-PO into one procedure. A probe on the held-out probe split runs all 12 fixed cells once, selects the best fixed cell, measures C_{floor}, instantiates the token value of the budget B from the selected cell's mean probe cost, raises an inadmissible budget to the admissibility threshold, and fits the min–max normalization constants that the controller's features use from then on. Each outer iteration freezes the current controller as the behaviour policy, draws G actions per query from it, and scores every action by the penalized reward $\tilde{r} = r - \lambda\, c/B$, so the price set at the previous iteration shapes the advantages of the current one. The inner epochs update the controller on the clipped surrogate with the entropy regularizer, after which one projected dual step moves λ by the relative gap between the rollout batch's mean cost, an unbiased estimate of the behaviour policy's expected cost on the training split, and B. At deployment one retriever call under the frozen normalization yields $\phi(q)$, the highest-probability joint cell is played, and the frozen pipeline executes it.

Compute environment and wall-clock. All experiments run on a single NVIDIA A100 80GB SXM GPU hosted on a 32-core AMD EPYC node with 256GB of system memory, under

CUDA 12.1 with driver 535.104.05. The software stack is transformers 5.14.1, tokenizers 0.23.1, huggingface-hub 1.25.1, accelerate 1.14.0, datasets 5.0.0, sentence-transformers 5.6.1, faiss-cpu 1.14.3, peft 0.19.1, vllm 0.26.0 for batched base-model inference, numpy 2.4.6, pandas 2.3.3, scikit-learn 1.9.0, matplotlib 3.10.9, and tqdm 4.67.3. The dominant cost is base-model inference rather than controller updates: one training run issues $K_{\text{outer}} \times N_q \times G = 7{,}680$ generation episodes, the per-benchmark probe issues $200 \times 12 = 2{,}400$, and one evaluation pass issues one episode per evaluation query for the single-pass methods, and one episode per ladder step executed for the confidence cascade. On this node a training run takes about 2.1 GPU-hours per benchmark, a probe about 0.6 per benchmark, and an evaluation pass about 0.3 on the 1,000-query QA splits and 0.2 on MATH-500. Taking the three benchmarks together, one trained configuration therefore costs about 6.3 GPU-hours and one evaluation pass about 0.8. The reported set comprises 15 trained configurations: the main method at three seeds, five ablation rows that require training (the two single-axis controllers, the frozen-price variant, the flat 12-way head, and the variant without the retrieval preview), the four non-adopted sensitivity settings, and the three scalarization weights. Four entries need no training run of their own and are excluded from that count: the best fixed cell is selected on the probe split, the score-threshold heuristic and the retrieve-or-not gate are hand-set rules, and the tuned-scalarization row reuses the selected run of the scalarization sweep; the three adaptive baselines are likewise tuned on the probe split rather than trained by gradient descent. Training therefore accounts for about $15 \times 6.3 \approx 95$ GPU-hours. The probes add 1.8 at one probe per benchmark, and the 37 evaluation passes add $37 \times 0.8 \approx 30$: the five methods of Table 1 at three seeds each, 15 passes in all, of which the best-fixed-cell row supplies one of the six fixed (e, k) cells of Table 2, whose accuracies and costs, the reference row of the ladder included, are evaluation-split values rather than probe readings; the remaining five cells of Table 2, the seven ablation rows other than the reference and the tuned-scalarization row, the two untrained floor rows, the seven sweep settings, and the transfer evaluation account for the other 22, the tuned-scalarization row reusing the evaluation of the selected sweep run. The full reported set is thus about 127 GPU-hours. Controller optimization itself accounts for under one percent of that: the 0.2M-parameter controller trains in seconds once the rollout rewards and costs are in hand.

Prompt templates. Two scaffolds are used, one for the QA benchmarks and one for MATH-500, identical across all methods, baselines, and ablations so that no comparison mixes prompts. The retrieved block and the effort line are the only slots that vary with the sampled action, and the retrieved block is omitted entirely when $k = 0$. The QA scaffold is:

```
You are a careful question-answering
assistant. Use the reference passages
when they help and ignore them when
they do not.

Reference passages:
[1] {passage_1}
...
[k] {passage_k}

Question: {question}

Think step by step for at most {u(e)}
tokens, then give the final answer on
a new line in the form
Answer: <answer>.
Keep the answer to a short phrase.
```

The MATH-500 scaffold differs only in the task line and the answer format:

```
You are a careful mathematical problem
solver. Use the reference material when
it helps and ignore it when it does not.

Reference material:
[1] {passage_1}
...
```

```
[k] {passage_k}

Problem: {problem}

Think step by step for at most {u(e)}
tokens, then give the final answer on a
new line in the form
Answer: \boxed{<answer>}.
```

The reasoning budget is enforced twice: the cap $u(e)$ is stated in the prompt and is also passed to the decoder as a hard stop, so $t^{\text{think}} \leq u(e)$ holds by construction and the prompt statement only helps the model plan within it. Token counts entering Eq. 10 are read from the serving counters after final serialization, so the scaffold, the retrieved passages, the reasoning tokens, and the answer tokens are all charged at their realized lengths. The per-query cap of Eq. 11 is explicit at this configuration: with the HIGH reasoning cap at 4,096, the answer cap $Y_{\max} = 256$, ten passages of 256 tokens each for D_{10}, and the scaffold cap $P_{\max} = 192$, it evaluates to $c_{\max} = 4{,}096 + 256 + 2{,}560 + 192 = 7{,}104$ tokens. No single query can be charged more than that, whichever cell the controller selects.

APPENDIX A.7: DISCUSSION

Why one budget resolves the coupling. The design reads Challenge #1 as a representation problem rather than a tuning problem. A controller that prices effort and depth on two independent dials does not represent that both draw on one shared token budget, so no axis-aligned rule can weigh a trade-off it never encodes. Casting the joint action as a one-step constrained Markov decision process with a single budget B (Eqs. 1–2) hands the decision the one quantity a decoupled controller lacks; hence the ablation ladder varies how the budget is used rather than which budget is set, every trained row being given the same B.

Why depth selection is conditioned on effort. Factoring the joint policy so that the depth choice reads the chosen effort level (Eq. 3) keeps the two dials coherent instead of letting each head optimize a private objective. The discrete 12-cell grid makes allocation a small, learnable choice, and the conditional structure lets the retrieval axis adapt to the reasoning axis rather than fixing a product of two independent marginals. The retrieval-score preview of Eq. 8 is what makes this adaptation cheap, since the controller reads substitution structure without spending any passage tokens.

What the dual variable contributes. The single Lagrangian dual variable of Eq. 14 is what turns a user budget into a training signal without a hand-set weight. By raising the price while the budget is violated and relaxing it when the budget is slack, the dual descent tracks the budget as the policy learns to move probability mass across the grid. The deployer therefore specifies a budget rather than a weight; the price emerges from training.

Allocation, not spend, as the operative lever. The advantage comes from where the budget is placed, not from extra spend, because the measured expected cost under dual descent stays near B over the final outer iterations. The ladder row that freezes the price at its final trained value shows the other side: its expected cost drifts off target, an eighteen-percent violation, and it loses 1.8 points.

APPENDIX A.8: FUTURE WORK

Several directions would extend this work. A first line would carry the joint controller into the multi-step agentic setting, where retrieval depth is replaced by a sequence of memory or tool operations and the single-turn credit assignment no longer holds exactly. A second could extend the single per-query budget to several coupled budgets, letting effort and depth trade against distinct ceilings rather than one shared allowance, so the dual variable of Eq. 14 generalizes to a small family of constraints. A third would enlarge the depth grid or make k continuous, testing how much of the accuracy gap to the per-query oracle a finer grid recovers. A fourth would add a budget-conditioning feature so that one controller serves a range of budgets without retraining, testing whether budget generalization can be learned directly.