

BORA-BUDGET: Learning When to Think Under a Token Budget

Zilong Wang¹ Kang Pan² Xiao Zhang² Erli Meng² Shuai Li^{1,3,*}

¹Shanghai Jiao Tong University

²Xiaomi Corporation ³Shanghai Innovation Institute

wangzilong@sjtu.edu.cn pankang@xiaomi.com zhangxiao16@xiaomi.com
mengerli@xiaomi.com shuaili8@sjtu.edu.cn

Abstract

Hybrid-thinking language models expose both cheap non-thinking and expensive thinking modes, creating a deployment problem: deciding when additional reasoning is worth its token cost and override risk. We formulate this problem as budgeted opportunity allocation, where a controller first obtains a cheap seed answer, estimates the marginal value and risk of paid thinking, and preserves the seed unless an adoption gate accepts the thinking result. We propose BORA-BUDGET, a budget-aware inference framework for frozen LLMs, and instantiate it as BORA-SWITCH-CALIBRATED. The controller uses seed-state features to predict helpfulness, harmfulness, and thinking cost, then allocates thinking through either frontier operating points or an AutoThreshold policy calibrated to a total token budget. Across LLM scales on math dataset, the method substantially improves over random allocation and approaches uniform-thinking accuracy under lower token budgets. Transfer results on multiple-choice, code, and hard-QA tasks show promising but uneven generalization: trace length remains a strong baseline, harmful overrides are reduced but not eliminated, and learning how much to think remains open. These results position BORA-BUDGET as a seed-preserving control layer for budgeted inference rather than a replacement for stronger always-thinking baselines.

1 Introduction

Large language models have made deliberate reasoning a test-time resource. Prompting methods can elicit intermediate reasoning steps (Wei et al., 2022; Kojima et al., 2022), sampling can aggregate multiple reasoning paths (Wang et al., 2023), and search or decomposition can spend additional computation on harder instances (Zhou et al., 2023; Yao et al., 2023). Recent work further emphasizes that

test-time compute itself can be scaled and budgeted (Snell et al., 2025; Muennighoff et al., 2025). These developments change the deployment question. A system no longer only asks whether a model can reason; it must decide *when* the extra reasoning is worth paying for.

This question becomes especially sharp for hybrid-thinking models. Such models expose both a cheap non-thinking mode and a more expensive thinking mode (Yang et al., 2025). Uniformly invoking the expensive mode is simple and often strong, but it wastes tokens on examples where the cheap seed answer is already correct. Uniformly avoiding the expensive mode is cheap, but leaves hard examples unsolved. Fixed test-time scaling strategies, including self-consistency and tree-style search, also tend to spend computation broadly: they prescribe how much to sample or search, but not which examples actually have positive marginal value. Cost-aware cascades and model routers address related budget questions across models (Chen et al., 2024; Ong et al., 2025), while adaptive computation studies input-dependent compute inside learned systems (Graves, 2016). Hybrid-thinking inference creates a complementary problem: route among reasoning *actions* inside a frozen model after observing a cheap seed.

We formulate this problem as *budgeted opportunity allocation*. For each input, a controller first obtains a cheap seed answer and a seed-state context. It then chooses among ACCEPT, which preserves the seed with zero extra reasoning cost, and paid THINK actions, which generate candidate answers under a finite token budget. The utility of a paid action is defined relative to the seed: it should receive credit for fixing a wrong seed, pay a cost for additional tokens, and be penalized for harmful seed-correct to final-wrong overrides. This seed-relative view differs from accuracy-only routing because it treats preserving a correct cheap answer as an explicit system objective.

*Corresponding author.

We introduce BORA-BUDGET, a controller layer for this allocation problem. Its main deployed instance, BORA-SWITCH-CALIBRATED, learns an opportunity score from seed-state features. The controller estimates the probability that paid thinking will help, the probability that it will harm, and the expected cost of invoking the thinking action. It then allocates thinking with either frontier operating points or a deployable AutoThreshold rule: given a total budget, a token price is selected and thinking is triggered only when predicted marginal value exceeds predicted cost and harmful-risk penalty. Finally, triggering and adoption are separated. Even after the system pays for a thinking candidate, an adoption gate can reject the candidate and return the seed. This separation is crucial because the expensive mode is not automatically safer than the cheap seed.

The technical contribution is therefore not a new base reasoner, nor a claim that expensive thinking should always be used. It is a budgeted control layer for frozen hybrid-thinking models. Our contributions are threefold:

1. We formulate hybrid-thinking inference as seed-preserved budgeted opportunity allocation, where ACCEPT is a zero-cost action and paid thinking is evaluated by marginal utility, token cost, and harmful overrides.
2. We propose BORA-BUDGET and its deployed binary instance BORA-SWITCH-CALIBRATED, which learns helpfulness, harmfulness, and cost from seed-state features, allocates thinking with AutoThreshold, and separates triggering from adoption.
3. We provide controller-level guarantees and experiments across model scales and transfer tasks, showing that learned allocation strongly improves over random allocation and approaches uniform-thinking accuracy under lower token budgets in the main calibrated setting.

The rest of the paper formalizes the allocation objective, describes the deployed controller, evaluates its accuracy–cost tradeoffs, and discusses where task-aware calibration, stronger adoption gates, and multi-budget arm selection remain open.

2 Related Work

Test-time reasoning and adaptive reasoning. Chain-of-thought and zero-shot reasoning elicit in-

termediate reasoning steps (Wei et al., 2022; Kojima et al., 2022), while self-consistency, least-to-most prompting, program-of-thoughts, and tree-of-thought search spend additional computation through sampling, decomposition, program execution, or search (Wang et al., 2023; Zhou et al., 2023; Chen et al., 2023a; Yao et al., 2023). Recent work studies test-time scaling and budget forcing (Snell et al., 2025; Muennighoff et al., 2025), and more directly, adaptive thinking methods learn when to invoke explicit reasoning (Zhang et al., 2025; Fang et al., 2025; Lou et al., 2025). Token-budget-aware methods such as TALE and Budget-Thinker control the length or remaining budget of the reasoning trace itself (Han et al., 2025; Wen et al., 2025). These works are closest in motivation, but BORA-BUDGET differs by treating a frozen hybrid-thinking model as exposing cheap seed and paid thinking actions: it learns an external seed-state opportunity model, allocates thinking under a total token budget, and explicitly models harmful seed-to-wrong overrides.

Routing, cascades, and hybrid-thinking models.

Cost-aware cascades and routers select among systems with different prices and capabilities (Chen et al., 2024; Ong et al., 2025), and reasoning cascades similarly escalate from cheaper to stronger inference paths using consistency signals (Yue et al., 2024). Hybrid-thinking models such as Qwen3 expose cheap non-thinking and expensive thinking modes (Yang et al., 2025), creating a related routing problem inside a single frozen model. BORA-BUDGET is complementary to model routing: after observing a cheap seed answer, it routes among reasoning actions within the same model and separates triggering paid thinking from adopting the resulting answer.

Signals, verification, and constrained allocation.

Our trigger features use trace length together with confidence and consistency signals such as answer consistency and self-consistency statistics (Wang et al., 2023; Yue et al., 2024). Verifier and process-supervision methods validate or rerank generated candidates (Lightman et al., 2024); in contrast, BORA-BUDGET first decides whether an expensive candidate should be generated and then applies an adoption gate before overriding the seed. The token-budget view is also related to adaptive computation and bandits with knapsacks (Graves, 2016; Badanidiyuru et al., 2013), though we use it as a controller-level inference framework rather than a

new regret-optimal bandit algorithm.

3 Problem Formulation and Method

3.1 Budgeted Opportunity Allocation

Hybrid-thinking models make test-time reasoning a budget-allocation problem. Given a batch of N problems $\mathcal{X} = \{x_1, \dots, x_N\}$, BORA-BUDGET first runs every problem through a cheap non-thinking seed policy π_0 :

$$(\tau_i^0, \hat{y}_i^0, c_i^0) = \pi_0(x_i), \quad (1)$$

where τ_i^0 is the seed trace, $\hat{y}_i^0$ is the seed answer, and c_i^0 is the seed token or latency cost. The system then extracts a seed-state context

$$s_i = \phi(x_i, \tau_i^0, \hat{y}_i^0). \quad (2)$$

The context contains only the problem and cheap seed output; it does not use gold labels or thinking-model candidates at inference time.

The general action set is

$$\mathcal{A} = \{\text{ACCEPT}, \text{THINK}(b_1), \dots, \text{THINK}(b_K)\}. \quad (3)$$

ACCEPT returns the seed with zero extra reasoning cost. A paid thinking action runs the same frozen model in thinking mode under budget b , producing a candidate answer that may or may not be adopted. Our main experimental instance uses the reliable binary action set

$$\mathcal{A}_{\text{main}} = \{\text{ACCEPT}, \text{THINK}(b_T)\}. \quad (4)$$

We keep the more general K -armed version as formulation and diagnostic because the oracle has large headroom, but learned arm selection is not yet stronger than the binary single-budget controller. The concrete budget used for b_T is an experimental setting, not part of the method definition.

Adoption gate. If $a_i = \text{ACCEPT}$, the final answer is $\hat{y}_i^0$. If the controller selects $\text{THINK}(b)$, the model first generates

$$(\tau_i^b, \hat{y}_i^b, c_i^b) = \pi_b(x_i). \quad (5)$$

An adoption gate g then decides whether the candidate is trusted:

$$\hat{y}_i^b = \begin{cases} \hat{y}_i^b, & g(\hat{y}_i^0, \hat{y}_i^b, \tau_i^b) = 1, \\ \hat{y}_i^0, & g(\hat{y}_i^0, \hat{y}_i^b, \tau_i^b) = 0. \end{cases} \quad (6)$$

Thus paying to think and overriding the seed are separate actions. This separation is essential: same-model thinking can confidently produce a wrong alternative, so a seed-preserved system should be allowed to fallback.

Marginal utility. During calibration, where y_i is known, we define action value relative to the seed:

$$R_i(a) = \mathbb{I}\{\hat{y}_i^a = y_i\} - \mathbb{I}\{\hat{y}_i^0 = y_i\} - \lambda C_i(a) - \eta H_i(a), \quad (7)$$

where $C_i(a) = c_i(a) - c_i^0$ is extra cost and

$$H_i(a) = \mathbb{I}\{\hat{y}_i^0 = y_i, \hat{y}_i^a \neq y_i\} \quad (8)$$

is a harmful seed-correct-to-final-wrong flip. For ACCEPT, $R_i(\text{ACCEPT}) = 0$. This reward makes the central deployment tradeoff explicit: a thinking call is valuable only when its expected improvement over the seed exceeds its token cost and override risk.

Budgeted objective. At inference time y_i is unavailable, so BORA-BUDGET learns $\hat{V}_i(a) \approx \mathbb{E}[R_i(a) | s_i]$. Given a total extra thinking budget B_{extra} , the batch objective is

$$\max_{a_1, \dots, a_N} \sum_{i=1}^N \hat{V}_i(a_i) \quad (9)$$

$$\text{s.t.} \quad \sum_{i=1}^N \hat{C}_i(a_i) \leq B_{\text{extra}}, \quad a_i \in \mathcal{A}. \quad (10)$$

This is a multiple-choice knapsack. For the binary ACCEPT/THINK instance with equal predicted costs, the constrained solution selects the highest-value examples. With variable predicted costs, we use a Lagrangian token-price policy or a value-per-token streaming heuristic; exact offline batch optimization can instead be treated as a 0–1 knapsack. For the general K -armed action space, we use the Lagrangian token-price view

$$a_i^\lambda = \arg \max_{a \in \mathcal{A}} [\hat{V}_i(a) - \lambda \hat{C}_i(a)], \quad (11)$$

with λ chosen to satisfy the target budget.

3.2 BORA-SWITCH-CALIBRATED: The Deployed Binary Instance

BORA-SWITCH-CALIBRATED is the deployed binary instance of BORA-BUDGET. It runs a cheap seed, extracts seed-state features, predicts opportunity and cost, and either preserves the seed or pays for thinking followed by a conservative adoption gate. Figure 1 summarizes the full control flow.

BORA-BUDGET: Budgeted Selective Thinking for Hybrid-Thinking LLMs

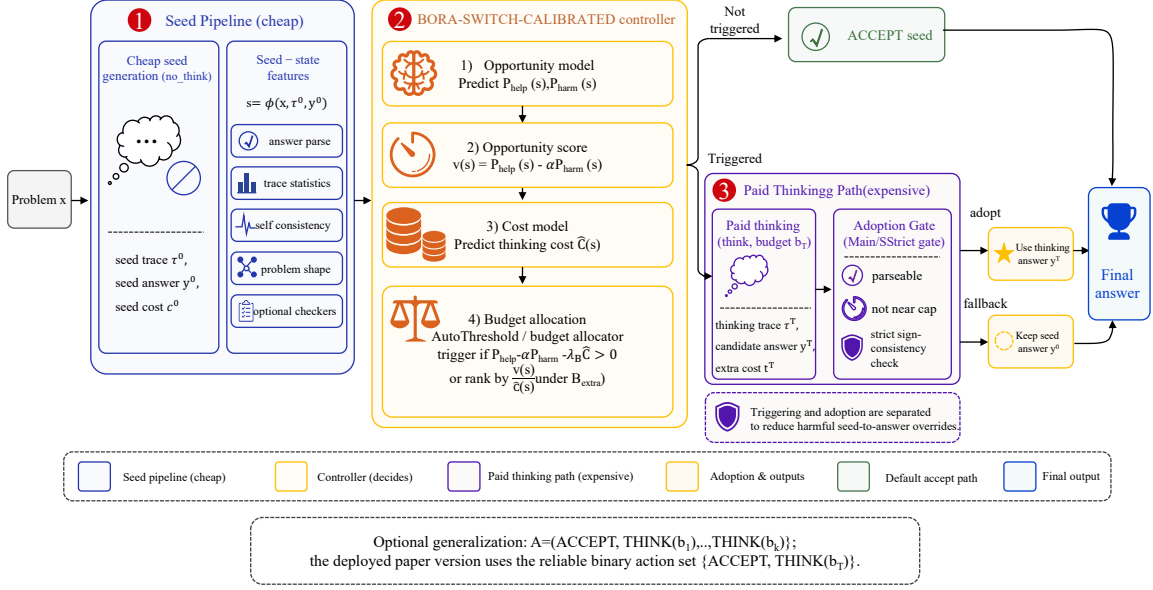


Figure 1: **BORA-BUDGET overview.** Each problem first receives a cheap /no_think seed answer, from which the controller extracts seed-state features. The opportunity model estimates helpfulness, harmfulness, and expected cost; the budget allocator then either accepts the seed or pays for a thinking candidate. Triggering and adoption are separated: even after a paid thinking call, the adoption gate may reject the candidate and preserve the seed. The general formulation supports multiple thinking budgets, while the deployed paper instance uses the reliable binary ACCEPT/THINK action set.

Seed-state features. We use five feature groups:

- **Answer parse features:** parse success, numeric value, sign, integer or decimal form, and multiple final-answer candidates.
- **Trace statistics:** seed token count, word and character counts, cap-hit indicators, and counts of numeric spans.
- **Seed self-consistency features:** counts of boxed or final-answer candidates in the seed trace, distinct candidate values, whether the stored prediction matches the first or last candidate, numeric spread among candidates, and correction or confidence markers.
- **Problem-shape features:** lexical indicators for algebra, geometry, number theory, fractions, comparisons, minima and maxima, and proof-like wording.
- **Optional independent checkers:** symbolic-answer format risks and conservative arithmetic consistency features extracted without gold labels or thinking outputs.

Trace length is therefore a feature, not the method itself. We use feature ablations to test whether learned allocation adds value beyond trace length.

Opportunity model. For the binary thinking arm, we train gradient-boosted decision-tree classifiers to estimate

$$P_{\text{help}}(s) = P(\hat{y}^0 \neq y, \hat{y}^T = y \mid s), \quad (12)$$

$$P_{\text{harm}}(s) = P(\hat{y}^0 = y, \hat{y}^T \neq y \mid s), \quad (13)$$

where $\hat{y}^T$ is the post-gate thinking answer. The opportunity value is

$$v(s) = P_{\text{help}}(s) - \alpha P_{\text{harm}}(s). \quad (14)$$

Calibration rollouts provide the labels for these probabilities. At inference time, the controller observes only the seed-state context; gold answers and thinking candidates are never used to compute the trigger score. The experimental protocol specifies whether calibration is task-local, problem-disjoint, or transferred across datasets.

3.3 Budget Allocation Variants

Frontier top- k . For analysis, we report top- ρ operating points: trigger the highest scoring ρ fraction of examples. These are useful for visualizing the accuracy-token frontier, but they are not the only deployable form of the algorithm.

Streaming threshold. To avoid relying on offline batch ranking, we also calibrate a scalar threshold on training seeds and apply it independently to held-out examples. This tests whether a fixed threshold can approximately maintain the intended budget on a held-out stochastic seed.

AutoThreshold under a total budget. The deployment policy is BORA-BUDGET-AutoThreshold. Given B_{extra} , the controller estimates a cost model $\hat{C}(s)$ for the paid thinking action and triggers if

$$P_{\text{help}}(s) - \alpha P_{\text{harm}}(s) - \lambda_B \hat{C}(s) > 0, \quad (15)$$

where λ_B is chosen so that the predicted extra thinking spend is near B_{extra} . Equivalently, for the binary action set, examples are ranked by

$$r_i = \frac{P_{\text{help}}(s_i) - \alpha P_{\text{harm}}(s_i)}{\hat{C}(s_i)} \quad (16)$$

and selected until the predicted budget is exhausted.

3.4 Adoption Gates

The *main* gate requires the thinking candidate to parse numerically and not be produced by a near-cap generation. The *strict* gate adds a sign consistency check to reduce sign-flip hazards. We report strict results as the main conservative operating points because they reduce observed harmful flips relative to raw override and expose the recall-risk tradeoff. This is an empirical filter, not a formal safety guarantee.

3.5 General K-Armed Extension

The same formulation supports

$$\mathcal{A} = \{\text{ACCEPT}, \text{THINK}(b_1), \text{THINK}(b_2), \dots, \text{THINK}(b_K)\}.$$

For each budget one can train arm-specific value and cost models and choose

$$a_i = \arg \max_{a \in \mathcal{A}} \hat{V}_i(a) - \lambda_B \hat{C}_i(a).$$

Appendix diagnostics show large oracle headroom for such multi-budget selection, but current learned arm selectors remain weaker than the binary controller; the main experiments therefore use the reliable $\{\text{ACCEPT}, \text{THINK}(b_T)\}$ instance.

4 Controller-Level Guarantees

We state lightweight guarantees for the controller layer. They do not assume that thinking is always useful, nor that selective thinking must outperform always-thinking. Instead, they formalize when a learned allocator approaches its oracle counterpart. Full proofs, plus knapsack and finite-class extensions, are in Appendix A.

Setup. Let $s = \phi(x, \tau^0, \hat{y}^0)$ be the seed-state context. For an action a , define the seed-relative correctness gain

$$G(s, a) = \mathbb{E}[\mathbf{1}\{\hat{y}^a = y\} - \mathbf{1}\{\hat{y}^0 = y\} \mid s],$$

the harmful-flip risk

$$H(s, a) = \Pr(\hat{y}^0 = y, \hat{y}^a \neq y \mid s),$$

and extra cost $C(s, a) = \mathbb{E}[c(a) - c^0 \mid s]$. With harmfulness penalty μ , write $V_\mu(s, a) = G(s, a) - \mu H(s, a)$.

Proposition 1 (Selective allocation advantage). *Consider the binary action set $\{\text{ACCEPT}, \text{THINK}\}$, fixed thinking cost, and trigger rate ρ . Let $v(s) = V_\mu(s, \text{THINK})$, let π_{top} trigger the top- ρ examples by true value, and let π_{rand} trigger independently with probability ρ . If q_ρ satisfies $\Pr(v(s) \geq q_\rho) = \rho$, then*

$$\begin{aligned} U(\pi_{\text{top}}) - U(\pi_{\text{rand}}) \\ = \rho \left(\mathbb{E}[v(s) \mid v(s) \geq q_\rho] - \mathbb{E}[v(s)] \right) \geq 0. \end{aligned}$$

The inequality is strict whenever the top- ρ region has strictly larger average value than the full population.

Corollary 1 (Top- ρ regret under score error). *Under the setting of Proposition 1, suppose a learned score $\hat{v}$ satisfies $|\hat{v}(s) - v(s)| \leq \epsilon$ uniformly. If $\hat{\pi}_{\text{top}}$ triggers the top- ρ examples by $\hat{v}$, then*

$$U(\pi_{\text{top}}) - U(\hat{\pi}_{\text{top}}) \leq 2\rho\epsilon.$$

Theorem 1 (AutoThreshold regret under estimation error). *For token price λ , define*

$$F_{\lambda, \mu}(s, a) = G(s, a) - \mu H(s, a) - \lambda C(s, a).$$

Let $\pi_{\lambda, \mu}^*$ maximize $F_{\lambda, \mu}$, and let $\hat{\pi}_{\lambda, \mu}$ maximize $\hat{F}_{\lambda, \mu}(s, a) = \hat{G}(s, a) - \mu \hat{H}(s, a) - \lambda \hat{C}(s, a)$. If, uniformly over s, a ,

$$|\hat{G} - G| \leq \epsilon_G, \quad |\hat{H} - H| \leq \epsilon_H, \quad |\hat{C} - C| \leq \epsilon_C,$$

then

$$\begin{aligned} & \mathbb{E}[F_{\lambda,\mu}(s, \pi_{\lambda,\mu}^*(s)) - F_{\lambda,\mu}(s, \hat{\pi}_{\lambda,\mu}(s))] \\ & \leq 2(\epsilon_G + \mu\epsilon_H + \lambda\epsilon_C). \end{aligned}$$

Proposition 2 (Conservative feasibility). *For a batch of contexts, suppose deployment uses upper confidence bounds $U_i^C(a)$ and $U_i^H(a)$ that cover true cost and harmful risk for every selected context-action pair. If the chosen allocation satisfies*

$$\sum_i U_i^C(a_i) \leq B, \quad \sum_i U_i^H(a_i) \leq \Gamma,$$

then, on the coverage event, true token cost is at most B , and the expected number of harmful flips is at most Γ .

Proposition 1 explains why opportunity-aware spending can beat random allocation; Corollary 1 shows that ranking error controls loss to the oracle top- ρ trigger; and Theorem 1 decomposes AutoThreshold regret into value, harmful-risk, and cost-estimation errors. Proposition 2 does not certify zero harmful flips, but it formalizes harmful overrides as a constrained resource.

5 Experiments

Setup. MATH500 (Hendrycks et al., 2021; Lightman et al., 2024) is the main benchmark; in our runs, it creates a large gap between cheap and thinking-mode reasoning. We evaluate Qwen3-4B, Qwen3-8B, and Qwen3-14B with explicit /no_think and thinking modes. All three model scales use three matched stochastic seeds for the main MATH500 selective-frontier analysis. Baselines include /no_think, uniform thinking, random triggers, trace-length triggers, and learned GBDT triggers. We report final-answer accuracy under the full-completion fixed evaluator, average total tokens, helpful rescues, harmful flips, and adoption statistics. Token count is the primary normalized cost metric; selective methods may not translate token savings directly into wall-clock savings because seed and thinking calls run sequentially on triggered examples.

We also include transfer and sanity checks on TheoremQA (Chen et al., 2023b), SVAMP (Patel et al., 2021), ARC-Challenge (Clark et al., 2018), CommonsenseQA (Talmor et al., 2019), MBPP (Austin et al., 2021), and GPQA Diamond (Rein et al., 2024). The main cross-task table

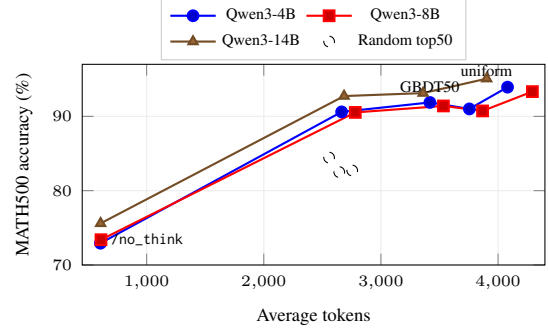


Figure 2: MATH500 accuracy–token frontier across model scales. Learned allocation moves substantially above /no_think and random triggering while using fewer average tokens than the strongest uniform thinking baselines.

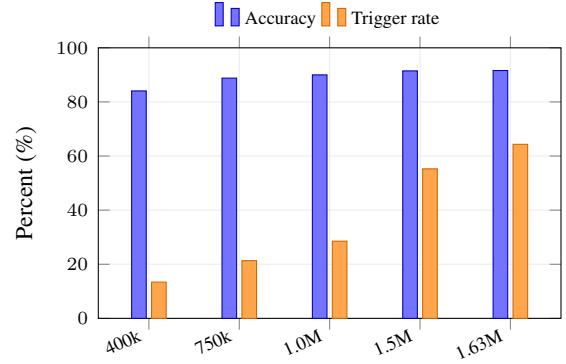


Figure 3: BORA-BUDGET under explicit extra-token budgets on Qwen3-8B MATH500. The allocator induces its trigger rate from the budget rather than from a hand-picked top- k fraction. Average total tokens for the five budgets are 1434.8, 2104.3, 2636.6, 3619.0, and 3866.8; harmful flips remain below one mean count per seed.

reports tasks for which we have matching accuracy, token, and harmful-flip summaries; additional math-transfer rows are kept in the appendix. Appendix B.8 gives implementation details for generation storage, answer extraction, trigger training, and budget allocation ¹.

5.1 Uniform Thinking Is Strong but Expensive

Table 1 shows the central tension. Across model scales, /no_think is cheap but far below uniform thinking. On Qwen3-8B, for example, /no_think reaches 73.40% at 610.2 tokens, whereas the strongest uniform thinking baseline reaches 93.33% at 4293.1 tokens. Qwen3-14B follows the same pat-

¹The code and artifacts are available at <https://anonymous.4open.science/r/BORA-BUDGET-E777>

Table 1: MATH500 cross-model accuracy–cost frontier under the fixed evaluator. Top30/top50 are frontier operating points of the same learned opportunity controller. All three model scales use three matched seeds. Harmful counts are observed seed-correct to final-wrong flips after strict adoption. All displayed rows report three-seed means and standard deviations; the medium-cap uniform baseline for Qwen3-14B was not completed and is therefore omitted rather than imputed.

Model	Method	Acc.	Std	Avg. tokens	Helpful	Harmful
Qwen3-4B	/no_think	72.93	0.46	605.6	–	–
	Always /think@8k	91.00	0.80	3754.9	–	–
	Always /think@12k	93.93	0.12	4080.9	–	–
	GBDT top30 strict	90.60	0.53	2664.6	266	1
	GBDT top50 strict	91.87	0.70	3418.1	286	2
	Random top50 strict	82.50	0.35	2644.4	145	1
Qwen3-8B	/no_think	73.40	0.53	610.2	–	–
	Always /think@8k	90.73	0.58	3867.8	–	–
	Always /think@12k	93.33	1.21	4293.1	–	–
	GBDT top30 strict	90.53	1.29	2781.0	258	1
	GBDT top50 strict	91.40	0.87	3533.5	272	2
	Random top50 strict	82.71	0.59	2755.1	140	0
Qwen3-14B	/no_think	75.60	0.72	607.0	–	–
	Always /think@12k	95.07	0.46	3902.7	–	–
	GBDT top30 strict	92.73	0.46	2685.3	257	0
	GBDT top50 strict	93.13	0.61	3360.1	265	2
	Trace top50 strict	92.33	0.70	3439.0	254	3
	Random top50 strict	84.45	0.50	2557.5	134	3

Table 2: Selected transfer and sanity checks beyond MATH500, now including cost and harmful-flip reporting. Results use task-appropriate evaluators. MBPP is a visible-test smoke test, not a hidden-test coding benchmark. GPQA uses an open mirror with a corrected boxed-first no-thinking prompt.

Dataset	Model	/no_think		Uniform		GBDT top50			Trace top50		
		Acc.	Tok.	Acc.	Tok.	Acc.	Tok.	Harm.	Acc.	Tok.	Harm.
ARC-Challenge	Qwen3-8B	92.67	271.9	94.67	1063.3	95.22	836.5	5	94.67	900.8	9
CommonsenseQA	Qwen3-8B	80.44	253.8	85.22	1109.5	84.89	840.6	10	83.89	917.2	19
MBPP-200 smoke	Qwen3-14B	82.50	51.9	86.00	1659.9	92.86	826.7	0	93.88	1210.6	0
GPQA Diamond	Qwen3-14B	40.91	105.8	58.59	5727.5	52.01	3015.6	0.4	49.01	3169.7	1.4

tern: three seeds average 75.60% for /no_think and 95.07% for uniform thinking, but the latter spends 3902.7 tokens on every problem. The model can reason much better when thinking is enabled; the system question is where to spend that budget.

5.2 BORA-SWITCH-CALIBRATED Improves the Cross-Model Frontier

BORA-SWITCH-CALIBRATED fills the middle of this accuracy–cost frontier. On Qwen3-4B, the top50 strict point reaches 91.87% at 3418.1 tokens, exceeding the lower-cost uniform thinking baseline while using fewer tokens and remaining below the strongest uniform-thinking baseline. On Qwen3-8B, top50 strict reaches 91.40% at 3533.5 tokens, again improving over the lower-cost uniform baseline under a smaller token budget. Qwen3-14B now provides method-level support rather than only baseline support: over three seeds, top50 strict reaches 93.13% at 3360.1 tokens, compared with 95.07% at 3902.7 tokens for uniform thinking and 84.45% for random top50. We therefore do not claim to beat the strongest always-thinking base-

line; the result is an accuracy–cost frontier that is consistent across model scales.

Figure 2 visualizes the same result. Learned triggering moves the frontier upward from /no_think and random allocation without spending the full uniform-thinking budget. Harmful flips are small but nonzero under the fixed evaluator, so the safety claim is empirical reduction rather than elimination. Because leave-one-seed-out training reuses the same problem identities across seeds, Appendix B.4 also reports qid-disjoint MATH500 folds. This stricter split weakens the learned-over-trace gap but preserves the central pattern: learned allocation remains far above random allocation and is competitive with the strong trace-length proxy.

5.3 Budget Allocation Removes Fixed Trigger Ratios

Top30 and top50 are useful for drawing a frontier, but deployment should accept a token budget and induce its own trigger rate. Figure 3 reports a Qwen3-8B budget-allocation sweep. With a 1.0M extra-token budget, the controller reaches 90.00%

at 2636.6 tokens with a 28.53% induced trigger rate. With a 1.5M budget, it reaches 91.47% at 3619.0 tokens with a 55.27% trigger rate. Near the lower-cost uniform-thinking budget, it reaches 91.60% at 3866.8 tokens. These rows support the algorithmic framing as token-price or value-per-token allocation rather than a hand-picked trigger-ratio heuristic.

The trigger diagnostic explains why the learned controller is useful: at top30, it covers 88.72% of seed-wrong examples while touching only 8.72% of seed-correct examples; at top50, seed-wrong coverage rises to 94.49% but seed-correct exposure increases to 33.88%. This is the expected efficiency–recall tradeoff. The full diagnostic table is in Appendix B.

5.4 Transfer and Diagnostic Tasks

Table 2 tests whether the same allocation idea gives useful signals beyond the main MATH500 setting. The answer is promising but uneven. On ARC-Challenge and CommonsenseQA, GBDT top50 improves over /no_think while using fewer tokens than uniform thinking, but harmful flips are nonzero and task-specific calibration matters. MBPP and GPQA are best read as sanity checks: MBPP uses visible tests rather than hidden tests, while GPQA required a prompt/evaluation audit for /no_think and remains far below uniform thinking. The Qwen3-14B GPQA row is still useful as a cost-sensitive signal: GBDT top50 improves over corrected /no_think by 11.10 points while using about half the tokens of uniform thinking. The table also shows why we do not overstate the learned trigger: trace length is a strong baseline on code and some hard-QA settings, so the safest claim is that learned allocation strongly beats random and is often competitive with or better than trace.

Additional feature, calibration, checker, GSM8K (Cobbe et al., 2021), AIME24 (Mathematical Association of America, 2026), and K-armed diagnostics are reported in Appendix B.

6 Conclusion

We introduced BORA-BUDGET, a budgeted opportunity-allocation framework for hybrid-thinking LLMs. Its deployed instance, BORA-SWITCH-CALIBRATED, first obtains cheap ACCEPT-baseline seeds, then allocates paid thinking only where predicted marginal value justifies cost and override risk. Empirically, learned allocation

improves strongly over random allocation, is competitive with strong trace-length triggers, approaches uniform-thinking accuracy under lower token budgets in the main calibrated setting, and uses strict adoption to reduce harmful seed overrides. AutoThreshold further shows that the policy can take an explicit token budget rather than a hand-picked trigger ratio. Future work should improve cross-task calibration and close the gap to K-armed oracles that choose how much to think.

7 Limitations

Allocation, not universal thinking. Always-thinking baselines are strong, but they do not answer the deployment question of where a finite thinking budget should be spent. BORA-BUDGET does not claim to beat the strongest uniform-thinking baseline or significantly exceed the lower-cost uniform-thinking baseline in raw accuracy. Its contribution is an accuracy–cost–safety frontier: learned allocation identifies high-value thinking opportunities, and AutoThreshold converts a total token budget into an induced trigger rate.

Safety is an adoption problem. The experiments show that deciding to spend tokens and deciding to trust the result should be separated. Strict adoption reduces harmful seed-to-wrong overrides, but harmful flips remain nonzero in the fixed-evaluator cross-model runs. The appropriate claim is therefore empirical risk control, not formal safety. Strict adoption also does not eliminate wrong-to-wrong answer changes, motivating future gates with independent arithmetic, symbolic, or verifier checks.

Calibration remains task-dependent. The strongest results use task-aware calibration. Cross-model experiments show that the allocation pattern can persist across model scales, but cross-task transfer is uneven and sometimes sensitive to prompting and evaluation design. Trace length is also a surprisingly strong proxy on code and some hard-QA settings, suggesting that future controllers should combine learned opportunity scores with task-specific difficulty proxies. The current claim is therefore budgeted allocation with task-aware calibration, not universal zero-shot trigger transfer. Appendix diagnostics further show that K-armed oracles reveal large headroom for learning how much to think, while current learned budget-arm selectors are not yet strong enough to

replace the simpler binary controller.

Acknowledgments

This research was supported by the AI Laboratory of Xiaomi Communications Co., Ltd.

References

- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021. [Program synthesis with large language models](#). *arXiv preprint arXiv:2108.07732*.
- Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. 2013. [Bandits with knapsacks](#). In *IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216.
- Lingjiao Chen, Matei Zaharia, and James Zou. 2024. [FrugalGPT: How to use large language models while reducing cost and improving performance](#). *Transactions on Machine Learning Research*. Featured Certification.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023a. [Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks](#). *Transactions on Machine Learning Research*.
- Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023b. [TheoremQA: A theorem-driven question answering dataset](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7901, Singapore. Association for Computational Linguistics.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. [Think you have solved question answering? try ARC, the AI2 reasoning challenge](#). *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. [Thinkless: Llm learns when to think](#). In *Advances in Neural Information Processing Systems*.
- Alex Graves. 2016. [Adaptive computation time for recurrent neural networks](#). *arXiv preprint arXiv:1603.08983*.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. [Token-budget-aware LLM reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855, Vienna, Austria. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. [Large language models are zero-shot reasoners](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *International Conference on Learning Representations*.
- Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. 2025. [Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning](#). *Preprint*, arXiv:2505.11896.
- Mathematical Association of America. 2026. [American invitational mathematics examination \(AIME\)](#). MAA Invitational Competitions. Accessed August 30, 2026.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. [s1: Simple test-time scaling](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20275–20321, Suzhou, China. Association for Computational Linguistics.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2025. [Routellm: Learning to route llms from preference data](#). In *International Conference on Learning Representations*.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. [Are NLP models really able to solve simple math word problems?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. [GPQA: A graduate-level google-proof q&a benchmark](#). In *First Conference on Language Modeling*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. [Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning](#). In *The Thirteenth International Conference on Learning Representations*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. [CommonsenseQA: A question answering challenge targeting commonsense knowledge](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. 2025. [Budgetthinker: Empowering budget-aware llm reasoning with control tokens](#). Preprint, arXiv:2508.17196.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). arXiv preprint arXiv:2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. [Tree of thoughts: Deliberate problem solving with large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2024. [Large language model cascades with mixture of thought representations for cost-efficient reasoning](#). In *The Twelfth International Conference on Learning Representations*.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025. [AdaptThink: Reasoning models can learn when to think](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3716–3730, Suzhou, China. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). In *International Conference on Learning Representations*.

A Proofs for Controller-Level Guarantees

This appendix provides full proofs for the controller-level guarantees stated in Section 4, together with two additional results: a batch knapsack view for the K -armed extension and a finite-class constrained policy-selection guarantee. The results do not assume that thinking is always useful, nor that a selective controller must outperform an always-thinking baseline. Instead, they justify the allocation layer: if marginal value, harmful risk, and token cost can be estimated from offline calibration rollouts, then the learned controller approaches the corresponding oracle budgeted allocation policy up to estimation error.

Setup. Let $s = \phi(x, \tau^0, \hat{y}^0)$ denote the seed-state context produced by the cheap seed policy. The action set is

$$\mathcal{A} = \{\text{ACCEPT}, \text{THINK}(b_1), \dots, \text{THINK}(b_K)\}.$$

The deployed binary instance uses

$$\mathcal{A}_{\text{bin}} = \{\text{ACCEPT}, \text{THINK}(b_T)\}.$$

For an action $a \in \mathcal{A}$, let $\hat{y}^a$ be the final answer after the action and the adoption gate. Define the conditional marginal correctness gain

$$G(s, a) = \mathbb{E}[\mathbf{1}\{\hat{y}^a = y\} - \mathbf{1}\{\hat{y}^0 = y\} \mid s],$$

the conditional harmful-flip risk

$$H(s, a) = \mathbb{P}(\hat{y}^0 = y, \hat{y}^a \neq y \mid s),$$

and the conditional extra token cost

$$C(s, a) = \mathbb{E}[c(a) - c^0 \mid s].$$

For the accept action,

$$G(s, \text{ACCEPT}) = 0, \tag{17}$$

$$H(s, \text{ACCEPT}) = 0, \tag{18}$$

$$C(s, \text{ACCEPT}) = 0. \tag{19}$$

Given a harmful-flip penalty $\mu \geq 0$, define the seed-relative utility

$$V_\mu(s, a) = G(s, a) - \mu H(s, a).$$

For the binary setting, this corresponds to the paper’s score $P_{\text{help}} - \alpha P_{\text{harm}}$, up to the choice of harmful-flip penalty.

A.1 Why Selective Thinking Can Beat Random Allocation

The first result formalizes the basic opportunity-allocation intuition. When the marginal value of thinking is heterogeneous across examples, the best selective trigger at a fixed trigger rate is better than random spending.

Proposition 3 (Selective allocation advantage). *Consider the binary action set $\{\text{ACCEPT}, \text{THINK}\}$, and suppose the thinking action has a fixed extra cost. Let*

$$v(s) = V_\mu(s, \text{THINK})$$

be the true seed-relative value of triggering thinking. For a target trigger rate $\rho \in (0, 1)$, let q_ρ be a threshold such that

$$\mathbb{P}(v(s) \geq q_\rho) = \rho,$$

ignoring ties for simplicity. Let π_{top} trigger exactly on the top- ρ examples:

$$\pi_{\text{top}}(s) = \begin{cases} \text{THINK}, & v(s) \geq q_\rho, \\ \text{ACCEPT}, & \text{otherwise.} \end{cases}$$

Let π_{rand} trigger independently with probability ρ , independent of s . Then with $m_\rho = \mathbb{E}[v(s) \mid v(s) \geq q_\rho]$,

$$U(\pi_{\text{top}}) - U(\pi_{\text{rand}}) = \rho(m_\rho - \mathbb{E}[v(s)]) \geq 0,$$

where $U(\pi) = \mathbb{E}[V_\mu(s, \pi(s))]$. The inequality is strict whenever the top- ρ region has strictly larger average value than the full population.

Proof. The random policy triggers with probability ρ , independently of the context, so

$$U(\pi_{\text{rand}}) = \rho \mathbb{E}[v(s)].$$

The top- ρ policy triggers exactly on the event $v(s) \geq q_\rho$, so

$$\begin{aligned} U(\pi_{\text{top}}) &= \mathbb{E}[v(s) \mathbf{1}\{v(s) \geq q_\rho\}] \\ &= \rho \mathbb{E}[v(s) \mid v(s) \geq q_\rho]. \end{aligned} \quad (20)$$

Subtracting the two expressions proves the identity. Since the top- ρ region contains the largest values of $v(s)$, its conditional mean is at least the population mean. $\square$

The next corollary shows that a learned score only needs to rank examples approximately well in order to approach the oracle top- ρ trigger.

Corollary 2 (Top- ρ regret under value-estimation error). *Let $\hat{v}(s)$ be a learned score satisfying*

$$|\hat{v}(s) - v(s)| \leq \epsilon \quad \text{for all } s.$$

Let $\hat{\pi}_{\text{top}}$ trigger on the top- ρ examples according to $\hat{v}(s)$, and let π_{top} be the oracle top- ρ policy according to $v(s)$. Then

$$U(\pi_{\text{top}}) - U(\hat{\pi}_{\text{top}}) \leq 2\rho\epsilon.$$

Proof. Let S^* be the oracle top- ρ trigger set and let $\hat{S}$ be the learned top- ρ trigger set. Both have probability mass ρ . Since $\hat{S}$ maximizes the estimated score among all sets of mass ρ ,

$$\mathbb{E}[\hat{v}(s) \mathbf{1}\{s \in \hat{S}\}] \geq \mathbb{E}[\hat{v}(s) \mathbf{1}\{s \in S^*\}].$$

Using $|\hat{v} - v| \leq \epsilon$, we obtain

$$\begin{aligned} \mathbb{E}[v(s) \mathbf{1}\{s \in S^*\}] &\leq \mathbb{E}[\hat{v}(s) \mathbf{1}\{s \in S^*\}] + \rho\epsilon \\ &\leq \mathbb{E}[\hat{v}(s) \mathbf{1}\{s \in \hat{S}\}] + \rho\epsilon \\ &\leq \mathbb{E}[v(s) \mathbf{1}\{s \in \hat{S}\}] + 2\rho\epsilon. \end{aligned} \quad (21)$$

This is exactly the desired utility-regret bound. $\square$

Implication. Proposition 3 explains why selective thinking can improve over random token spending: the gain comes from heterogeneity in the marginal value of thinking. Corollary 2 then says that a learned trigger approaches the oracle trigger whenever its opportunity score is accurate enough to preserve the high-value region.

A.2 AutoThreshold Under Estimation Error

The deployed BORA-BUDGET policy is not limited to a fixed trigger ratio. Given a token price $\lambda \geq 0$, it chooses the action with the largest estimated Lagrangian value. This is the population analogue of the AutoThreshold rule

$$P_{\text{help}}(s) - \alpha P_{\text{harm}}(s) - \lambda \hat{C}(s) > 0.$$

For action a , define the true token-price objective

$$F_{\lambda, \mu}(s, a) = G(s, a) - \mu H(s, a) - \lambda C(s, a),$$

and its learned estimate

$$\hat{F}_{\lambda, \mu}(s, a) = \hat{G}(s, a) - \mu \hat{H}(s, a) - \lambda \hat{C}(s, a).$$

Let

$$\pi_{\lambda, \mu}^*(s) \in \arg \max_{a \in \mathcal{A}} F_{\lambda, \mu}(s, a)$$

be the oracle token-price policy, and let

$$\hat{\pi}_{\lambda, \mu}(s) \in \arg \max_{a \in \mathcal{A}} \hat{F}_{\lambda, \mu}(s, a)$$

be the learned AutoThreshold policy.

Theorem 2 (AutoThreshold regret under estimation error). Write $\pi^* = \pi_{\lambda,\mu}^*$ and $\hat{\pi} = \hat{\pi}_{\lambda,\mu}$. Assume that, for all s and all $a \in \mathcal{A}$,

$$|\hat{G}(s, a) - G(s, a)| \leq \epsilon_G,$$

$$|\hat{H}(s, a) - H(s, a)| \leq \epsilon_H,$$

and

$$|\hat{C}(s, a) - C(s, a)| \leq \epsilon_C.$$

Let

$$\Delta_{\lambda,\mu}(s) = F_{\lambda,\mu}(s, \pi^*(s)) - F_{\lambda,\mu}(s, \hat{\pi}(s)).$$

Then

$$\mathbb{E}[\Delta_{\lambda,\mu}(s)] \leq 2(\epsilon_G + \mu\epsilon_H + \lambda\epsilon_C).$$

Proof. For any action a , the estimation assumptions imply

$$\left| \hat{F}_{\lambda,\mu}(s, a) - F_{\lambda,\mu}(s, a) \right| \leq \epsilon_G + \mu\epsilon_H + \lambda\epsilon_C.$$

Let

$$\epsilon_\lambda = \epsilon_G + \mu\epsilon_H + \lambda\epsilon_C.$$

For a fixed context s , write $a^* = \pi_{\lambda,\mu}^*(s)$ and $\hat{a} = \hat{\pi}_{\lambda,\mu}(s)$. Since $\hat{a}$ maximizes $\hat{F}_{\lambda,\mu}$,

$$\hat{F}_{\lambda,\mu}(s, \hat{a}) \geq \hat{F}_{\lambda,\mu}(s, a^*).$$

Therefore,

$$\begin{aligned} F_{\lambda,\mu}(s, a^*) &\leq \hat{F}_{\lambda,\mu}(s, a^*) + \epsilon_\lambda \\ &\leq \hat{F}_{\lambda,\mu}(s, \hat{a}) + \epsilon_\lambda \\ &\leq F_{\lambda,\mu}(s, \hat{a}) + 2\epsilon_\lambda. \end{aligned} \quad (22)$$

Taking expectation over s proves the result. $\square$

Implication. Theorem 2 directly analyzes the controller used by BORA-BUDGET. If the learned helpfulness, harmfulness, and cost models are accurate, then the learned token-price policy is close to the oracle token-price policy. The regret decomposes additively into three interpretable sources: value-estimation error, harmful-risk estimation error, and cost-model error.

A.3 Conservative Budget and Harm Control

The previous theorem analyzes the token-price objective. We next show how conservative upper bounds on cost and harmful risk can turn the same allocation view into high-probability budget and risk control.

Consider a batch of N contexts $s_1, \dots, s_N$. For each context-action pair, suppose the controller has an upper confidence bound on cost, $\mathcal{U}_i^C(a)$, and an upper confidence bound on harmful risk, $\mathcal{U}_i^H(a)$. Let $Z_i(a) \in \{0, 1\}$ denote the realized harmful flip indicator if action a is selected for example i , and let

$$h_i(a) = \mathbb{E}[Z_i(a) \mid s_i]$$

be its conditional harmful-flip probability.

Theorem 3 (Conservative budget and harmful-risk feasibility). Fix a batch $s_1, \dots, s_N$. Suppose that with probability at least $1 - \delta_{\text{cov}}$, the following coverage event holds simultaneously for all i and a :

$$C_i(a) \leq \mathcal{U}_i^C(a), \quad h_i(a) \leq \mathcal{U}_i^H(a).$$

Let the controller choose actions $a_1, \dots, a_N$ satisfying the conservative constraints

$$\sum_{i=1}^N \mathcal{U}_i^C(a_i) \leq B$$

and

$$\sum_{i=1}^N \mathcal{U}_i^H(a_i) \leq \Gamma.$$

Then, on the coverage event,

$$\sum_{i=1}^N C_i(a_i) \leq B$$

and

$$\mathbb{E} \left[\sum_{i=1}^N Z_i(a_i) \mid s_1, \dots, s_N \right] \leq \Gamma.$$

If, additionally, the harmful indicators are conditionally independent given the contexts and selected actions, then for any $\delta \in (0, 1)$, with probability at least $1 - \delta_{\text{cov}} - \delta$,

$$\sum_{i=1}^N Z_i(a_i) \leq \Gamma + \sqrt{2\Gamma \log(1/\delta)} + \frac{1}{3} \log(1/\delta).$$

Proof. On the coverage event,

$$\sum_{i=1}^N C_i(a_i) \leq \sum_{i=1}^N \mathcal{U}_i^C(a_i) \leq B.$$

Similarly,

$$\sum_{i=1}^N h_i(a_i) \leq \sum_{i=1}^N \mathcal{U}_i^H(a_i) \leq \Gamma.$$

Since

$$\mathbb{E}[Z_i(a_i) \mid s_i, a_i] = h_i(a_i),$$

we have

$$\mathbb{E}\left[\sum_{i=1}^N Z_i(a_i) \mid s_1, \dots, s_N\right] = \sum_{i=1}^N h_i(a_i) \leq \Gamma.$$

The final high-probability bound follows from Bernstein's inequality for a sum of conditionally independent Bernoulli variables whose total conditional mean is at most Γ , combined with the coverage event by a union bound. $\square$

Implication. Theorem 3 gives a formal interpretation of conservative deployment. If the controller uses upper confidence bounds for token cost and harmful risk, then budget feasibility holds on the cost-coverage event, and harmful flips are controlled in expectation, with a standard high-probability deviation bound for realized flips. This does not imply that zero harmful flips can be certified from a finite test set. Instead, it formalizes how harmful overrides can be treated as a constrained resource.

A.4 Batch Knapsack View

For equal-cost binary triggering, selecting the top-scoring examples is the exact constrained solution at a fixed trigger count. With variable token costs, value-per-token ranking should be interpreted as a Lagrangian or streaming threshold policy rather than an exact solution to the finite 0–1 knapsack. In an offline batch setting, one may instead solve the estimated multiple-choice knapsack directly.

For a batch, define the true batch utility

$$U(a_1, \dots, a_N) = \sum_{i=1}^N V_\mu(s_i, a_i),$$

and true batch cost

$$C(a_1, \dots, a_N) = \sum_{i=1}^N C_i(a_i).$$

Let $\widehat{V}_{\mu,i}(a)$ and $\widehat{C}_i(a)$ be empirical estimates.

Proposition 4 (Estimated knapsack feasibility under cost tightening). *Suppose that for all i, a ,*

$$|\widehat{V}_{\mu,i}(a) - V_\mu(s_i, a)| \leq \epsilon_V, \quad (23)$$

$$|\widehat{C}_i(a) - C_i(a)| \leq \epsilon_C. \quad (24)$$

Let $\widehat{a}_1, \dots, \widehat{a}_N$ solve the estimated tightened problem

$$\max_{a_1, \dots, a_N} \sum_{i=1}^N \widehat{V}_{\mu,i}(a_i) \quad (25)$$

$$\text{s.t.} \quad \sum_{i=1}^N \widehat{C}_i(a_i) \leq B - N\epsilon_C. \quad (26)$$

Then the selected allocation is feasible for the true budget:

$$\sum_{i=1}^N C_i(\widehat{a}_i) \leq B.$$

Moreover, if $a_1^*, \dots, a_N^*$ is any allocation satisfying the stricter true budget

$$\sum_{i=1}^N C_i(a_i^*) \leq B - 2N\epsilon_C,$$

then

$$\sum_{i=1}^N V_\mu(s_i, a_i^*) - \sum_{i=1}^N V_\mu(s_i, \widehat{a}_i) \leq 2N\epsilon_V.$$

Proof. The estimated cost constraint gives

$$\sum_{i=1}^N \widehat{C}_i(\widehat{a}_i) \leq B - N\epsilon_C.$$

Using $C_i(a) \leq \widehat{C}_i(a) + \epsilon_C$, we get

$$\sum_{i=1}^N C_i(\widehat{a}_i) \leq \sum_{i=1}^N \widehat{C}_i(\widehat{a}_i) + N\epsilon_C \leq B.$$

Now consider any allocation $a_1^*, \dots, a_N^*$ satisfying

$$\sum_{i=1}^N C_i(a_i^*) \leq B - 2N\epsilon_C.$$

Since $\widehat{C}_i(a) \leq C_i(a) + \epsilon_C$,

$$\sum_{i=1}^N \widehat{C}_i(a_i^*) \leq \sum_{i=1}^N C_i(a_i^*) + N\epsilon_C \leq B - N\epsilon_C.$$

Thus a^* is feasible for the estimated tightened problem. By optimality of $\widehat{a}$ for the estimated problem,

$$\sum_{i=1}^N \widehat{V}_{\mu,i}(\widehat{a}_i) \geq \sum_{i=1}^N \widehat{V}_{\mu,i}(a_i^*).$$

Using the value-estimation error on both sides,

$$\begin{aligned} \sum_{i=1}^N V_{\mu}(s_i, a_i^*) &\leq \sum_{i=1}^N \widehat{V}_{\mu,i}(a_i^*) + N\epsilon_V \\ &\leq \sum_{i=1}^N \widehat{V}_{\mu,i}(\widehat{a}_i) + N\epsilon_V \\ &\leq \sum_{i=1}^N V_{\mu}(s_i, \widehat{a}_i) + 2N\epsilon_V. \end{aligned} \quad (27)$$

This proves the utility bound. $\square$

Implication. Proposition 4 connects the general $\{\text{ACCEPT}, \text{THINK}(b_1), \dots, \text{THINK}(b_K)\}$ formulation to a finite-batch multiple-choice knapsack. It also clarifies the role of K-armed oracle diagnostics: if arm-specific value and cost estimates were accurate, the budgeted selector would approach the oracle allocation; the remaining gap is therefore an estimation and ranking challenge, not a formulation issue.

A.5 Finite-Class Policy Selection

The previous results analyze top- ρ , token-price, and batch-knapsack controllers directly. We now give a more general finite-class guarantee that covers top- k frontiers, threshold grids, and finite collections of K-armed budget policies.

Let Π be a finite class of policies $\pi : s \mapsto a$. Define the bounded seed-relative reward

$$\begin{aligned} \widetilde{R}(s, a) &= \mathbf{1}\{\hat{y}^a = y\} - \mathbf{1}\{\hat{y}^0 = y\} \\ &\quad - \mu \mathbf{1}\{\hat{y}^0 = y, \hat{y}^a \neq y\}. \end{aligned} \quad (28)$$

For a policy π , let

$$\begin{aligned} U(\pi) &= \mathbb{E}[\widetilde{R}(s, \pi(s))], \\ C(\pi) &= \mathbb{E}[C(s, \pi(s))], \end{aligned}$$

and

$$H(\pi) = \mathbb{E}[\mathbf{1}\{\hat{y}^0 = y, \hat{y}^{\pi(s)} \neq y\}].$$

Suppose a calibration set contains n i.i.d. examples and, for every example and every action $a \in \mathcal{A}$, counterfactual labels for $\widetilde{R}(s, a)$, $C(s, a)$, and the harmful indicator. For any $\pi \in \Pi$, let $\widehat{U}(\pi)$, $\widehat{C}(\pi)$, and $\widehat{H}(\pi)$ be the corresponding empirical averages.

The empirical controller chooses

$$\widehat{\pi} \in \arg \max_{\pi \in \Pi} \widehat{U}(\pi)$$

subject to the slackened empirical constraints

$$\widehat{C}(\pi) \leq B - \epsilon_C, \quad \widehat{H}(\pi) \leq \gamma - \epsilon_H.$$

Theorem 4 (Finite-class constrained policy selection). Assume that for all s, a ,

$$\widetilde{R}(s, a) \in [-M, M], \quad (29)$$

$$C(s, a) \in [0, C_{\max}], \quad (30)$$

$$\mathbf{1}\{\hat{y}^0 = y, \hat{y}^a \neq y\} \in [0, 1]. \quad (31)$$

For any $\delta \in (0, 1)$, define

$$\epsilon_U = 2M \sqrt{\frac{\log(6|\Pi|/\delta)}{2n}},$$

$$\epsilon_C = C_{\max} \sqrt{\frac{\log(6|\Pi|/\delta)}{2n}},$$

and

$$\epsilon_H = \sqrt{\frac{\log(6|\Pi|/\delta)}{2n}}.$$

Let π^* be the best policy in Π satisfying the stricter true constraints

$$C(\pi^*) \leq B - 2\epsilon_C, \quad H(\pi^*) \leq \gamma - 2\epsilon_H.$$

Then, with probability at least $1 - \delta$,

$$C(\widehat{\pi}) \leq B, \quad H(\widehat{\pi}) \leq \gamma,$$

and

$$U(\pi^*) - U(\widehat{\pi}) \leq 2\epsilon_U.$$

Proof. Fix any $\pi \in \Pi$. Hoeffding's inequality gives

$$|\widehat{U}(\pi) - U(\pi)| \leq \epsilon_U$$

with failure probability at most $\delta/(3|\Pi|)$. The same argument gives

$$|\widehat{C}(\pi) - C(\pi)| \leq \epsilon_C$$

and

$$|\widehat{H}(\pi) - H(\pi)| \leq \epsilon_H$$

with the same failure probability for cost and harmful flips. A union bound over all policies and all three quantities implies that these inequalities hold simultaneously for every $\pi \in \Pi$ with probability at least $1 - \delta$.

On this event, empirical feasibility of $\widehat{\pi}$ implies

$$C(\widehat{\pi}) \leq \widehat{C}(\widehat{\pi}) + \epsilon_C \leq B$$

and

$$H(\widehat{\pi}) \leq \widehat{H}(\widehat{\pi}) + \epsilon_H \leq \gamma.$$

Moreover, the stricter true constraints on π^* imply

$$\widehat{C}(\pi^*) \leq B - \epsilon_C, \quad \widehat{H}(\pi^*) \leq \gamma - \epsilon_H,$$

so π^* is feasible for the empirical optimization. Since $\hat{\pi}$ maximizes empirical utility over the empirical feasible set,

$$\hat{U}(\hat{\pi}) \geq \hat{U}(\pi^*).$$

Applying the uniform utility bound twice,

$$U(\pi^*) \leq \hat{U}(\pi^*) + \epsilon_U \leq \hat{U}(\hat{\pi}) + \epsilon_U \leq U(\hat{\pi}) + 2\epsilon_U.$$

This proves the theorem. $\square$

Discussion. The theory supports the controller layer of BORA-BUDGET. Proposition 3 shows why opportunity-aware selection improves over random allocation when thinking value is heterogeneous. Theorem 2 shows that AutoThreshold approaches the oracle token-price policy up to value, cost, and harmful-risk estimation errors. Theorem 3 explains how conservative upper bounds can enforce token-budget feasibility and control harmful-risk exposure. Proposition 4 connects the K-armed extension to multiple-choice knapsack allocation. Finally, Theorem 4 gives a general finite-class empirical-selection guarantee for top- k , threshold-grid, and finite budgeted-policy classes.

These results are intentionally controller-level guarantees. They do not prove that thinking is always beneficial, that selective thinking must outperform the strongest always-thinking baseline, or that a finite evaluation with zero observed harmful flips is a formal safety certificate. Instead, they formalize the central claim of BORA-BUDGET: given reliable calibration rollouts, a seed-preserving controller can allocate paid thinking actions under token and harmful-risk constraints, and its loss relative to the corresponding oracle allocation is controlled by estimation error.

B Additional Diagnostics and Reproducibility

B.1 Main-Text Scope

We intentionally keep the main paper focused on fixed-evaluator MATH500 (Hendrycks et al., 2021; Lightman et al., 2024) budgeted thinking allocation. Conservative BORA on GSM8K (Cobbe et al., 2021) gives a small directionally positive signal but is not statistically decisive, so it is used here as a baseline-preservation diagnostic. Additional cross-task details, GPQA prompt-audit notes, checker features, high-cap thinking, and learned K-armed selectors are reported as diagnostics rather than central claims.

B.2 GSM8K Conservative BORA

Table 4 summarizes five matched GSM8K (Cobbe et al., 2021) test seeds. Conservative BORA improves pooled correctness from 6132/6595 for direct CoT and 6129/6595 for strongseed to 6150/6595, but paired bootstrap intervals cross zero. The action diagnostic is more important: verifier adoption produces all ten helpful rescues, while branch overrides produce three of four harmful flips. This motivates the seed-preserving formulation and the later adoption gate.

B.3 Fixed-Evaluator MATH500 Details

Table 5 expands the main MATH500 (Hendrycks et al., 2021; Lightman et al., 2024) table with trace and random allocation baselines. These rows use the same fixed evaluator as the main text. The learned trigger consistently improves over random allocation and is often competitive with or better than trace-length allocation, although trace length remains a strong baseline.

B.4 Problem-Disjoint MATH500 Robustness

Our main protocol on MATH500 (Hendrycks et al., 2021; Lightman et al., 2024) is leave-one-seed-out: it tests whether a trigger calibrated on two stochastic seeds transfers to a held-out seed. This is the appropriate task-aware calibration setting for the main frontier, but it does reuse the same problem identities across seeds. To audit problem-level generalization, Table 6 instead splits MATH500 by qid into five folds, trains the trigger only on disjoint qids, and evaluates on the held-out qids. Under this stricter protocol, GBDT remains well above random allocation across all three model scales. The learned-over-trace gap is smaller and sometimes nearly tied, confirming that trace length is a strong difficulty proxy; we therefore treat qid-disjoint results as robustness evidence for non-random opportunity allocation rather than as proof that learned triggers universally dominate trace heuristics.

B.5 Cross-Task and Checker Diagnostics

Table 8 gives additional transfer rows for TheoremQA (Chen et al., 2023b), SVAMP (Patel et al., 2021), ARC-Challenge (Clark et al., 2018), CommonsenseQA (Talmor et al., 2019), MBPP (Austin et al., 2021), and GPQA Diamond (Rein et al., 2024). The non-math multiple-choice tasks show useful transfer, but harmful flips are nonzero and task-specific calibration matters. GPQA uses the corrected boxed-first no-thinking prompt described

in the main text. Optional symbolic and arithmetic checker features were explored as refinements; they are not promoted to the frozen main method because the full cross-model analysis uses the simpler seed-state trigger.

B.6 Multi-Budget and Hard-Math Diagnostics

The optional K-armed extension over ACCEPT and multiple thinking budgets is useful as a formulation and oracle diagnostic: it shows headroom if the system could identify the smallest useful budget. We do not promote those learned selectors to the main table because the current fixed-evaluator cross-model story is stronger and cleaner for the binary controller.

High-cap always-thinking diagnostics show that more thinking budget can still improve raw accuracy in the legacy evaluator, but they are not part of the fixed-evaluator main claim. On our AIME24 diagnostic (Mathematical Association of America, 2026), thinking improves hard-math accuracy, but the trigger is nearly always-on there, so AIME24 is not evidence of selective control.

B.7 Artifact, Compute, and Ethics Details

Calibration protocols. The main MATH500 experiments use task-aware leave-one-seed-out calibration: for each held-out stochastic seed, the trigger is trained on the other two seeds and evaluated only on the held-out seed. This protocol tests stochastic-seed generalization under a fixed benchmark distribution. Transfer tasks follow the same principle when multiple seeds are available. We do not claim that a single universal trigger transfers zero-shot across all tasks. When a setting uses a dataset-local trigger, problem-disjoint split, or diagnostic/smoke-test protocol, we label it accordingly.

Artifact licenses and terms of use. We use publicly released model checkpoints, benchmark datasets, and evaluation tools according to their respective licenses and terms of use. The Qwen3 checkpoints are used only for inference; we do not fine-tune, modify, or redistribute model weights. Benchmark datasets are used only for research evaluation, and we report aggregate metrics rather than redistributing dataset contents. For resources with access conditions or mirrors, such as GPQA Diamond, we disclose the source used in the experimental setup and restrict usage to evaluation and aggregate reporting.

Artifact use and intended use. We use all existing artifacts consistently with their intended research and evaluation purposes. The model checkpoints are used as frozen inference models. The benchmark datasets are used to evaluate model reasoning, question answering, and code generation performance. Derived artifacts, including model outputs, opportunity rows, trigger labels, and summary tables, are used only to analyze budgeted inference behavior and reproduce aggregate results. We do not use research-only datasets or derivatives outside the research context.

Data content, privacy, and offensive content. We do not collect new human-subject data. All experiments use publicly released research benchmarks for mathematical reasoning, multiple-choice question answering, hard QA, and code generation. These datasets are intended for aggregate model evaluation and are not expected to contain personal user data. During evaluator and prompt development, we inspected benchmark metadata and sampled examples for obvious personally identifying information or offensive content. We report only aggregate metrics and diagnostic counts in the paper. Stored model outputs and opportunity rows are used only for research analysis and reproducibility, and we do not release or display information intended to identify individual people.

Compute infrastructure and budget. All experiments use frozen Qwen3 checkpoints. Qwen3-4B and Qwen3-8B inference used single-GPU workers, while Qwen3-14B used multi-GPU tensor parallelism when needed. The experiments were run on NVIDIA RTX 3090 GPUs. Across rollout generation, uniform-thinking baselines, transfer experiments, and offline trigger/budget analyses. Token counts are reported as the primary normalized compute metric in the main tables, while wall-clock latency depends on batching, hardware, and deployment setup.

Package versions and implementation parameters. The learned trigger uses scikit-learn HistGradientBoostingClassifier models for P_{help} and P_{harm} , with `max_iter=200`, learning rate 0.05, and ℓ_2 regularization 0.01. Cost prediction uses HistGradientBoostingRegressor with the same iteration, learning-rate, and regularization settings. The fixed evaluator uses normalization rules, `math_verify` when available, and SymPy simplification as a fallback. The released

Setting	Training data	Evaluation data	Claim supported
MATH500 LOSO	two seeds, same benchmark	held-out seed	main calibrated frontier
MATH500 qid-disjoint	train qids	held-out qids	robustness check
Cross-model MATH500	model-specific seeds	held-out seed	protocol transfers across model scales
ARC/CommonsenseQA	task-local seeds	held-out seed	non-math transfer signal
MBPP-200	task-local seed features	held-out seed	code-pipeline smoke test
GPQA Diamond	corrected prompt, task-local rows	held-out seed	hard-QA sanity check

Table 3: Calibration and evaluation protocols. The main claim is task-aware budgeted allocation, not universal zero-shot trigger transfer.

repository provides the rollout-construction scripts, answer-evaluation scripts, trigger-frontier analysis scripts, fixed-evaluator budget-allocation scripts, and exact environment/package versions.

Human subjects and annotators. This work does not involve human participants, crowdworkers, or human annotators. We do not collect new human-subject data, recruit participants, pay annotators, or request human consent. Therefore participant instructions, recruitment procedures, compensation details, consent procedures, and ethics-board approval are not applicable.

Potential risks. The main risk of the method is that the controller may spend additional computation and still adopt an incorrect thinking answer. Strict adoption reduces harmful seed-correct to final-wrong overrides, but it does not eliminate them, and wrong-to-wrong answer changes may still occur. The controller is also calibrated to task distributions, and performance may degrade under distribution shift or when answer extraction is unreliable. We therefore view BORA-BUDGET as a budgeted inference tool for research and low-risk deployment settings, not as a formal safety guarantee for high-stakes decisions.

Use of AI assistants. We used AI assistants to support language polishing, explanatory-text drafting, checklist preparation, code debugging, and preliminary figure drafting. All scientific claims, experimental results, analyses, code changes, and final manuscript content were reviewed and verified by the authors. AI assistants were not used to generate benchmark labels, modify evaluation data, or make autonomous research decisions.

B.8 Implementation Details

AutoThreshold pseudocode. Algorithm 1 gives the deployable binary budget allocator used in the main experiments.

Generation and stored artifacts. All experiments use frozen Qwen3 checkpoints; no model weights are fine-tuned. For each benchmark, model scale, inference mode, and seed, we store one record per example containing the problem id, full completion when available, extracted prediction, gold answer, correctness flag, token counts, and generation metadata. Selective rows are computed from these saved counterfactual outputs: a seed record is paired with the corresponding paid thinking record for the same problem id and stochastic seed. This offline pairing ensures that random, trace, learned, and budget-allocation policies are compared on exactly the same candidate generations.

Splits and stochastic seeds. The main MATH500 Qwen3-4B, Qwen3-8B, and Qwen3-14B results use three matched seeds (17, 7, 23), with leave-one-seed-out training: for each held-out seed, the trigger is trained on the other two seeds and evaluated only on the held-out seed. Transfer tasks follow the same principle when multiple seeds are available; otherwise they are reported as sanity or smoke-test rows rather than primary claims.

Prompt templates. For math benchmarks, the user prompt is the standard CoT template Please reason step by step, and put your final numeric answer within `\boxed{}`. Problem: {question}. The runner then appends a mode suffix: `/no_think` for the cheap seed condition and `/think` for paid thinking conditions. We render the resulting message with the model tokenizer’s chat template when available, using the system prompt You are a careful math reasoner.. For GPQA, we use a separate boxed-first multiple-choice prompt after auditing that the older long no-thinking prompt caused extraction errors.

Answer extraction and correctness. All main MATH500 numbers use a full-completion fixed

Algorithm 1 BORA-BUDGET-AutoThreshold

Require: Problems $\{x_i\}_{i=1}^N$, extra budget B_{extra} , seed policy π_0 , thinking policy π_T , opportunity models $P_{\text{help}}, P_{\text{harm}}$, cost model $\hat{C}$, harm penalty α , adoption gate g

Ensure: Final answers $\{\hat{y}_i\}_{i=1}^N$

- 1: **for** $i = 1, \dots, N$ **do**
- 2: Run seed policy: $(\tau_i^0, \hat{y}_i^0, c_i^0) \leftarrow \pi_0(x_i)$
- 3: Extract seed-state context: $s_i \leftarrow \phi(x_i, \tau_i^0, \hat{y}_i^0)$
- 4: Estimate marginal value: $v_i \leftarrow P_{\text{help}}(s_i) - \alpha P_{\text{harm}}(s_i)$
- 5: Estimate extra cost: $\hat{C}_i \leftarrow \hat{C}(s_i)$
- 6: **end for**
- 7: Choose token price λ_B so predicted spend is at most B_{extra}
- 8: $T \leftarrow \{i : v_i - \lambda_B \hat{C}_i > 0\}$
- 9: **for** $i = 1, \dots, N$ **do**
- 10: **if** $i \notin T$ **then**
- 11: $\hat{y}_i \leftarrow \hat{y}_i^0$
- 12: **else**
- 13: Run thinking policy: $(\tau_i^T, \tilde{y}_i^T, c_i^T) \leftarrow \pi_T(x_i)$
- 14: **if** $g(\hat{y}_i^0, \tilde{y}_i^T, \tau_i^T) = 1$ **then**
- 15: $\hat{y}_i \leftarrow \tilde{y}_i^T$
- 16: **else**
- 17: $\hat{y}_i \leftarrow \hat{y}_i^0$
- 18: **end if**
- 19: **end if**
- 20: **end for**
- 21: **return** $\{\hat{y}_i\}_{i=1}^N$

evaluator rather than the earlier pre-extracted-answer audit. The evaluator first normalizes exact text, compact text, decimal values, and top-level numeric sets. It then calls `math_verify` when available, with LaTeX and expression extraction, and falls back to SymPy simplification for simple symbolic expressions. To avoid false positives from sub-expression extraction, scalar predictions are safely rejected against structured gold answers unless the gold format is a harmless scalar wrapper such as a unit. For GPQA, we use a separate multiple-choice evaluator and reran the no-thinking condition with a boxed-first prompt after auditing that the older long prompt caused extraction errors.

Opportunity rows and labels. For each seed-thinking pair, the rollout builder records seed correctness, thinking correctness, token costs, parse success, near-cap status, and three transition la-

bels: HELPFUL is seed-wrong to thinking-correct, HARMFUL is seed-correct to thinking-wrong, and WRONG-TO-WRONG is seed-wrong to thinking-wrong after adoption. Main and strict gate pass flags are also stored in the opportunity row, so all trigger policies can be replayed with the same adoption logic.

Seed-state features. The trigger uses only information available after the cheap seed. Feature groups include answer-parse features (parse success, numeric parse success, sign, integrality, digit count, and log absolute value), trace statistics (seed tokens, prompt/completion tokens, numeric counts, and question length), problem-shape indicators (geometry, algebra, number theory, fraction/decimal, comparison, and proof-like keywords), and seed self-consistency features extracted from the seed trace. The self-consistency features count boxed/final-answer candidates, distinct candidate values, whether the stored prediction matches the first or last candidate, numeric spread among candidates, and correction or confidence markers such as “wait”, “mistake”, “check”, and “confirm”. Legacy BORA signals are retained only for historical diagnostics and are not required by the frozen main controller.

Trigger model. The main learned trigger fits two HistGradientBoostingClassifier models from scikit-learn: one for $P_{\text{help}}(s)$ and one for $P_{\text{harm}}(s)$. We use `max_iter=200`, learning rate 0.05, ℓ_2 regularization 0.01, and the held-out seed as the random state for reproducibility. If a training fold has only one class for a label, the predictor falls back to the empirical class frequency. The score used for frontier rows is $P_{\text{help}}(s) - 2P_{\text{harm}}(s)$. Trace baselines rank examples by trace length or seed tokens; heuristic baselines replay the legacy BORA trigger; random baselines average over repeated random selections.

Adoption gates. The main gate accepts a thinking candidate only if the extracted candidate is parseable and the generation is not near the token cap. The strict gate adds a sign-consistency check between seed and thinking answers when both are scalar numeric values; for structured non-scalar answers the sign check is treated as not applicable rather than as a rejection. If the gate fails, the final answer falls back to the seed, but the triggered thinking cost is still counted.

Budget allocation. For top-rate frontier rows, we rank held-out examples by the learned opportunity score and select the highest-scoring fraction. For deployable budget rows, we additionally fit a `HistGradientBoostingRegressor` with the same iteration, learning-rate, and regularization settings to predict the extra thinking cost. The allocator then uses a Lagrangian/value-per-predicted-token streaming heuristic to select examples until the requested extra-token budget is exhausted. Exact finite-batch optimization can be viewed as a 0–1 knapsack; our reported deployable budget rows use the streaming heuristic. Diagnostic rows using actual held-out costs or oracle correctness are used only to isolate cost-model error and upper-bound performance; they are not used as deployable policies.

Reproducibility. The repository contains separate scripts for rollout construction, robust math answer evaluation, trigger-frontier analysis, and fixed-evaluator budget allocation.

B.9 Additional Tables and Figures

This final block collects the supporting tables used to audit the main claims. Table 4 is a historical diagnostic rather than a central performance claim: the pooled gain on GSM8K is small, but the action-level breakdown shows why baseline preservation and adoption gates matter. Table 5 expands the MATH500 cross-model frontier with trace and random allocation baselines; it is the most direct evidence that learned triggering is far above random spending, while also showing that trace length is a strong competing difficulty proxy. Table 6 is the strictest leakage audit: problem ids are held out across folds, so the controller cannot rely on repeated benchmark items across stochastic seeds. The learned trigger remains well above random on all model scales, but its margin over trace becomes smaller, which is why the paper claims robust non-random allocation rather than universal dominance over trace heuristics. Table 7 explains the mechanism of the Qwen3-8B trigger: low-budget triggers cover most seed-wrong examples while touching relatively few seed-correct examples, and larger budgets trade more seed-correct exposure for higher recall. Finally, Table 8 reports transfer and sanity rows; these support the broader applicability of the allocation view, but their uneven behavior and nonzero harmful flips motivate the task-calibration limitation stated in the main

text.

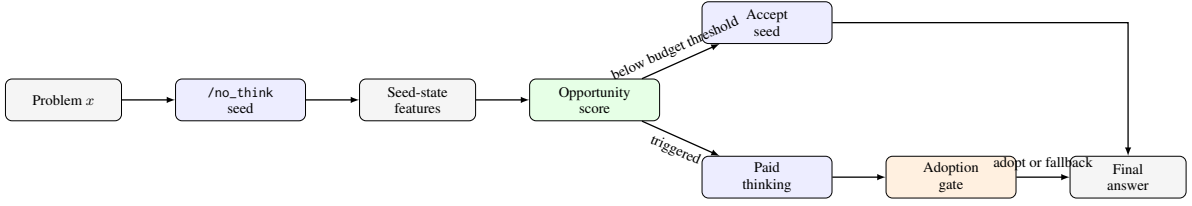


Figure 4: BORA-BUDGET treats thinking as a paid action after a cheap seed and separates triggering from adoption.

Table 4: GSM8K test results over five matched seeds. The gain is directionally positive but not statistically decisive; we use this setting mainly as a baseline-preservation and adoption-risk diagnostic.

Method	Mean accuracy	Pooled correct	Avg. tokens
Standard direct CoT	92.98	6132 / 6595	289.33
Strongseed@1024	92.93	6129 / 6595	289.72
BORA no-highrisk	93.25	6150 / 6595	293.60

Action	Helpful rescues	Harmful flips
VERIFY-ADOPT	10	1
BRANCH-2	0	3

Table 5: Additional MATH500 selective-allocation details under the fixed-completion evaluator. All three model scales use three matched seeds.

Model	Method	Acc.	Std	Avg. tokens	Helpful	Harmful
Qwen3-4B	GBDT top30 strict	90.60	0.53	2664.6	266	1
Qwen3-4B	GBDT top50 strict	91.87	0.70	3418.1	286	2
Qwen3-4B	Trace top50 strict	90.93	0.23	3494.2	273	3
Qwen3-4B	Random top50 strict	82.50	0.35	2644.4	145	1
Qwen3-8B	GBDT top30 strict	90.53	1.29	2781.0	258	1
Qwen3-8B	GBDT top50 strict	91.40	0.87	3533.5	272	2
Qwen3-8B	Trace top50 strict	90.40	1.06	3655.2	256	1
Qwen3-8B	Random top50 strict	82.71	0.59	2755.1	140	0
Qwen3-14B	GBDT top30 strict	92.73	0.46	2685.3	257	0
Qwen3-14B	GBDT top50 strict	93.13	0.61	3360.1	265	2
Qwen3-14B	Trace top50 strict	92.33	0.70	3439.0	254	3
Qwen3-14B	Random top50 strict	84.45	0.50	2557.5	134	3

Table 6: Problem-disjoint MATH500 robustness. We split by problem id into five folds, train the trigger on disjoint qids, and evaluate on held-out qids. GBDT and trace rows use thresholds calibrated on the training qids; random rows select the target fraction on the held-out fold as a negative control. Harmful is the mean count per fold.

Model	Policy	Target	Acc.	Std	Avg. tokens	Trigger	Harm.
Qwen3-4B	GBDT threshold	30%	89.47	1.07	2856.7	34.3	0.4
Qwen3-4B	Trace threshold	30%	89.13	1.12	2683.5	29.7	0.0
Qwen3-4B	Random	30%	78.70	0.63	1827.2	30.0	0.3
Qwen3-4B	GBDT threshold	50%	91.27	1.12	3691.3	58.5	0.8
Qwen3-4B	Trace threshold	50%	90.93	1.55	3501.2	50.0	0.6
Qwen3-4B	Random	50%	82.50	0.94	2641.1	50.0	0.5
Qwen3-8B	GBDT threshold	30%	89.67	0.85	3059.4	37.8	0.2
Qwen3-8B	Trace threshold	30%	89.27	0.72	2788.5	30.0	0.0
Qwen3-8B	Random	30%	79.01	0.37	1903.7	30.0	0.2
Qwen3-8B	GBDT threshold	50%	90.73	0.89	3741.9	57.1	0.2
Qwen3-8B	Trace threshold	50%	90.40	0.76	3642.8	50.1	0.2
Qwen3-8B	Random	50%	82.66	0.42	2764.3	50.0	0.4
Qwen3-14B	GBDT threshold	30%	92.20	1.30	2876.5	35.8	0.6
Qwen3-14B	Trace threshold	30%	91.47	0.73	2657.3	30.1	0.0
Qwen3-14B	Random	30%	80.92	0.40	1780.1	30.0	0.4
Qwen3-14B	GBDT threshold	50%	92.60	1.28	3456.9	53.9	1.0
Qwen3-14B	Trace threshold	50%	92.33	1.13	3442.5	50.0	0.6
Qwen3-14B	Random	50%	84.32	0.56	2552.6	50.0	0.7

Table 7: Qwen3-8B MATH500 trigger diagnostic under the fixed evaluator. The learned trigger concentrates on seed-wrong examples at low budgets.

Point	Seed wrong	Seed correct
top10	34.84	1.00
top20	68.92	2.27
top30	88.72	8.72
top50	94.49	33.88

Table 8: Additional transfer and sanity rows. MBPP uses visible tests and GPQA uses the corrected boxed-first no-thinking prompt; these rows support transfer patterns but are not the main calibrated claim.

Dataset	Model	/no_think	Uniform thinking	GBDT top30	GBDT top50	Trace top50	Random top50
ARC-Challenge	Qwen3-4B	91.78	94.22	94.67	94.44	–	–
CommonsenseQA	Qwen3-4B	79.44	82.67	82.56	83.00	–	–
ARC-Challenge	Qwen3-8B	92.67	94.67	95.33	95.22	94.67	93.78
CommonsenseQA	Qwen3-8B	80.44	85.22	84.33	84.89	83.89	82.77
MBPP-200 smoke	Qwen3-4B	75.00	81.00	85.71	87.76	–	–
MBPP-200 smoke	Qwen3-8B	75.00	84.00	82.65	84.69	–	–
MBPP-200 smoke	Qwen3-14B	82.50	86.00	90.82	92.86	93.88	90.84
GPQA Diamond	Qwen3-4B	38.89	45.96	46.47	50.51	45.95	46.22
GPQA Diamond	Qwen3-8B	32.32	47.98	38.38	42.92	43.46	42.48
GPQA Diamond	Qwen3-14B	40.91	58.59	46.97	52.01	49.01	49.04