Nice Fold OR *Hero Call*: LEARNING BUDGET-EFFICIENT THINKING UNDER POLICY-DEPENDENT SOLVABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models (LRMs) improve problem solving through extended reasoning, but often misallocate test-time compute. Existing efficiency methods reduce cost by compressing reasoning traces or conditioning budget on perceived difficulty, yet read the resulting pass rate as a difficulty score, leaving the zero-return regime unmodeled. As a result, they overspend on queries beyond the model's capability while compressing hard-but-solvable ones that need deeper reasoning. In this work, we formulate adaptive reasoning as a computational investment under uncertainty, where budget follows the expected return of reasoning rather than perceived difficulty. To instantiate this principle, we propose **B**udget-**E**fficient **T**hinking (BET), a two-stage framework that combines behavioral cold-start with GRPO under an investment-cost-aware reward. By aligning solve-or-fold decisions with rollout-derived solvability, BET learns three behaviors: (1) *short solve*, answering easy queries concisely; (2) *nice fold*, abstaining early when continued reasoning has near-zero expected return; and (3) *hero call*, preserving sufficient compute for hard-but-solvable queries. Across seven benchmarks and three base models, BET reduces reasoning tokens by **54%** while improving up to **3.2%** accuracy, and transfers zero-shot to scientific QA and logical reasoning with comparable efficiency gains. Code is available at `https://anonymous.4open.science/r/BET-CCEC/`.

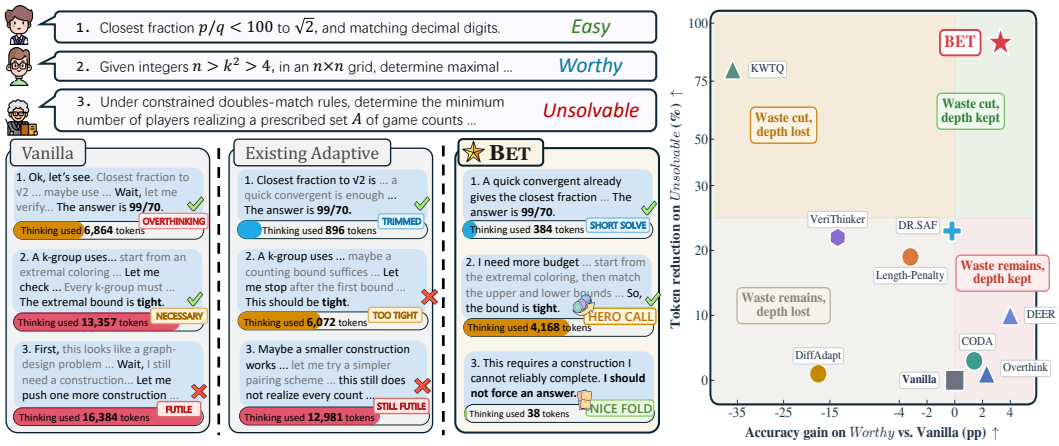
1 INTRODUCTION

LRMs such as OpenAI-o1 (El-Kishky, 2024) and DeepSeek-R1 (Guo et al., 2025) have advanced complex problem solving through extended reasoning with multi-step verification (Wei et al., 2022; Zhang et al., 2025). As these models become central to real-world agentic systems (Watanabe et al., 2025; Wang et al., 2026; Zhou et al., 2025), their inference-time cost is becoming unsustainable (Sui et al., 2025; Feng et al., 2025). Part of this cost is the genuine price of solving hard problems, and the rest stems from overthinking on easy queries and futile exploration beyond the model's competence (Chen et al., 2024; Liu et al., 2025b; Wu et al., 2025). The central challenge is to *allocate test-time compute according to its expected return without sacrificing problem-solving capability.*

Recent work addresses this burden in two main ways: *uniform reasoning compression* reduces generation cost through concise-chain distillation (Chen et al., 2024), length shaping (Shi et al., 2025; Tu et al., 2025), or target-chain supervision (Chen et al., 2025b; Arora & Zanette, 2025), while *difficulty-conditioned budget control* acts more finely by injecting early-exit signals during decoding (Yang et al., 2025) or learning query-aware depth policies (Zhu et al., 2025; Huang et al., 2025a; Chen et al., 2025a). These methods reduce token usage by compressing reasoning globally or scaling its depth to perceived difficulty, turning adaptive reasoning into a problem of how much compute to spend on each query.

In realistic workloads, however, difficulty is not solvability. Problems with similar nominal difficulty can have opposite return profiles under the current policy, with one yielding to deeper reasoning and another remaining unsolved even at full budget (Snell et al., 2024). Existing methods often treat hard queries as longer versions of easy ones, which reflects a conflation of solvability with difficulty. For a given policy, hard queries include both hard-but-solvable cases, where deeper reasoning can change the answer, and effectively unsolvable cases, where success remains negligible even with large budgets (Brown et al., 2024). As Fig. 1(a) illustrates, this missing signal leads to two errors. **(1)** Models spend near-maximal budgets on unsolvable queries with near-zero empirical return. **(2)**

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.



(a) Comparison of model reasoning behaviors on different tasks. (b) Performance of adaptive methods.

Figure 1: **Reasoning behavior and accuracy-efficiency trade-offs on Omni-Math. (a)** *Easy*, *worthy*, and *unsolvable* correspond to 16/16, 1–15/16, and 0/16 correct vanilla rollouts. Vanilla LRMs overthink on easy queries and over-allocate on unsolvable ones, while prior adaptive methods often curtail worthy reasoning prematurely. **(b)** BET lies on the Pareto frontier, preserving *worthy* reasoning while reducing waste on *unsolvable* queries.

Under global or difficulty-conditioned compression, they also shorten hard-but-solvable queries, losing correctness that sufficient reasoning would have preserved.

These observations motivate a shift from difficulty-conditioned compression to solvability-aware budget investment. We instantiate this view in **B**udget-**E**fficient **T**hinking (BET), a two-stage framework that allocates test-time compute by policy-dependent solvability. BET estimates whether further investment is likely to pay off under the current policy and learns three target behaviors: **(1)** *short solve*, answering easy queries concisely; **(2)** *nice fold*, abstaining early when continued reasoning has near-zero expected return; and **(3)** *hero call*, investing sufficient compute in hard-but-solvable queries where deeper reasoning can still change the answer.

Realizing this policy is non-trivial because solvability and efficient cost are posterior signals, while budget investment must be committed before reasoning begins. BET bridges this gap by converting current-policy rollouts into a group profile and optimizing a composite investment-cost-aware reward. The reward gates solve-or-abstain value by empirical solvability, applies query-adaptive efficiency shaping to shorten redundancy without penalizing necessary exploration, and finally calibrates pre-reasoning solvability and budget estimates to the rollout-derived profile. The resulting policy plans its token budget before reasoning, folds selectively on queries it cannot solve, and preserves depth on hard-but-solvable ones. Together, these behaviors improve the accuracy-efficiency trade-off and place BET on the Pareto frontier in Fig. 1(b). We summarize our contributions as follows:

- ❶ **Solvability-Aware Compute Allocation.** We recast adaptive reasoning as compute investment under uncertainty, giving each query a budget that follows the return it can still yield under the current policy, estimated online from that policy's own rollouts.
- ❷ **Investment-Aware Training.** We propose BET, which pairs behavioral cold-start with investment-cost-aware GRPO. An asymmetric solvability gate, efficiency shaping, and budget calibration jointly induce *short solve*, *nice fold*, and *hero call* reasoning behavior.
- ❸ **A Better Accuracy-Efficiency Frontier.** Across seven benchmarks and three base models, BET is **(I)** ***cost-effective***, reducing tokens by 54% with accuracy gains reaching 3.2%; **(II)** ***waste-reducing***, cutting unsolvable-query tokens by 92% while preserving solvable depth; and **(III)** ***generalizable***, transferring to scientific QA and logical reasoning with 56% token savings and consistent gains.

2 Related Work

Uniform reasoning compression. Overthinking in LRMs has motivated methods that reshape reasoning toward concise solutions. QFFT (Liu et al., 2025a) preserves short-chain instincts by

removing SFT questions, CoT-Valve (Ma et al., 2025) interpolates between long- and short-CoT models, and VeriThinker (Chen et al., 2025b) suppresses redundant self-reflection through auxiliary verification. Others identify overthinking transitions or optimize preferences toward concise paths (Zhao et al., 2025a; An et al., 2025), while length-based RL uniformly penalizes generation cost (Arora & Zanette, 2025; Luo et al., 2025; Aggarwal & Welleck, 2025) and inference-time variants steer activations (Huang et al., 2025b) or compress prompts (Xu et al., 2025; Han et al., 2024). These methods set the budget by a single global pressure that ignores the individual query. *Such uniform pressure shortens reasoning, but it applies the same discount to redundant verification and to exploration that would still have changed the answer.*

Difficulty-conditioned budget control. A second family of methods makes the budget query-dependent. Preflight controllers decide in advance: ThinkSwitcher (Liang et al., 2025) and DiffAdapt (Liu et al., 2025b) route queries across modes, and others preset token budgets (Zhao et al., 2025b; Wen et al., 2025). In-generation controllers act during decoding: DEER (Yang et al., 2025) exits on confidence stabilization, while KWTQ (Davidov et al., 2026) abstains below an auxiliary value threshold. Training-based controllers write difficulty into the objective: DR.SAF (Chen et al., 2025a) assigns incentives from grouped pass rates, AdaCtrl (Huang et al., 2025a) thresholds them into tagged modes, and CODA (Wu et al., 2026) gates the reward on online difficulty. Effort still scales along a single difficulty axis, where hard-but-solvable and effectively unsolvable queries share the low end. BET *instead gives the policy an explicit abstain action, gates it asymmetrically on rollout evidence, and calibrates it jointly with the token budget, so effort follows attainable return.*

3 METHOD

3.1 PROBLEM SETUP

Consider a reasoning policy π_θ that, given a query $x \in \mathcal{X}$, generates a response $y \sim \pi_\theta(\cdot \mid x)$ at a token cost $c(y) \in \mathbb{R}_+$. Queries differ in how amenable they are to the model, which we capture through the *policy-dependent solvability*:

$$s_\pi(x) \;=\; \Pr_{y\sim\pi(\cdot|x)}\big[\,\textsc{Correct}(y)\,\big], \tag{1}$$

which is the probability that the current policy produces a correct solution for x. This quantity is policy-dependent rather than query-intrinsic. It evolves during training and may shift from near zero to positive as the model improves. Accordingly, we call a query *unsolvable* when its empirical success rate under the current policy is negligible, indicating a capability mismatch rather than the absence of a valid solution. We formalize the threshold in § 3.3 and justify it in Appendix G.

Under this view, each query becomes a *compute investment decision* in which the policy balances the expected value of a correct solution against failed exploration and token cost. We formalize this trade-off as maximizing the expected net return over the query distribution $\mathcal{D}$,

$$\max_{\pi}\;\; \mathbb{E}_{x\sim\mathcal{D}}\Big[\, s_\pi(x)\cdot r^+ \;-\; \big(1-s_\pi(x)\big)\cdot \phi\big(\bar{c}_\pi(x)\big) \;-\; \alpha_{\text{cost}}\cdot \bar{c}_\pi(x)\,\Big], \tag{2}$$

where r^+ is the reward for a correct solution, $\phi(\cdot)$ is a cost-dependent penalty for failed attempts, $\bar{c}_\pi(x) = \mathbb{E}_{y\sim\pi}[c(y)]$ is the expected token cost, and $\alpha_{\text{cost}} > 0$ controls the global cost pressure.

Budget investment depends on expected reasoning return. When $s_\pi(x) \approx 0$, the expected value of solving vanishes and failed attempts dominate, so the policy should stop early. When $s_\pi(x) > 0$, additional budget is justified only while reasoning can still improve the answer. The hard case lies between these extremes, where possible gains must be weighed against token cost. Since both $s_\pi(x)$ and marginal return are unknown before generation, Eq. 2 requires the policy to estimate its own competence and budget need before reasoning begins.

3.2 WHEN TO INVEST: POLICY-DEPENDENT SOLVABILITY

Key Observation 1. Solvability is policy-dependent. The same query may be solvable under one policy yet beyond reach under another, since success depends jointly on the query and on current model capability. Thus, fixed difficulty labels cannot provide stable allocation targets.

Solvability must therefore be estimated from policy behavior. External difficulty labels, whether human ratings or competition hierarchies (Gao et al., 2024; Lightman et al., 2023; Balunović et al., 2025), do not track a model-specific success boundary (Chen et al., 2025a; Liu et al., 2025b), while recent adaptive methods read grouped rollout pass rates as query-hardness proxies (Chen et al., 2025a; Huang et al., 2025a; Wu et al., 2026). BET instead reads this statistic as current-policy solvability, measuring whether continued reasoning is likely to yield return under the policy being optimized, and recomputes it from π_θ throughout training. Given K independent rollouts $\{y_1, \ldots, y_K\}$ on query x, we derive two statistics that characterize compute-worthiness.

Group success rate. We estimate solvability with its Monte Carlo counterpart,

$$\hat{s}(x) \;=\; \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}\big[\textsc{Correct}(y_k)\big]. \tag{3}$$

Rollouts come from the current π_θ, so $\hat{s}(x)$ is non-stationary. Queries may move from $\hat{s}(x) \approx 0$ to $\hat{s}(x) > 0$ as the model improves, leaving the zero-return regime without relabeling. The label is one-sided. It requires all K rollouts to fail, an event of probability $(1-s)^K$ that decays geometrically in K. At $K{=}16$ this probability falls below 5% once $s \geq 0.17$, so a query wrongly assigned there has low true solvability and, by Eq. 2, low expected return. Empirically, only 0.43% of zero labels flip when K grows from 16 to 24 (Fig. 5(c)), as analyzed in Appendix G.

Efficient solution cost. For queries with $\hat{s}(x) > 0$, we estimate a lower-envelope cost of successful reasoning under the current policy. Let $\mathcal{C}(x) = \{y_k : \textsc{Correct}(y_k)\}$ denote correct rollouts sorted by token length. We define the *efficient solution cost* $\hat{c}^*(x)$ as the mean length of the shortest $m(x) = \max\{1, \lceil p|\mathcal{C}(x)|\rceil\}$ trajectories in $\mathcal{C}(x)$, with $p = 30\%$. This lower-envelope estimate filters long correct trajectories and marks the cost beyond which further reasoning has diminishing return.

3.3 How to Invest: An Asymmetric Compute-Aware Reward

> ***Key Observation 2.*** Within the solvable range, additional reasoning eventually becomes low-return, creating a natural budget ceiling between productive and redundant exploration.

The two observations together split the query space into a zero-return zone, where abstention dominates, and a diminishing-return zone, where reasoning is useful only up to a finite cost ceiling. An efficient policy should solve easy queries concisely (*short solve*), preserve depth for hard-but-solvable ones (*hero call*), and abstain on unsolvable ones (*nice fold*), so the reward must encode which action to take before reasoning begins as well as correctness and cost.

As shown in Fig. 2, we carry Eq. 2 into training as a per-trajectory reward $\mathcal{R}(y \mid x) = R_{\text{VAL}} + R_{\text{EFF}} + R_{\text{CAL}}$. R_{VAL} takes up its solution value and failed-attempt penalty, R_{EFF} replaces its global cost term with query-adaptive efficiency shaping, and R_{CAL} supplies the estimates it presupposes. All terms use the group profile $(\hat{s}(x), \hat{c}^*(x))$, so the reward evolves with competence rather than static annotations.

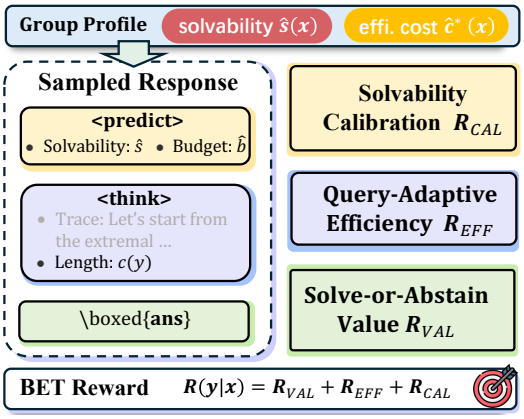


Figure 2: **Composite reward structure in BET.** A sampled response splits into `<predict>`, `<think>`, and answer components, shaped by R_{CAL}, R_{EFF}, and R_{VAL} under the group profile.

Solution value with abstention gate. We augment the standard correctness reward with an explicit *nice fold* action. A correct solution earns reward $+1$, and an incorrect attempt incurs the cost-proportional penalty $-\phi(c(y))$. Abstention is gated asymmetrically by solvability:

$$R_{\text{VAL}}(y \mid x) = \begin{cases} +1, & y \text{ is correct}, \\ -\phi(c(y)), & y \text{ is incorrect}, \\ R_{\text{ABSTAIN}}(x), & y \text{ abstains}, \end{cases} \qquad R_{\text{ABSTAIN}}(x) = \begin{cases} +\delta, & \hat{s}(x) < \epsilon_{\text{abs}}, \\ -\lambda, & \hat{s}(x) \geq \epsilon_{\text{abs}}. \end{cases} \tag{4}$$

Here $\delta > 0$ rewards identifying an unsolvable query, $\lambda \gg \delta$ penalizes premature surrender, $\phi(\cdot)$ is a monotone cost penalty on failed attempts, and $\epsilon_{\text{abs}} = 1/K$ makes abstention rewarding only when all K rollouts fail (§ 3.2). We instantiate $\phi(c) = \alpha_{\text{fail}} \cdot c/L_{\max}$, with $L_{\max}$ the maximum completion length and α_{fail} the failed-attempt cost weight.

This asymmetry keeps fold from becoming an absorbing shortcut. With all rollouts failed, abstention yields $+\delta$. Beyond leaving each rollout free to reason in depth, we keep one attempting whenever all K would abstain, so a success stays observable (Appendix E). Once any rollout succeeds, $\hat{s}(x) \geq \epsilon_{\text{abs}}$ and abstention receives $-\lambda$, while correct solutions receive $+1$. Since $\lambda \gg \delta$, a single success flips abstention from reward to penalty, preserving incentives for *hero call* as verified in § 3.5.

Query-adaptive efficiency shaping. Efficiency pressure should shorten redundancy without penalizing necessary exploration. We therefore condition length shaping on correctness and efficient solution cost, rewarding only correct solutions shorter than the current cost estimate,

$$R_{\text{EFF}}(y \mid x) = \begin{cases} \beta \cdot \max\Big(0,\ 1 - \dfrac{c(y)}{\hat{c}^*(x)}\Big) & y \text{ is correct } \wedge\ \hat{s}(x) > \tau, \\ 0 & \text{otherwise}, \end{cases} \tag{5}$$

where $\hat{c}^*(x)$ is the efficient solution cost from § 3.2, τ is a confidence threshold, and β controls bonus strength. For easy queries with small $\hat{c}^*(x)$, the bonus is positive only for short correct solutions, encouraging concision. For hard-but-solvable queries whose $\hat{c}^*(x)$ approaches the budget, the bonus vanishes and deep reasoning remains unpenalized. R_{EFF} activates only for correct solutions with $\hat{s}(x) > \tau$, since failed attempts are already cost-shaped by R_{VAL}. This avoids double penalties, while $\tau > \epsilon_{\text{abs}}$ prevents unstable low-solvability estimates from setting the efficiency target.

Solvability calibration. Eq. 2 requires allocation before true solvability and efficient cost are known, so the model must commit to a structured pre-reasoning estimate with predicted solvability $\hat{s}$ and budget $\hat{b}$ in $[0, 1]$. R_{CAL} calibrates these estimates against the group profile $(\hat{s}(x), \hat{c}^*(x))$,

$$R_{\text{CAL}}(y \mid x) = \begin{cases} -\gamma_s \cdot \big|\hat{s} - \hat{s}(x)\big| \ -\ \gamma_b \cdot \ell\big(\hat{b},\ b^*(x)\big) & \hat{s}(x) \geq \epsilon_{\text{abs}}, \\ -\gamma'_s \cdot \hat{s} \ -\ \gamma'_b \cdot \hat{b} & \hat{s}(x) < \epsilon_{\text{abs}}, \end{cases} \tag{6}$$

where $b^*(x) = \hat{c}^*(x)/L_{\max}$ is the normalized budget target, $\ell(\hat{b}, b^*)$ penalizes budget underestimation more than overestimation, and $\gamma'_s > \gamma_s$ strengthens zero-return calibration. When all rollouts fail, this term pushes predicted solvability toward zero and suppresses declared budget. R_{CAL} shapes capability representation and supplies the planning signal for compute allocation, leaving the solve-or-abstain choice to R_{VAL}. Definitions of $b^*(x)$ and $\ell(\hat{b}, b^*)$ are in Appendix A.3.

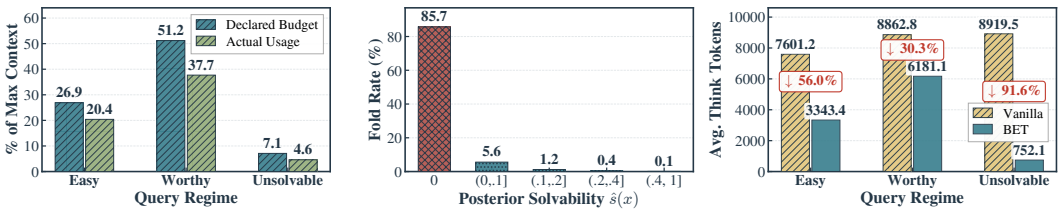
3.4 Two-Stage Training

Stage 1: Behavioral cold-start. Directly optimizing the composite reward from a vanilla model is impractical because the policy lacks the behaviors reward differentiation needs: structured solvability estimation, budget-aware generation, and rational abstention. We address this with SFT on curated demonstrations that split correctly answered queries into *short solve* and *hero call* by offline cost profiling, and add *nice fold* for the queries a single base-policy sample fails, where the model emits a designated `<Unsolvable>` token. This stage is purely behavioral. It exposes the policy to the full action space, including formatting, estimation syntax, and abstention, while the statistics that drive the reward are estimated from K rollouts online in Stage 2 (Appendix D).

Stage 2: RL with investment-cost-aware reward. We instantiate the reward of § 3.3 within GRPO (Shao et al., 2024). At each step the current policy generates $K = 16$ rollouts per query, $\hat{s}(x)$ and $\hat{c}^*(x)$ are computed on the fly, the composite reward is evaluated per trajectory, and the policy is updated with clipped surrogate optimization and group-relative advantages, so the reward tracks the evolving capability boundary. Algorithm 2 gives the procedure.

3.5 Emergence of the Three Behaviors

We verify that the three target behaviors emerge after training using a fixed pre-BET diagnostic. With the frozen base Qwen3-4B policy (*vanilla*) on Omni-Math (Gao et al., 2024), we run K=16 rollouts



(a) Budget Calibration (b) Fold Activation (c) Token Consumption by Regime

Figure 3: **Behavioral diagnostics of BET on Omni-Math. (a)** Declared and realized budget by query regime, as percentages of the maximum context length. **(b)** Fold rate versus vanilla posterior solvability $\hat{s}_0(x)$, concentrated at $\hat{s}_0(x)$=0. **(c)** Average think tokens for vanilla and BET by regime.

per query, matching the training-time profiling granularity, and partition queries into *easy* (16/16), *worthy* (1–15/16), and *unsolvable* (0/16) regimes. This split marks where reasoning is already reliable, potentially answer-changing, or futile, and tests whether BET reallocates compute accordingly.

❶ **BET plans and differentiates compute by query solvability.** The vanilla policy spends nearly the same budget on *worthy* and *unsolvable* queries, treating both as similarly demanding. In contrast, BET separates them in Fig. 3(c), eliminating 91.6% of *unsolvable* expenditure through learned abstention while retaining 70% of the *worthy* budget for productive reasoning. It also shortens *easy* query generation through *short solve*. Declared budgets track usage (Fig. 3(a)), so the model commits to a budget before it starts generating.

❷ **BET concentrates its folds at the zero-return boundary.** Folding is sharply localized where Observation 1 predicts little return. Fig. 3(b) shows 85.7% of rollouts abstaining at $\hat{s}_0(x)$=0, with the rate falling almost to zero once $\hat{s}_0(x) > 0.2$. This step pattern localizes fold at the empirical capability boundary. Fold and *hero call* also coexist without interference. Under the same *worthy* partition, BET obtains the largest net gain in correct answers among all compared methods (Fig. 4(b), § 4.5), so abstention reclaims budget from futile queries while leaving productive reasoning intact.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Training details. We use Qwen3-4B-Thinking-2507 (Qwen Team, 2025), DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI, 2025b), and DeepSeek-R1-Distill-Qwen-14B (DeepSeek-AI, 2025a). Both stages train on a stratified 20K subset of DeepMath-103K (He et al., 2025), sampled uniformly across difficulty labels and decontaminated against all evaluation benchmarks. Stage 1 uses LoRA (Hu et al., 2021) SFT to induce structured solvability estimation and the three target behaviors. Stage 2 applies GRPO (Shao et al., 2024) with TRL (von Werra et al., 2020), using K=16 rollouts, temperature 0.8, and maximum length 16,384, with β and p kept at their defaults of 0.30 and 30% throughout. Length-Penalty, CODA, and DR.SAF use the same group size and rollout protocol for fair comparison. Details are in Appendix A.

Baselines. We compare BET against eight baselines, all re-implemented with the same backbones and decoding configuration: **(i)** *preflight routing* (DiffAdapt (Liu et al., 2025b)), **(ii)** *in-generation intervention* (DEER (Yang et al., 2025), KWTQ (Davidov et al., 2026)), **(iii)** *offline adaptation* (VeriThinker (Chen et al., 2025b), Overthink (Chen et al., 2024)), and **(iv)** *online post-training* (Length-Penalty (Arora & Zanette, 2025), CODA (Wu et al., 2026), DR.SAF (Chen et al., 2025a)).

4.2 EVALUATION

Benchmarks. We evaluate on seven reasoning benchmarks in two groups. For *in-domain* mathematical reasoning, we use Omni-Math (Gao et al., 2024), MATH500 (Lightman et al., 2023), AMC-23 (math-ai, 2025), and AIME-25 (Balunović et al., 2025), spanning intermediate to olympiad-level difficulty. For *out-of-domain* transfer, we evaluate zero-shot on GPQA-Diamond (Rein et al., 2023), MUSR (Sprague et al., 2023), and LSAT-AR (Wang et al., 2021), covering scientific QA,

Table 1: **Main results on seven reasoning benchmarks** (†in-domain mathematical, ‡out-of-domain transfer). Δ on the Avg. columns is relative to the base model. Row colors: preflight routing, in-generation intervention, offline adaptation, online post-training. Qwen3-4B covers all four families; the DeepSeek backbones report the online family (all families on all backbones in Tab. 5).

Model	Accuracy (%) (↑)									#Tokens (K) (↓)								
	Math†				Avg.	General‡			Avg.	Math†				Avg.	General‡			Avg.
	Omni	MATH	AMC	AIME		GPQA	MUSR	LSAT		Omni	MATH	AMC	AIME		GPQA	MUSR	LSAT	
Qwen3-4B	54.5	88.8	88.0	63.3	73.6	50.9	64.4	71.7	62.4	8.6	6.8	12.9	13.8	10.5	6.5	6.1	7.2	6.6
+DiffAdapt	48.0	84.6	84.5	63.3	70.1	48.6	53.8	48.1	50.2	6.8	6.0	10.3	11.0	8.5	6.0	6.8	3.8	5.5
+DEER	55.3	89.4	90.5	65.3	75.1	53.2	62.0	72.1	62.4	8.1	5.2	9.7	10.3	8.3	4.8	5.4	6.2	5.4
+KWTQ	41.4	87.8	86.0	54.0	67.3	32.1	57.5	67.1	52.2	5.4	6.7	12.6	10.3	8.8	**3.8**	5.3	6.9	5.3
+VeriThinker	49.4	86.2	86.0	63.3	71.2	50.1	47.6	59.8	52.5	6.3	5.8	9.3	10.6	8.0	6.0	3.9	4.5	4.8
+Overthink	55.2	90.1	86.5	64.7	74.1	48.8	63.8	72.3	61.6	8.2	5.1	9.4	13.5	9.1	7.6	4.1	8.5	6.7
+Length-Penalty	52.9	87.4	85.5	60.0	71.4	49.5	64.2	69.2	61.0	6.6	5.4	7.2	10.9	7.5	4.7	2.5	4.5	3.9
+CODA	55.0	89.1	88.5	64.7	74.3	51.2	64.5	73.0	62.9	6.9	5.1	9.8	11.8	8.4	5.5	3.9	4.9	4.8
+DR.SAF	54.1	89.8	90.0	64.0	74.5	50.7	63.4	72.0	62.0	5.1	4.4	6.6	9.1	6.3	4.2	2.1	3.9	3.4
+BET (δ=0.10)	55.6	90.0	89.5	64.7	74.9 (+1.3)	52.4	63.9	74.3	63.5 (+1.2)	**3.8**	**3.3**	**5.0**	**6.1**	**4.6** (−57%)	**3.8**	**1.2**	**3.6**	**2.9** (−56%)
+BET (δ=0.05)	**55.9**	**90.8**	**91.0**	**66.0**	**75.9** (+2.3)	**53.9**	**64.7**	**75.7**	**64.8** (+2.4)	5.0	4.2	6.1	7.7	5.8 (−45%)	4.6	2.0	4.5	3.7 (−44%)
DeepSeek-7B	44.4	90.2	80.0	30.0	61.2	41.5	**44.4**	30.4	38.8	8.6	3.8	11.0	10.4	8.5	4.6	5.2	6.9	5.5
+Length-Penalty	42.8	89.4	83.0	33.3	62.1	38.5	39.2	32.4	36.7	6.2	2.2	7.4	8.2	6.0	4.5	4.8	5.9	5.1
+CODA	45.8	90.3	84.5	34.7	63.8	41.8	44.3	34.8	40.3	6.0	2.5	7.6	7.9	6.0	3.8	4.6	5.6	4.7
+DR.SAF	45.2	89.6	85.0	**36.0**	64.0	40.2	41.8	35.4	39.1	5.3	1.4	6.2	7.3	5.0	2.8	4.2	4.5	3.8
+BET (δ=0.10)	46.9	91.4	84.5	34.7	64.4 (+3.2)	41.0	43.6	34.7	39.8 (+1.0)	**3.7**	**1.2**	**5.6**	**5.9**	**4.1** (−52%)	**2.0**	**2.8**	**2.1**	**2.3** (−58%)
+BET (δ=0.05)	**47.1**	**92.0**	**85.5**	**36.0**	**65.2** (+4.0)	**42.0**	44.3	**35.6**	**40.6** (+1.9)	5.3	1.8	6.4	7.0	5.2 (−39%)	2.9	4.4	4.0	3.8 (−32%)
DeepSeek-14B	51.3	91.3	86.0	43.3	68.0	47.4	46.2	42.8	45.5	7.4	3.6	9.8	8.7	7.4	4.3	4.6	5.9	4.9
+Length-Penalty	49.5	90.7	88.5	42.0	67.7	45.0	42.3	47.0	44.7	5.9	2.4	6.8	7.2	5.6	3.9	3.6	4.8	4.1
+CODA	52.7	90.7	87.0	43.3	68.4	47.7	46.4	48.5	47.5	5.6	2.2	7.1	6.9	5.5	3.4	3.4	4.3	3.7
+DR.SAF	52.7	91.6	87.0	44.7	69.0	46.4	43.4	**49.3**	46.3	5.2	1.6	5.4	5.9	4.5	2.6	3.4	3.5	3.2
+BET (δ=0.10)	53.8	91.9	89.0	44.0	69.7 (+1.7)	47.4	46.5	48.9	47.6 (+2.1)	**3.9**	**1.4**	**4.9**	**5.3**	**3.9** (−48%)	**2.0**	**2.2**	**2.7**	**2.3** (−53%)
+BET (δ=0.05)	**54.7**	**92.5**	**89.5**	**46.0**	**70.7** (+2.7)	**47.9**	**47.0**	**49.3**	**48.1** (+2.6)	6.1	2.5	5.9	6.5	5.2 (−29%)	3.3	3.9	4.0	3.7 (−25%)

multi-step narrative reasoning, and logical reasoning, respectively. All results average five stochastic evaluation runs, with per-run variance in Appendix C.

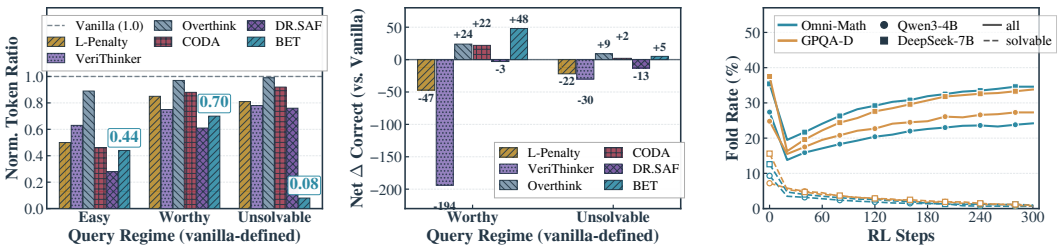
Metrics and inference. We report task accuracy (ACC) and the average number of think tokens (TOK), counted inside the `<think>...</think>` block. All methods share the same decoding configuration and a maximum completion length of 16,384. Correctness is judged only from the answer in `\boxed{}`, and an abstention receives no credit while remaining in the denominator, so every query is scored regardless of whether the method chose to answer it.

4.3 Main Results

❶ BET attains the best accuracy-efficiency frontier on every backbone. Since BET cuts budget only where further reasoning has stopped paying off, in-domain accuracy improves on all three backbones at δ=0.10 while token cost falls by roughly half. On Qwen3-4B, accuracy rises from 73.6% to 74.9% as tokens fall by 57%, and the two distilled DeepSeek models gain 3.2 and 1.7 points at 52% and 48% savings. DR.SAF is the most efficient baseline throughout, yet BET exceeds its average accuracy on every backbone while spending 14% to 28% fewer tokens.

❷ BET saves by selective compression rather than uniform shortening. Baselines that apply pressure without a solvability signal lose accuracy somewhere on the in-domain suite: DiffAdapt drops to 48.0% on Omni-Math, Length-Penalty to 60.0% on AIME-25, and KWTQ, whose halting rule fires once an auxiliary value estimate falls below a threshold, forfeits a quarter of the vanilla accuracy on Omni-Math. BET improves every in-domain benchmark on both Qwen3-4B and DeepSeek-7B at roughly half the token cost, a combination no baseline achieves, with its largest gain on the AIME-25 split of DeepSeek-7B, where vanilla is weakest (30.0% to 34.7%).

❸ BET exposes a single accuracy-efficiency knob through δ. The fold reward δ sets how readily BET abstains on low-solvability queries. Abstention earns $+\delta$ only in groups without a success, so raising δ extends folding from the least solvable queries upward. At δ=0.05, BET holds the highest in-domain average on every backbone and the best AIME-25 accuracy of 66.0%, while still removing 29% to 45% of the tokens. Raising δ to 0.10 trades about a point of accuracy for a further fifth to a quarter of its remaining tokens, and even then Qwen3-4B matches the most accurate baseline, DEER, at 74.9% against 75.1%, on 45% fewer tokens. BET therefore defines a tunable policy family, and Appendix F.2 reports the sweep over δ together with stability across β and p.



(a) Regime-wise Token Ratio (b) Net Correct Change (c) Fold-Rate Dynamics (δ=0.10)

Figure 4: **Where BET reallocates compute. (a)** Per-regime token ratio and **(b)** net correctness change on Omni-Math under the vanilla-derived partition in § 3.5. **(c)** Fold rate over RL steps on Omni-Math and GPQA-Diamond, for all queries and for those solved at least once by vanilla.

4.4 Zero-Shot Transfer to Out-of-Domain Reasoning

Learned allocation transfers without adaptation. Trained only on mathematics, BET keeps its accuracy-efficiency profile out of domain. At δ=0.05 it attains the highest general average on every backbone, and at δ=0.10 it still improves while removing 53% to 58% of the vanilla budget. The margin is widest on LSAT-AR, with DeepSeek-14B rising from 42.8% to 48.9%, while on MUSR BET stays within a point of vanilla at a fifth to half of its cost. CODA comes closest, edging past δ=0.10 on DeepSeek-7B at twice the tokens, and falls behind at δ=0.05. Signals fitted in domain fare worse: DiffAdapt and VeriThinker surrender a sixth to a fifth of the vanilla average on Qwen3-4B, and KWTQ, whose value probe is trained per backbone, retains barely a third on DeepSeek-7B. BET instead learns its criterion from the policy's own rollouts, so the criterion moves with the policy rather than with the domain it was fitted to.

4.5 Where BET Reallocates Compute

We analyze per-regime behavior on Omni-Math under the fixed vanilla-derived partition from § 3.5, comparing token allocation and correctness changes.

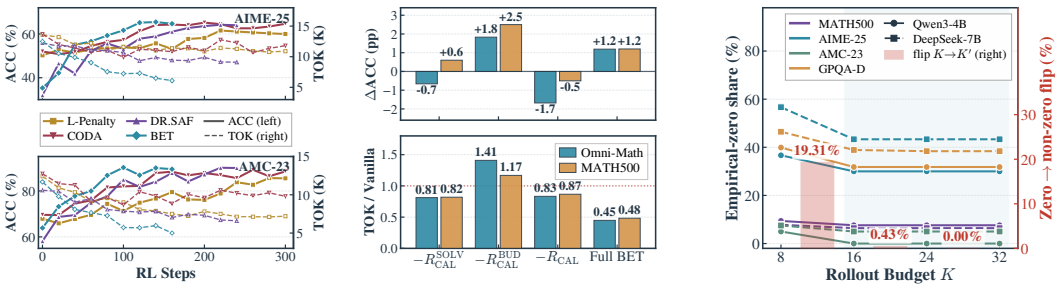
❶ BET induces the clearest solvability-aware allocation profile. Most baselines compress nearly uniformly (Fig. 4(a)): the two offline methods vary by at most 0.15 across regimes, and Length-Penalty, despite compressing easy queries to 0.50, still spends 0.81 of the vanilla budget on unsolvable ones. DR.SAF and CODA separate easy queries more clearly, at 0.28 and 0.46, yet keep 0.76 and 0.92 of it on unsolvable cases, never isolating the zero-return regime. In contrast, BET matches the target structure, compressing easy queries to 0.44, preserving 0.70 on worthy ones, and collapsing unsolvable queries to 0.08. This profile directly reflects the *short solve*, *hero call*, and *nice fold* behavior induced by the asymmetric reward.

❷ BET reduces waste and preserves solvable depth. Compression without a solvability signal takes part of its savings from reasoning that worthy queries still need, at the cost of lost answers (Fig. 4(b)). Length-Penalty and DR.SAF lose 47 and 3 worthy instances at budgets of 0.85 and 0.61, whereas BET gains 48 at 0.70. On vanilla zero queries, explicit fold modeling removes most futile exploration while recovering 5 answers, indicating selective folding rather than blanket refusal.

❸ BET converges to a selective and stable fold. Fig. 4(c) tracks the fold rate over RL steps on one in-domain and one out-of-domain benchmark, for both backbones. The cold-start policy already emits folds, but not yet tied to solvability, since Stage 1 exposes the action without optimizing its use. As Stage 2 proceeds, the overall rate levels off at a dataset-dependent level, while on queries vanilla solves at least once it falls from 9.3% to 0.7% on Omni-Math for Qwen3-4B, and from 12.6% to 1.0% on DeepSeek-7B. This separation holds on both benchmarks and backbones. Stage 2 shapes the fold decision itself, and the resulting rate follows each dataset's solvability composition.

4.6 Training Dynamics of BET

We compare BET with Length-Penalty, CODA, and DR.SAF on AIME-25 and AMC-23 throughout RL training; Fig. 5(a) reports accuracy and think-token cost per checkpoint.



(a) RL Dynamics on AIME-25 and AMC-23 (b) R_{CAL} Ablation (Qwen3-4B) (c) Finite-K Boundary Stability

Figure 5: **Training dynamics, boundary reliability, and calibration ablation. (a)** Accuracy and think tokens per checkpoint, ending at convergence. **(b)** Ablation of the two calibration branches of R_{CAL} on Qwen3-4B. **(c)** Empirical-zero share and zero-to-non-zero flip rate under nested rollout budgets on MATH500, AIME-25, AMC-23, and GPQA-Diamond with Qwen3-4B and DeepSeek-7B.

BET reaches its accuracy earlier and at a lower token cost. BET compresses only the reasoning its own rollouts show to be redundant, so accuracy and token cost improve together during training. The cold-start checkpoint starts far below the base model, at 35.3% and 63.9% against 63.3% and 88.0%, so these gains come from Stage 2. On AIME-25, BET settles at its final accuracy by step 160, while DR.SAF needs 240 steps and CODA and Length-Penalty the full 300, an ordering AMC-23 preserves. CODA eventually matches BET's 64.7% on AIME-25, but only at the end of training and at nearly twice the tokens per query. By step 160, BET has already established a stable advantage, halving its token cost from 12.5K to 6.1K, the AIME-25 cost it retains at evaluation in Table 1.

4.7 Ablation and Boundary Reliability

Solvability calibration is necessary for differentiated allocation. Fig. 5(b) ablates the two branches of R_{CAL} on Omni-Math and MATH500. Removing $R_{\text{CAL}}^{\text{SOLV}}$ weakens partitioning, leaving near-uniform compression at 0.81 and 0.82 of vanilla and Omni-Math accuracy 0.7 points below it. Removing $R_{\text{CAL}}^{\text{BUD}}$ detaches planned budget from cost, raising Omni-Math consumption to 1.41 times vanilla for 0.7 points over full BET. Removing both collapses BET into undifferentiated compression, dropping below vanilla accuracy at 0.83 of its budget. Full BET restores the best trade-off at 0.45 and 0.48 of vanilla, so the `<predict>` field becomes a planning signal only under R_{CAL}.

A rollout budget of 16 suffices to fix the boundary. Since $\hat{s}(x)$ comes from finitely many rollouts, a query enters the zero-return regime only if none of its K samples succeeds, which a small budget can produce by chance. We therefore re-estimate it on four benchmarks and two backbones under strictly nested budgets from $K{=}8$ to $K{=}32$, where each larger budget extends the same prefix and a label can only leave the zero set. The conditional rate at which a zero label acquires a success falls from 19.31% between 8 and 16 rollouts to 0.43% between 16 and 24, with no further flips observed at $K{=}32$ (Fig. 5(c)). Rollout cost grows linearly in K while the correction it buys decays by an order of magnitude, so we adopt $K{=}16$ as the operating point. K thus decides whether success evidence is observed, and the asymmetric gate of Eq. 4 decides how abstention is scored once it is.

5 Conclusion

Cost-efficient reasoning methods control compute without modeling policy-dependent solvability, so they spend budget beyond the model's capability while under-investing in hard-but-solvable queries. We propose BET, a solvability-aware framework that treats test-time compute as an investment under uncertainty. By combining behavioral cold-start with investment-cost-aware GRPO, BET aligns solve-or-fold decisions with rollout-derived solvability and induces *short solve*, *nice fold*, and *hero call* behavior. Across seven benchmarks and three base models, it cuts reasoning tokens by 54% on average while improving accuracy, and transfers zero-shot to scientific and logical reasoning.

Reproducibility Statement

The training pipeline, reward formulation, and evaluation protocol needed to reproduce our results are described in § 3.3, § 3.4, and § 4.2. The appendices give the complete hyperparameter, hardware, and serving configuration for both stages (Appendix A), the offline profiling procedure and output template behind the cold-start data (Appendix D), the full algorithm (Appendix E), every backbone and baseline behind Table 1 (Appendix B), and the per-run variance of the five evaluation runs that all reported numbers average (Appendix C).

AI Use Statement

We used generative AI tools as a writing and presentation aid, to edit and polish prose written by the authors and to prepare the scripts that render our measured results into figures and tables. The authors reviewed all AI-assisted output and take responsibility for this paper.

References

Pranjal Aggarwal and Sean Welleck. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. *ArXiv*, abs/2503.04697, 2025. URL https://api.semanticscholar.org/CorpusID:276813519.

Sohyun An, Ruochen Wang, Tianyi Zhou, and Cho-Jui Hsieh. Don't Think Longer, Think Wisely: Optimizing Thinking Dynamics for Large Reasoning Models. *ArXiv*, abs/2505.21765, 2025. URL https://api.semanticscholar.org/CorpusID:278959343.

Daman Arora and Andrea Zanette. Training Language Models to Reason Efficiently. *ArXiv*, abs/2502.04463, 2025. URL https://api.semanticscholar.org/CorpusID:276235717.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on Uncontaminated Math Competitions, February 2025. URL https://matharena.ai/.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *ArXiv*, abs/2407.21787, 2024. URL https://api.semanticscholar.org/CorpusID:271571035.

Qiguang Chen, Dengyun Peng, Jinhao Liu, Huikang Su, Jiannan Guan, Libo Qin, and Wanxiang Che. Aware First, Think Less: Dynamic Boundary Self-Awareness Drives Extreme Reasoning Efficiency in Large Language Models. *ArXiv*, abs/2508.11582, 2025a. URL https://api.semanticscholar.org/CorpusID:280671520.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT Think That Much for 2+3=? On the Overthinking of o1-Like LLMs. *ArXiv*, abs/2412.21187, 2024. URL https://api.semanticscholar.org/CorpusID:275133600.

Zigeng Chen, Xinyin Ma, Gongfan Fang, Ruonan Yu, and Xinchao Wang. VeriThinker: Learning to Verify Makes Reasoning Model Efficient. *ArXiv*, abs/2505.17941, 2025b. URL https://api.semanticscholar.org/CorpusID:278886376.

Hen Davidov, Nachshon Cohen, Oren Kalinsky, Yaron Fairstein, Guy Kushilevitz, Ram Yazdi, and Patrick Rebeschini. Knowing when to quit: A principled framework for dynamic abstention in llm reasoning. *ArXiv*, abs/2604.18419, 2026. URL https://api.semanticscholar.org/CorpusID:287634255.

DeepSeek-AI. DeepSeek-R1-Distill-Qwen-14B. https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B, 2025a. Hugging Face model card, accessed 2026-04-17.

DeepSeek-AI. DeepSeek-R1-Distill-Qwen-7B. https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B, 2025b. Hugging Face model card, accessed 2026-04-17.

Ahmed El-Kishky. OpenAI o1 System Card. *ArXiv*, abs/2412.16720, 2024. URL https://api.semanticscholar.org/CorpusID:272648256.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient Reasoning Models: A Survey. *Trans. Mach. Learn. Res.*, 2025. URL https://api.semanticscholar.org/CorpusID:277786677.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A Universal Olympiad Level Mathematic Benchmark For Large Language Models. *ArXiv*, abs/2410.07985, 2024. URL https://api.semanticscholar.org/CorpusID:273233571.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyun Zhao, Shiqing Ma, and Zhenyu Chen. Token-Budget-Aware LLM Reasoning. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://api.semanticscholar.org/CorpusID:274992044.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. DeepMath-103K: A Large-Scale, Challenging, Decontaminated, and Verifiable Mathematical Dataset for Advancing Reasoning. *ArXiv*, abs/2504.11456, 2025. URL https://api.semanticscholar.org/CorpusID:277787455.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv*, abs/2106.09685, 2021. URL https://api.semanticscholar.org/CorpusID:235458009.

Shijue Huang, Hongru Wang, Wanjun Zhong, Zhaoyu Su, Jiazhan Feng, Bowen Cao, and Yi R. Fung. AdaCtrl: Towards Adaptive and Controllable Reasoning via Difficulty-Aware Budgeting. *ArXiv*, abs/2505.18822, 2025a. URL https://api.semanticscholar.org/CorpusID:278905898.

Yao Huang, Huanran Chen, Shouwei Ruan, Yichi Zhang, Xingxing Wei, and Yinpeng Dong. Mitigating Overthinking in Large Reasoning Models via Manifold Steering. *ArXiv*, abs/2505.22411, 2025b. URL https://api.semanticscholar.org/CorpusID:278959734.

Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. ThinkSwitcher: When to Think Hard, When to Think Fast. In *Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://api.semanticscholar.org/CorpusID:278769768.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *ArXiv*, abs/2305.20050, 2023. URL https://api.semanticscholar.org/CorpusID:258987659.

Wanlong Liu, Jun Xu, Fei Yu, Yukang Lin, Ke Ji, Wenyu Chen, Yan Xu, Yasheng Wang, Lifeng Shang, and Benyou Wang. QFFT, Question-Free Fine-Tuning for Adaptive Reasoning. *ArXiv*, abs/2506.12860, 2025a. URL https://api.semanticscholar.org/CorpusID:279402374.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. DiffAdapt: Difficulty-Adaptive Reasoning for Token-Efficient LLM Inference. *ArXiv*, abs/2510.19669, 2025b. URL https://api.semanticscholar.org/CorpusID:282272126.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-Pruner: Length-Harmonizing Fine-Tuning for O1-Like Reasoning Pruning. *ArXiv*, abs/2501.12570, 2025. URL https://api.semanticscholar.org/CorpusID:275790112.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. CoT-Valve: Length-Compressible Chain-of-Thought Tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://api.semanticscholar.org/CorpusID:276317564.

math-ai. AMC-23. https://huggingface.co/datasets/math-ai/amc23, 2025. Hugging Face dataset. 40 problems from the 2023 AMC 12A/12B benchmark; accessed 2026-04-16.

Qwen Team. Qwen3-4B-Thinking-2507. https://huggingface.co/Qwen/Qwen3-4B-Thinking-2507, 2025. Hugging Face model card, accessed 2026-04-17.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *ArXiv*, abs/2311.12022, 2023. URL https://api.semanticscholar.org/CorpusID:265295009.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *ArXiv*, abs/2402.03300, 2024. URL https://api.semanticscholar.org/CorpusID:267412607.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient Reinforcement Finetuning via Adaptive Curriculum Learning. *ArXiv*, abs/2504.05520, 2025. URL https://api.semanticscholar.org/CorpusID:277628042.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *ArXiv*, abs/2408.03314, 2024. URL https://api.semanticscholar.org/CorpusID:271719990.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. MuSR: Testing the Limits of Chain-of-thought with Multistep Soft Reasoning. *ArXiv*, abs/2310.16049, 2023. URL https://api.semanticscholar.org/CorpusID:264439655.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models. *Trans. Mach. Learn. Res.*, 2025. URL https://api.semanticscholar.org/CorpusID:277150783.

Songjun Tu, Jiahao Lin, Qichao Zhang, Xiangyu Tian, Linjing Li, Xiangyuan Lan, and Dongbin Zhao. Learning When to Think: Shaping Adaptive Reasoning in R1-Style Models via Multi-Stage RL. *ArXiv*, abs/2505.10832, 2025. URL https://api.semanticscholar.org/CorpusID:278714618.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Siyuan Wang, Zhongkun Liu, Wanjun Zhong, Ming Zhou, Zhongyu Wei, Zhumin Chen, and Nan Duan. From lsat: The progress and challenges of complex reasoning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:2201–2216, 2021. URL https://api.semanticscholar.org/CorpusID:236772032.

Yin-Jie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking. *ArXiv*, abs/2603.10165, 2026. URL https://api.semanticscholar.org/CorpusID:286457252.

Miku Watanabe, Hao Li, Yutaro Kashiwa, Brittany Reid, Hajimu Iida, and Ahmed E. Hassan. On the Use of Agentic Coding: An Empirical Study of Pull Requests on GitHub. *ACM Transactions on Software Engineering and Methodology*, 2025. URL https://api.semanticscholar.org/CorpusID:281394787.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *ArXiv*, abs/2201.11903, 2022. URL https://api.semanticscholar.org/CorpusID:246411621.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. BudgetThinker: Empowering Budget-aware LLM Reasoning with Control Tokens. *ArXiv*, abs/2508.17196, 2025. URL https://api.semanticscholar.org/CorpusID:280711841.

Siye Wu, Jian Xie, Yikai Zhang, and Yang Xiao. Coda: Difficulty-aware compute allocation for adaptive reasoning. *ArXiv*, abs/2603.08659, 2026. URL https://api.semanticscholar.org/CorpusID:286377727.

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When More is Less: Understanding Chain-of-Thought Length in LLMs. *ArXiv*, abs/2502.07266, 2025. URL https://api.semanticscholar.org/CorpusID:276259519.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of Draft: Thinking Faster by Writing Less. *ArXiv*, abs/2502.18600, 2025. URL https://api.semanticscholar.org/CorpusID:276618268.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Zheng Lin, Li Cao, and Weiping Wang. Dynamic Early Exit in Reasoning Models. *ArXiv*, abs/2504.15895, 2025. URL https://api.semanticscholar.org/CorpusID:277994255.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

Haoran Zhao, Yuchen Yan, Yongliang Shen, Haolei Xu, Wenqi Zhang, Kaitao Song, Jian Shao, Weiming Lu, Jun Xiao, and Yueting Zhuang. Let lrms break free from overthinking via self-braking tuning. *Advances in Neural Information Processing Systems 38*, 2025a. URL https://api.semanticscholar.org/CorpusID:278769932.

Kai Zhao, Yanjun Zhao, Jiaming Song, Shien He, Lusheng Zhang, Qiang Zhang, and Tianjiao Li. SABER: Switchable and Balanced Training for Efficient LLM Reasoning. *ArXiv*, abs/2508.10026, 2025b. URL https://api.semanticscholar.org/CorpusID:280649722.

Zhaomeng Zhou, Lan Zhang, Junyang Wang, and Mu Yuan. IoT-Brain: Grounding LLMs for Semantic-Spatial Sensor Scheduling. *Proceedings of the 2025 ACM Workshop on Access Networks with Artificial Intelligence*, 2025. URL https://api.semanticscholar.org/CorpusID:283459628.

Yubo Zhu, Dongrui Liu, Zecheng Lin, Wei Tong, Sheng Zhong, and Jing Shao. The LLM Already Knows: Estimating LLM-Perceived Question Difficulty via Hidden Representations. In *Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://api.semanticscholar.org/CorpusID:281325426.

A Training Pipeline and Implementation Details

A.1 Overview of the Training Pipeline

BET is trained with a two-stage pipeline. Stage 1 performs supervised fine-tuning on a compact cold-start dataset constructed from offline profiling of the base model. This stage establishes the structured output format and the three target behaviors used throughout training and evaluation. Stage 2 further optimizes the policy with GRPO under the composite reward defined in § 3.3.

Both stages use the same stratified 20K subset of DeepMath-103K (He et al., 2025), sampled uniformly across difficulty labels. Stage 1 initializes the response format and behavioral repertoire, while Stage 2 learns the corresponding selection policy by recomputing the policy-dependent statistics $\hat{s}(x)$ and $\hat{c}^*(x)$ online at each optimization step.

Tables 2–4 summarize the full configuration used in our experiments. Table 2 reports the Stage 1 supervised fine-tuning hyperparameters alongside the rollout, optimization, decoding, and serving configuration for Stage 2 GRPO training. Table 3 lists the reward-related hyperparameters introduced in § 3.3. Table 4 defines the auxiliary quantities used by R_{CAL}, including the calibration targets and the asymmetric budget loss.

A.2 Training Configuration

Implementation notes. Stage 1 uses LoRA-based supervised fine-tuning. Stage 2 uses a trainer–server architecture in which policy optimization and rollout generation run asynchronously. We use an 8×A40 training node together with a single RTX PRO 6000 vLLM server for rollout generation. Unless otherwise stated, the maximum completion length is 16,384 tokens, and each prompt in Stage 2 is sampled with K=16 rollouts. BET uses the same structured output template throughout Stage 2 training and evaluation.

Table 2: Training configuration shared by all base models. Left: Stage 1 supervised fine-tuning. Right: Stage 2 GRPO training, rollout, and evaluation.

Stage 1 (SFT)	**Value**
Training data	20K DeepMath-103K
Objective	SFT on cold-start data
PEFT method	LoRA
LoRA target modules	Q, K, V, O
LoRA rank r	64
LoRA scaling α_{LoRA}	128
LoRA dropout	0.05
Optimizer	AdamW
Learning rate	2×10^{-5}
Weight decay	0.01
Scheduler	Cosine
Warmup ratio	0.03
Epochs	2
Micro-batch size	1
Gradient accumulation	16
Effective batch size	128
Max sequence length	16,384
Validation split	5%
Checkpoint selection	Lowest val. loss

Stage 2 (GRPO)	**Value**
Initialization	Stage 1 checkpoint
RL objective	GRPO (§ 3.3)
Trainer framework	TRL + vLLM server
Training GPUs	8×A40
Inference server	1×RTX PRO 6000
Rollouts per query K	16
Prompt batch size	64 queries
Trajectories per step	1,024
Rollout temperature	0.8
Rollout top-p	1.0
Max completion length	16,384
Optimizer	AdamW
Learning rate	1×10^{-6}
Weight decay	0.0
GRPO clip ϵ_{clip}	0.0625
KL coefficient	0.0
Training steps	300
Evaluation cadence	Every 20 steps
Evaluation decoding	T=0.8, top-p=1.0
Evaluation repeats	5 per query
Output template	App. D.2
Format checker	Enforced

A.3 Reward Hyperparameters and Auxiliary Definitions

Table 3: Reward-related hyperparameters used in § 3.3.

Symbol	Meaning	Value	Role in BET
δ	Reward for correct abstention	0.10	Encourages Nice Fold on effectively unsolvable queries
λ	Penalty for premature abstention	0.80	Discourages folding on queries that warrant deeper investment
β	Efficiency bonus strength	0.30	Rewards Short Solve when a concise correct path exists
α_{fail}	Failed-attempt cost weight	0.20	Scales the length-normalized penalty $\phi(c) = \alpha_{\text{fail}} c / L_{\max}$ on incorrect attempts
τ	Solvability confidence threshold	0.20	Activates R_{EFF} only when solvability is reliable
ϵ_{abs}	Abstention guard threshold	$1/K$	Makes fold rewarding only when all K rollouts fail
p	Efficient-cost percentile	30%	Defines the lower-envelope estimate of $\hat{c}^*(x)$
γ_s	Solvability calibration weight	0.10	Aligns predicted solvability with empirical group solvability
γ_b	Budget calibration weight	0.20	Aligns requested budget with efficient solution cost
γ'_s	Zero-return solvability weight	0.20	Sharpens solvability calibration in the zero-return regime
γ'_b	Unsolvable-regime budget weight	0.10	Suppresses over-budgeting when Nice Fold is preferred
μ	Underestimation penalty multiplier	2.0	Makes under-budgeting costlier than over-budgeting in $\ell(\hat{b}, b^*)$

Table 4 provides the definitions of the calibration targets and the asymmetric budget loss used in Eq. equation 6. In the solvable regime, R_{CAL} aligns the model's predicted solvability $\hat{s}$ directly with the group estimate $\hat{s}(x)$. The budget target $b^*(x)$ normalizes the efficient solution cost by the maximum completion length $L_{\max}$. The asymmetric loss $\ell(\hat{b}, b^*)$ penalizes budget underestimation more heavily than overestimation through the coefficient $\mu > 1$, reflecting the observation that under-budgeting truncates productive reasoning while over-budgeting only leaves unused headroom.

Table 4: Auxiliary symbols in the calibration reward R_{CAL} (Eq. 6).

Symbol	Definition	Role
$b^*(x)$	$\hat{c}^*(x)/L_{\max}$	Budget target expressing efficient cost as a context fraction
$\ell(\hat{b}, b^*)$	$\begin{cases} \mu\lvert\hat{b} - b^*\rvert, & \hat{b} < b^* \\ \lvert\hat{b} - b^*\rvert, & \hat{b} \geq b^* \end{cases}$	Asymmetric budget loss with heavier penalty on underestimation

A.4 Baseline Implementation

The eight baselines are grouped by the stage at which they control compute. *Preflight routing* (DiffAdapt) selects a reasoning mode before decoding begins. *In-generation intervention* (DEER, KWTQ) halts decoding on a runtime signal. DEER exits once answer confidence stabilizes, and KWTQ abstains once an auxiliary value estimate falls below its threshold, with that estimator trained separately for each backbone. *Offline adaptation* (VeriThinker, Overthink) fine-tunes the backbone on its published supervision and then decodes without further intervention. *Online post-training* (Length-Penalty, CODA, DR.SAF) starts from the same base checkpoint as BET and uses the same rollout group size K=16, decoding configuration, maximum completion length, training steps, and evaluation protocol. Method-specific coefficients follow the original papers when specified, and are otherwise chosen by a small validation sweep on accuracy and think-token count, then held fixed across all evaluation benchmarks. Every method shares the same external prompt, answer extraction, and scoring rule, and is evaluated under the same five stochastic runs.

B Complete Main Results

Table 1 in the main text reports all four method families on Qwen3-4B and the online post-training family on the two DeepSeek backbones. Table 5 below completes that comparison: every backbone carries all eight baselines across the four families, on all seven benchmarks. Numbers, formatting, and row colors follow Table 1, and every entry averages the same five stochastic evaluation runs. With all baselines included, BET at δ=0.05 still holds the best in-domain and general average on all three backbones, and among the methods that improve on the base model's accuracy, BET at δ=0.10 spends the fewest tokens on every backbone.

Table 5: **Complete results on seven reasoning benchmarks for all three backbones** (†in-domain mathematical, ‡out-of-domain transfer). This table extends Tab. 1, which reports the four method families on Qwen3-4B and the online family on the two DeepSeek backbones. Δ on the Avg. columns is relative to the base model. Row colors: preflight routing, in-generation intervention, offline adaptation, online post-training.

Model	Accuracy (%) (↑)									#Tokens (K) (↓)								
	Math†				Avg.	General‡			Avg.	Math†				Avg.	General‡			Avg.
	Omni	MATH	AMC	AIME		GPQA	MUSR	LSAT		Omni	MATH	AMC	AIME		GPQA	MUSR	LSAT	
Qwen3-4B	54.5	88.8	88.0	63.3	73.6	50.9	64.4	71.7	62.4	8.6	6.8	12.9	13.8	10.5	6.5	6.1	7.2	6.6
+DiffAdapt	48.0	84.6	84.5	63.3	70.1	48.6	53.8	48.1	50.2	6.8	6.0	10.3	11.0	8.5	6.0	6.8	3.8	5.5
+DEER	55.3	89.4	90.5	65.3	75.1	53.2	62.0	72.1	62.4	8.1	5.2	9.7	10.3	8.3	4.8	5.4	6.2	5.4
+KWTQ	41.4	87.8	86.0	54.0	67.3	32.1	57.5	67.1	52.2	5.4	6.7	12.6	10.3	8.8	**3.8**	5.3	6.9	5.3
+VeriThinker	49.4	86.2	86.0	63.3	71.2	50.1	47.6	59.8	52.5	6.3	5.8	9.3	10.6	8.0	6.0	3.9	4.5	4.8
+Overthink	55.2	90.1	86.5	64.7	74.1	48.8	63.8	72.3	61.6	8.2	5.1	9.4	13.5	9.1	7.6	4.1	8.5	6.7
+Length-Penalty	52.9	87.4	85.5	60.0	71.4	49.5	64.2	69.2	61.0	6.6	5.4	7.2	10.9	7.5	4.7	2.5	4.5	3.9
+CODA	55.0	89.1	88.5	64.7	74.3	51.2	64.5	73.0	62.9	6.9	5.1	9.8	11.8	8.4	5.5	3.9	4.9	4.8
+DR.SAF	54.1	89.8	90.0	64.0	74.5	50.7	63.4	72.0	62.0	5.1	4.4	6.6	9.1	6.3	4.2	2.1	3.9	3.4
+**BET** (δ=0.10)	55.6	90.0	89.5	64.7	74.9 (+1.3)	52.4	63.9	74.3	63.5 (+1.2)	**3.8**	**3.3**	**5.0**	**6.1**	**4.6** (−57%)	**3.8**	**1.2**	**3.6**	**2.9** (−56%)
+**BET** (δ=0.05)	**55.9**	**90.8**	**91.0**	**66.0**	**75.9** (+2.3)	**53.9**	**64.7**	**75.7**	**64.8** (+2.4)	5.0	4.2	6.1	7.7	5.8 (−45%)	4.6	2.0	4.5	3.7 (−44%)
DeepSeek-7B	44.4	90.2	80.0	30.0	61.2	41.5	**44.4**	30.4	38.8	8.6	3.8	11.0	10.4	8.5	4.6	5.2	6.9	5.5
+DiffAdapt	42.0	90.8	80.0	30.7	60.9	37.4	36.2	23.4	32.3	9.0	3.0	11.4	9.7	8.3	4.0	5.7	5.0	4.9
+DEER	43.4	89.0	74.5	33.3	60.1	41.7	34.4	28.7	34.9	7.3	2.8	8.4	8.3	6.7	3.8	4.0	5.6	4.5
+KWTQ	29.9	88.7	76.5	14.7	52.4	16.7	20.9	4.3	13.9	4.1	3.7	10.2	**3.8**	5.5	**1.7**	**2.1**	**1.7**	**1.8**
+VeriThinker	45.2	**92.2**	83.0	34.0	63.6	40.2	29.8	30.6	33.5	7.8	2.4	9.2	8.6	7.0	3.9	4.9	6.2	5.0
+Overthink	43.9	91.6	82.5	33.3	62.8	40.9	39.0	29.6	36.5	8.1	2.6	9.3	9.4	7.3	4.4	5.0	6.5	5.3
+Length-Penalty	42.8	89.4	83.0	33.3	62.1	38.5	39.2	32.4	36.7	6.2	2.2	7.4	8.2	6.0	4.5	4.8	5.9	5.1
+CODA	45.8	90.3	84.5	34.7	63.8	41.8	44.3	34.8	40.3	6.0	2.5	7.6	7.9	6.0	3.8	4.6	5.6	4.7
+DR.SAF	45.2	89.6	85.0	**36.0**	64.0	40.2	41.8	35.4	39.1	5.3	1.4	6.2	7.3	5.0	2.8	4.2	4.5	3.8
+**BET** (δ=0.10)	46.9	91.4	84.5	34.7	64.4 (+3.2)	41.0	43.6	34.7	39.8 (+1.0)	**3.7**	**1.2**	**5.6**	5.9	**4.1** (−52%)	2.0	2.8	2.1	2.3 (−58%)
+**BET** (δ=0.05)	**47.1**	92.0	**85.5**	**36.0**	**65.2** (+4.0)	**42.0**	44.3	**35.6**	**40.6** (+1.9)	5.3	1.8	6.4	7.0	5.2 (−39%)	2.9	4.4	4.0	3.8 (−32%)
DeepSeek-14B	51.3	91.3	86.0	43.3	68.0	47.4	46.2	42.8	45.5	7.4	3.6	9.8	8.7	7.4	4.3	4.6	5.9	4.9
+DiffAdapt	49.2	91.7	86.0	43.3	67.6	46.4	39.1	36.0	40.5	7.3	3.0	9.7	8.0	7.0	3.9	5.0	4.3	4.4
+DEER	50.5	90.9	83.0	43.3	67.0	47.5	37.7	41.6	42.3	6.6	2.8	7.8	7.1	6.1	3.6	3.8	5.0	4.1
+KWTQ	37.8	90.5	85.0	26.7	60.0	23.7	25.1	9.6	19.5	4.2	3.5	9.3	**4.3**	5.3	**2.0**	2.5	**2.3**	**2.3**
+VeriThinker	51.9	**92.7**	87.0	44.0	68.9	47.7	33.2	42.2	41.0	6.6	2.6	8.2	7.3	6.2	3.8	4.1	5.1	4.3
+Overthink	50.9	92.2	87.0	43.3	68.4	47.5	41.9	42.4	43.9	7.1	2.7	8.3	8.1	6.5	4.3	4.2	5.8	4.8
+Length-Penalty	49.5	90.7	88.5	42.0	67.7	45.0	42.3	47.0	44.7	5.9	2.4	6.8	7.2	5.6	3.9	3.6	4.8	4.1
+CODA	52.7	90.7	87.0	43.3	68.4	47.7	46.4	48.5	47.5	5.6	2.2	7.1	6.9	5.5	3.4	3.4	4.3	3.7
+DR.SAF	52.7	91.6	87.0	44.7	69.0	46.4	43.4	**49.3**	46.3	5.2	1.6	5.4	5.9	4.5	2.6	3.4	3.5	3.2
+**BET** (δ=0.10)	53.8	91.9	89.0	44.0	69.7 (+1.7)	47.4	46.5	48.9	47.6 (+2.1)	**3.9**	**1.4**	**4.9**	5.3	**3.9** (−48%)	**2.0**	**2.2**	2.7	**2.3** (−53%)
+**BET** (δ=0.05)	**54.7**	92.5	**89.5**	**46.0**	**70.7** (+2.7)	**47.9**	**47.0**	**49.3**	**48.1** (+2.6)	6.1	2.5	5.9	6.5	5.2 (−29%)	3.3	3.9	4.0	3.7 (−25%)

C Statistical Significance

Table 6 reports the mean and standard deviation of accuracy and think-token count over the five stochastic evaluation runs behind Table 1, on four benchmarks with the Qwen3-4B backbone. Three are in-domain and one is out-of-domain, and the set includes AMC-23 and AIME-25, the two smallest evaluation sets and those with the largest run-to-run variance. All runs use the same trained checkpoint with independent stochastic sampling (T=0.8, top-p=1.0), and the standard deviation is computed with Bessel's correction ($N-1$ denominator) and reported as 1σ. The means match the corresponding entries of Table 1.

Table 6: Mean ± standard deviation (1σ, Bessel-corrected) over the five stochastic evaluation runs behind Tab. 1, on the Qwen3-4B backbone. Accuracy in % and think tokens in K, matching Tab. 1. Row colors follow the same four method families.

Model Name	Omni-Math†		AMC-23†		AIME-25†		GPQA-Diamond‡	
	Acc. (%) ↑	#Tok. (K) ↓	Acc. (%) ↑	#Tok. (K) ↓	Acc. (%) ↑	#Tok. (K) ↓	Acc. (%) ↑	#Tok. (K) ↓
Qwen3-4B-Thinking-2507	54.5 ± 0.38	8.6 ± 0.13	88.0 ± 1.12	12.9 ± 0.69	63.3 ± 2.36	13.8 ± 0.70	50.9 ± 2.18	6.5 ± 0.29
+DiffAdapt	48.0 ± 0.47	6.8 ± 0.14	84.5 ± 2.09	10.3 ± 0.62	63.3 ± 2.36	11.0 ± 0.66	48.6 ± 2.71	6.0 ± 0.27
+DEER	55.3 ± 0.36	8.1 ± 0.11	90.5 ± 1.12	9.7 ± 0.59	65.3 ± 1.83	10.3 ± 0.63	53.2 ± 2.07	4.8 ± 0.25
+KWTQ	41.4 ± 0.50	5.4 ± 0.09	86.0 ± 1.37	12.6 ± 0.68	54.0 ± 3.65	10.3 ± 0.68	32.1 ± 2.71	3.8 ± 0.22
+VeriThinker	49.4 ± 0.43	6.3 ± 0.13	86.0 ± 1.37	9.3 ± 0.58	63.3 ± 2.36	10.6 ± 0.63	50.1 ± 2.46	6.0 ± 0.28
+Overthink	55.2 ± 0.39	8.2 ± 0.12	86.5 ± 1.37	9.4 ± 0.58	64.7 ± 1.83	13.5 ± 0.69	48.8 ± 2.25	7.6 ± 0.31
+Length-Penalty	52.9 ± 0.48	6.6 ± 0.15	85.5 ± 2.09	7.2 ± 0.50	60.0 ± 2.36	10.9 ± 0.65	49.5 ± 2.77	4.7 ± 0.24
+CODA	55.0 ± 0.41	6.9 ± 0.11	88.5 ± 1.37	9.8 ± 0.60	64.7 ± 2.98	11.8 ± 0.73	51.2 ± 2.38	5.5 ± 0.26
+DR.SAF	54.1 ± 0.41	5.1 ± 0.10	90.0 ± 1.77	6.6 ± 0.47	64.0 ± 1.49	9.1 ± 0.57	50.7 ± 2.36	4.2 ± 0.23
+**BET** (δ=0.10)	55.6 ± 0.34	3.8 ± 0.08	89.5 ± 1.12	5.0 ± 0.40	64.7 ± 1.83	6.1 ± 0.43	52.4 ± 1.97	3.8 ± 0.22
+**BET** (δ=0.05)	55.9 ± 0.36	5.0 ± 0.09	91.0 ± 1.37	6.1 ± 0.46	66.0 ± 1.49	7.7 ± 0.44	53.9 ± 2.06	4.6 ± 0.24

D Cold-Start Data Construction and Output Format

D.1 Offline Profiling and Demonstration Construction

The cold-start data in Stage 1 is constructed by offline profiling of the base policy on the same 20K stratified subset of DeepMath-103K used throughout training. Stage 1 is behavioral supervision that exposes the model to the output format and to the three target behaviors. For each query x we draw a single sample y_x from the base policy under the same maximum completion length as in the main experiments, and record whether it is correct together with its think-token cost $c(y_x)$.

The policy-dependent statistics that drive the reward are not taken from this pass. The solvability estimate $\hat{s}(x)$ and the efficient solution cost $\hat{c}^*(x)$ of § 3.2 are computed from $K{=}16$ rollouts and are recomputed online at every Stage 2 step from the evolving policy, so BET learns its decision boundary in Stage 2. Stage 1 supplies format and behavioral coverage, and Stage 2 supplies the criterion.

We partition profiled queries into three buckets and construct one demonstration for each selected query:

- ***Short Solve.*** Queries whose sampled solution is correct and inexpensive, taken from the lower-cost portion of the correctly answered set. We retain the sampled trace and assign a low predicted budget.
- ***Hero Call.*** Queries whose sampled solution is correct but expensive, taken from the higher-cost portion of that set. We retain the sampled trace and assign a higher predicted budget, exposing the model to the amount of exploration needed in the high-cost solvable regime.
- ***Nice Fold.*** Queries whose sampled solution is incorrect. For these queries, the target output contains an explicit abstention decision, together with a low predicted budget and no reasoning chain beyond the abstention.

In practice, we rank the correctly answered queries by $c(y_x)$ and take the lower and upper portions of this list to form the *short solve* and *hero call* buckets, so Stage 1 teaches distinct budget regimes. *Nice fold* examples are taken from the queries the base policy fails on this pass. The `Solvability` and `Budget` fields of a Stage 1 target carry behavioral values. They are set from a monotone mapping of the query's difficulty label and the cost of its sampled solution, and to zero for nice fold targets, which establishes the semantics of each field and the ordering across the three behaviors. In Stage 2, R_{CAL} aligns them with the online $\hat{s}(x)$ and $\hat{c}^*(x)$. Table 7 summarizes the resulting dataset.

D.2 Structured Output Template

All Stage 1 demonstrations use the same structured template shown in Fig. 6. The `Solvability` and `Budget` fields are scalar values in $[0, 1]$, representing the behavioral solvability value and the assigned budget fraction for the query. For *nice fold* targets, `<predict>` carries a zero budget, the final answer is replaced with `<Unsolvable>`, and the `<think>` block is kept empty or reduced

Table 7: Statistics of the Stage 1 cold-start dataset.

Demonstration type	Selection rule	# Examples	Avg. think tokens	Avg. total tokens
Short Solve	correct sample, low $c(y_x)$	7,641	1093	1172
Hero Call	correct sample, high $c(y_x)$	5,428	5524	5621
Nice Fold	incorrect sample	6,931	38	61
Total	–	20,000	1930	1994

to a minimal abstention statement. The same template is retained throughout Stage 2 training and evaluation for BET.

Structured Output Template

```
<predict>
Solvability: [0,1]
Budget: [0,1]
</predict>
<think>
... reasoning trace ...
</think>
\boxed{final answer}
```

Figure 6: Structured output template used in Stage 1 demonstrations and retained in Stage 2.

D.3 EXAMPLE STAGE 1 TARGETS

We illustrate how a raw query is converted into a Stage 1 target.

Raw Query

```
How many distinct arrangements of the letters in MISSISSIPPI have no
    two S's adjacent?
```

Constructed *Short Solve* Target

```
<predict>
Solvability: 0.69
Budget: 0.24
</predict>
<think>
We want arrangements of the letters in MISSISSIPPI with no two S's
    adjacent.

First count the letters other than S. Removing the four S's leaves
M, I, I, I, I, P, P. These 7 letters can be arranged in

7! / (4! 2!)

ways, since the four I's are identical and the two P's are identical.

Now place the four S's into the gaps around this arrangement.
For any fixed arrangement of the non-S letters, there are 8 available
slots where an S can go: one before the first letter, one after the
last letter, and one between each pair of consecutive letters.
```

```
To ensure that no two S's are adjacent, we may place at most one S in
each slot. So we only need to choose 4 of the 8 slots for the 4
identical S's. The number of valid choices is

C(8,4).

Therefore the total number of valid arrangements is

7! / (4! 2!) * C(8,4).

Compute each factor:
7! / (4! 2!) = 5040 / 48 = 105,
C(8,4) = 70.

Hence the total number of arrangements is

105 * 70 = 7350.
</think>
\boxed{7350}
```

Hero call targets use the same template with a larger budget value and a longer retained reasoning trace. *Nice fold* targets instead take the following form.

Constructed Nice Fold Target

```
<predict>
Solvability: 0.00
Budget: 0.00
</predict>
<think>
This query is beyond my current reliable capability.
</think>
\boxed{<Unsolvable>}
```

These examples illustrate the Stage 1 supervision targets for *short solve* and *nice fold*.

E Full Bet Algorithm

For compactness, we introduce the following notation. Given a query x, let $\textsc{Sample}(x;\pi)$ denote a single sample from policy π together with its correctness flag, used only for the Stage 1 cold start, and let $\textsc{Profile}(x;\pi,K,p)$ denote the online profiling operator of Stage 2, which samples K rollouts from π, computes the solvability estimate $\hat{s}_x$, the correct-rollout set $\mathcal{C}_x$, and the efficient solution cost $\hat{c}_x^*$ from the shortest p-fraction of $\mathcal{C}_x$. Let $\tilde{s}_x$ denote the behavioral solvability value written into a solvable Stage 1 target, obtained from a monotone mapping of the query's difficulty label and the cost of its sampled solution. Let $\textsc{Declare}(x;\pi,K)$ return the abstain-or-attempt decision of each of the K rollouts, which Bet commits in the <predict> block, with a zero budget marking abstention, before any reasoning is generated. Let $\textsc{BudgetTarget}(\hat{c}_x^*)$ denote the continuous budget target used for solvability calibration, and let $\textsc{Format}(\hat{s},b,z,a)$ denote the structured output template consisting of a predicted solvability $\hat{s}$, a predicted budget b, a reasoning trace z, and a final answer a.

Algorithm 1 gives the complete Stage 1 cold-start construction procedure. Every profiled query is assigned. Queries the single sample fails go to *nice fold*, and the remaining queries go to *short solve* or *hero call* according to the cost of that sample before optional balancing. Algorithm 2 gives the full Stage 2 GRPO training loop with dynamic solvability estimation and reward aggregation.

Guaranteed attempt within a group. The asymmetric gate of Eq. 4 takes effect only when a group contains an attempt. If all K rollouts of a query abstain, then $\hat{s}(x) = 0$ holds by construction and every rollout receives $+\delta$, so the group yields no reward contrast and the query stays in the zero set for

Algorithm 1 BET Stage 1: Cold-Start Demonstration Construction

Require: stratified pool $\mathcal{D}_{\text{pool}}$, base policy π_{θ_0}, split fraction ρ
Ensure: cold-start dataset $\mathcal{D}_{\text{cs}}$

$\mathcal{S}_{\text{solv}} \leftarrow \varnothing, \quad \mathcal{S}_{\text{unsolv}} \leftarrow \varnothing$
for all $x \in \mathcal{D}_{\text{pool}}$ **do**
  $(y_x, \text{ok}_x) \leftarrow \textsc{Sample}(x; \pi_{\theta_0})$ ▷ one sample per query
  if $\neg\, \text{ok}_x$ **then**
    $\mathcal{S}_{\text{unsolv}} \leftarrow \mathcal{S}_{\text{unsolv}} \cup \{x\}$ ▷ zero-return regime
  else
    $\mathcal{S}_{\text{solv}} \leftarrow \mathcal{S}_{\text{solv}} \cup \{(x, y_x, c(y_x))\}$
  end if
end for
$q_{\text{split}} \leftarrow \textsc{Quantile}_{\rho}\big(\{c(y_x) : (x, y_x, c(y_x)) \in \mathcal{S}_{\text{solv}}\}\big)$
$\mathcal{D}_{\text{cs}} \leftarrow \varnothing$
for all $(x, y_x, c(y_x)) \in \mathcal{S}_{\text{solv}}$ **do**
  $b_x^* \leftarrow \textsc{BudgetTarget}(c(y_x))$
  if $c(y_x) \le q_{\text{split}}$ **then**
    $t_x \leftarrow \textsc{Format}(\tilde{s}_x, b_x^*, y_x, a_x^{\text{box}})$
    $\mathcal{D}_{\text{cs}} \leftarrow \mathcal{D}_{\text{cs}} \cup \{(x, t_x, \textit{Short Solve})\}$
  else
    $t_x \leftarrow \textsc{Format}(\tilde{s}_x, b_x^*, y_x, a_x^{\text{box}})$
    $\mathcal{D}_{\text{cs}} \leftarrow \mathcal{D}_{\text{cs}} \cup \{(x, t_x, \textit{Hero Call})\}$
  end if
end for
for all $x \in \mathcal{S}_{\text{unsolv}}$ **do**
  $\hat{s}_x \leftarrow 0, \quad b_x^* \leftarrow 0$
  $t_x \leftarrow \textsc{Format}(\hat{s}_x, b_x^*, \texttt{abstain}, \texttt{Unsolvable})$
  $\mathcal{D}_{\text{cs}} \leftarrow \mathcal{D}_{\text{cs}} \cup \{(x, t_x, \textit{Nice Fold})\}$
end for
$\mathcal{D}_{\text{cs}} \leftarrow \textsc{SubsampleOrBalance}(\mathcal{D}_{\text{cs}})$ ▷ optional class balancing
return $\mathcal{D}_{\text{cs}}$

as long as the policy keeps abstaining on it. We therefore constrain each group to retain one attempt. When all K rollouts declare abstention, one of them, chosen uniformly at random, is overridden at the `<predict>` step before any reasoning is generated and completes a full reasoning trace. The group size stays at K, and the overridden decision is excluded from the policy-gradient term while its reasoning and answer are learned normally. On a query that remains beyond the policy, the forced attempt fails, $\hat{s}(x)$ stays at zero, and the abstaining rollouts keep their $+\delta$ against a penalized attempt, which sharpens the fold contrast. On a query that has become solvable, a single success lifts $\hat{s}(x)$ to $1/K$ and flips all abstentions to $-\lambda$.

Here a_x^{box} denotes the final answer in boxed form. Algorithm 1 corresponds to the cold-start construction of § 3.4 and Appendix D, and Algorithm 2 to the Stage 2 loop of §§ 3.3 and 3.4.

F Reward Ablation and Hyperparameter Sensitivity

We further analyze each reward component in BET and examine robustness to key hyperparameters. Our ablations cover four reward components: the solvability and budget calibration branches of R_{CAL}, the abstention gate of R_{VAL}, and the query-adaptive efficiency reward R_{EFF}. Within R_{VAL} we ablate the fold component, since removing correctness as well collapses RL training. All ablations are conducted on Omni-Math (Gao et al., 2024) and MATH500 (Lightman et al., 2023) using Qwen3-4B-Thinking.

F.1 Ablation of Reward Components

Table 8 reports the effect of removing each branch of R_{CAL} individually or jointly, the fold gate of R_{VAL}, and the efficiency reward R_{EFF}. Full BET yields the best Pareto position on both benchmarks, while each ablation exposes a distinct failure mode.

Algorithm 2 BET Stage 2: GRPO with Dynamic Solvability Estimation

Require: cold-start initialized policy π_{θ_1}, training subset $\mathcal{D}_{\text{sub}}$, rollout count K, percentile p, clipping coefficient ϵ_{clip}, normalization constant ϵ_{norm}, learning rate η_{lr}, reward hyperparameters $(\delta, \lambda, \beta, \tau, \gamma_s, \gamma_b, \gamma_s', \gamma_b')$, abstention threshold ϵ_{abs}
Ensure: trained BET policy π_{θ^*}

$\pi_\theta \leftarrow \pi_{\theta_1}$
while not converged **do**
  $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$ ▷ rollout policy for this update
  sample prompt batch $\mathcal{B} = \{x^{(1)}, \ldots, x^{(B)}\} \sim \mathcal{D}_{\text{sub}}$
  for all $x \in \mathcal{B}$ **do**
    $\mathcal{A}_x \leftarrow \textsc{Declare}(x; \pi_{\theta_{\text{old}}}, K)$ ▷ abstain or attempt, read off the `<predict>` block
    if every entry of $\mathcal{A}_x$ abstains **then**
      override one entry of $\mathcal{A}_x$, chosen uniformly at random, to attempt ▷ guaranteed attempt
    end if
    $(\mathcal{Y}_x, \hat{s}_x, \mathcal{C}_x, \hat{c}_x^*) \leftarrow \textsc{Profile}(x; \pi_{\theta_{\text{old}}}, K, p \mid \mathcal{A}_x)$
    for $k = 1$ to K **do**
      $y_k \leftarrow \mathcal{Y}_x[k]$
      $\textsc{Parse}(y_k) \Rightarrow (\hat{\hat{s}}_k, \hat{b}_k, z_k, a_k)$
      $R_k^{\text{VAL}} \leftarrow R_{\text{VAL}}(y_k \mid x; \hat{s}_x, \epsilon_{\text{abs}}, \delta, \lambda)$ ▷ Eq. 4
      $R_k^{\text{EFF}} \leftarrow R_{\text{EFF}}(y_k \mid x; \hat{s}_x, \hat{c}_x^*, \beta, \tau)$ ▷ Eq. 5
      $R_k^{\text{CAL}} \leftarrow R_{\text{CAL}}(y_k \mid x; \hat{s}_x, \hat{c}_x^*, \hat{\hat{s}}_k, \hat{b}_k)$ ▷ Eq. 6
      $\mathcal{R}_k(x) \leftarrow R_k^{\text{VAL}} + R_k^{\text{EFF}} + R_k^{\text{CAL}}$
    end for
    $\bar{\mathcal{R}}_x \leftarrow \frac{1}{K} \sum_{k=1}^{K} \mathcal{R}_k(x)$
    $\sigma_x \leftarrow \text{Std}(\{\mathcal{R}_k(x)\}_{k=1}^{K})$
    for $k = 1$ to K **do**
      $\hat{A}_k(x) \leftarrow \dfrac{\mathcal{R}_k(x) - \bar{\mathcal{R}}_x}{\sigma_x + \epsilon_{\text{norm}}}$ ▷ group-relative normalization
    end for
  end for
  form token-level clipped GRPO objective

$$\mathcal{J}_{\text{BET}}(\theta) \leftarrow \mathbb{E}_{x \in \mathcal{B}} \left[\frac{1}{K} \sum_{k=1}^{K} \frac{1}{T_k} \sum_{t=1}^{T_k} \min\Big(r_{k,t}(\theta) \hat{A}_k(x),\ \text{clip}(r_{k,t}(\theta), 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}) \hat{A}_k(x) \Big) \right],$$

  where T_k is the number of generated tokens in y_k and

$$r_{k,t}(\theta) = \frac{\pi_\theta(y_{k,t} \mid x, y_{k,<t})}{\pi_{\theta_{\text{old}}}(y_{k,t} \mid x, y_{k,<t})}.$$

  $\theta \leftarrow \theta + \eta_{\text{lr}} \nabla_\theta \mathcal{J}_{\text{BET}}(\theta)$
end while
return $\pi_{\theta^*} \leftarrow \pi_\theta$

Removing the solvability branch $R_{\text{CAL}}^{\text{SOLV}}$. This variant weakens BET from a capability-aware allocator into a milder cost-sensitive compressor. On Omni-Math, tokens drop from the vanilla 8.6K to 7.0K, a 19% reduction, but the model no longer learns a reliable regime partition and accuracy slips below vanilla. Full BET removes 56% of the tokens while improving accuracy. The same pattern appears on MATH500, where the model still reacts to cost but no longer aligns compression with current-policy solvability.

Removing the budget branch $R_{\text{CAL}}^{\text{BUD}}$. This produces the opposite failure mode. The model retains a solvability signal and therefore continues to pursue hard-but-promising queries, but it loses the constraint that ties declared budget to empirical competence. As a result, it over-requests compute almost everywhere. This raises raw accuracy slightly, but at a prohibitive token cost. On Omni-Math, accuracy increases to 56.3% while tokens surge to 12.1K, 41% above the vanilla budget, and on MATH500 the same variant reaches 91.3% accuracy but consumes 8.0K tokens, 17% above vanilla.

Removing R_{CAL} entirely. When both calibration branches are removed, the model loses capability-aware partitioning and budget calibration, degrading into a coarse compressor driven only by R_{VAL}

Table 8: Ablation of the reward components on Omni-Math and MATH500, averaged over the same five stochastic evaluation runs as Table 1. $R_{\text{CAL}}^{\text{SOLV}}$ and $R_{\text{CAL}}^{\text{BUD}}$ denote the solvability and budget calibration branches of R_{CAL}, $R_{\text{VAL}}^{\text{FOLD}}$ denotes the abstention gate of R_{VAL}, and R_{EFF} is the query-adaptive efficiency reward.

Variant	Omni-Math		MATH500	
	Acc. (%) ↑	#Tok. (K) ↓	Acc. (%) ↑	#Tok. (K) ↓
Vanilla	54.5	8.6	88.8	6.8
w/o $R_{\text{CAL}}^{\text{SOLV}}$	53.8	7.0	89.4	5.6
w/o $R_{\text{CAL}}^{\text{BUD}}$	56.3	12.1	91.3	8.0
w/o R_{CAL}	52.8	7.1	88.3	5.9
w/o $R_{\text{VAL}}^{\text{FOLD}}$	53.4	7.5	88.8	6.2
w/o R_{EFF}	55.8	6.2	90.2	5.3
Full BET	55.6	3.8	90.0	3.3

and R_{EFF}. On Omni-Math, accuracy falls to 52.8%, below vanilla, while tokens remain at 7.1K, only 17% below the vanilla budget. MATH500 follows the same pattern, with accuracy at 88.3%, below vanilla, with tokens at 5.9K, a 13% reduction. Unlike single-branch ablations, which retain partial regime awareness, this variant shortens reasoning without separating the regimes (Fig. 7), keeping unsolvable queries at 0.93 of the vanilla budget against 0.82 for worthy ones.

Removing the fold gate of R_{VAL}. The fold branch of R_{VAL} grants $+\delta$ for abstaining on queries with $\hat{s}(x) < \epsilon_{\text{ABS}}$ but penalizes premature surrender by $-\lambda$ with $\lambda \gg \delta$. Removing this gate (i.e., setting $R_{\text{ABSTAIN}} \equiv 0$) eliminates the positive incentive for fold. Within a few optimization steps, the policy stops emitting `<Unsolvable>` altogether and reverts to forced answering on every query, including those with $\hat{s}_0(x) = 0$. It is *less* willing to fold than w/o R_{CAL}, which still retains the $+\delta$ signal. The effect is a modest accuracy increase from sporadic lucky guesses on previously folded queries (53.4% on Omni-Math, 88.8% on MATH500), paid for by a substantial token rebound on unsolvable cases (7.5K and 6.2K tokens, with the unsolvable regime ratio in Fig. 7 climbing back to ≈ 1.0), leaving 88% and 91% of the vanilla budget in place. The fold gate is thus necessary for converting zero-return queries into near-zero token expenditure, and removing it reproduces the vanilla failure mode of treating unsolvable and worthy queries as similarly demanding. The gate alone does not produce the full profile. Fig. 7 shows that the variants which keep the gate but drop a calibration branch still spend 0.84 to 1.70 of the vanilla budget on unsolvable queries, so the cut also depends on R_{CAL} supplying a reliable solvability estimate.

Removing the efficiency reward R_{EFF}. R_{EFF} shapes correct-trajectory length toward the efficient solution cost $\hat{c}^*(x)$. Removing it ($\beta = 0$) leaves R_{VAL} and R_{CAL} intact, so fold and hero call behaviors persist. The model still abstains on zero-return queries and still calibrates declared budget against empirical cost. What disappears is the pressure that compresses overthinking on easy and worthy queries, evidenced in Fig. 7 by the easy and worthy regime ratios climbing back to near-vanilla levels while the unsolvable ratio remains close to full BET. Consequently, accuracy is preserved or marginally improved (55.8% on Omni-Math, 90.2% on MATH500), since the policy is free to spend additional reasoning on solvable queries, but token consumption climbs to 6.2K and 5.3K. Both numbers stay well below vanilla because fold still suppresses the unsolvable regime, yet remain far above full BET. At 73% and 78% of the vanilla budget against 45% and 48% for full BET, R_{EFF} is confirmed as the source of the query-adaptive compression that complements fold-driven waste reduction. Removing it leaves regime awareness intact but eliminates *short solve* on easy queries.

Putting the four components together. Full BET restores the strongest trade-off on both benchmarks, confirming that the four reward components play complementary roles. $R_{\text{VAL}}^{\text{FOLD}}$ activates abstention on zero-return queries, R_{EFF} compresses redundant reasoning on solvable ones, $R_{\text{CAL}}^{\text{SOLV}}$ supplies the solvability signal needed for regime separation, and $R_{\text{CAL}}^{\text{BUD}}$ anchors planned compute to the empirical cost profile. The two new ablations form a clean dual. Removing $R_{\text{VAL}}^{\text{FOLD}}$ collapses the unsolvable regime back toward vanilla while leaving easy and worthy regimes partially compressed, whereas removing R_{EFF} preserves the unsolvable cut but allows easy and worthy queries to drift back

Table 9: Sensitivity of BET to key hyperparameters on Omni-Math, MATH500, and AIME-25. Unless varied, all hyperparameters are fixed to the default setting $(\delta, \beta, p) = (0.10, 0.30, 30\%)$. All entries are averaged over the same five stochastic evaluation runs as Table 1.

Param.	Value	Omni-Math		MATH500		AIME-25	
		Acc. (%) ↑	#Tok. (K) ↓	Acc. (%) ↑	#Tok. (K) ↓	Acc. (%) ↑	#Tok. (K) ↓
δ	0.02	56.2	6.1	90.8	4.7	66.7	9.4
	0.05	55.9	5.0	90.8	4.2	66.0	7.7
	0.10	**55.6**	**3.8**	**90.0**	**3.3**	**64.7**	**6.1**
β	0.20	55.7	4.2	90.1	3.6	64.7	6.8
	0.30	**55.6**	**3.8**	**90.0**	**3.3**	**64.7**	**6.1**
	0.40	55.1	3.6	89.8	3.1	62.7	5.7
p	20%	53.4	3.7	87.9	3.2	63.3	6.0
	30%	**55.6**	**3.8**	**90.0**	**3.3**	**64.7**	**6.1**
	40%	55.5	4.2	89.9	3.8	62.7	7.2

toward overthinking. The three behaviors of §3.5 depend on different subsets of the reward. *Short solve* requires efficiency shaping on solvable queries, *nice fold* requires an explicit abstention value on zero-return queries, and *hero call* requires calibration signals that prevent solvable hard queries from being folded or over-compressed. Reproducing the sharp allocation profile reported in §3.5 requires all four components acting jointly.

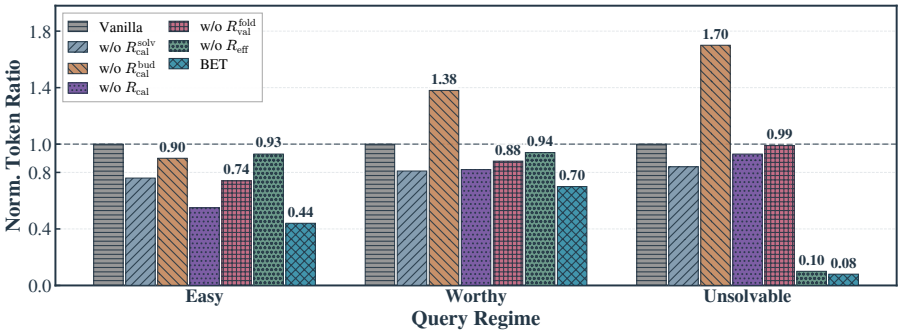


Figure 7: Regime-wise token ratio on Omni-Math under reward-component ablations. The unsolvable cut requires the fold gate and calibration together. Only the two variants that keep both, full BET and w/o R_{EFF}, collapse the unsolvable regime to 0.08 and 0.10 of the vanilla budget, whereas removing the gate leaves 0.99 and removing either calibration branch leaves 0.84 to 1.70. R_{EFF} controls compression on easy and worthy queries. Only full BET separates all three regimes.

F.2 Sensitivity to Key Hyperparameters

We further examine the sensitivity of BET to three key hyperparameters: the fold reward δ, the efficiency bonus strength β, and the efficient-cost percentile p. Table 9 reports the resulting trade-offs. Increasing δ encourages more aggressive folding, which trades a small amount of accuracy for a lower token budget. Increasing β strengthens compression on solvable queries, improving efficiency at the cost of a mild accuracy trade-off. Moving p away from its default costs in either direction. At 20% the cost estimate is too tight and accuracy falls by up to 2.2 points, while at 40% it is too permissive and tokens rise by a tenth to a fifth with no accuracy gain. Overall, δ and β trade accuracy against budget smoothly, p has a clear optimum at the default, and that setting gives the best cost-quality balance on all three benchmarks.

G FOLD DECISIONS AND PRESERVED EXPLORATION

G.1 DESIGN GOAL

BET chooses among solving a query concisely, exploring it in depth, and declining to answer, so that stopping an unproductive investment is a behavior the policy selects for itself. A fold reports low expected return under the current model, budget, and reward configuration, and a stronger model or a later checkpoint may still solve the query. A query is a solving opportunity whenever a correct path exists that the current policy could find, and it keeps that status whether the path is long or seldom sampled. Finite sampling that misses such a path produces an error of identification, and § G.4 measures how large it is. The design goal is to reduce futile attempts while preserving as much as possible of the successful reasoning the current policy can realize, including solutions that are expensive to reach or rarely found.

G.2 HOW THE REWARD PRESERVES EXPLORATION

Four properties follow from that goal. Abstention is gated asymmetrically, worth $+\delta$ in a group with no success and $-\lambda$ with $\lambda \gg \delta$. The evidence required is one rollout, since $\epsilon_{\text{abs}} = 1/K$ is the lowest threshold the estimator can express, so nothing in the design asks a query to be easy before it is worth attempting. Correct solutions receive $+1$ at any length, and R_{EFF} rewards those under $\hat{c}^*(x)$ without penalizing those above it, so depth on a hard query is not itself costly. Finally, when all K rollouts abstain one is overridden at the `<predict>` step (Appendix E), so a success stays observable.

These settings also bound how low solvability may fall before the reward stops preferring an attempt. Measured against a fixed abstention value of δ, attempting a query with true solvability s has expected value $s - (1 - s)\,\alpha_{\text{fail}}\hat{c}$ under Eq. 4, with $\hat{c} = c/L_{\max}$ the share of the completion budget consumed, so an attempt is preferred when

$$s \;>\; s^*(\hat{c}) \;=\; \frac{\delta + \alpha_{\text{fail}}\hat{c}}{1 + \alpha_{\text{fail}}\hat{c}}. \tag{7}$$

With $\delta = 0.10$ and $\alpha_{\text{fail}} = 0.20$ this reference threshold is 0.14 at a quarter of the budget and 0.25 at the full budget. The gate makes the actual comparison more favorable to attempting still, because abstention is worth $+\delta$ only when no rollout in the group succeeds. Against n attempting peers of success rate q its expected value is $(\delta + \lambda)(1 - q)^n - \lambda$, which at n=15 and q=0.2 is -0.77 against $+0.04$ for a full-budget attempt. Eq. 7 therefore bounds from above the solvability at which the reward stops preferring an attempt, and δ moves that bound, which is why it acts as a single accuracy-efficiency knob (§ 4.3, Appendix F.2).

Table 10 places the design among existing ways of controlling test-time compute. Among the baselines, only KWTQ abstains, declining below a threshold on an auxiliary value head as a decoding-time rule outside the training objective. In BET the decision is taken inside the policy and graded by the reward, with the value of abstaining set by whether success evidence appears and trained jointly with the cost estimate and the budget calibration.

G.3 EVIDENCE FROM THE EXPERIMENTS

A fold counts as incorrect throughout our evaluation and receives no accuracy credit. Under the vanilla-derived partition of § 4.5, BET spends 0.08 of the vanilla budget on queries vanilla never solves, against 0.76 to 0.99 across the baselines (Fig. 4(a)), so the savings of § 4.3 concentrate in the regime the fold action targets. The worthy regime retains 0.70 of the budget, and there BET makes the largest net correctness gain, adding 48 solved instances where Length-Penalty and DR.SAF lose 47 and 3. It also recovers 5 on queries vanilla never solved (Fig. 4(b)). Accuracy rises under the reduced budget, consistent with folds landing mostly on queries that would not have been solved.

The ablations in Appendix F show that this allocation depends on the reward as a whole: removing $R_{\text{VAL}}^{\text{FOLD}}$ returns the unsolvable regime to 0.99, removing R_{EFF} holds the cut at 0.10 but lets easy and worthy queries drift back toward overthinking, and removing either calibration branch leaves the unsolvable regime between 0.84 and 1.70. In the settings evaluated, BET reaches a more favorable accuracy-cost trade-off by cutting computation where it does not pay, while retaining effective reasoning on queries that remain solvable.

Table 10: **How compute-control mechanisms allocate and stop reasoning.** *Reasoner training*: the reasoning model's own parameters are updated. *Length-aware training*: the objective carries a generation-cost term. *Explicit adaptive control*: allocation is conditioned on the individual query. *Abstain action*: declining to answer is a possible outcome. Row colors mark the four families of Tab. 1.

Method	**Controlling signal**	Reasoner training	Length-aware training	Explicit adaptive control	Abstain action
DiffAdapt	Difficulty predicted before decoding	✗	✗	✓	✗
DEER	Decoding confidence	✗	✗	✓	✗
KWTQ	Auxiliary value head	✗	✗	✓	✓
VeriThinker	Static verification supervision	✓	✗	✗	✗
Overthink	Static short-chain supervision	✓	✓	✗	✗
Length-Penalty	Uniform length cost	✓	✓	✗	✗
CODA	Online difficulty estimate	✓	✓	✓	✗
DR.SAF	Grouped pass rate	✓	✓	✓	✗
BET *(ours)*	Group success $\hat{s}(x)$ and cost $\hat{c}^*(x)$	✓	✓	✓	✓

G.4 Decision Reliability under Finite Sampling

We first measure how stable K-rollout labels are. Under nested rollout budgets on four benchmarks (Fig. 5(c)), 19.31% of empirical-zero labels acquire a success between $K{=}8$ and 16, 0.43% between 16 and 24, and none between 24 and 32, so labels that survive a group of 16 are largely stable against further sampling. The rate follows from the estimator. An empirical zero requires all K rollouts to fail, an event of probability $(1-s)^K$, which at $K{=}16$ falls below 5% once $s \geq 0.17$ and below 0.3% once $s \geq 0.30$.

What this leaves is the low-solvability tail. When true solvability is very small an empirical zero remains a likely outcome, so a rare success can still go unobserved. Such queries also carry the lowest expected return under Eq. 2, so the residual costs little. K governs whether success evidence can be observed and the asymmetric gate governs how abstention is treated once it has been. We set $K{=}16$ at the point where additional rollouts stop repaying their cost.

H Limitations

BET has several limitations. First, training relies on K-rollout grouped sampling for online solvability estimation, and this grouped rollout requirement may become costly for larger models or longer contexts. Second, the current solvability estimate treats each query independently and does not model cross-query structure or curriculum effects during training. Third, our formulation focuses on single-query reasoning with explicit token-budget trade-offs. Extending the same investment-based principle to multi-turn agent settings, tool use, or environments with delayed external feedback remains an important direction for future work.

I Broader Impacts

This work studies how to reduce unnecessary test-time computation in large reasoning models while preserving problem-solving capability. A positive impact is that more efficient reasoning can reduce inference cost, latency, and energy consumption, making reasoning models more accessible to users and organizations with limited computational budgets. The learned *nice fold* behavior may also expose capability-aware uncertainty, which can be useful in cascaded systems where difficult queries are escalated to stronger models.

Lowering the cost of reasoning models can also make both beneficial and harmful uses easier to scale, including automated generation of misleading content, spam, or low-quality decision support. In addition, an incorrectly calibrated fold decision may withhold useful answers, while an overconfident answer after insufficient reasoning may mislead users. These risks are especially important in high-stakes domains such as medicine, law, finance, or safety-critical decision making. Deployment

in sensitive settings should include domain-specific validation, monitoring, human oversight, and appropriate escalation policies.