CURVE: Dynamic Cost-Accuracy Control for LLM Reasoning via Online Constrained Optimization

Sunggeun La [1] **Jinhyeon Kim** [2] **Sangmin Lee** [3] **Kyungwoo Song** [4]

Abstract

Recent reasoning frameworks for large language models (LLMs) improve accuracy by generating multiple candidates and applying verification, but typically rely on fixed reasoning budgets, offering limited control over the cost–accuracy trade-off. We propose Constrained Update for Risk-aware Verification Ensemble (CURVE), an online, label-free decision-making framework that enables explicit test-time control of this trade-off. CURVE formulates answer acceptance as an online constrained optimization problem, adaptively allocating verification effort by updating acceptance thresholds from proxy risk estimates and reweighting verifiers via inter-verifier consensus. Across diverse reasoning benchmarks, CURVE consistently achieves a more favorable cost–accuracy trade-off than fixed-budget baselines under comparable computational budgets. We provide an empirical analysis suggesting a regime where verification signals become weakly informative, clarifying when additional test-time verification is likely to help. [2]

1. Introduction

Prior studies have shown that large language models (LLMs) can generate reasoning sequences involving multiple intermediate steps, leading to substantial improvements in problem-solving performance across domains. (Brown et al., 2024; Snell et al., 2024; Burns et al., 2022) A line of work beginning with Chain-of-Thought prompting (Wei et al., 2022) and extending to Tree-of-Thought (Yao et al., 2023) and Iteration-of-Thought (Radha et al., 2024) has demonstrated that structured reasoning and iterative deliberation can significantly enhance accuracy. More recently, approaches such as Self-Consistency (Wang et al., 2023) and iterative refinement (Madaan et al., 2023; Gou et al., 2024) have shown that generating multiple candidate solutions and aggregating or refining them further improves reliability.

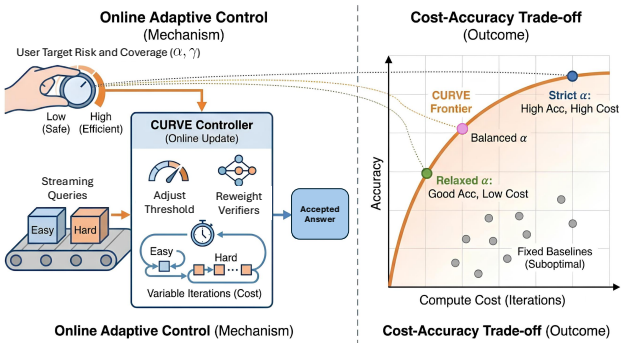

Figure 1. Conceptual Illustration of CURVE. *Left:* CURVE adaptively updates acceptance thresholds and verifier weights based on user-specified risk α and coverage γ, allocating variable computation across queries. *Right:* This yields a cost–accuracy frontier, where stricter risk targets increase accuracy at higher cost, while relaxed targets reduce computation. Fixed-budget baselines fail to follow this frontier.

A central component of these methods is *verification*: selecting, filtering, or reweighting candidate answers based on auxiliary signals. Recent systems such as Weaver(Saad-Falcon et al., 2025) and MAV(Lifshitz et al., 2025) employ ensembles of heterogeneous verifiers to assess candidate solutions from multiple perspectives, achieving notable gains in accuracy (Lifshitz et al., 2025; Dhuliawala et al., 2024). In practice, improved reasoning performance is commonly obtained by increasing test-time computational budgets. Most existing methods rely on a fixed reasoning budget, leading to wasted computation on easy problems and insufficient effort on harder ones. Recent work explores adaptive compute allocation via learned cascade policies (Zhang et al., 2024), but these approaches typically require training and do not explicitly control reliability at test time. As a result, users have little control over the accuracy–cost trade-off, despite varying reliability and resource requirements across applications. In practical settings, the desired level of reliability and computational expense varies across applica-

[1]Asiana Airlines [2]Hyundai KEFICO [3]LG CNS [4]Yonsei University. Correspondence to: Sunggeun La <sg.la@yonsei.ac.kr>, Kyungwoo Song <kyungwoo.song@yonsei.ac.kr>.

Presented at the ICML 2026 Workshop on FAGEN.,

[2]Code and data are available at https://github.com/SG-Marco/CURVE.

tions, yet current methods offer no principled mechanism to adapt reasoning effort accordingly. This observation raises a natural question: can reasoning systems adapt their computational effort online, in response to user-specified tolerance for error and resource constraints, rather than committing to a rigid budget?

Motivated by this question, we propose **CURVE**, an online decision making framework that enables dynamic, test-time control of reasoning cost and accuracy. CURVE operates in a label-free setting and allows users to specify target risk levels. Rather than fixing the number of samples or iterations, it adaptively allocates verification effort based on model confidence and instance difficulty.

In summary, our key contributions are as follows:

- **Dynamic Control Framework.** We propose CURVE, a test-time reasoning framework that dynamically controls answer acceptance during inference, allowing reasoning behavior to adapt online without a fixed inference budget.
- **Label-Free Online Risk Control.** CURVE enables online control without access to ground-truth labels, using inter-verifier agreement as a proxy signal to enforce user-specified risk constraints.
- **Empirical Performance.** Across multiple reasoning benchmarks, CURVE outperforms fixed-budget baselines, achieving a more favorable cost–accuracy trade-off through adaptive test-time reasoning.

2. Preliminaries

2.1. Background and Motivation.

Prior work on test-time reasoning and selective prediction provides the theoretical foundation for our approach. Self-consistency (Wang et al., 2023) assumes that correct answers accumulate greater probability mass than incorrect ones, making majority voting more reliable than single samples. In parallel, classical reject-option theory (Chow, 1970) establishes that error can be traded for rejection via thresholding posterior confidence. Subsequent work on selective classification (Geifman & El-Yaniv, 2017) shows that the achievable risk–coverage trade-off is governed by how well confidence scores order correct and incorrect predictions.

2.2. Online Optimization and No-Regret Updates

CURVE builds on standard results from online learning, which we briefly review for completeness. Consider an online decision process over T rounds. At each round $t \in \{1, \ldots, T\}$, an algorithm maintains a weight distribution $\pi_t = (\pi_{i,t})_{i=1}^m$ over a finite set of experts (or verifiers). Each expert i receives a bounded reward $r_{i,t} \in [0, 1]$.

Multiplicative weight updates (MWU) guarantee the following no-regret bound (Arora et al., 2012; Zinkevich, 2003; Cesa-Bianchi & Lugosi, 2006; Hazan et al., 2016):

$$\sum_{t=1}^{T} \sum_{i=1}^{m} \pi_{i,t} r_{i,t} \;\geq\; \max_{i \in \{1,\ldots,m\}} \sum_{t=1}^{T} r_{i,t} \;-\; O(\sqrt{T}). \quad (1)$$

This bound ensures that the cumulative reward of the algorithm is asymptotically no worse than that of the best fixed expert chosen in hindsight. In CURVE, this guarantee serves as a theoretical safety baseline for verifier weight updates. However, its practical usefulness depends on the informativeness of the verifier pool: when verification signals lack weak stochastic dominance, the bound remains formally valid but becomes vacuous, as matching an uninformative verifier yields no performance gain.

3. CURVE Framework

Constrained Update for Risk-aware Verification Ensemble (CURVE) is a test-time, online, label-free framework that enables explicit user control over the accuracy–cost trade-off by prioritizing risk control. Users specify a target risk level α as a hard constraint, while the coverage target γ is optimized only after the risk constraint is satisfied.

Rather than fixing a reasoning budget, CURVE adapts its acceptance behavior online to ensure reliability first and then improve efficiency. Operating without access to ground-truth labels, CURVE leverages inter-verifier consensus signals and online constrained optimization (Yu et al., 2017) to (i) exploit observations from past queries in a sequential setting, (ii) adapt decision policies under explicit risk constraints, and (iii) improve accuracy without monotonically increasing computational cost.

Framework Overview. Figure 2 provides a schematic of the CURVE framework. CURVE is initialized with a target risk α and a baseline coverage γ, which determine the conservativeness of the acceptance policy.

At each iteration, the generator proposes a candidate answer that is evaluated by multiple verifiers. Their outputs are aggregated into an ensemble score and compared against an adaptive threshold τ_t to decide acceptance. In the absence of ground-truth labels, CURVE estimates risk and coverage from inter-verifier agreement and updates τ_t online.

Verifier weights $w_{i,t}$ are updated only on accepted rounds to prevent reinforcing low-confidence noise. To improve robustness across iterations, CURVE additionally exploits historical signals by re-scoring previously proposed answers and selecting the strongest candidate observed. If no answer

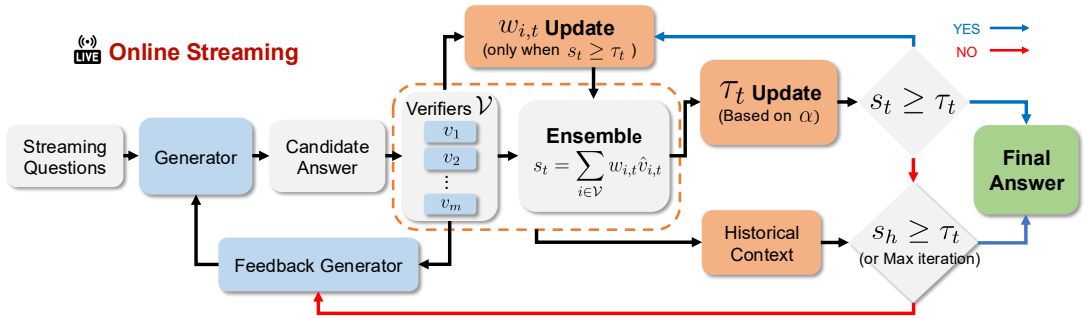

Figure 2. Overview of the CURVE framework. CURVE operates in an online streaming setting. For each incoming query, the generator proposes a candidate answer, which is evaluated by a set of heterogeneous verifiers. Their outputs are aggregated into an ensemble score, which is compared against an adaptive acceptance threshold. The threshold is updated online based on user-specified risk constraints, while verifier weights are updated only on accepted rounds. If the candidate is rejected, verifier feedback is used to generate guidance for the next iteration. A historical context module aggregates past signals to refine final answer selection. Blue blocks denote calls to the underlying LLM, while orange blocks represent CURVE-specific control and update mechanisms.

is accepted, verifier feedback guides further refinement until acceptance or a maximum iteration is reached.

3.1. Problem Setting

We consider a test-time reasoning setting in which an LLM is queried with a sequence of problems $\{x_t\}_{t=1}^{T}$. For each problem x_t, the model generates one candidate answer, which is evaluated by a set of verifiers $\mathcal{V} = \{v_1, \ldots, v_m\}$. Each verifier produces a correctness assessment denoted as $\hat{v}_{i,t} \in [0, 1]$, which represents either a binary judgment or a continuous confidence score. The true label $y_t \in \{0, 1\}$ is not observed during inference.

The system must decide whether to *accept* or *reject* a candidate answer based on the aggregated verifier scores. Accepting a candidate introduces risk, while rejection incurs computational or coverage costs. The objective is to dynamically optimize the trade-off between accuracy and these associated costs.

3.2. Risk and Coverage

We measure performance through two quantities. *Coverage* reflects the rate at which the system decides to accept and output a solution. *Risk* is defined as the empirical error rate among accepted answers. At test time, risk is estimated through proxy signals derived from verifier scores. CURVE aims to maintain risk below a user-specified target while maximizing coverage and minimizing computational effort.

3.3. Ensemble Scoring

At each round t, the generator proposes a candidate answer. Each verifier $i \in \mathcal{V}$ produces a correctness assessment denoted as $\hat{v}_{i,t} \in [0, 1]$, representing either a binary judgment or a continuous confidence score (Freund & Schapire, 1997; Kunapuli, 2023; Bartlett et al., 1998; Dietterich, 2000). The ensemble score is defined as a weighted average:

$$s_t = \sum_{i \in \mathcal{V}} w_{i,t} \hat{v}_{i,t}, \tag{2}$$

where $w_{i,t}$ denotes the weight of the verifier i at round t, satisfying $w_{i,t} \geq 0$ and $\sum_{i \in \mathcal{V}} w_{i,t} = 1$. The system accepts the candidate answer according to

$$A_t = \mathbb{I}(s_t \geq \tau_t), \tag{3}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function and τ_t is an adaptive acceptance threshold. Accepted rounds act as a selector for verifier weight updates, while threshold updates are performed at every round.

3.4. Risk and Coverage Estimation via EWMA

To update the acceptance threshold τ_t in a principled and label-free manner, CURVE maintains online estimates of coverage and risk.

A key component of CURVE is the adaptive update of the acceptance threshold that governs candidate answer selection. This update relies on estimates of risk and coverage, which we describe first. To track system behavior under non-stationary conditions, we estimate coverage and risk using exponentially weighted moving averages (EWMA)(Robbins & Monro, 1951; Zinkevich, 2003).

Here, $\hat{C}_t$ and $\hat{R}_t$ represent the system's running estimates of *coverage* and *risk* at step t, respectively. Coverage $\hat{C}_t$ measures how often the system chooses to accept a candidate answer, while risk $\hat{R}_t$ estimates how likely an accepted answer is to be incorrect. Since ground-truth labels are unavailable at test time, both quantities are tracked using proxy signals derived from inter-verifier agreement.

We index the process by discrete steps $t = 1, 2, \ldots$. The estimates are initialized as $\hat{R}_1 = \alpha$ and $\hat{C}_1 = \gamma$, corre-

sponding to the user-specified target risk and coverage. This initialization anchors the early dynamics of threshold updates to the desired operating regime and avoids unstable behavior in the initial steps.

The updates are given by:

$$\hat{C}_{t+1} = (1-\beta)\hat{C}_t + \beta A_t, \tag{4}$$

$$\hat{R}_{t+1} = \begin{cases} (1-\beta)\hat{R}_t + \beta(1-\hat{y}_t), & \text{if } A_t = 1, \\ \hat{R}_t, & \text{if } A_t = 0, \end{cases} \tag{5}$$

where $\beta \in (0,1)$ is a smoothing factor.

Here, $\hat{y}_t \in [0,1]$ denotes a *proxy correctness signal* rather than a ground-truth label. It is computed from inter-verifier agreement as follows

$$\hat{y}_t = \frac{1}{m}\sum_{i=1}^{m} \hat{v}_{i,t}. \tag{6}$$

where m is the number of verifiers. Note that for confidence-score verifier, the continuous confidence output is binarized via a predefined cutoff value, and the resulting binary decision is used as $\hat{v}_{i,t}$ in the above aggregation. $\hat{y}_t$ summarizes how strongly the verifiers agree on the candidate answer at step t. Rather than indicating whether the answer is truly correct, $\hat{y}_t$ measures the *degree of consensus* among the verifiers: values close to 1 indicate broad agreement, while values near 0 indicate substantial disagreement.

Accordingly, we define the *instantaneous proxy risk* as $1-\hat{y}_t$, which serves as a soft estimate of how unreliable the accepted answer may be. Intuitively, when most verifiers concur, the answer is likely to be correct, whereas widespread disagreement signals a higher chance of error.

3.5. Constraint-Aware Threshold Update via Hinge-Based Subgradient

We now turn to the threshold update rule, which uses the estimated risk $\hat{R}_t$ and coverage $\hat{C}_t$ to adapt the acceptance threshold. Our objective is to maximize coverage while maintaining empirical risk below a target level α. We view this setting through the lens of online constrained decision making, using a Lagrangian relaxation (Yu et al., 2017).

The instantaneous update signal is defined as:

$$\Delta_t = (\hat{R}_t - \alpha)_+ - \lambda_{\text{rc}}(\gamma - \hat{C}_t)_+, \tag{7}$$

where $(\cdot)_+ = \max(0,\cdot)$, γ is a baseline coverage target, and $\lambda_{rc} \geq 0$ controls the risk–coverage trade-off. The update signal Δ_t combines risk and coverage constraint violations into a single scalar. The hinge form updates the threshold only when a constraint is violated: risk excess increases τ_t, while coverage shortfall relaxes it. This yields a stable, label-free mechanism for controlling the risk–coverage trade-off.

The threshold is updated as:

$$\tau_{t+1} = \text{clip}\big(\tau_t + \eta_{\tau,t}\,\Delta_t,\ 0,\ 1\big), \tag{8}$$

where $\eta_{\tau,t} > 0$ is a learning rate and $\text{clip}(\cdot,0,1)$ keeps the threshold within [0,1]. We use an inverse-time decaying learning rate. ($\eta_{\tau,t} = \eta_{\tau,1}/(1+\lambda_{\text{decay}}\,t)$, λ_{decay} is decay rate) This update rule prioritizes correcting risk violations by increasing τ_t, while relaxing the threshold to improve coverage when the system is safely below the risk constraint.

Theorem 3.1 (Convergence of Time-Averaged Constraint Violation)**.** *Consider the update rule in Eq. 7 with $\lambda_{rc} = 0$, a constant step size $\eta > 0$ and an initial threshold $\tau_1 \in [0,1]$. Under the feasibility assumption that $(\hat{R}_t - \alpha)_+ = 0$ whenever $\tau_t = 1$, and the boundedness assumption that there exists a constant $G \in (0,\infty)$ such that $0 \leq (\hat{R}_t - \alpha)_+ \leq G$ for all t, the time-averaged constraint violation satisfies, for any $T \geq 1$,*

$$\frac{1}{T}\sum_{t=1}^{T}(\hat{R}_t - \alpha)_+ \leq \frac{1-\tau_1+\eta G}{\eta T}. \tag{9}$$

Remark. Theorem 3.1 does not guarantee that the proxy risk $\hat{R}_t$ stays below α at every round. Rather, it ensures that violations of the target level do not persist or accumulate over time. Such average constraint satisfaction is standard in online optimization with noisy, label-free feedback, and aligns well with CURVE's adaptive test-time setting.

3.6. Selective Weight Update

After updating the acceptance threshold, we next turn to the verifier weight updates. Verifier weights $w_{i,t}$ are updated only on accepted rounds ($A_t = 1$), which stabilizes learning by filtering out low-confidence noise. Each verifier receives a bounded agreement signal $r_{i,t} \in [0,1]$ indicating its consistency with the accepted answer.

$$r_{i,t} := \hat{y}_t\,\mathbb{I}\big(\hat{v}_{i,t} \geq \tfrac{1}{2}\big) \ + \ (1-\hat{y}_t)\,\mathbb{I}\big(\hat{v}_{i,t} < \tfrac{1}{2}\big), \tag{10}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. This reward assigns higher weight to verifiers whose judgments are consistent with the inter-verifier consensus. When verifier i supports the accepted answer (i.e., $\hat{v}_{i,t} \geq 1/2$), it receives a reward proportional to the overall agreement level $\hat{y}_t$. Conversely, when it disagrees, it is assigned a complementary reward $1-\hat{y}_t$. This design amplifies reliable verifiers under strong consensus while attenuating noisy signals when agreement is weak. so that $r_{i,t} \in [0,1]$ assigns higher reward to verifiers that agree with the consensus when agreement is strong, and lower reward otherwise. Note that acceptance is determined by the weighted ensemble score,

Algorithm 1 CURVE algorithm[3]

Input: Verifiers $\mathcal{V}$, risk target α, coverage baseline γ, initial threshold τ_1, trade-off λ_{rc}, learning rates $\eta_{\tau,1}$, $\eta_{w,1}$, decay rateλ_{decay}, smoothing factor β, maximum iterations T_{max}
Initialize: $w_{i,1} = 1/|\mathcal{V}|$ for all $i \in \mathcal{V}$, EWMA estimates $\hat{R}_1 = \alpha$, $\hat{C}_1 = \gamma$
for $t = 1$ to T_{max} **do**
  Generator proposes candidate answer
  Evaluate verifier outputs score $\hat{v}_{i,t} \in [0, 1]$
  $s_t = \sum_{i \in \mathcal{V}} w_{i,t} \hat{v}_{i,t}$
  $A_t = \mathbb{I}(s_t \geq \tau_t)$
  $\hat{C}_{t+1} = (1 - \beta)\hat{C}_t + \beta A_t$
  if $A_t = 1$ **then**
    Compute inter-verifier consensus signal $\hat{y}_t$
    $\hat{R}_{t+1} = (1 - \beta)\hat{R}_t + \beta(1 - \hat{y}_t)$
    for $i \in \mathcal{V}$ **do**
      $\eta_{w,t} = \eta_{w,1} / (1 + \lambda_{\text{decay}}\, t)$
      $w_{i,t+1} = w_{i,t} \exp(\eta_{w,t} r_{i,t})$
    end for
    Normalize w_{t+1}
  else
    $\hat{R}_{t+1} = \hat{R}_t$, $w_{t+1} = w_t$
  end if
  $\Delta_t = \max(0, \hat{R}_t - \alpha) - \lambda_{\text{rc}} \max(0, \gamma - \hat{C}_t)$
  $\eta_{\tau,t} = \eta_{\tau,1} / (1 + \lambda_{\text{decay}}\, t)$
  $\tau_{t+1} = \min(1, \max(0, \tau_t + \eta_{\tau,t} \Delta_t))$
end for

whereas weight updates are driven by inter-verifier agreement. Thus, a verifier that dominates acceptance due to a large weight may still be down-weighted if it disagrees with the majority, which prevents persistent overconfidence and stabilizes learning under noisy, label-free feedback.

Weights are updated multiplicatively, following the MWU algorithm (Arora et al., 2012):

$$w_{i,t+1} \propto w_{i,t} \exp(\eta_{w,t} r_{i,t}), \quad \sum_{i \in \mathcal{V}} w_{i,t+1} = 1, \tag{11}$$

where $\eta_{w,t} > 0$ is the weight learning rate with an inverse-time decaying. ($\eta_{w,t} = \eta_{w,1} / (1 + \lambda_{\text{decay}}\, t)$, λ_{decay} is decay rate)

3.7. Historical Context as an Inductive Bias

When the instantaneous ensemble score s_t fails to meet the acceptance threshold τ_t, relying solely on the current step's signal is insufficient for confident selection. However, treating each iteration independently disregards the valuable trajectory of candidate answers and verification outcomes accumulated over time. To address this ambiguity without discarding prior computational effort, CURVE leverages historical context as an inductive bias to stabilize decision-making.

Classical analyses often rely on independence assumptions, whereas practical reasoning tasks exhibit systematic and recurring failure modes, such as adversarial distractors that consistently mislead both generators and verifiers (Jia & Liang, 2017; Zellers et al., 2019). To mitigate such effects, we incorporate historical context not as a formal prior, but as a lightweight heuristic inductive bias applied only at the selection stage.

Let $\mathcal{A}$ denote the set of unique candidate answers accumulated for a given problem across iterations, and let $a \in \mathcal{A}$ be one such candidate. We define a history-based adjustment score for each candidate a as:

$$\begin{aligned} s_h(a) = \text{MeanScore}(a) + w_{\text{freq}} \log N_a \\ - w_{\text{pen}} \frac{V_a}{N_a} + w_{\text{rescue}} \frac{F_a}{K}, \end{aligned} \tag{12}$$

where $\text{MeanScore}(a)$ denotes the average ensemble score assigned to candidate a across all iterations in which it appears, reflecting its overall verification confidence. Here, N_a denotes the number of iterations in which candidate a appears for the same problem, V_a the number of vetoes issued by designated critical verifiers (see Appendix F), F_a the number of iterations in which a is supported by the forward verifier (which solves the problem independently without feedback), and K the total number of iterations performed for that problem. The coefficients w_{freq}, w_{pen}, and w_{rescue} control the relative strength of each component.

Intuitively, this heuristic favors candidates that recur consistently across iterations, penalizes those repeatedly rejected by critical verifiers, and recovers answers that receive repeated support during generation. The resulting adjustment stabilizes selection when ensemble scores alone are insufficient to discriminate between candidates.

Importantly, this historical bias does not affect risk estimation or threshold updates, thereby preserving the theoretical properties of the core algorithm. Rather than serving as a principled defense against adversarial inputs, it acts as a practical stabilizer against recurring error patterns observed during iterative reasoning.

4. Experiments

4.1. Experimental Setup

Datasets. We assess CURVE across a diverse set of multiple-choice and open-ended reasoning tasks.

Multiple-choice. We use GPQA-Diamond (Rein et al., 2024), a challenging subset of GPQA consisting of 198

[3]Historical context is applied only at the final selection stage and omitted here for clarity.

expert-level questions. For MMLU-Pro, we randomly sample 500 problems from the full benchmark of approximately 12,000 questions (Wang et al., 2024).

Open-ended. For mathematical reasoning, we evaluate on two datasets. We use 228 rule-based problems with difficulty levels 6–7 from Omni-MATH (Gao et al., 2025), and MATH-500, a standard benchmark of competition-level mathematics problems (Lightman et al., 2023). We use rule-based mathematical equivalence evaluators for both Omni-MATH and MATH-500. The evaluators normalize answers and check equivalence via numeric, symbolic, and structural matching (e.g., fractions, tuples, and LaTeX expressions), following standard practices used in prior math benchmarks (e.g., MATH and PRM800K). For Omni-MATH, we adopt the evaluation protocol provided in the Omni-MATH repository.

Models. We consider both a legacy model and recent frontier models: GPT-4o-mini (OpenAI, 2024), GPT-5.1 (OpenAI, 2025), and Gemini-2.5-flash (Comanici et al., 2025). Unless otherwise specified, all models are evaluated using default inference settings.

Verifiers. We employ a heterogeneous set of eight lightweight verifiers that assess candidate answers from complementary perspectives, capturing distinct and partially non-overlapping sources of verification signals. These include backward consistency checks (Kazemi et al., 2023), step-wise reasoning validation (Lightman et al., 2023), distractor refutation, and confidence-based scoring. All verifiers operate via simple prompting and do not require additional training, reward models, or supervision. Additional details are provided in Appendix F and G.

Data Pool Construction via Offline Logging. To ensure reproducible and consistent comparisons across methods, we construct a shared data pool using an offline logging procedure, inspired by common practices in offline evaluation for sequential decision-making (Li et al., 2011). For each problem, we generate an initial solution and collect verifier evaluations. Disagreements among verifiers are aggregated into agreement-based signals, which are used to prompt the generator to refine the answer. This process is repeated for 16 iterations, resulting in a logged pool of candidate solutions and associated verification signals. Using a fixed data pool enables controlled comparisons by decoupling candidate generation from downstream selection and control strategies.

Baselines. We compare CURVE against fixed-budget baselines: (i) CoT, (ii) Self-Consistency ($n = 40$), (iii) Refined Self-Consistency (Refined SC), which applies self-consistency over the shared refined data pool, (iv) Self-Refine ($t = 4, 16$), and (v) MAV where available. All baselines operate under fixed inference budgets, in contrast to CURVE's adaptive test-time control.

Implementation Details. CURVE maintains exponentially weighted moving averages (EWMA) with a smoothing factor $\beta = 0.2$ for (i) coverage, defined as the acceptance rate, and (ii) proxy risk, estimated from inter-verifier agreement. The acceptance threshold is updated using a hinge-style constrained rule: the threshold increases when the estimated risk exceeds the target $\alpha \in [0, 0.3]$, and relaxes when coverage falls below the baseline γ.

Verifier weights are learned via multiplicative updates on accepted rounds only, using a risk-conditioned agreement signal. Specifically, when the running proxy risk is confidently below α, inter-verifier agreement with the ensemble decision is treated as a reliable pseudo-label for updating verifier weights; otherwise, learning is deferred to avoid reinforcing unreliable signals.

The learning rates are initialized at $\eta_w = 0.2$ and $\eta_\tau \in \{0.01, 0.02\}$, and follow an inverse-time schedule $\eta_t = \eta_0/(1 + \lambda_{\text{decay}} t)$ for stability. Hyperparameters are selected from a small predefined grid, using a coarse search over risk–coverage targets to identify a stable operating region, followed by finer sweeps around the best region. Historical context, when enabled, is used only as an auxiliary inductive bias at the selection stage and is excluded from proxy-risk estimation and threshold updates.

5. Results

5.1. Cost–Accuracy Trade-off

Figure 3 shows the accuracy–cost trade-off induced by CURVE on GPQA-Diamond and MMLU-Pro using GPT-

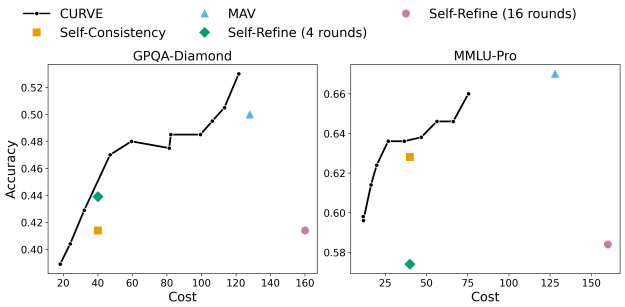

Figure 3. Cost–accuracy trade-off using GPT-4o-mini. Cost is measured as the average number of API calls per problem. CURVE traces a continuum of operating points under different target risk levels α, while fixed-budget methods (Self-Consistency, MAV, Self-Refine) appear as isolated points. CURVE achieves competitive or superior accuracy across a wide range of computational costs without committing to a fixed inference budget.

Table 1. Comparison with fixed-budget reasoning baselines. All numbers report accuracy (%). Baseline methods operate under fixed inference budgets, including Chain-of-Thought decoding (CoT), Self-Consistency and iterative refinement with a fixed number of rounds. Across nearly all model–benchmark combinations, CURVE achieves the best performance, demonstrating the advantage of adaptive, risk-aware control over fixed-budget reasoning strategies. GPQA-D denotes GPQA-Diamond[4]

	GPT-4o-mini				GPT-5.1			Gemini-2.5-flash	
Method	GPQA-D	MMLU-Pro	Omni-MATH	MATH-500	GPQA-D	MMLU-Pro	Omni-MATH	GPQA-D	MMLU-Pro
CoT	36.9	59.2	31.3	68.0	68.7	80.4	39.6	78.6	79.0
Self-Consistency	41.4	62.8	29.4	74.2	68.7	84.4	49.1	82.8	83.6
Self-Refine (4 rounds)	43.9	57.4	23.8	68.8	68.2	79.6	40.5	75.4	74.4
Self-Refine (16 rounds)	41.4	58.4	24.2	66.0	63.6	81.8	51.5	76.5	75.2
Refined SC[5]	50.0	63.2	33.0	73.6	74.8	85.4	50.0	82.8	80.6
CURVE	**53.0**	**66.0**	**33.5**	**75.6**	**76.8**	**85.6**	**55.5**	**84.0**	**84.0**

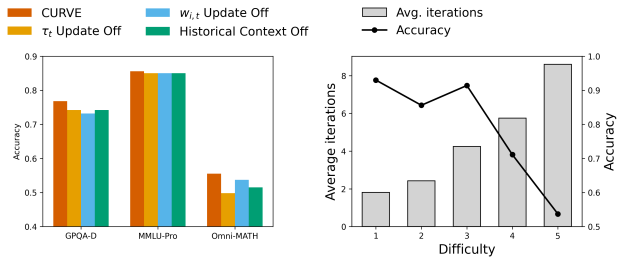

Figure 4. Left: Accuracy ablation results on GPQA-Diamond, MMLU-Pro, and Omni-MATH using GPT-5.1, showing that disabling any individual CURVE component (τ_t update, $w_{i,t}$ update, or historical context) consistently degrades performance. Right: Iteration scaling with problem difficulty on MATH-500 using GPT-4o-mini, where CURVE allocates more reasoning iterations to harder problems while accuracy decreases accordingly, illustrating difficulty-aware control of test-time computation.

4o-mini. The target risk α serves as an explicit control parameter that determines how conservatively the system verifies candidate answers across benchmarks. Computational cost is measured as the average number of API calls per problem, while accuracy is evaluated on the corresponding benchmark.

As α increases, CURVE systematically reduces computational cost by accepting answers earlier, accompanied by a gradual decrease in accuracy. This monotonic trend illustrates a clear and interpretable accuracy–cost trade-off: conservative settings (small α) prioritize reliability at higher cost, whereas relaxed settings (large α) reduce cost with graceful degradation in accuracy. This behavior is consistently observed across both GPQA-Diamond and MMLU-Pro, despite their differing difficulty profiles.

Importantly, this trade-off is not achieved by tuning a fixed iteration budget. Instead, CURVE adapts the effective number of verification steps online, allowing the system to trace a continuous cost–accuracy curve rather than operating at a single fixed point. Each operating point along the curve corresponds to a distinct user-specified reliability constraint, demonstrating that cost and accuracy can be flexibly balanced at test time.

This behavior is consistent with the difficulty-dependent trends observed in the right panel of Figure 4. As difficulty increases, CURVE allocates more iterations, while accuracy remains stable at lower difficulty levels and degrades on the hardest problems, reflecting the model's intrinsic capacity limits rather than insufficient search.

In summary, this result highlights the central advantage of CURVE: it exposes the accuracy–cost landscape as a controllable continuum across multiple benchmarks, enabling users to select operating points that match their resource constraints and tolerance for risk, rather than relying on a single, rigid inference budget.

Additional analyses on performance sensitivity to the maximum iteration budget $T_{\max}$ are provided in Appendix E.1.

5.2. Empirical Validation of Risk–Coverage Control

We empirically validate the theoretical risk–coverage control guarantees of CURVE on GPQA-Diamond using GPT-5.1. Risk and coverage are estimated in a fully label-free manner, based on *inter-verifier consensus* signals rather than gold labels, which directly matches the assumptions of our theoretical analysis.

Figure 5 illustrates the evolution of the EWMA proxy-risk $\hat{R}_t$ and coverage $\hat{C}_t$ under different target risks $\alpha \in \{0.05, 0.15, 0.30\}$, with the target coverage fixed at $\gamma = 0.2$. The goal is not to track these targets exactly, but to keep $\hat{R}_t$ below α while allowing $\hat{C}_t$ to grow above γ whenever

[4]A few samples were excluded due to malformed structured outputs returned by the model, including 11 GPQA-Diamond questions with Gemini-2.5-flash and one Omni-MATH question. Accuracies were computed over the remaining samples.

[5]Refined Self-Consistency (Refined SC) is an internally designed baseline that applies self-consistency over iteratively refined candidates under a fixed budget, and is not directly available in prior work. See Appendix C.

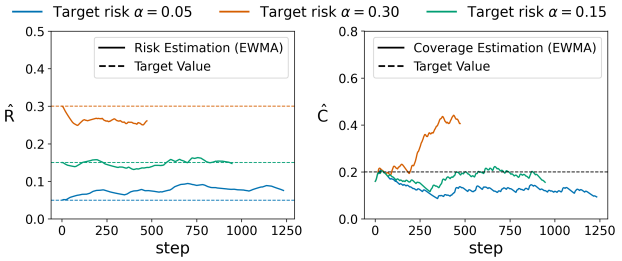

Figure 5. Risk–coverage dynamics on GPQA-Diamond with GPT-5.1 under different target risks $\alpha \in \{0.05, 0.15, 0.30\}$. Left: EWMA proxy-risk estimate $\hat{R}_t$ (solid) with target levels (dashed). Right: EWMA coverage estimate $\hat{C}_t$ (solid) with fixed target coverage $\gamma = 0.2$ (dashed). Different α values induce different stopping behaviors, leading to different step horizons. The EWMA proxy-risk trajectories remain stable around their target levels over time, illustrating the non-accumulation of proxy-risk violations predicted by Theorem 3.1.

possible.

As predicted by theory, looser constraints (larger α) lead to earlier acceptance and shorter verification horizons, resulting in higher coverage while maintaining controlled risk, whereas stricter constraints require sustained verification and limit coverage growth. Different constraint levels therefore induce different stopping behaviors and effective step horizons. In aggregate, these results empirically confirm that the CURVE threshold update mechanism enforces the intended asymmetric risk–coverage constraints in practice, supporting the guarantees established in Theorem 3.1.

5.3. Comparison with Prior Methods in Accuracy and Efficiency

Table 1 reports accuracy under fixed inference budgets. Across nearly all model–benchmark combinations, CURVE achieves the best or tied-best performance, consistently outperforming fixed-budget baselines including Chain-of-Thought, Self-Consistency, and Self-Refine. Notably, CURVE also improves upon *Refined Self-Consistency*, which discussed in Appendix C applies majority voting over the same pool of iteratively refined candidates under a fixed budget, highlighting the benefit of adaptive, risk-aware control beyond candidate quality alone.

Unlike fixed-budget reasoning methods that operate at a single cost level, CURVE exposes the cost–accuracy trade-off as a controllable continuum by adapting its reasoning budget online, allocating more computation to difficult instances while terminating early on easier ones.

We compare CURVE against representative fixed-budget baselines. Chain-of-Thought serves as a minimal baseline, while Self-Consistency (SC) samples 40 independent responses and aggregates them via majority voting. To enable a matched comparison under a fixed budget, *Refined Self-Consistency* applies majority voting over the same set of 16 iteratively refined candidates generated during data construction; while it does not always outperform naive SC, it provides a stronger fixed-budget baseline. For MAV, we report results from the original work, where up to 20 verifiers are used, and conservatively estimate its computational cost assuming only the same eight verifiers as CURVE.

Figure 3 illustrates the resulting cost–accuracy trade-offs. Fixed-budget methods appear as isolated points, whereas CURVE traces a smooth frontier, matching or exceeding baseline accuracy at comparable or lower cost.

5.4. Ablation Study of CURVE Components

We analyze the contribution of CURVE's core components through an ablation study, as shown in left panel of Figure 4. Specifically, we evaluate variants that disable adaptive threshold updates (τ_t), verifier weight updates ($w_{i,t}$), or the use of historical context, while keeping all other components unchanged.

Across benchmarks, disabling any single component consistently degrades accuracy relative to the full CURVE configuration. Removing threshold updates weakens risk-aware control and leads to suboptimal acceptance decisions, while disabling weight updates limits the model's ability to emphasize informative verifiers. Removing the historical context leads to the largest degradation in performance, underscoring the value of incorporating evidence accumulated across iterations.

6. Conclusion and Limitations

We introduced CURVE, an online, label-free framework for test-time reasoning that enables explicit user control over the cost–accuracy trade-off. By formulating answer acceptance as a constrained optimization problem, CURVE adaptively updates verification thresholds and verifier weights based on proxy risk and inter-verifier agreement, avoiding the inefficiencies of fixed-budget reasoning.

Across multiple benchmarks and models, CURVE adapts reasoning effort to problem difficulty, often achieving higher accuracy under equal or lower computational budgets than fixed-budget baselines. These results suggest that dynamic, user-controlled test-time reasoning provides a practical alternative to rigid inference policies.

CURVE relies on proxy risk estimates derived from inter-verifier agreement, which may become biased when verifiers share correlated failure modes. While update gating improves robustness, verifier quality and diversity remain important, and automated verifier construction and initialization are promising directions for future work.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning, with a particular focus on controllable and reliable test-time reasoning for large language models. Our proposed framework enables users to explicitly manage the trade-off between computational cost and reliability, which may support more efficient and transparent deployment of language models under diverse resource constraints.

We do not anticipate significant negative societal impacts arising directly from this work. As with all advances in language model reasoning and deployment, potential downstream uses should be considered carefully, but we believe the contributions of this paper primarily serve to improve the safety, efficiency, and interpretability of machine learning systems.

References

Angelopoulos, A. N., Bates, S., Fisch, A., Lei, L., and Schuster, T. Conformal risk control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=33XGfHLtZg.

Arora, S., Hazan, E., and Kale, S. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012. doi: 10.4086/toc.2012.v008a006. URL https://theoryofcomputing.org/articles/v008a006.

Bartlett, P., Freund, Y., Lee, W. S., and Schapire, R. E. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5): 1651–1686, 1998.

Bates, S., Angelopoulos, A., Lei, L., Malik, J., and Jordan, M. Distribution-free, risk-controlling prediction sets. *Journal of the ACM (JACM)*, 68(6):1–34, 2021.

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Burns, C., Ye, H., Klein, D., and Steinhardt, J. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge university press, 2006.

Chow, C. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970. doi: 10.1109/TIT.1970.1054406.

Comanici, G. et al. Gemini 2.5: Pushing the Frontier with Advanced Reasoning ..., 2025. URL https://arxiv.org/abs/2507.06261.

Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., and Weston, J. Chain-of-verification reduces hallucination in large language models. In *Findings of the association for computational linguistics: ACL 2024*, pp. 3563–3578, 2024.

Dietterich, T. G. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

El-Yaniv, R. and Wiener, Y. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010. URL http://jmlr.org/papers/v11/el-yaniv10a.html.

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Gao, B., Song, F., Yang, Z., Cai, Z., Miao, Y., Dong, Q., Li, L., Ma, C., Chen, L., Xu, R., Tang, Z., Wang, B., Zan, D., Quan, S., Zhang, G., Sha, L., Zhang, Y., Ren, X., Liu, T., and Chang, B. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=yaqPf0KAlN.

Geifman, Y. and El-Yaniv, R. Selective classification for deep neural networks, 2017. URL https://arxiv.org/abs/1705.08500.

Gou, Z., Shao, Z., Gong, Y., yelong shen, Yang, Y., Duan, N., and Chen, W. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Hazan, E. et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Jia, R. and Liang, P. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031, 2017.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Kazemi, M., Kim, N., Bhatia, D., Xu, X., and Ramachandran, D. LAMBADA: Backward chaining for automated reasoning in natural language. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6547–6568, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.361. URL https://aclanthology.org/2023.acl-long.361/.

Kunapuli, G. *Ensemble methods for machine learning*. Simon and Schuster, 2023.

Li, L., Chu, W., Langford, J., and Wang, X. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pp. 297–306, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450304931. doi: 10.1145/1935826.1935878. URL https://doi.org/10.1145/1935826.1935878.

Lifshitz, S., McIlraith, S. A., and Du, Y. Multi-agent verification: Scaling test-time compute with multiple verifiers. *CoRR*, 2025.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *CoRR*, 2023.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

OpenAI. GPT-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, 2024. Accessed: 2025-12-23.

OpenAI. GPT-5.1 Instant and GPT-5.1 Thinking System Card. https://cdn.openai.com/pdf/4173ec8d-1229-47db-96de-06d87147e07e/5_1_system_card.pdf, 2025. Accessed: 2025-12-23.

Radha, S. K., Nouri Jelyani, Y., Ghukasyan, A., and Goktas, O. Iteration of thought: Leveraging inner dialogue for autonomous large language model reasoning. *arXiv e-prints*, pp. arXiv–2409, 2024.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Saad-Falcon, J., Buchanan, E. K., Chen, M. F., Huang, T.-H., McLaughlin, B., Bhathal, T., Zhu, S., Athiwaratkun, B., Sala, F., Linderman, S., et al. Shrinking the generation-verification gap with weak verifiers. *arXiv preprint arXiv:2506.18203*, 2025.

Schapire, R. E. et al. A brief introduction to boosting. In *Ijcai*, volume 99, pp. 1401–1406, 1999.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., and Chen, W. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of Thoughts: Deliberate problem solving with large language models, 2023.

Yu, H., Neely, M., and Wei, X. Online convex optimization with stochastic constraints. *Advances in Neural Information Processing Systems*, 30, 2017.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Zhang, X., Huang, Z., Taga, E. O., Joe-Wong, C., Oymak, S., and Chen, J. Efficient contextual llm cascades through budget-constrained policy learning. *Advances in Neural Information Processing Systems*, 37:91691–91722, 2024.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

A. Related Work and Additional Discussion

This appendix provides additional context for CURVE by situating it within prior work on test-time scaling, verification-based reasoning, and risk-controlled decision making. We emphasize both conceptual connections and key distinctions.

Fixed-Budget Test-Time Scaling. A large body of prior work improves reasoning accuracy by increasing test-time computation under a fixed budget. Chain-of-Thought (CoT) prompting (Wei et al., 2022) encourages explicit intermediate reasoning, while extensions such as Tree-of-Thought (Yao et al., 2023) and Iteration-of-Thought (Radha et al., 2024) structure this process into search or refinement procedures. Self-Consistency (SC) (Wang et al., 2023) improves robustness by marginalizing over multiple independently sampled reasoning traces and aggregating their outcomes.

While effective, these methods apply a uniform computational budget across all instances (e.g., sampling $k = 40$ paths per query). As a result, easy instances are often over-computed, while difficult instances may remain under-verified. Although varying the budget exposes an implicit cost–accuracy trade-off, the control variable is the raw compute budget itself, rather than a reliability-oriented criterion.

Verification and Multi-Agent Reasoning. Recent work has explored verification-based reasoning to improve robustness. Training-based approaches such as Process Reward Models (Lightman et al., 2023) require large amounts of annotated supervision. Prompt-based self-correction methods, including Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023), use iterative feedback loops but may suffer from error reinforcement when the model is confidently wrong.

To mitigate this, multi-agent verification frameworks such as MAV (Lifshitz et al., 2025) and Weaver (Saad-Falcon et al., 2025) employ ensembles of heterogeneous verifiers. These methods typically aggregate verifier outputs using static rules or fixed weights. CURVE aligns with this multi-agent perspective but differs fundamentally by treating verification as an *online decision problem*, dynamically reweighting verifiers and adapting acceptance thresholds based on observed proxy risk.

Adaptive Control and Risk-Aware Decision Making. CURVE is conceptually related to selective prediction and risk control (Geifman & El-Yaniv, 2017; Bates et al., 2021; Angelopoulos et al., 2024; El-Yaniv & Wiener, 2010), which study the trade-off between error and abstention. Unlike conformal prediction, which relies on labeled calibration data and exchangeability assumptions, CURVE operates in a fully label-free, test-time setting. Risk is estimated via inter-verifier agreement rather than ground-truth supervision.

From an optimization standpoint, CURVE builds on ideas from online constrained optimization and no-regret learning (Robbins & Monro, 1951; Zinkevich, 2003). By framing answer acceptance as a sequential constrained decision, CURVE provides a principled mechanism for navigating the cost–accuracy frontier dynamically, rather than fixing computation *a priori*.

Limits of Verification-Based Reasoning. Several recent studies have noted that verification signals may become unreliable when generators and verifiers share correlated failure modes (Kadavath et al., 2022; Saad-Falcon et al., 2025). Our analysis complements these observations by empirically identifying a regime in which even the strongest available verifier fails to outperform the base generator. Below this regime—which we term the *Verifiability Threshold*—any form of ensembling or adaptive control, including CURVE, becomes ineffective. This limitation reflects a fundamental constraint of verification-based reasoning rather than a shortcoming of a specific algorithm.

B. Proof of Proxy-Risk Control with Hinge Threshold Updates

Theorem B.1 (Theorem 3.1 (restated))**.** *Consider the update rule in Eq. 8 with $\lambda_{rc} = 0$, a constant step size $\eta > 0$ and an initial threshold $\tau_1 \in [0, 1]$. Under the feasibility assumption that $(\hat{R}_t - \alpha)_+ = 0$ whenever $\tau_t = 1$, and the boundedness assumption that there exists a constant $G \in (0, \infty)$ such that $0 \le (\hat{R}_t - \alpha)_+ \le G$ for all t, the time-averaged constraint violation satisfies, for any $T \ge 1$,*

$$\frac{1}{T}\sum_{t=1}^{T}(\hat{R}_t - \alpha)_+ \le \frac{1 - \tau_1 + \eta G}{\eta T}. \tag{13}$$

Proof. Let $\tilde{\tau}_{t+1} = \tau_t + \eta(\hat{R}_t - \alpha)_+$ denote the unprojected update at step t. By definition, the violation term $(\hat{R}_t - \alpha)_+$ is non-negative. Thus, $\tilde{\tau}_{t+1} \ge \tau_t$, and the projection onto $[0, 1]$ implies $\tau_{t+1} = \min(1, \tilde{\tau}_{t+1})$. We can express the relationship

between the projected and unprojected updates as:

$$\tau_{t+1} = \tau_t + \eta(\hat{R}_t - \alpha)_+ - \xi_t, \tag{14}$$

where $\xi_t = \max(0, \tilde{\tau}_{t+1} - 1) \geq 0$ represents the saturation loss (residual) caused by the upper bound constraint. Rearranging the terms to isolate the constraint violation, we get:

$$\eta(\hat{R}_t - \alpha)_+ = (\tau_{t+1} - \tau_t) + \xi_t. \tag{15}$$

Summing this equality over the horizon $t = 1, \ldots, T$:

$$\eta \sum_{t=1}^{T} (\hat{R}_t - \alpha)_+ = \sum_{t=1}^{T} (\tau_{t+1} - \tau_t) + \sum_{t=1}^{T} \xi_t. \tag{16}$$

The first term on the right-hand side is a telescoping sum:

$$\sum_{t=1}^{T} (\tau_{t+1} - \tau_t) = \tau_{T+1} - \tau_1. \tag{17}$$

Since $\tau_t \in [0, 1]$ for all t, we have the deterministic bound $\tau_{T+1} - \tau_1 \leq 1 - \tau_1$. Regarding the saturation term $\sum_{t=1}^{T} \xi_t$, note first that $\{\tau_t\}$ is non-decreasing since $\tau_{t+1} = \min\{1, \tau_t + \eta(\hat{R}_t - \alpha)_+\} \geq \tau_t$. If $\xi_t > 0$, then $\tilde{\tau}_{t+1} > 1$, hence $\tau_{t+1} = 1$. By monotonicity and the upper bound $\tau_t \leq 1$, this implies $\tau_k = 1$ for all $k \geq t + 1$. Then the feasibility assumption yields $(\hat{R}_k - \alpha)_+ = 0$ for all $k \geq t + 1$, so $\xi_k = \max(0, \tau_k + \eta(\hat{R}_k - \alpha)_+ - 1) = 0$ for all $k \geq t + 1$. Therefore, at most one index can have $\xi_t > 0$, and thus

$$\sum_{t=1}^{T} \xi_t \leq \max_{1 \leq t \leq T} \xi_t.$$

Moreover, since $\tau_t \leq 1$ and $(\hat{R}_t - \alpha)_+ \geq 0$,

$$\xi_t = (\tau_t + \eta(\hat{R}_t - \alpha)_+ - 1)_+ \leq (\eta(\hat{R}_t - \alpha)_+)_+ = \eta(\hat{R}_t - \alpha)_+ \leq \eta G,$$

where the last inequality uses the boundedness assumption. Hence, $\sum_{t=1}^{T} \xi_t \leq \eta G$. Substituting these into the summation yields:

$$\eta \sum_{t=1}^{T} (\hat{R}_t - \alpha)_+ \leq (1 - \tau_1) + \eta G. \tag{18}$$

Dividing both sides by ηT completes the proof. □

Remark (Sign convention and interpretation). Although Eq. (14) is written in an ascent form, $\tau_{t+1} = \tau_t + \eta(\hat{R}_t - \alpha)_+$, this is consistent with projected subgradient descent on the surrogate loss $f_t(\tau)$. Increasing τ corresponds to a more conservative acceptance policy, which monotonically reduces the induced proxy risk. Thus, the update direction coincides with descending the convex surrogate violation with respect to the policy parameter τ, up to a change of coordinates. This sign convention is standard in online selective prediction and constraint-control formulations.

C. Refined Self-Consistency

Definition and Motivation. We introduce *Refined Self-Consistency* (Refined SC) as an internally designed fixed-budget baseline to evaluate the effectiveness of CURVE under iterative refinement. Unlike conventional Self-Consistency, which aggregates multiple independent generations produced from the same prompt, Refined SC aggregates a sequence of answers obtained through iterative refinement, where each candidate is generated by conditioning on feedback from previous iterations.

The motivation behind Refined SC is to isolate the effect of *adaptive control* in CURVE from the benefits of iterative improvement alone. By applying a simple majority-voting rule over refined candidates under a fixed budget, Refined SC provides a strong yet transparent baseline that captures the benefits of iteration without adaptive stopping or risk-aware control.

Computation Procedure. For each problem, we run the same iterative refinement procedure used to construct the CURVE candidate pool, up to a fixed maximum of $K = 16$ iterations. At each iteration, the model generates a candidate answer conditioned on feedback from previous rounds, resulting in a sequence of refined candidates $\{a_1, \ldots, a_{16}\}$.

Following the self-consistency principle, the final prediction is selected via majority voting over these refined candidates. That is, the answer that appears most frequently across the 16 refinement rounds is chosen as the final output. The Refined SC score is then computed as the standard accuracy of these majority-selected answers with respect to ground-truth solutions.

Interpretation. Refined SC measures the extent to which iterative feedback leads to stable and correct convergence, rather than isolated correct predictions. By aggregating over a fixed number of refinement rounds, it emphasizes robustness and agreement among refined solutions, while remaining strictly within a fixed-budget setting.

As shown in Table 1, Refined SC already provides a strong baseline across multiple benchmarks. However, CURVE consistently outperforms Refined SC, demonstrating that simple iteration combined with majority voting is not sufficient to match the gains achieved by adaptive, risk-aware test-time control. This comparison highlights that CURVE's improvements stem not merely from iterative refinement, but from its ability to dynamically regulate acceptance decisions and verification effort.

D. Assumption: Weakly Informative Verification Ensembles

Verification-based reasoning can improve accuracy only if verification signals are *informative in aggregate*. At a minimum, an ensemble of verification signals must be able to distinguish correct answers from incorrect ones better than random guessing. From a decision-theoretic perspective, this requires that there exists a scoring function whose true positive rate (TPR) is at least as large as its false positive rate (FPR) over some operating range. Equivalently, accepted answers should be correct more often than incorrect ones in expectation.

This condition does not require any single verifier to be strong. Rather, it only requires that *a combination of verifiers* can produce a weakly informative signal, i.e., a signal that is positively correlated with correctness. This principle underlies classical ensemble methods, including boosting, where multiple weak learners are combined to achieve reliable discrimination. This intuition is reminiscent of classical ensemble methods such as boosting, though we do not rely on any specific boosting assumptions here (Freund & Schapire, 1997; Schapire et al., 1999).

Ensemble-level formulation. We formalize this minimal requirement at the level of an ensemble score. Let S denote an ensemble scoring function obtained by aggregating one or more verification signals (e.g., via weighted averaging or any other combination rule). Let S^+ and S^- denote the random variables corresponding to ensemble scores assigned to correct (positive) and incorrect (negative) answers, respectively. We write F_{S^+} and F_{S^-} for their cumulative distribution functions, where $F_{S^+}(x) = \mathbb{P}(S^+ \leq x)$.

We assume that the ensemble score satisfies *weak stochastic dominance*:

$$F_{S^+}(x) \leq F_{S^-}(x), \qquad \forall x \in [0, 1]. \tag{19}$$

This condition states that correct answers tend to receive higher ensemble scores than incorrect ones in distribution. Importantly, it does *not* require perfect separation, a fixed decision boundary, or a calibrated probability estimate. It merely ensures that thresholding the ensemble score yields a region where TPR $\geq$ FPR, making selective acceptance and risk control meaningful.

Relation to CURVE. Equation (19) is defined over an arbitrary ensemble score S and is independent of a particular algorithm. In CURVE, the ensemble score corresponds to the adaptive weighted verifier aggregation $s_t = \sum_i w_{i,t}\hat{v}_{i,t}$, but this represents only a particular instance of the broader formulation introduced above. CURVE does not assume that each verifier is informative on its own; instead, it aims to *construct* an informative ensemble online by reweighting verifiers and adapting acceptance thresholds.

Empirical validation of weak stochastic dominance. We empirically examine whether the weak stochastic dominance (WSD) condition discussed in this section is supported by observed test-time behavior. Specifically, we analyze the empirical distributions of ensemble scores assigned to correct and incorrect candidate answers.

Since selective acceptance decisions are only meaningful for candidates that are ultimately accepted, we restrict the analysis to accepted rounds ($A_t = 1$) Eq.(3). For each evaluated model–benchmark pair, we collect ensemble scores of final accepted candidates, partition them by ground-truth correctness, and compute their empirical cumulative distribution functions (CDFs). Figure 6 visualizes representative examples of these distributions.

The empirical CDF provides a non-parametric summary of the score distribution without assuming a specific functional form. Under weak stochastic dominance, the CDF corresponding to correct answers tends to lie below that of incorrect answers, indicating that correct candidates receive higher ensemble scores in distribution. Crossings between the two CDFs indicate score regions where this ordering is locally ambiguous, suggesting regimes in which verification signals may become less discriminative.

Across the evaluated settings, we observe that the empirical CDFs of correct answers are generally shifted to the right of those of incorrect answers, with no prominent or systematic crossings in the examined cases. While this analysis does not exhaustively characterize all possible model–task regimes, the observed trends suggest that the ensemble score provides a weakly informative signal for separating correct and incorrect candidates in practice.

Importantly, this evaluation is conducted using ground-truth correctness and is not used by the CURVE controller during test-time operation. To isolate the behavior of the raw ensemble score, the historical context module is disabled when collecting these statistics. Taken together, these results provide empirical support for the applicability of selective acceptance and adaptive control under weakly informative verification signals.

Failure modes and limits. When Eq. (19) fails, ensemble scores for correct and incorrect answers become statistically indistinguishable, implying $\mathrm{TPR} \approx \mathrm{FPR}$ for all thresholds. In this regime, threshold updates and verifier reweighting carry no directional information, and additional computation cannot reliably improve performance. We refer to this boundary as the *Verifiability Threshold*. This limitation is fundamental and applies to verification-based reasoning methods in general, not only to CURVE.

Verifiability Threshold . The Verifiability Threshold captures a gradual transition in the usefulness of verification signals as problem difficulty increases relative to model and verifier capability. Beyond this boundary, score distributions for correct and incorrect answers substantially overlap, and selective acceptance becomes unreliable. Importantly, this threshold does not represent a sharp phase change, nor an assumption required by CURVE. Rather, it provides an empirical lens for interpreting when and why verification-based test-time scaling succeeds or fails. CURVE operates effectively when weakly informative signals exist, but cannot overcome the fundamental limits imposed when verification becomes uninformative.

Figure 7 provides an illustrative view of this phenomenon. Below the threshold, verification signals behave close to chance-level, and combining multiple verifiers offers little benefit beyond stochastic variation. Above the threshold, at least one weakly informative signal exists, allowing ensembles to amplify verification-induced improvements through aggregation. CURVE operates effectively in the latter regime by adaptively exploiting such weak signals, but cannot overcome the fundamental limitations imposed when verification becomes uninformative.

We emphasize that the Verifiability Threshold is not an assumption required by our method, but an empirical lens for interpreting when and why verification-based test-time scaling succeeds or fails.

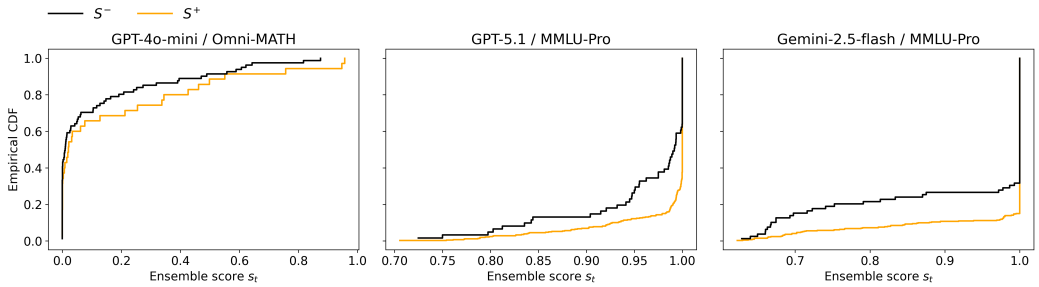

Figure 6. Empirical CDFs of ensemble scores for correct (S^+) and incorrect (S^-) accepted answers across representative model–benchmark pairs. Ensemble scores are computed on accepted rounds ($A_t = 1$) and evaluated using ground-truth correctness. Across settings, the CDF of correct answers lies consistently below that of incorrect answers, indicating weak stochastic dominance and a weakly informative verification signal.

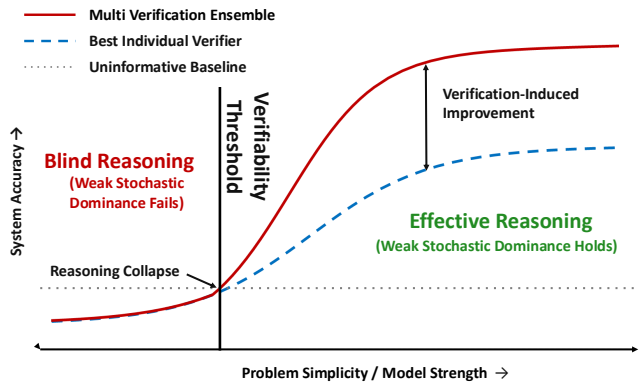

Figure 7. Verifiability regimes in multi-verification ensembles (conceptual). Below a verifiability threshold, verification signals are effectively uninformative and aggregation provides little benefit. Above the threshold, weak but informative signals can be amplified through ensembling, enabling consistent gains over the best individual verifier. The threshold is intended as an illustrative boundary rather than a sharp theoretical phase transition.

E. Additional Experimental Results

E.1. Impact of Maximum Iteration Budget

To understand the scaling properties of CURVE, we analyze how the maximum iteration budget ($T_{\max}$) influences the final accuracy. Figure 8 illustrates the cumulative accuracy gain (Δ Accuracy) relative to the CoT baseline (Iteration 0) as $T_{\max}$ increases from 1 to 16.

Positive Scaling Laws. As shown in the figure, for the majority of model–benchmark combinations, we observe a monotonic increase in accuracy as the iteration budget expands. Notably, for **GPT-4o-mini on GPQA-Diamond** and **GPT-5.1 on Omni-MATH**, the performance curves show a steep upward slope without signs of saturation even at $T = 16$. This indicates that CURVE effectively converts additional test-time compute into reliability gains, suggesting that further performance improvements could be realized with larger computational budgets.

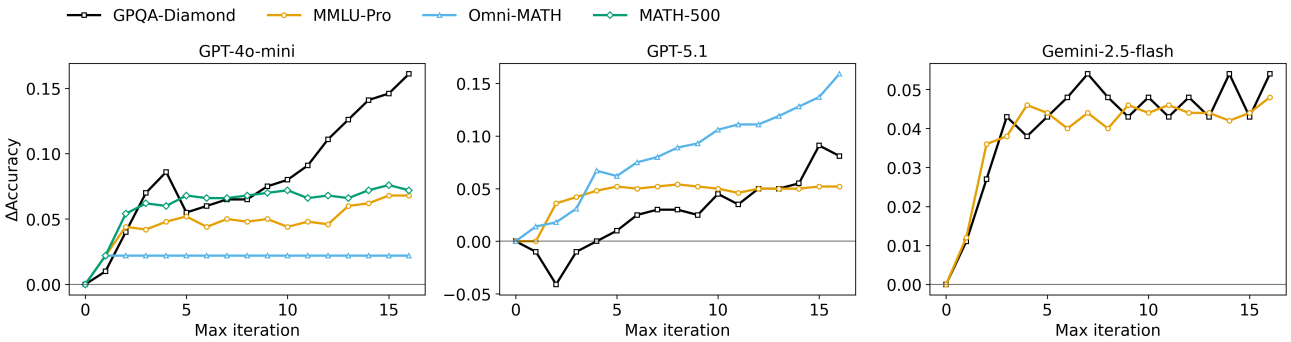

Figure 8. Accuracy gain as a function of the maximum number of verification iterations. Each curve reports the change in accuracy (ΔAccuracy) relative to the baseline CoT (iteration $= 0$) as the allowed maximum iteration increases. Results are shown across multiple benchmarks and model backbones. For most model–benchmark combinations, accuracy improves monotonically with additional verification iterations, indicating that CURVE effectively exploits extra test-time computation when informative verification signals are available.

Saturation and Limited Scaling. Conversely, some configurations (e.g., Gemini-2.5-flash on MMLU-Pro) exhibit early saturation, suggesting that the model reaches its effective performance ceiling after only a small number of iterations. Notably, **GPT-4o-mini on Omni-MATH** is the only setting that shows no consistent accuracy gain, with performance remaining nearly flat as the iteration budget increases. This indicates that, for this model–benchmark pair, the model's inherent

reasoning limitations are reached before additional iterations can translate into measurable performance improvements.

E.2. Evaluating the Effect of Verification Ensembling via Single-Verifier Comparisons

To assess whether verification signals are sufficiently informative to support ensembling, we analyze the performance of individual verifiers in isolation. For each model–benchmark pair, we apply the CURVE framework using only one verifier at a time (out of eight), without ensembling, for up to 16 iterations, and report the accuracy of the best-performing single verifier. Table 2 summarizes the results.

Table 2. Final accuracy (%) of the best single verifier, the Chain-of-Thought (CoT) baseline, and the full CURVE ensemble. The *single verifier* result corresponds to the best-performing verifier when CURVE is applied using only one verifier at a time (out of eight), without ensembling, for up to 16 iterations. Across all settings, the best single verifier consistently outperforms the CoT baseline, indicating that individual verification signals are informative. The full CURVE ensemble achieves accuracy that is consistently equal to or higher than the best single verifier, demonstrating that aggregating multiple verification signals yields additional or non-degrading gains. The Omni-MATH / GPT-4o-mini case represents a limiting regime where verification signals are weak, and CURVE matches but does not substantially exceed the best single verifier.

Model	Benchmark	CoT	Single Verifier	**CURVE**
GPT-4o-mini	GPQA-Diamond	36.9	48.0	**53.0**
	MMLU-Pro	59.2	62.6	**66.0**
	Omni-MATH	31.3	33.5	**33.5**
	MATH-500	68.0	71.2	**75.6**
GPT-5.1	GPQA-Diamond	68.7	74.7	**76.8**
	MMLU-Pro	80.4	84.4	**85.6**
	Omni-MATH	39.6	48.0	**55.5**
Gemini-2.5-flash	GPQA-Diamond	78.6	80.7	**84.0**
	MMLU-Pro	79.0	83.2	**84.0**

Across all evaluated settings, the best single verifier consistently outperforms the Chain-of-Thought (CoT) baseline. This indicates that at least one verifier provides a weakly informative signal beyond the generator alone, satisfying the minimal condition required for verification-based reasoning to be effective.

Moreover, the full CURVE ensemble achieves accuracy that is consistently equal to or higher than that of the best single verifier. This demonstrates that aggregating multiple verification signals does not degrade performance and, in most cases, yields additional gains by amplifying informative signals through ensembling.

Taken together, these results provide empirical evidence that verification signals in our evaluated settings are sufficiently informative to support multi-verifier aggregation. Rather than relying on any single strong verifier, CURVE leverages the presence of at least one weakly informative verifier to improve performance through adaptive ensembling.

E.3. Empirical Risk–Coverage Frontier

We provide additional empirical evidence that the adaptive control mechanism of CURVE induces a structured trade-off not only in proxy quantities, but also in realized test-time outcomes. Figure 9 visualizes the empirical risk–coverage (RC) plane for GPT-4o-mini on GPQA-Diamond and MMLU-Pro, constructed using ground-truth correctness rather than proxy signals. Each gray point corresponds to an operating point induced by a distinct choice of target parameters (α, γ). For each setting, we measure the empirical risk (error rate among accepted answers) and empirical coverage (fraction of accepted queries) over the full evaluation set. Importantly, these quantities are computed using true labels and are not involved in the online control process.

The solid black curve highlights the empirical Pareto frontier formed by these operating points. Across both benchmarks, the observed points concentrate along a monotonic frontier in the RC plane, demonstrating that CURVE systematically exposes a controllable relationship between reliability (risk) and selectivity (coverage) in realized performance. This behavior closely aligns with classical analyses of selective classification and abstention, which characterize achievable regions and frontiers in the risk–coverage plane (El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017), while extending these ideas to a label-free, test-time adaptive setting.

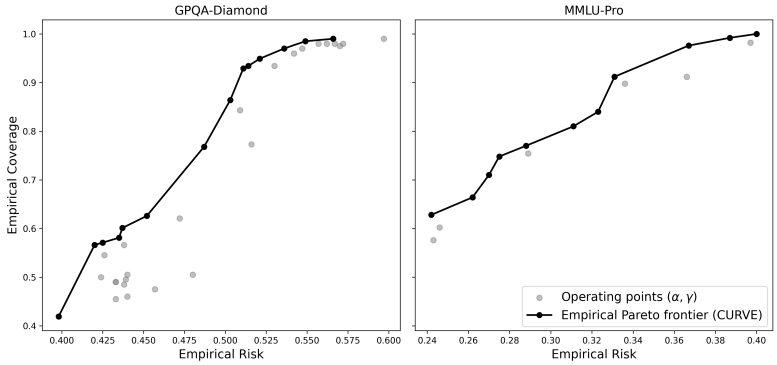

Figure 9. Risk–coverage planes for GPT-4o-mini on GPQA-Diamond (left) and MMLU-Pro (right). Each gray point corresponds to an operating point induced by a distinct choice of target risk–coverage parameters (α, γ), plotted using empirically measured risk and coverage. The solid black curve highlights the boundary of achievable risk–coverage trade-offs observed in practice. Together, these plots show that CURVE, using the same backbone model, systematically exposes a controllable relationship between reliability (risk) and selectivity (coverage) across different benchmarks.

Taken together, this result confirms that the trade-offs induced by CURVE are not artifacts of proxy estimation. Although the controller operates solely on inter-verifier agreement signals, the resulting operating points form a coherent RC frontier when evaluated against ground-truth outcomes. Taken together with the main results, this analysis shows that CURVE induces frontier behavior at multiple levels: computational cost versus accuracy, proxy risk versus coverage, and realized risk versus coverage. This consistency supports the interpretation of CURVE as an online control mechanism that traces a meaningful decision frontier, rather than tuning isolated operating points.

F. Verifier Architecture and Design

We instantiate a suite of eight lightweight verifiers for each task, each reflecting a distinct verification principle. The verifier suite is task-dependent: while the total number of verifiers is fixed to eight, the specific composition differs between multiple-choice and open-ended tasks. The verifiers are designed to be fast, complementary, and weakly informative in isolation, while yielding a stronger aggregate signal when combined. Except for the confidence-score verifier, which outputs a scalar value in $[0, 1]$, all verifiers produce binary judgments (`correct`/`incorrect`).

Overview of Verifiers.

- FORWARD VERIFIER: Re-solves the problem independently and checks exact agreement with the candidate answer.
- BACKWARD VERIFIER: Verifies consistency by reasoning backward from the candidate answer to the given conditions.
- STEPWISE VERIFIER: Validates each provided solution step for local logical and arithmetic correctness.
- ORDER-OF-MAGNITUDE VERIFIER: Checks whether the candidate answer lies within a plausible quantitative range.
- CONFIDENCE-SCORE VERIFIER: Outputs a scalar confidence score based on lightweight consistency and sanity cues.
- PARAPHRASE VERIFIER: Tests robustness of the candidate answer under semantically equivalent paraphrases.
- SELF-REVIEW VERIFIER: Performs a checklist-style audit for common logical, numerical, and conceptual errors.
- TASK-SPECIFIC VERIFIER:
  - DISTRACTOR REFUTATION VERIFIER (multiple-choice): Verifies correctness by refuting all non-candidate options.
  - SOLUTION-NECESSITY VERIFIER (open-ended): Checks whether all necessary problem constraints are actually used in the solution.

Critical Verifiers and Veto Policy. Among the verifier suite, a subset of verifiers are designated as *critical* and are allowed to issue vetoes that incur an explicit penalty in the historical context re-scoring stage. By default, the SELF-REVIEW VERIFIER is treated as critical, as it targets general and systematic failure modes. The ORDER-OF-MAGNITUDE VERIFIER is also treated as critical when applicable, since large numerical deviations typically indicate irrecoverable errors.

Task-dependent critical verifiers are additionally used. For multiple-choice questions, the DISTRACTOR REFUTATION VERIFIER is treated as critical, as incorrect elimination of alternatives directly invalidates the candidate. For open-ended questions, the SOLUTION-NECESSITY VERIFIER is treated as critical, as it checks whether essential problem constraints are actually used in the solution.

G. Verifier Prompts

We instantiate eight verifier prompts reflecting distinct verification principles. Among them, only the confidence-based verifier outputs a scalar score in the range [0,1], while the remaining verifiers produce binary (true/false) decisions.

G.1. FORWARD VERIFIER

Concept. This verifier evaluates candidate answers through forward reasoning by independently re-solving the problem from the original question. It derives its own solution using a standard forward problem-solving process and compares the resulting answer with the candidate answer produced by the generator. The verifier issues a binary judgment based on exact agreement between the two answers.

Verification Objective. The objective of this verifier is to determine whether the candidate answer is reproducible under an independent forward solution process. If the answer obtained by the verifier matches the generator's output, the candidate is marked as correct; otherwise, it is deemed incorrect. This verifier is designed to detect errors arising from flawed reasoning or accidental correctness by enforcing solution reproducibility.

Prompt.

```
"""
You are an expert mathematician.
Solve subjective mathematical reasoning question.

Constraints for the "final_answer" field:
1. Write **ONLY** the final mathematical result in LaTeX format.
2. Do not include any explanatory text, sentences, or words (e.g., "The answer
is").
3. Do not include variable assignments (e.g., write "5" instead of "x=5").
4. Ensure all backslashes in LaTeX are properly escaped for JSON (e.g., use
"\\frac" instead of "\frac").

Think step by step.
Output your response strictly in the following JSON format:
{{
  "solution_steps": ["step 1", "step 2", "..."],
  "final_answer": <Use LaTeX format for the final answer>
}}


You MUST follow the final answer format.
You MUST NOT output anything outside the JSON.


# Now solve the problem.
Question:
```

```
{question}

"""
```

G.2. BACKWARD VERIFIER

Concept. This verifier is based on explicit backward reasoning. It assumes the candidate answer is correct and traces the reasoning process in reverse, starting from the proposed answer and step-by-step verifying consistency with the initial conditions specified in the problem. Each numeric and logical constraint is checked explicitly during the backward verification process.

Verification Objective. The objective of this verifier is to determine whether a given candidate answer is logically and numerically consistent with all conditions stated in the problem. If any condition is violated during backward verification, the answer is marked as incorrect; otherwise, it is deemed correct.

Prompt.

```
"""
You are a backward-reasoning verifier for problems.

Given:
- Question: {question}
- Candidate Answer: {candidate_answer}

Your job is to verify the candidate answer by **reasoning backward step-by-step**
clearly from the candidate answer to initial given conditions.

Follow these instructions **exactly**:
1. Assume the candidate answer is correct.
2. Clearly show each step backward, verifying each numeric condition explicitly
given in the question.
3. If at any point, any condition fails, conclude "incorrect" clearly stating
which condition failed.
4. If all conditions are satisfied consistently, conclude "correct".

Strictly follow this JSON format:
{{
  "reasoning": "<step-by-step backward verification reasoning>",
  "candidate_answer": <candidate_answer>,
  "verdict": "correct" or "incorrect"
}}

No additional keys, no extra text.
"""
```

G.3. STEPWISE VERIFIER

Concept. This verifier performs strict step-by-step validation of a provided solution. Rather than re-deriving the solution, it evaluates each intermediate step exactly as given, checking for logical and arithmetic correctness. The verifier treats the sequence of steps as fixed and assesses internal consistency without introducing alternative reasoning paths.

Verification Objective. The objective of this verifier is to determine whether all provided solution steps are individually correct and collectively consistent. If every step satisfies the required logical and arithmetic constraints, the candidate answer is deemed correct; if any step is incorrect or inconsistent, the entire solution is marked as incorrect, with the offending step explicitly identified.

Prompt.

```
"""
You are a stepwise verifier.

Given:
- Problem: "{question}"
- Candidate Answer: {candidate_answer}
- Solution Steps: {solution_steps}

Instructions:
1. Strictly verify each given step individually exactly as provided.
   - Do NOT recalculate differently.
2. If every step provided is logically and arithmetically correct, conclude
"correct".
3. If any provided step is incorrect or inconsistent, clearly identify it, and
conclude "incorrect".

Strictly follow this JSON format:
{{
  "reasoning": "<Explain clearly any incorrect step.>",
  "overall_verdict": "correct" or "incorrect",
  "stepwise_results": [
    {{"step": 1, "verdict": "correct" or "incorrect"}},
    ...
  ]
}}

No additional keys, no extra text.
"""
```

G.4. ORDER-OF-MAGNITUDE VERIFIER

Concept. This verifier assesses the plausibility of a candidate answer through order-of-magnitude reasoning. When applicable, it derives a coarse expected value range using standard physical constants, dimensional analysis, and domain-agnostic heuristics, without fully re-solving the problem. The verifier emphasizes unit consistency and explicit assumptions, providing a conservative plausibility band typically spanning one to two orders of magnitude.

Verification Objective. The objective of this verifier is to determine whether a candidate answer lies within a physically or quantitatively plausible range implied by the problem context. If the answer falls outside the derived order-of-magnitude band, it is deemed incorrect; otherwise, it is considered correct. For questions where magnitude reasoning is not applicable, the verifier abstains from judgment.

Prompt.

```
"""
You are an order-of-magnitude (OOM) verifier. Determine whether the candidate
answer lies in a plausible band.

Given:
- Question: {question}
- Candidate Answer: {candidate_answer}
- Solution Steps: {solution_steps}

Instructions:
```

```
1) Decide if an OOM/band check is applicable. If the quantity is categorical or
purely verbal, set "applicability":"not_applicable".
2) If applicable, derive a coarse expected band using standard constants and
heuristics only
(e.g., k_B T at 300 K ~ 0.026 eV; pH in [0, 14];
 human-scale densities ~ 1e3 kg/m^3;
 diffusion ~ sqrt(2 D t);
 half-life / exponential scales).
   Keep assumptions explicit. Perform unit checks and convert all values to a
   consistent unit.
3) Provide a conservative band [lower, upper] (typically within 1-2 orders of
magnitude of your estimate).
4) Evaluate the candidate_answer value against the band; decide
"plausible/implausible/uncertain".
5) Do not fully re-solve the problem; only magnitude/dimension reasoning.
6) Return ONLY the following JSON:

{{
  "reasoning": "<Your OOM reasoning with assumptions and unit checks>",
  "verdict": "correct" or "incorrect"
}}
"""
```

G.5. Confidence-Score Verifier

Concept. This verifier functions as a lightweight confidence calibrator rather than a full correctness checker. It estimates the likelihood that a candidate answer is correct by examining high-level cues such as internal consistency, alignment with definitions, unit and dimensional compatibility, category plausibility, and basic sanity checks. The verifier deliberately avoids re-solving the problem and relies on minimal, fast-to-evaluate signals.

Verification Objective. The objective of this verifier is to produce a calibrated scalar confidence score in the range [0,1] that reflects the estimated correctness of a candidate answer. The score is intended to capture coarse-grained reliability rather than definitive correctness, providing a probabilistic assessment based on weak but informative verification cues.

Prompt.

```
"""
You are a confidence calibrator. Estimate how likely the candidate answer is
correct based on internal consistency, cue alignment, and minimal sanity checks.
Do NOT fully re-solve the problem.

Given:
- Question: {question}
- Candidate Answer: {candidate_answer}
- Solution Steps: {solution_steps}

Instructions:
1) Consider quick cues (definition fit, unit/dimension compatibility, category
plausibility, known bounds) without full derivation.
2) List 2-4 short signals that support or undermine the candidate.
3) Output a calibrated confidence in [0,1].
4) Return ONLY the following JSON (no extra text):

{{
  "confidence": <float between 0 and 1>
```

```
}}
"""
```

G.6. DISTRACTOR REFUTATION VERIFIER

Concept. This verifier assesses candidate answers by ruling out all non-candidate options through concise, fact-based refutations, without re-solving the problem. Each alternative is eliminated based on a concrete violation such as a definition conflict, unit or dimensional mismatch, order-of-magnitude error, or logical inconsistency.

Verification Objective. The objective of this verifier is to verify that the candidate answer remains valid after all competing options are correctly refuted. If any refutation is invalid or applies to the candidate itself, the answer is marked as incorrect; otherwise, it is deemed correct.

Prompt.

```
"""
You are a rigorous verifier. Your job is to REJECT wrong options with brief,
factual refutations.

Given:
- Question: {question}
- Candidate Answer: {candidate_answer}
- Solution Steps: {solution_steps}

Instructions:
1) For EACH NON-CANDIDATE option, write ONE concise refutation sentence that
states a concrete reason it cannot be correct
   (e.g., violated law/principle, unit/dimension mismatch, definition conflict,
   order-of-magnitude error, logical inconsistency).
2) Mark each refutation as "pass" if the reason is valid and specific; otherwise
"fail".
3) Sanity-check the candidate: ensure that none of your refutation reasons
actually apply to the candidate.
4) DO NOT re-solve the whole problem; focus on ruling out distractors with
minimal facts.
5) Return ONLY the following JSON (no extra text):

{{
  "refutations":
    {{"A": "<reason>",
     "B": "<reason>", ... <for all non-candidate options>}},
  "verdict": "correct" or "incorrect"
}}
"""
```

G.7. SOLUTION-NECESSITY VERIFIER

Concept. This verifier checks whether the solution explicitly or implicitly uses the necessary conditions required for the candidate answer to be correct, without re-solving the problem.

Verification Objective. The objective is to detect answers that are accidentally correct or derived via invalid shortcuts by verifying that all critical constraints are actually used.

Prompt.

```
"""
You are a Solution-Necessity verifier.

Your task is NOT to re-solve the problem.
Your task is to check whether the proposed solution necessarily uses
the critical conditions required to reach the candidate answer.

Given:
- Question:
{question}
- Candidate Answer:
{candidate_answer}
- Solution Steps:
{solution_steps}

Instructions:
1. Identify 1{3 NECESSARY conditions or facts from the question
   that MUST be used for the candidate answer to be correct.
   (e.g., specific constraints, key assumptions, boundary conditions,
   required equations, or defining properties.)
2. Check whether each necessary condition is explicitly or implicitly
   used in the provided solution steps.
3. If ANY necessary condition is missing, ignored, or contradicted,
   conclude the answer is "incorrect".
4. If all necessary conditions are used consistently, conclude "correct".
5. Do NOT introduce new facts and do NOT re-derive the full solution.

Return ONLY the following JSON format (no extra text):

{{
  "necessary_conditions": [
    "<condition 1>",
    "<condition 2>", ...
  ],
  "usage_check": [
    {{"condition": "<condition 1>", "used": true or false}},
    {{"condition": "<condition 2>", "used": true or false}}, ...
  ],
  "verdict": "correct" or "incorrect"
}}
"""
```

G.8. PARAPHRASE VERIFIER

Concept. This verifier evaluates the robustness of a candidate answer to surface-level linguistic variation. It generates multiple concise paraphrases of the original question that preserve all factual content, constraints, and quantities, and assesses whether the candidate answer remains valid under these semantically equivalent reformulations. The verifier does not re-solve the problem and focuses on invariance to phrasing rather than derivational correctness.

Verification Objective. The objective of this verifier is to determine whether a candidate answer is stable under paraphrasing of the problem statement. If the answer remains consistent across all paraphrased versions, it is deemed correct; otherwise, it is marked as incorrect. This verifier targets errors arising from misinterpretation or sensitivity to superficial wording changes rather than substantive reasoning flaws.

Prompt.

```
"""
You are a paraphrase-invariance verifier. Lightly rephrase the question and check
whether the candidate answer remains the same under paraphrase. Do NOT change the
factual content.

Given:
- Question: {question}
- Candidate Answer: {candidate_answer}

Instructions:
1) Produce TWO semantically equivalent, concise paraphrases that preserve all
constraints and quantities.
2) For each paraphrase, judge whether the candidate answer still appears correct
("same") or not ("different") WITHOUT fully re-solving.
3) If both judgments are "same", set consistency="pass"; otherwise "fail".
4) If consistency="pass" set verdict="correct"; else "incorrect".
5) Return ONLY the following JSON (no extra text):

{{
  "paraphrases": ["<paraphrase #1>", "<paraphrase #2>"],
  "judgments": ["<same|different>", "<same|different>"],
  "consistency": "<pass|fail>",
  "verdict": "correct" or "incorrect"
}}
"""
```

G.9. SELF-REVIEW VERIFIER

Concept. This verifier performs a critical self-audit of a candidate answer by checking for common failure modes without re-solving the problem from scratch. It applies a concise checklist covering unit and dimensional consistency, adherence to relevant laws or principles, order-of-magnitude plausibility, logical coherence, and consistency across provided solution steps. The verifier emphasizes error detection through systematic scrutiny rather than full derivation.

Verification Objective. The objective of this verifier is to identify whether a candidate answer contains any critical flaws that invalidate its correctness. If any check in the self-review process fails, the answer is marked as incorrect; otherwise, it is deemed correct. This verifier is designed to catch subtle but consequential errors that may evade more localized or single-principle verification methods.

Prompt.

```
"""
You are a self-review verifier. Critically audit the candidate answer for common
pitfalls without re-solving from scratch.

Given:
- Question: {question}
- Candidate Answer: {candidate_answer}
- Solution Steps: {solution_steps}

Instructions:
1) Run a short checklist: unit/dimension, law/principle, order-of-magnitude,
logical coherence, and step consistency (if steps given).
2) If ANY critical check fails, set verdict="incorrect"; else "correct".
3) Return ONLY the following JSON (no extra text):
```

```
{{
  "review": "<Your critical evaluation of the solution steps and final answer>",
  "verdict": "correct" or "incorrect"
}}
"""
```

H. Practical Guidelines for Deployment

A common question in deploying online control frameworks is how to initialize the user targets (α, γ) on new, unseen tasks. In CURVE, α is a *user-specified safety constraint* (maximum tolerated proxy risk among accepted answers), while γ is a *coverage preference* that is pursued only when the system is safely below the risk target. Accordingly, (α, γ) should be chosen based on application requirements rather than task difficulty.

- **High-stakes scenarios (e.g., medical, legal):** Use a strict risk target (e.g., $\alpha \leq 0.05$) and a conservative coverage baseline (e.g., $\gamma \leq 0.1$). If the task is difficult or the verifier pool is weak, CURVE will behave conservatively by increasing the acceptance threshold, which lowers coverage to reduce errors—a desirable fail-safe behavior.
- **General-purpose / creative scenarios:** Use a relaxed risk target (e.g., $\alpha \in [0.1, 0.2]$) with a higher coverage baseline (e.g., $\gamma \in [0.2, 0.4]$). Under looser constraints, CURVE can accept earlier and more often, improving coverage and reducing compute on easy instances.

While CURVE adapts online, a short warm-up phase (e.g., 10–20 steps) can be used to calibrate the initial threshold τ_1 and to check whether the requested coverage level is feasible for the current model–task pair.