Re-FORC: Adaptive Reward Prediction for Efficient Chain-of-Thought Reasoning

Renos Zabounidis[1,2]*, **Aditya Golatkar**[1], **Michael Kleinman**[1],

Alessandro Achille[1], **Wei Xia**[1], **Stefano Soatto**[1]

[1] AWS Agentic AI, [2]Carnegie Mellon University

Abstract

We propose Re-FORC, an adaptive reward prediction method that, given a context, enables prediction of the expected future rewards as a function of the number of future thinking tokens. Re-FORC trains a lightweight adapter on reasoning models, demonstrating improved prediction with longer reasoning and larger models. Re-FORC enables: 1) *early stopping* of unpromising reasoning chains, reducing compute by 26% while maintaining accuracy, 2) *optimized model and thinking length selection* that achieves 4% higher accuracy at equal compute and 55% less compute at equal accuracy compared to the largest model, 3) *adaptive test-time scaling*, which increases accuracy by 11% in high compute regime, and 7% in low compute regime. Re-FORC allows dynamic reasoning with length control via cost-per-token thresholds while estimating computation time upfront.

1 Introduction

Modern reasoning models can dynamically use inference-time computation to improve answer quality, but determining the optimal amount of inference-time computation for a given query remains an open challenge. While an inference pipeline can generate longer reasoning traces, backtrack to explore alternatives, or even delegate computation to specialized models, the large action space makes it difficult to identify the best strategy for any given query. This challenge is compounded by the diversity of user requirements. Different users have varying tolerance for latency and assign different values to output quality. A strategy that works well for a time-sensitive application may be entirely inappropriate for a high-stakes decision where accuracy is paramount. How can we adaptively optimize inference-time compute based on both query difficulty and user-specific constraints?

The question can be framed more generally as maximizing the net utility of inference:

$$J = \mathbb{E}[R^*] - \lambda T_{\text{total}}$$

where $\mathbb{E}[R^*]$ represents the expected utility of the best answer found during inference-time reasoning,[2] T_{total} is the total compute cost incurred, λ captures the user's cost sensitivity and the expectation is over the randomness in inference.

While models can be trained for specific trade-offs, dynamically adapting to user constraints at inference time remains largely unexplored.[3] Our key insight is that current models lack the **ability to**

*Work done during an internship with AWS Agentic AI. Correspondence to agolatka@amazon.com

[3]As pointed out in [1], there is no canonical choice of λ when evaluating J, since the value of an agent depends on the user and the environment. The example provided therein is of an agent tasked with determining whether a fruit is edible: if an agent takes so long that the fruit has spoiled, providing the correct answer is of

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

predict the marginal benefit of additional computation for a given query. Lacking this, we cannot make informed decisions about *when to continue reasoning, when to backtrack, or when the expected improvement no longer justifies the computational cost.*

To address this gap, we introduce Re-FORC, the first method for forecasting the reward-versus-compute trade-off both before and during inference in reasoning-based LLMs. Given a partial reasoning trace, Re-FORC predicts the distribution of expected rewards from generating additional thinking tokens. This capability is achieved through a lightweight adapter fine-tuned on top of existing models, enabling predictions for both the base model and external black-box systems.

Building on Re-FORC, we then introduce a **greedy algorithm for optimal decision-making** inspired by the theory of Pandora's box problems [2]. This development is motivated by the following practical application requirements:

- **Adaptive early termination:** By identifying unpromising reasoning trajectories early, Re-FORC reduces computational cost by 26% while maintaining accuracy by adjusting to query difficulty.
- **Joint model and compute optimization:** When multiple model sizes are available, Re-FORC jointly selects both the optimal model and the number of reasoning tokens to maximize net utility for each query, going beyond traditional model routing by explicitly considering compute budgets.
- **User-controlled inference:** Users can specify their cost-performance trade-off λ at inference time without model retraining, providing a more intuitive alternative to token-count specifications while adapting inference to query complexity.
- **Transparent compute estimation:** Re-FORC can provide users with upfront estimates of expected computation time, improving the user experience for latency-sensitive applications.

2 Related Work

Our work intersects two key research areas: forecasting methods that predict LLM performance, and efficient reasoning strategies that optimize compute-accuracy trade-offs during inference.

Forecasting, Probing, and Verifier-Guided Inference-Time Scaling. Most forecasting research addresses non-chain-of-thought contexts, predicting restart benefits [3] or initial correctness [4]. Recent work shows models encode future correctness [5, 6] and factuality signals [7, 8] during reasoning. While some approaches use external verifiers for intermediate step evaluation [9, 10, 11], our method forecasts future expected reward as a continuous function of additional reasoning tokens, enabling utility-based decisions across reasoning horizons. Concurrent work on adaptive test-time compute allocation shows that compute-optimal scaling policies [12] and reward-guided adaptive reasoning depth [13] can outperform naive scaling approaches. Tree-of-thought methods enable deliberate multi-branch search over reasoning states [14, 15] and adaptive budget allocation across multiple trajectories [16], though recent work identifies efficiency challenges in naive verifier-guided exploration [17]. Recent iterative refinement methods repeatedly generate and aggregate reasoning traces, showing near-monotonic accuracy gains but rapidly growing compute cost [18]. Our forecaster differs in that it predicts a continuous marginal value curve of extra thinking tokens, rather than heuristically expanding and pruning reasoning paths.

Efficient Reasoning and Model Selection. A parallel line of work has developed strategies for efficient reasoning under compute constraints. Self-consistency [19] spawned compute-aware variants that halt when votes converge [20, 21] or use confidence-weighted aggregation [22]. Theoretical analyses characterize diminishing marginal returns from additional samples [23]. Recent work on adaptive reasoning length control includes token-budget-aware policies [24] and learned stopping mechanisms that predict when further reasoning becomes redundant [25], including dynamic early-exit policies that truncate reasoning once additional steps yield diminishing returns [26]. Structured exploration increases search breadth under larger budgets [27]. At the token level, early exit mechanisms enable anytime generation with calibrated stopping [28], while agentic frameworks learn when to plan [29, 30] and strategic decomposition approaches like ReAct [31], least-to-most prompting [32], and plan-and-solve strategies [33] allocate deliberate planning compute. Separately, model routing systems choose among different-sized models to optimize accuracy-cost trade-offs [34, 35, 36, 37, 38].

no use. But if the agent is a botanist tasked with categorizing species for an encyclopedia project, it matters little whether the fruit is edible by the time the correct answer is provided. In general, an agent needs to learn to modulate the cost of time depending on the characteristics of the environment and the stated preference of the user.

In addition to model selection, recent works aim to control the reasoning length either through training-free [39] or training-based approaches [40]. Our approach bridges these areas by using forecasted reward curves to make principled decisions about both when to stop reasoning and which model to use, grounded in metareasoning theory [41, 2, 42]. Recent work frames modern LLM inference as bounded-optimal decision-making over cognitive effort [43], providing theoretical grounding for treating compute allocation as an economically principled optimization problem.

It has been argued that accounting for the cost of time during training is key to developing transductive inference capabilities, *i.e.,* for *learning to reason.* In order to do so during training, a reward prediction model is a necessary component. Therefore, this work can be viewed as enabling the vision set forth in [44], which shows that without accounting for the cost of time, optimal inference can be achieved *brute-force* with no insight, even without learning. On the other hand, an explicit and tight relation is established between the decrease in inference time and the algorithmic mutual information in the trained model. Algorithmic mutual information is well defined but not computable, and can be considered a measure of transductive inference (*i.e.*, reasoning) skills. Inference time, on the other hand, is computable and, as we show in this paper, predictable on-line during inference computation and can be used for training [45].

3 Methodology

Inference-time compute allocation in reasoning models presents a sequential decision problem where an agent must decide at each step whether to generate additional thinking tokens or terminate. Let $\mathcal{X}$ denote the query space, $\mathcal{Z}$ the space of partial reasoning traces (possibly multiple trajectories), and $\mathcal{Y}$ the output space, and $\Pi = \{\pi_i\}_{i=1}^{N}$ be a collection of reasoning models. At each time step, the system state is (x, z) where $x \in \mathcal{X}$ is the query, $z \in \mathcal{Z}$ is the collection of partial traces generated so far, and the agent needs to decide which trajectory to continue or terminate the search.

This problem exhibits the structure of a *Markov chain selection* problem [46], where the agent must choose which of multiple stochastic processes (reasoning trajectories for LLMs) to advance. Each reasoning continuation corresponds to a transient Markov chain with reward structure, and the agent must select which chain to progress based on expected net utility. However, unlike classical settings where reward distributions are known, reasoning models operate with unknown, state-dependent reward distributions that depend on query complexity and current reasoning progress. Recently, [1] showed that universal search [47] can be formulated as a Pandora's box problem which can be solved with the Gittins policy. We use the framework from [1] in section 4.2 where we propose the Pandora's box greedy search for reasoning models using our forecaster $\psi(t \mid x, z, \pi)$.

The optimal policy for such problems [48] takes the form of a Gittins index policy [46, 49], which assigns each possible continuation a reservation value—the minimum expected reward improvement needed to justify its computational cost. In the Gittins index policy[46], we compute the Gittins index of each continuation, and compare it against our estimate of current best reward. We terminate search if none of the continuations improve the current best reward, otherwise we choose the trajectory with highest Gittins index.

The central challenge is that computing Gittins indices requires knowledge of the reward distributions for different reasoning continuations, which are unknown and must be learned. We address this by introducing **Re-FORC**, which learns to predict the forecasting functional $\psi(t \mid x, z, \pi)$ that estimates expected rewards from generating t additional thinking tokens from state (x, z) using model π. Using our forecaster we can approximate the Gittins index to choose the next action in the Markov chain.

We now introduce the necessary preliminaries and define the forecasting functional that enables training our predictor to approximate the Gittins index.

3.1 Sequential Compute Allocation

We formalize the inference-time compute allocation problem in LLMs by defining the decision space, objectives, and constraints. The core challenge is modeling how reasoning models generate thinking tokens and how the quality of their final answers depends on the computational resources allocated.

We use a Markov decision process where the state space is $\mathcal{S} = \mathcal{X} \times \mathcal{Z}$ and each state $s = (x, z)$ represents a query with a partial reasoning trace. The action space $\mathcal{A}$ includes:

- **Continue reasoning**: Generate Δ additional thinking tokens
- **Terminate**: Stop reasoning and output final answer y
- **Switch model**: Transfer to a different reasoning model π' when available

Each reasoning continuation from state (x, z) can be modeled as advancing a transient Markov chain with reward function $R : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ and cost function $c : \mathcal{A} \to \mathbb{R}_+$. The agent's objective is to maximize expected net utility:

$$\boxed{J = \mathbb{E}[R^*] - \lambda \cdot T_{\text{total}}} \tag{1}$$

where R^* is the reward of the best answer discovered, T_{total} is the total computational cost, and $\lambda > 0$ represents the cost sensitivity parameter. This objective balances exploration of potentially better solutions against computational expenditure measured in (necessarily subjective, environment- and user-dependent) units of λ.

The key challenge distinguishing our setting from classical Markov chain selection problems lies in computing the Gittins index itself. For a Markov chain in state s, the Gittins index g for a reasoning trajectory is defined as the solution to the following equation:

$$\boxed{\mathbb{E}\Big[(R(x, y) - g)_+ \Big| s = (x, z)\Big] - \lambda t = 0} \tag{2}$$

where t is the number of future reasoning tokens, and $(x)_+$ denotes $\max(x, 0)$. Let $\{z_i\}_{i=1}^n$ be n different reasoning trajectories our agent is exploring, where the current best reward is R^*. Using the Gittins index policy we compute the Gittins index, g_i for each trajectory z_i, and terminate search if $g_i < R^*$, otherwise choose the trajectory $\operatorname{argmax}_i g_i$.

We notice that computing the Gittins index requires calculating the expected reward $\mathbb{E}[R(x, y)]$ (we skip the conditional term for ease of notation) when generating t additional thinking tokens from state (x, z). This expectation depends on the stochastic reasoning process and the final answer quality, neither of which have closed-form expressions for language models. We address this by learning the forecasting functional $\psi(t \mid x, z, \pi) = \mathbb{E}[R(x, y)]$ for t-token continuations. This functional enables approximate computation of Gittins indices and principled decision-making in our adaptive setting.

3.2 Adaptive Reward Prediction

Given a query $x \in \mathcal{X}$, a partial chain-of-thought $z \in \mathcal{Z}$, and a reasoning model π, we define two modes of inference: $\pi^{(r)}$ for thinking token generation and $\pi^{(o)}$ for final output generation. The output given t additional thinking tokens is obtained by first sampling additional reasoning tokens $z_t \sim \pi^{(r)}(\cdot|x, z, t)$ where $|z_t| \leq t$, then sampling the output $y \sim \pi^{(o)}(\cdot|x, z, z_t)$.

Given a reward function $R(x, y) : \mathcal{X} \times \mathcal{Y} \to [0, 1]$, the adaptive forecasting functional is:

$$\psi(t \mid x, z, \pi) \triangleq \mathbb{E}_{z_t \sim \pi^{(r)}(\cdot|x,z,t), y \sim \pi^{(o)}(\cdot|x,z,z_t)}[R(x, y)] \tag{3}$$

This functional represents the expected reward after allocating exactly t additional thinking tokens, starting from the current reasoning state (x, z). Critically, ψ is query-dependent, path-dependent, and accounts for the stochastic nature of both reasoning generation and final answer sampling.

To predict $\psi(t \mid x, z, \pi)$, we design a lightweight forecasting module that can be attached to existing reasoning models. We model the forecaster output using a $\mathrm{Beta}(\alpha, \beta)$ distribution, which provides natural bounded support matching the $[0, 1]$ reward constraint while capturing both predicted mean reward and confidence through its variance. The Beta distribution also offers favorable mathematical properties for sequential updating and calibration.

The forecaster predicts Beta parameters $[\alpha_\theta(x, z, t), \beta_\theta(x, z, t)]^T$ for each thinking token budget t. At inference, we use the Beta mean as our point estimate:

$$\hat{\psi}(t \mid x, z, \pi) = \frac{\alpha_\theta(x, z, t)}{\alpha_\theta(x, z, t) + \beta_\theta(x, z, t)} \tag{4}$$

Since ψ is defined over discrete token budgets $t \in \mathbb{N}$, we predict it on a uniform grid $\mathcal{T} = \{0, \Delta, 2\Delta, \ldots, t_{\max}\}$ and obtain values at arbitrary t by linear interpolation between adjacent grid points. This approach balances computational efficiency with forecasting accuracy.

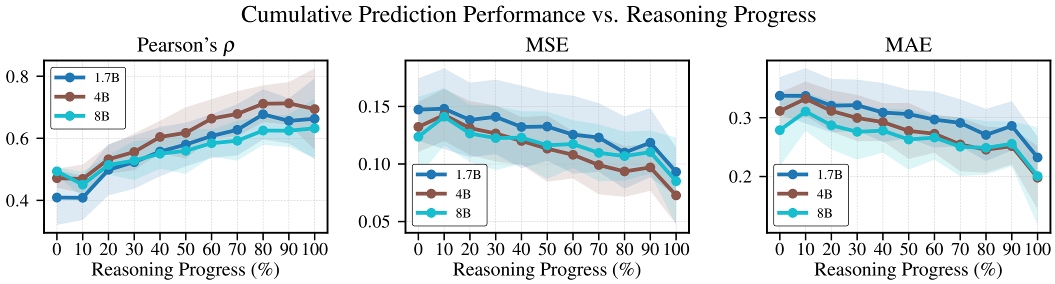

Figure 1: **Forecast performance with reasoning progress.** Correlation between Re-FORC (in eq. (4)) and the true reward (in eq. (3)) as CoT progresses for Qwen3 [50] models, averaged across five evaluation datasets (see section 5.2). **(left)** Pearson correlation (ρ) (higher is better), **(middle)** mean squared error (MSE) (lower is better), and **(right)** mean absolute error (MAE) (lower is better). Performance steadily improves with reasoning length, and larger models (e.g. 8B) achieve higher ρ and lower error throughout.

3.3 Training the predictor

Training the adaptive reward forecaster requires generating a dataset of (x, z, t, r) tuples where r represents the true expected reward $\psi(t \mid x, z, \pi)$ from continuing reasoning for t additional tokens from state (x, z). We generate training data by sampling problem instances (x_i, y_i) and generating full unconstrained reasoning trajectories up to the maximum context length. From each complete trajectory, we extract partial traces z by truncating at regular token intervals corresponding to our forecasting grid $\mathcal{T} = \{0, \Delta, 2\Delta, \ldots, t_{\max}\}$. For each partial trace z truncated at position ℓ, we sample the model's answer directly from state (x, z).

When constructing the adaptive forecasting functional $\psi(t \mid x, z, \pi)$ for different continuation lengths t, we reuse these sampled answers by taking all trajectory segments that extend exactly t tokens beyond the truncation point ℓ. This provides an efficient Monte Carlo approximation through trajectory reuse compared to the alternative of generating multiple continuations for every partial trajectory and every forecasting horizon would require $O(|\mathcal{T}| \times N \times L)$ trajectory samples, where N is the number of Monte Carlo samples per estimate and L is the maximum trajectory length. Our reuse strategy reduces this to $O(N)$ samples total while maintaining unbiased estimates of $\psi(t \mid x, z, \pi)$ across all forecasting horizons.

The forecaster is trained by maximizing the likelihood of observed rewards under the predicted Beta distributions:

$$\mathcal{L}_{\text{forecast}} = \mathbb{E}_{(x,z,t,r)\sim D_{\text{forecast}}}\big[-\log \text{Beta}(\alpha_\theta(x, z, t), \beta_\theta(x, z, t))(r)\big] \tag{5}$$

We provide more training details in section 5. Note that our forecasters are lightweight adapters attached to frozen base reasoning models.

4 Compute-Aware Inference

In this section, we explore the applications of our forecaster $\psi(t \mid x, z, \pi)$ for optimal inference-time decision-making. We apply the Gittins index-inspired greedy algorithm [46, 49] to evaluate the expected improvement from continuing each available reasoning trace and select the action with highest expected net utility at each step in the search. This approach enables four key applications: (1) early termination decisions that halt partially-completed reasoning traces when marginal improvement no longer justifies compute costs, (2) initial model selection that chooses the optimal model for query processing based on expected cost-accuracy trade-offs, (3) dynamic reselection that transfers unsuccessful queries to models with potentially better capabilities, and (4) exploration-exploitation trade-offs that balance sampling new reasoning traces against continuing existing ones.

4.1 Early stopping

Let z^* be the best (partial) thinking trace in our agent's inference search, with expected reward $\bar{R} = \mathbb{E}_{y\sim\pi^{(o)}(\cdot|x,z,z_t), z_0\sim\pi^{(r)}(\cdot|x,z^*,t=0)}\big[R(x, y)\big]$. At the start of the inference search the reward

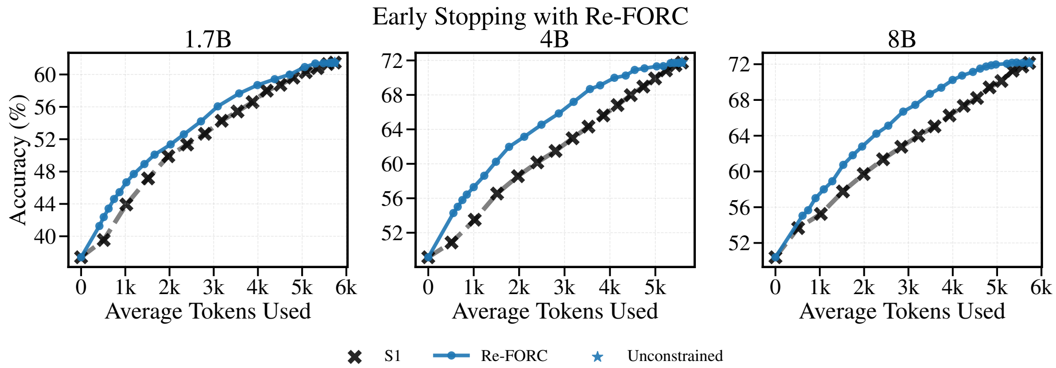

Figure 2: **Early Stopping with Re-FORC.** We plot the accuracy vs tokens trade-off for various Qwen-3 models: 1.7B **(left)**, 4B **(middle)**, and 8B **(right)**; averaged across five reasoning datasets (see section 5.2). We show that Re-FORC improves the trade-off frontier over baseline inference-time scaling methods like S1[39] across all model sizes and benchmarks. Results for individual datasets are in Appendix Fig. 12.

$\bar{R} = 0$. Let z be the current thinking trace of our agent (potentially $z = z^*$), then stopping here gives an expected net reward of: $J = \mathbb{E}[R^*] - \lambda T$.

If the agent decides to extend z for additional t steps, the expected improvement in reward will be:

$$\boxed{\Delta_t J \;=\; \mathbb{E}_{y\sim\pi^{(o)}(\cdot|x,z,z_t),z_t\sim\pi^{(r)}(\cdot|x,z,t)}\left[(R(x,y) - R^*)_+\right] - \lambda t.} \tag{6}$$

where $(\cdot)^+$ denotes the positive part, which we need since even if the reward ends up being lower than the current best R^* we can simply drop that attempt, but we still have to pay for the additional compute λt. Note that eq. (6) resembles the Gittins index definition from eq. (2) which is used to determine early stopping.

For a binary reward distribution, the expected improvement simplifies to the following expression:

$$\Delta_t J = \psi(t \mid x, z, \pi)(1 - \bar{R}) - \lambda t \tag{7}$$

We want to expand the current thinking trace for t steps only if the expected net reward is positive $\Delta_t J > 0$, i.e., only if there exists a t such that:

$$\boxed{\psi(t \mid x, z, \pi) \geq \frac{\lambda t}{1 - \bar{R}}} \tag{8}$$

We call this method Re-FORC-stopping. Note that if the cost-per-token λ is high, we are more likely to decide to stop the reasoning early. We also stop early if the current best expected reward $\bar{R}$ is close to 1. In practice we approximate ψ with our reward forecaster eq. (4) trained with eq. (5).

In fig. 2 we show that early stopping with Re-FORC significantly improves the reward compute trade-off across different sizes of reasoning models averaged across 5 math datasets (section 5.2). For instance, we can save 26% compute for the 4B reasoning model with Re-FORC stopping while obtaining maximum accuracy.

4.2 Pandora's Box Greedy Search

We can extend the approach of the previous section to make more general decisions [1]. Suppose we have a collection $\{\pi_1, ..., \pi_k\}$ of reasoning models, each with associated cost λ_i per token, and a set $\{z_1, ..., z_n\}$ of partial thinking traces. As before, let $\bar{R}$ be the best expected reward obtained so far in the search. Given a partial reasoning trace z_j, define the expected improvement:

$$\boxed{G(z_j|R^*, \pi_i) = \max_t \psi(t \mid x, z, \pi_i)(1 - \bar{R}) - \lambda t} \tag{9}$$

as the best improvement in net reward we can obtain if we extend z_j by t steps using model π_i. At each point in the search, we pick the combination of trace z_j and model π_i that has the best expected

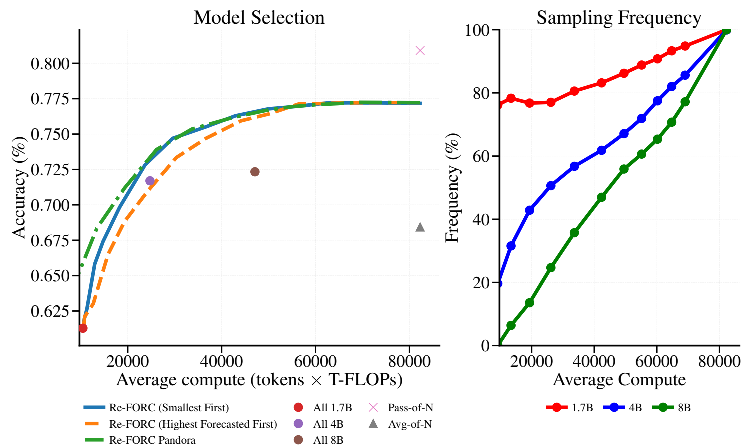

Figure 3: **Model and thinking length selection with Re-FORC. (Left)** We plot accuracy-compute trade-off for Qwen-3 models averaged across 5 reasoning benchmarks. We use the method proposed in eq. (10) and eq. (11) to perform routing with model selection, and compare against baselines. We show that the accuracy-compute frontier obtained by Re-FORC outperforms the individual models, including Avg-of-N baseline. **(Right)** We plot the sampling frequency of Re-FORC Pandora for different model sizes as a function of compute. The sampling frequency corresponds to the percentage questions Re-FORC chooses to sample a given model. At maximum compute budgets, Re-FORC samples from all the models (from small to large), while at minimum compute budget Re-FORC preferentially samples from the smallest 1.7B model. We show that the model routes a majority of the queries to the smaller model (especially in the low compute region), and starts routing more to the larger models only in high compute range. Results for individual datasets are in Appendix Fig. 13.

improvement $G(z_j|\bar{R}, \pi_i)$ and continue it for a fixed number of tokens. We add the new thinking trace to the set of currently explored trajectories and update the current best expected reward $\bar{R}$ if needed. We terminate the search when there are no combinations with positive expected improvement. In practice, we use $\hat{\psi}(t|x, z, \pi)$ to approximate the forecasting functional in eq. (9).

This algorithm has several important special cases which we discuss in the next subsections. However, if we have a single model and a single trace z under consideration, the search trivially reduces to the *early stopping* rule considered earlier in eq. (8).

4.3 Model selection

At the start of greedy search, $\bar{R} = 0$ and z is empty, so the first step of search reduces to selecting the best model π_i to route the query x to in order to maximize the net reward:

$$\boxed{G(z = \emptyset \mid \bar{R} = 0, \pi_i) = \max_t \psi(t \mid x, z, \pi_i) - \lambda_i t} \tag{10}$$

i.e. route to model π_j where $j = \text{argmax}_i G(z = \emptyset \mid \bar{R} = 0, \pi_i)$. This strategy is especially useful when the agent only has black-box access to reasoning models as it does not involve moving between partial reasoning trajectories during inference, instead we select a model, and perform inference to completion.

Dynamic re-selection. Alternatively, we can route the query to a cheap model π_{small} first, and obtain expected reward $\bar{R} = R_{\text{small}}$. Now we want to decide whether we want to attempt improving the current result by trying a larger model π_{large}. The decision is non-trivial, since it doesn't just matter what is the expected reward of the larger model, but the *improvement* over the reward we just obtained. Following the strategy, we decide to try the new model only if:

$$\boxed{G(z = \emptyset \mid \bar{R} = 0, \pi_{\text{large}}) = \max_t \psi(t \mid x, z, \pi_{\text{large}})(1 - R_{\text{small}}) - \lambda_{\text{large}} t > 0} \tag{11}$$

In fig. 3 we show the benefits of using Re-FORC for model selection, for instance, with equal compute our method obtains 4% higher accuracy on average over 5 datasets (see section 5.2), and 50% less compute at equal accuracy with respect to the largest model.

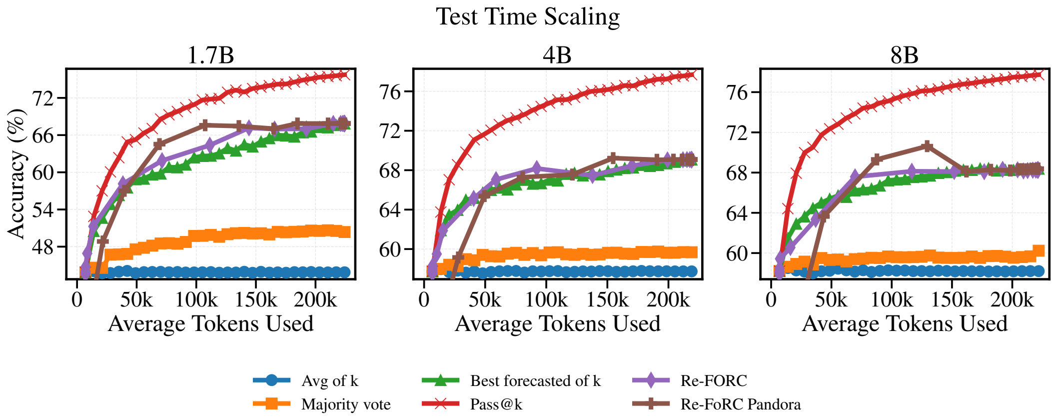

Figure 4: **Test-time scaling using Re-FORC.** We plot the accuracy-token trade off for Pandora's box greedy search based Re-FORC algorithms for Qwen-3 reasoning models averaged across 3 reasoning benchmarks: AIME 24/25 and AMC24. We also compare our method against repeated-sampling test-time scaling baselines Avg-of-k and majority-vote. Re-FORC based scaling outperforms the baselines only to be outperformed by Pass-at-k which requires oracle access to the true solution. Our Re-FORC Pandora method in eq. (12) outperforms the model-selection-based procedure in eq. (10) (purple) in low compute regime, which depicts the benefits of switching between multiple trajectories during inference. Note that we sample $32\times$ for each query during inference for all the methods. Results for individual datasets are in Appendix Fig. 14.

4.4 Test-Time Scaling

Finally, we provide an inference-time scaling rule to explore the full potential of the Pandora's box greedy search strategy through test-time scaling. Given a query x, a full reasoning trace z_i (sampled using eq. (8)) with its expected reward R_i, we decide whether to draw another fresh sample for test-time scaling, if and only if

$$\boxed{G(z = \emptyset \mid \bar{R} = R_i, \pi_{\mathrm{i}}) = \max_t \hat{\psi}(t \mid x, \emptyset, \pi_i)(1 - R_i) - \lambda_i t \; > \; 0} \tag{12}$$

In fig. 4, we show that Re-FORC based scaling improves accuracy by 7% in low compute regime, and 11% in high compute regime, especially using Pandora's box greedy search.

5 Experiment details

5.1 Training Setup

We implement our adaptive reward forecaster as a lightweight adapter attached to pretrained reasoning models from the Qwen3 family (1.7B, 4B, 8B parameters) [50]. The base reasoning models remain frozen during forecaster training to preserve their reasoning capabilities while learning to predict future performance. The forecaster architecture consists of a self-attention pooling layer that takes penultimate-layer activations $h_{1:n} \in \mathbb{R}^{n\times d}$ and aggregates sequence information into a fixed-size representation, followed by a linear projection head $g_\theta : \mathbb{R}^d \to \mathbb{R}^{2|\mathcal{T}|}$ that outputs Beta distribution parameters (α_t, β_t) for each time horizon $t \in \mathcal{T}$.

We use a uniform forecasting grid $\mathcal{T} = \{0, 512, 1024, \ldots, 4096\}$ with linear interpolation for intermediate values, where Beta parameters are obtained via softplus activation to ensure positivity. The forecaster introduces minimal computational overhead, requiring only a single forward pass through the base model during training to extract activations, amortizing forecasts over all horizons without additional thinking tokens during inference.

Training data is generated by sampling problem instances (x_i, y_i) from DeepScaleR-Preview [51] and creating full unconstrained reasoning trajectories up to maximum context length. We extract partial traces at regular intervals and use Monte Carlo estimation with $N = 8$ samples to compute empirical success rates, with rewards clipped to $(\varepsilon, 1 - \varepsilon)$ where $\varepsilon = 10^{-6}$ for numerical stability.

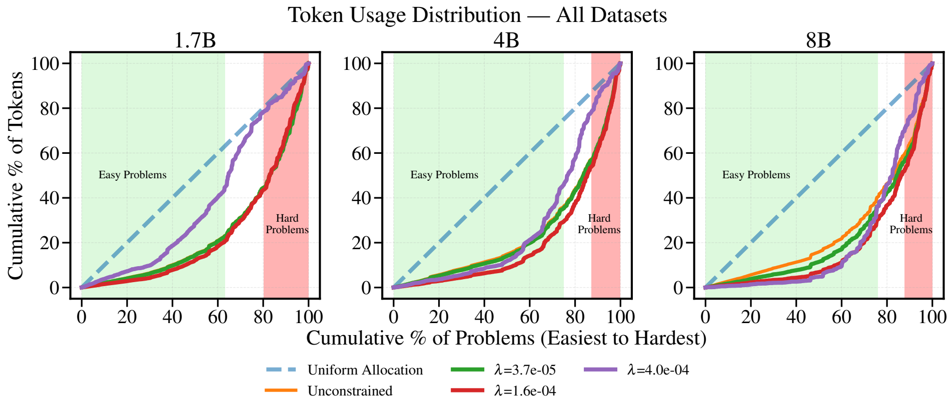

Figure 5: **Token distribution and problem difficulty for Qwen3 models averaged across datasets.** Problems are ordered by per-model solve rate across datasets (Minerva, MATH500, AMC2024, AIME2024, AIME2025); "easy" problems are solved in $\geq$90% of trials and "hard" ones in $<$50%. Each curve shows the cumulative share of total tokens spent versus cumulative problem difficulty, with the dashed diagonal indicating uniform allocation. Increasing λ represents a higher cost of time, encouraging more selective compute use. At high cost of time (λ=4.0×10^{-4}, purple), the small 1.7B model allocates proportionally more tokens to easier problems while largely avoiding the hardest ones—an economically efficient strategy given its limited capability. In contrast, larger models (4B, 8B) can still benefit from deeper reasoning on difficult problems and thus concentrate their compute on the hardest $\sim$20% of tasks while conserving effort on easy cases.

5.2 Evaluation Setup

We evaluate on five mathematics reasoning datasets: AMC 2024 [52, 53, 54, 55], Minerva Math [56], Math500 [57], and AIME 2024/25 [58, 59, 60, 61]. Forecasting performance is measured using Pearson correlation (ρ), mean squared error (MSE), and mean absolute error (MAE) between predicted and true reward values, while compute-aware inference is evaluated on accuracy-compute trade-offs measuring both final accuracy and total thinking tokens consumed.

Our experimental comparisons include unconstrained generation as a standard reasoning baseline without early stopping, fixed token limits representing simple cutoffs without adaptive decision-making, single-model baselines using only the largest or smallest available model, and oracle routing with ground-truth access (Pass@k) as a theoretical upper bound. These baselines allow us to isolate the contributions of adaptive forecasting versus simpler heuristic approaches.

6 Results

User-controlled inference: Users can dynamically control computational expenditure by selecting an appropriate λ value at inference time based on their accuracy and cost requirements. Re-FORC then automatically optimizes the reward-compute trade-off, as demonstrated in fig. 2, fig. 3, and fig. 4. The method proves especially advantageous in intermediate compute regimes, where users desire meaningful quality improvements without prohibitive costs—consistently achieving superior accuracy compared to baselines at equivalent compute budgets. For instance, while using early stopping (in fig. 2), Re-FORC provides maximum accuracy improvements in the 2k-4k token range, similarly, while test-time scaling (in fig. 4), Re-FORC with 8B model provides maximum improvements in the 100k tokens range ($100k \approx 32 \times 3k$, since we sample 32$\times$ per query). Furthermore, in resource-constrained settings, our early stopping criterion provides substantial efficiency gains, reducing computational overhead by approximately 25% while preserving accuracy.

Compute-aware applications: Our experiments demonstrate the effectiveness of Re-FORC across three key applications. First, Re-FORC-stopping (Eq. 8) provides smooth accuracy-compute frontiers, where moderate λ values preserve most peak accuracy while substantially reducing reasoning tokens. Second, Re-FORC-selection (Eq. 10) consistently outperforms all baselines across all compute budgets, achieving the same accuracy as the largest model with 55% less compute and 4% higher accuracy at the same compute level. Finally, Re-FORC-scaling (Eq. 12) achieves superior accuracy-compute trade-offs, improving accuracy by 11% at high compute budgets and 7% at low compute budgets while approaching oracle-level Pass@k performance and enabling practical per-token cost control.

Flexible base model training: Re-FORC maintains complete independence from the base reasoning model's training procedure, enabling practitioners to optimize the underlying model using any algorithm suited to their application. This stands in contrast to methods such as L1 [40] that require modifying input prompts during training, which can compromise the model's reasoning performance and accuracy. By operating solely at inference time, our approach traces the entire accuracy-compute trade-off curve while providing fine-grained cost control—all without altering the base model's training or architecture.

Difficulty-based allocation. For each model, we order problems by empirical solve rate (higher = easier) and plot the cumulative share of thinking tokens as a function of cumulative problem difficulty. Across methods, compute generally shifts from easy to hard problems. The key effect appears at higher λ (higher cost of time) and differs by model size: for the 1.7B model, high λ rationally concentrates effort on easier instances and largely avoids the hardest tail; for larger models (4B, 8B), the same high λ drives strong triage—very little compute on easy problems and substantially more on the hardest $\sim$20%—yielding a markedly more convex allocation curve. (We define "easy" as solve rate $\geq$90% and "hard" as $<$50% for each model.)

Forecast accuracy improves with reasoning progress: Forecast quality improves as chain-of-thought tokens increase, with higher ρ and lower MSE/MAE (see fig. 1). This effect is amplified in larger models, with the 8B model outperforming smaller variants across all prefixes. For this work, we only use the mean of the Beta distribution; however, incorporating the variance will improve forecaster calibration, and overall results, which we leave as potential future work.

7 Limitations

Data collection overhead: Collecting training data for the forecaster presents a significant computational challenge, especially at larger model scales. The process requires three computationally intensive steps: (1) partitioning reasoning tokens into fixed intervals of 512 tokens (0, 512, 1024, etc.), (2) sampling multiple reasoning trajectories for each length interval, and (3) generating multiple outputs conditioned on each reasoning trajectory. Although efficient parallelization can mitigate costs, the overall expense scales with both dataset size and model capacity, requiring high compute before training the forecaster.

Forecaster overconfidence: The forecaster occasionally exhibits overconfidence, predicting higher rewards than the base reasoning model can realistically achieve. This miscalibration can lead to wasteful computation, where the system continues sampling additional reasoning trajectories rather than terminating early when further improvement is unlikely. We observe that extended training with larger datasets helps mitigate this issue, though complete calibration remains an ongoing challenge.

8 Conclusion

We introduced Re-FORC, an adaptive reward prediction approach that enables efficient control of compute (both model size and reasoning length) over chain-of-thought reasoning by thresholding the forecasting functional using the Gittins index policy. We formulate the reward-compute prediction problem using Pandora's box greedy search [48, 46] and provide empirical techniques to approximate the Gittins index policy for reasoning models [1] in practice. Our method trains lightweight forecasters (adapters) on top of frozen reasoning models to predict future reward-token trade-off (reasoning trajectory outcomes). Our forecaster enables three key inference-time applications: (1) early stopping of unpromising reasoning trajectories, (2) compute-aware model selection from a pool of reasoning models, and (3) cost-aware test-time scaling. Results across five mathematics datasets demonstrate that forecaster-guided strategies consistently outperform baseline approaches, achieving superior accuracy-compute trade-offs.

References

[1] Alessandro Achille and Stefano Soatto. Ai agents as universal task solvers. *arXiv preprint arXiv:2510.12066*, 2025.

[2] Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979.

[3] Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation, 2024.

[4] Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of lm computation, 2024.

[5] Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification, 2025.

[6] Dongkeun Yoon, Seungone Kim, Sohee Yang, Sunkyoung Kim, Soyeon Kim, Yongil Kim, Eunbi Choi, Yireun Kim, and Minjoon Seo. Reasoning models better express their confidence. *arXiv preprint arXiv:2505.14489*, 2025.

[7] Giovanni Servedio, Alessandro De Bellis, Dario Di Palma, Vito Walter Anelli, and Tommaso Di Noia. Are the hidden states hiding something? testing the limits of factuality-encoding capabilities in llms, 2025.

[8] Yuqing Yang and Robin Jia. When do llms admit their mistakes? understanding the role of model belief in retraction, 2025.

[9] Theo Uscidda, Matthew Trager, Michael Kleinman, Aditya Chattopadhyay, Wei Xia, and Stefano Soatto. Latts: Locally adaptive test-time scaling. *arXiv preprint arXiv:2509.20368*, 2025.

[10] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[11] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

[12] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[13] Yingqian Cui, Zhenwei Dai, Pengfei He, Bing He, Hui Liu, Xianfeng Tang, Jingying Zeng, Suhang Wang, Yue Xing, Jiliang Tang, and Benoit Dumoulin. Adaptive test-time reasoning via reward-guided dual-phase search, 2025.

[14] Jieyi Long. Large language model guided tree-of-thought, 2023.

[15] Maciej Besta, Florim Memedi, Zhenyu Zhang, Robert Gerstenberger, Guangyuan Piao, Nils Blach, Piotr Nyczyk, Marcin Copik, Grzegorz Kwaśniewski, Jurgen Müller, Lukas Gianinazzi, Ales Kubicek, Hubert Niewiadomski, Aidan O'Mahony, Onur Mutlu, and Torsten Hoefler. Demystifying chains, trees, and graphs of thoughts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–20, 2025.

[16] Baohao Liao, Hanze Dong, Yuhui Xu, Doyen Sahoo, Christof Monz, Junnan Li, and Caiming Xiong. Fractured chain-of-thought reasoning, 2025.

[17] Ante Wang, Linfeng Song, Ye Tian, Dian Yu, Haitao Mi, Xiangyu Duan, Zhaopeng Tu, Jinsong Su, and Dong Yu. Don't get lost in the trees: Streamlining llm reasoning by overcoming tree search exploration pitfalls, 2025.

[18] Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R. Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, Nikolay Malkin, and Moksh Jain. Recursive self-aggregation unlocks deep thinking in large language models, 2025.

[19] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

[20] Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning, 2024.

[21] Xin Liu and Lu Wang. Answer convergence as a signal for early stopping in reasoning, 2025.

[22] Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, page 20090–20111. Association for Computational Linguistics, 2025.

[23] Junpei Komiyama, Daisuke Oba, and Masafumi Oyamada. Best-of-∞ – asymptotic performance of test-time compute, 2025.

[24] Anonymous. Reasoning at the right length: Adaptive budget forcing for efficient and accurate LLM inference. In *Submitted to The Fourteenth International Conference on Learning Representations*, 2025. under review.

[25] Renliang Sun, Wei Cheng, Dawei Li, Haifeng Chen, and Wei Wang. Stop when enough: Adaptive early-stopping for chain-of-thought reasoning, 2025.

[26] Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic early exit in reasoning models, 2025.

[27] Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*, 2024.

[28] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling, 2022.

[29] Davide Paglieri, Bartłomiej Cupiał, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning when to plan: Efficiently allocating test-time compute for llm agents, 2025.

[30] Bo Pan and Liang Zhao. Can past experience accelerate llm reasoning?, 2025.

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[32] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models, 2023.

[33] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, 2023.

[34] Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient llm inference, 2025.

[35] Neel Guha, Mayee F. Chen, Trevor Chow, Ishan S. Khare, and Christopher Ré. Smoothie: Label free language model routing, 2024.

[36] Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. Best-route: Adaptive llm routing with test-time optimal compute, 2025.

[37] Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. Masrouter: Learning to route llms for multi-agent systems, 2025.

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

[39] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[40] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

[41] Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1):361–395, 1991.

[42] Ali Aouad, Jingwei Ji, and Yaron Shaposhnik. The pandora's box problem with sequential inspections, 2025.

[43] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025.

[44] Alessandro Achille and Stefano Soatto. Ai agents as universal task solvers, 2025.

[45] Michael Kleinman, Matthew Trager, Alessandro Achille, Wei Xia, and Stefano Soatto. e1: Learning to control reasoning effort. *To Appear*, 2025.

[46] Ziv Scully and Alexander Terenin. The gittins index: A design principle for decision-making under uncertainty. *arXiv preprint arXiv:2506.10872*, 2025.

[47] Leonid A Levin. Universal sequential search problems. *Problems of information transmission*, 9(3):265–266, 1973.

[48] Martin Weitzman. *Optimal search for the best alternative*, volume 78. Department of Energy, 1978.

[49] Qian Xie, Raul Astudillo, Peter Frazier, Ziv Scully, and Alexander Terenin. Cost-aware bayesian optimization via the pandora's box gittins index. *Advances in Neural Information Processing Systems*, 37:115523–115562, 2024.

[50] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[51] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. `https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca3030`, 2025. Notion Blog.

[52] Art of Problem Solving. 2024 amc 12b — problems and solutions. `https://artofproblemsolving.com/wiki/index.php/2024_AMC_12B`, 2024. Administered November 12, 2024.

[53] Art of Problem Solving. 2024 amc 12a — problems and solutions. `https://artofproblemsolving.com/wiki/index.php/2024_AMC_12A`, 2024. Administered November 6, 2024.

[54] Art of Problem Solving. 2024 amc 10b — problems and solutions. `https://artofproblemsolving.com/wiki/index.php/2024_AMC_10B`, 2024. Administered November 12, 2024.

[55] Art of Problem Solving. 2024 amc 10a — problems and solutions. https://artofproblemsolving.com/wiki/index.php/2024_AMC_10A, 2024. Administered November 6, 2024.

[56] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[57] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

[58] Art of Problem Solving. 2024 aime i — problems and solutions. https://artofproblemsolving.com/wiki/index.php/2024_AIME_I, 2024. Held January 31–February 1, 2024.

[59] Art of Problem Solving. 2024 aime ii — problems and solutions. https://artofproblemsolving.com/wiki/index.php/2024_AIME_II, 2024. Held February 7, 2024.

[60] Art of Problem Solving. 2025 aime i — problems and solutions. https://artofproblemsolving.com/wiki/index.php/2025_AIME_I, 2025. Held February 6, 2025.

[61] Art of Problem Solving. 2025 aime ii — problems and solutions. https://artofproblemsolving.com/wiki/index.php/2025_AIME_II, 2025. Held February 12, 2025.

Appendix

A Additional Experiments

A.1 Problem Difficulty

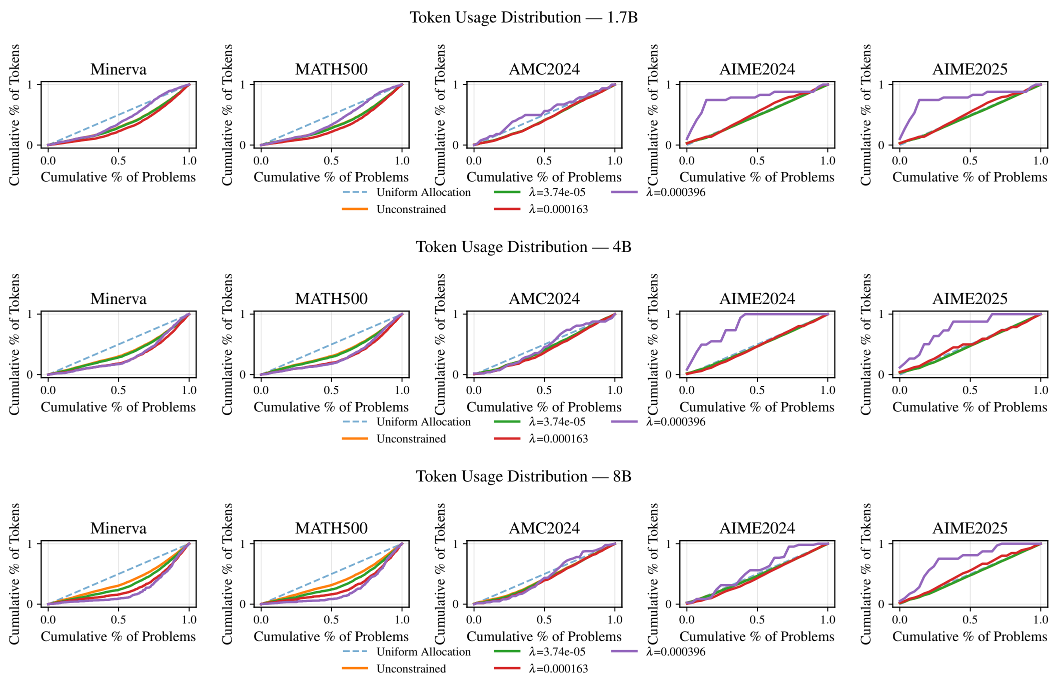

Figure 6: **Token distribution and problem difficulty per dataset.** We order problems based on the difficulty of a problem (based on the fraction of times the problem was correctly solved) and plot the percentage of tokens needed for solving those problems. A diagonal line indicates equal allocation regardless of problem difficulty and increasingly convex curves indicate fewer token allocation to easier problems. On difficult datasets (like AIME 2024 or AIME 2025) the model only allocates computation to easy problems that it solves correctly for large values of λ. In contrast for easier datasets like Minerva or Math500, the curves for 4B and 8B models are increasingly convex for increasing λ, implying that the model allocates fewer tokens to easier problems.

A.2 Forecasting performance

In this section we show the forecasting performance for all the five evaluation datasets considered in the main paper. This corresponds to fig. 1.

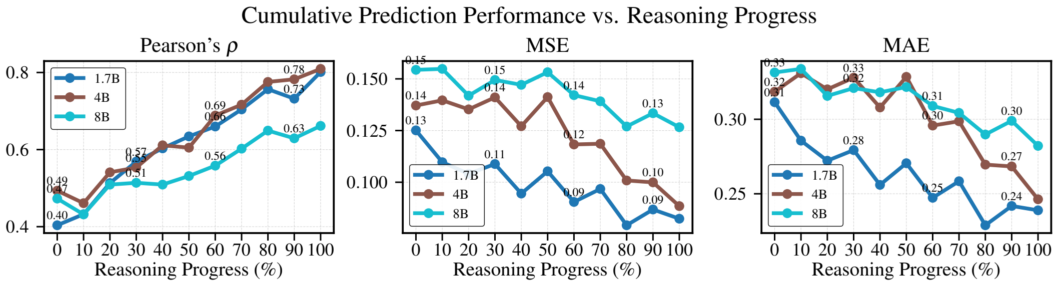

Figure 7: Forecasting performance for AIME2024

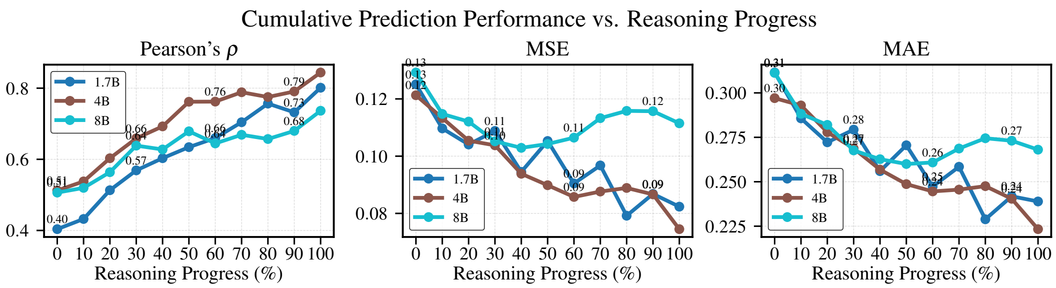

Figure 8: Forecasting performance for AIME2025

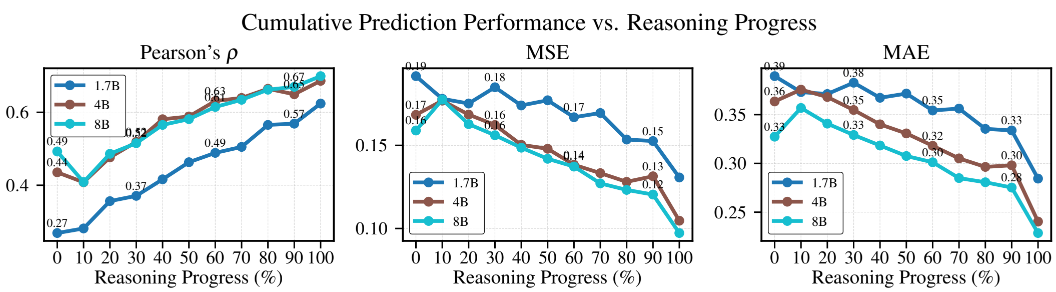

Figure 9: Forecasting performance for AMC

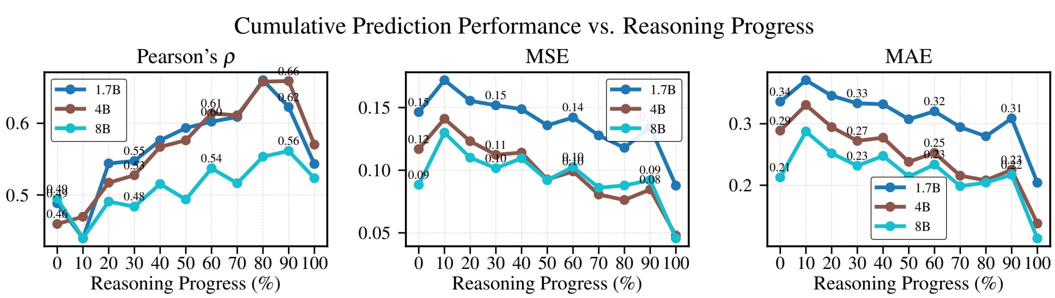

Figure 10: Forecasting performance for Math500

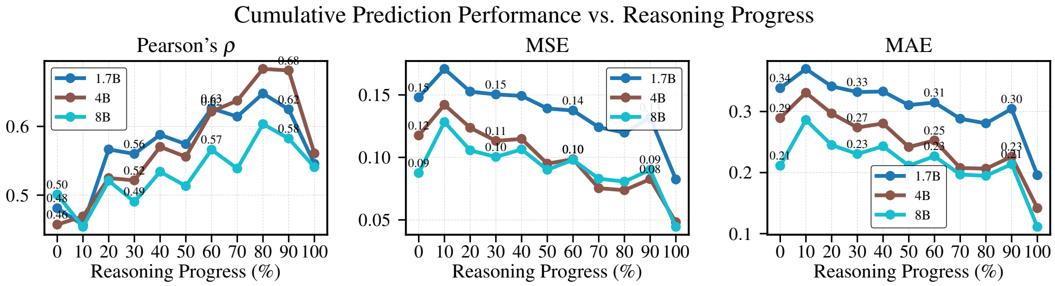

Figure 11: Forecasting performance for Minerva

A.3 Early Stopping

In this section we show the early stopping performance for all the five evaluation datasets considered in the main paper, corresponding to fig. 2.

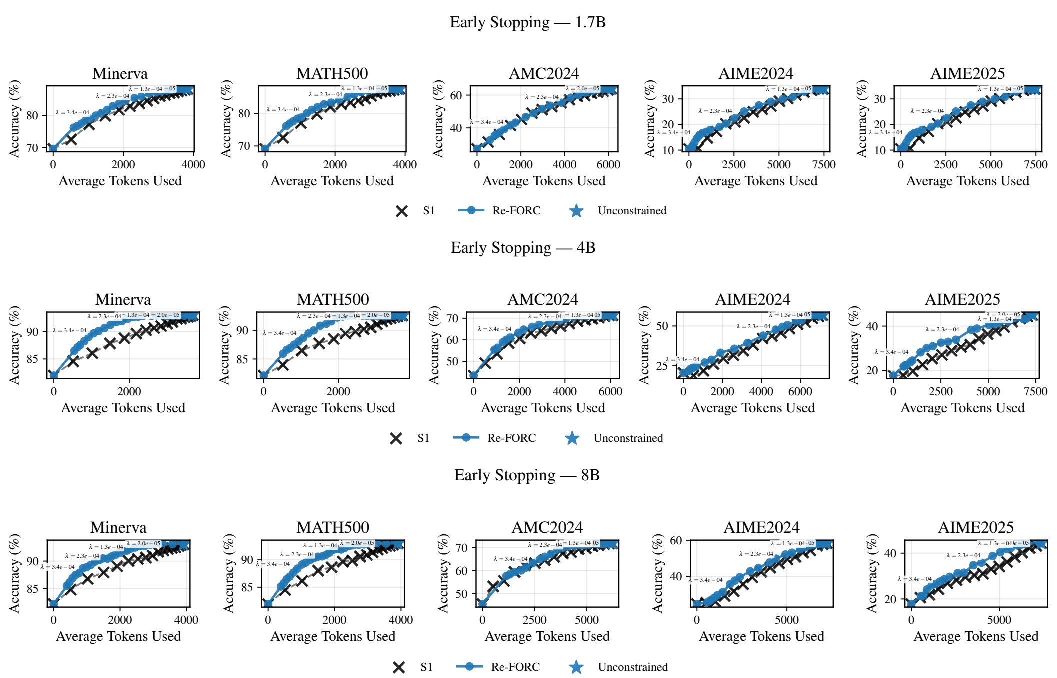

Figure 12: Early Stopping performance for 1.7B (top); 4B (middle); and 8B (bottom) Qwen 3 models.

A.4 Model Selection

In this section we show the model-selection performance for all the five evaluation datasets considered in the main paper, corresponding to fig. 3.

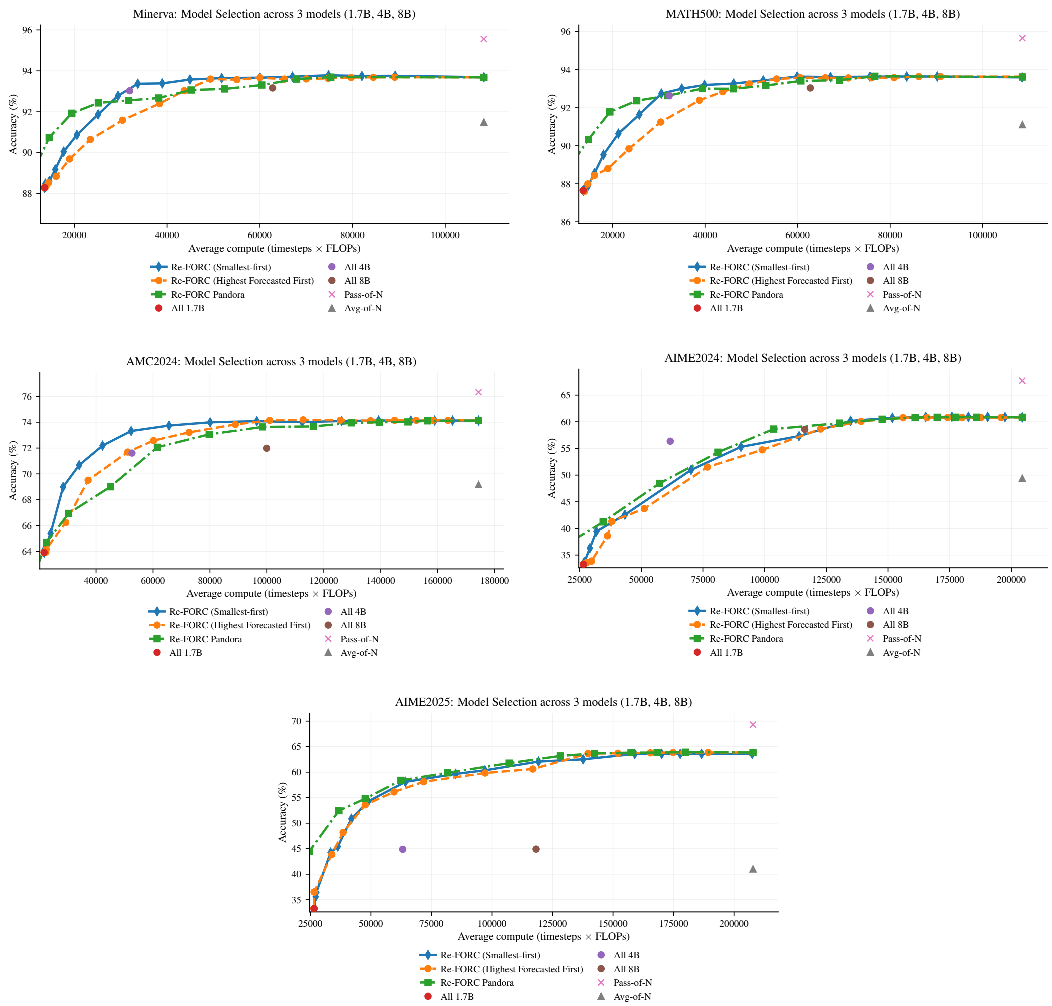

Figure 13: Cost-aware selection comparing accuracy versus token usage, where Pass@k represents the theoretical maximum with ground truth access, while Re-FORC based model selection (from eq. (10)) enables practical per-token cost control while improving accuracy.

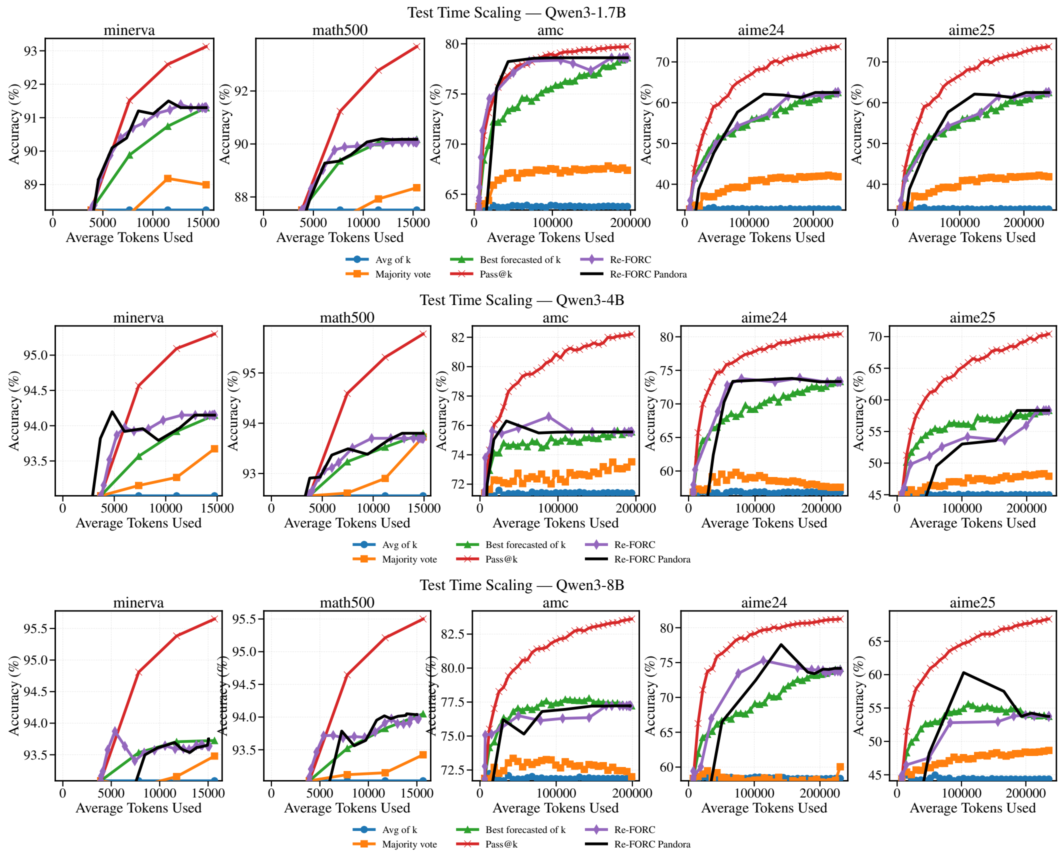

Figure 14: Test time scaling performance for 1.7B (top); 4B (middle); and 8B (bottom) Qwen 3 models.