BEACON: Bayesian Optimal Stopping for Efficient LLM Sampling

Anonymous ACL submission

Abstract

Sampling multiple responses is a common way to improve LLM output quality, but it comes at the cost of additional computation. The key challenge is deciding when to stop generating new samples to balance accuracy gains against efficiency. To address this, we introduce BEACON (Bayesian Efficient Adaptive Criterion for Optimal N-stopping), a principled adaptive sampling framework grounded in Sequential Search with Bayesian Learning. BEACON sequentially generates responses from the policy LLM, updates posterior belief over reward distributions in real time without further training, and determines when to stop by weighing expected gains against computational cost. Sampling terminates once the marginal utility of further exploration no longer justifies the expense. We establish both theoretical optimality guarantees and practical tractability, and show empirically that BEACON reduces average sampling by up to 80% while maintaining response quality. We further demonstrate BEACON's utility for cost-efficient preference data generation and outline practical extensions, offering actionable insights for future researchers.

1 Introduction

Large Language Models (LLMs) have shown human-like abilities across diverse tasks such as mathematics, coding, and creative writing (Ke et al., 2025; Hendrycks et al., 2021). Yet, they often produce inconsistent outputs, occasionally hallucinated on queries they could solve correctly across different runs (Manakul et al., 2023; Xu et al., 2025). To address this, **sampling** has been widely adopted: by generating multiple responses and selecting one based on specific criteria, it improves performance in tasks like complex reasoning (Wang et al., 2022; Snell et al., 2025), safety alignment (Ichihara et al., 2025), and preference data generation (Yuan et al., 2024). However, blindly scaling computational resources is suboptimal and impractical, particularly in settings such as streaming or real-time LLM applications (Xiao et al., 2024), where efficiency is as critical as **response quality** (Yehudai et al., 2025). This highlights the need for a deeper understanding of the **economy of inference**—balancing computational cost against performance gains.

Existing adaptive sampling methods are mainly based on sample-consistency heuristics to estimate task difficulty or confidence (Aggarwal et al., 2023; Wang et al., 2022; Wan et al., 2025b; Taubenfeld et al., 2025; Wan et al., 2025a). While training-free and easy to implement, these approaches often fail to generalize (Fu et al., 2024; Wang et al., 2025a) because multiple incorrect responses can exhibit consistency, and measuring consistency remains challenging for **open-ended tasks** with multiple valid answers. An alternative direction focuses on making Best-of-N sampling adaptive by learning when to stop generating candidates based on **reward model feedback** (Cobbe et al., 2021; OpenAI, 2022; Zhang et al., 2024). While these adaptive BoN methods show effectiveness across diverse scenarios, they rely on data-centric, training-heavy pipelines to learn auxiliary stopping models (Damani et al., 2025), which limits adaptability to new domains while potentially introducing bias and reducing output diversity. Critically, both approaches rely on heuristics or learned approximations without theoretical guarantees of optimality, making their stopping decisions inherently ad-hoc.

To bridge this theory-practice gap, we leverage principles in Bayesian optimal stopping (Baucells and Zorc, 2024) and reformulate LLM sampling as a sequential search problem. This framework ensures stopping decisions achieve **Bayesian optimality** given currently observed data, eliminating reliance on heuristic approximations (Rothschild, 1974). Rather than learning reward distributions *offline*, we conceptualize them as latent processes for online updating: each generated response reveals

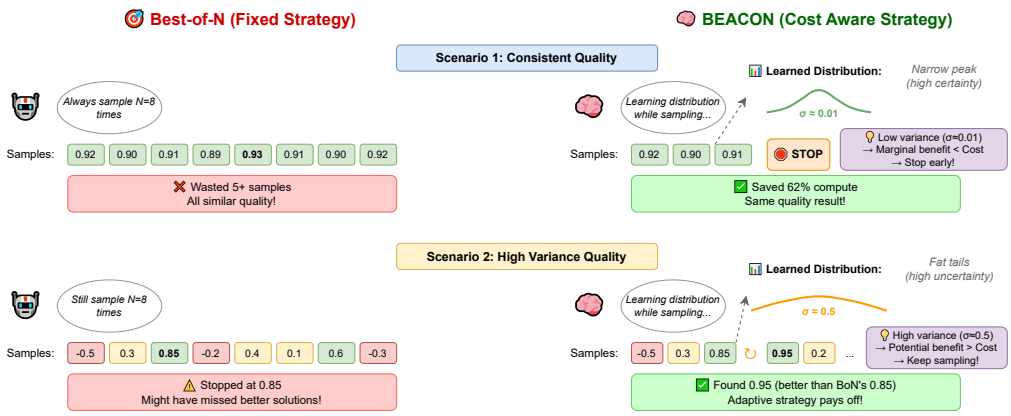

Figure 1: **Comparison of BEACON adaptive sampling versus Best-of-N fixed sampling.** BEACON adaptively determines sample size by learning the reward distribution and determine if additional sampling is worth the cost. Intuitively, BEACON stops earlier for consistent samples and continues sampling to find better solutions for variable reward samples, while Best-of-N always uses fixed samples (in this case, 8) regardless.

.

information about the underlying reward distribution while incurring computational costs (Toth and Oberhauser, 2020). The fundamental challenge becomes determining the **optimal stopping point** where expected benefits from additional samples no longer justify associated costs, which can be addressed with Bayesian learning theory (Christensen, 1986; Bikhchandani and Sharma, 1996).

We therefore introduce the **Bayesian Efficient Adaptive Criterion for Optimal N-stopping (BEACON)**, a novel adaptive sampling framework that makes optimal stopping decisions computationally practical while enabling real-time deployment without additional offline training requirements. Our approach can be understood through two synergistic components: **sequential search** addresses the adaptivity challenge, while **Bayesian learning** provides a principled framework for online reward distribution learning. Together, these components enable derivation of adaptive sampling policies without pre-training while guaranteeing **theoretical optimality**. BEACON sequentially collects responses, updates sufficient statistics of the posterior reward distribution, and employs an index-based sampling policy that compares a quality index against a cost threshold. Intuitively, BEACON terminates when reward evaluations exhibit minimal variation, indicating stable characterization of the quality distribution, or when additional computation is unlikely to yield superior rewards. Figure 1 contrasts BEACON's adaptive stopping with conventional Best-of-N sampling, illustrating how our Bayesian criterion adaptively allocates computation across variable-reward queries. Our empirical evaluations on reasoning and alignment benchmarks demonstrate that BEACON substantially reduces average inference costs compared to fixed BoN while maintaining comparable performance, with demonstrated utility for cost-effective preference data generation, practical hyperparameter selection guidance, and extensions to batch sampling for enhanced efficiency. In sum, our main contributions are: (**1**) We propose BEACON, an **adaptive sampling framework** that reformulates LLM sampling as a sequential Bayesian search problem for theoretically optimal stopping without training additional auxiliary models. (**2**) We provide rigorous analysis of its **theoretical guarantees** and computational complexity. (**3**) We conduct comprehensive experiments across diverse benchmarks, demonstrating its effectiveness against established baselines and highlighting practical extensions for post-training and real-world deployment.

2 Modeling LLM Sampling as Sequential Search Problem

2.1 Methodology Overview

Reward Models as Quality Assessment Tools. Reward Models (RMs) (Zhang et al., 2024; Zhong et al., 2025) provide scalar assessments of response quality that serve as evaluation signals for adaptive sampling. Trained on pairwise preference data $D = \{(x_i, y_{w,i}, y_{l,i})\}$, these models encode both preference direction and certainty in the magnitude of reward differences, making them ideal quality signals for probabilistic modeling.

Sequential Search for Optimal Stopping. We

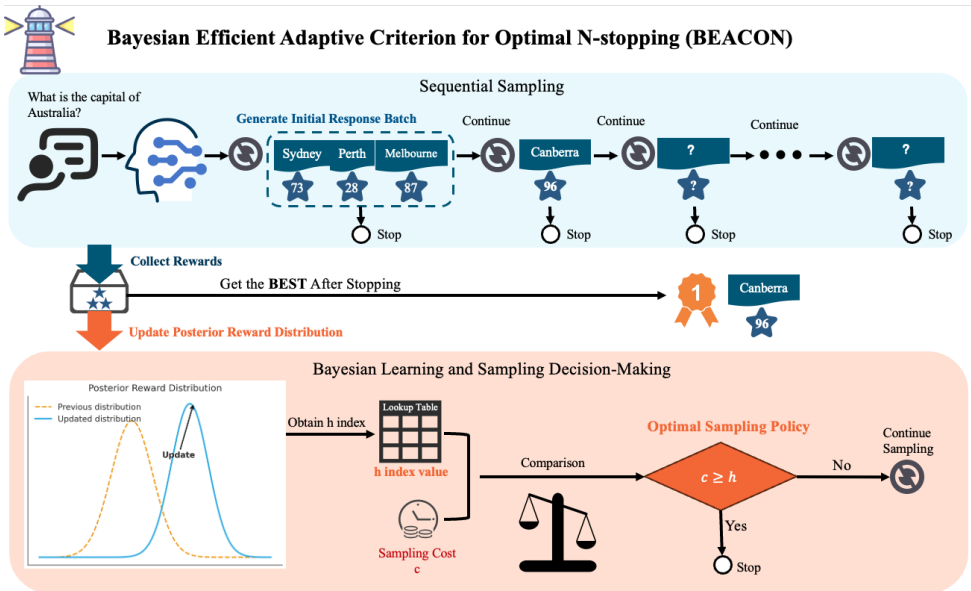

Figure 2: **BEACON framework**: The top layer shows sequential sampling of LLM responses with reward model evaluation. The bottom layer illustrates the optimal stopping mechanism, which updates Bayesian posterior beliefs about reward distribution parameters after each sample and determines when to stop based on optimal sampling policy, comparing the index-based threshold to the sampling cost.

reformulate LLM sampling as a sequential search problem to maximize net gain—balancing the highest quality against sampling cost. This approach replaces heuristics with theoretically grounded guarantees for deciding when additional samples are no longer economically justified. Sequential Search examines alternatives one by one, deciding after each observation whether to accept the current best outcome or continue sampling (Stigler, 1961; Weitzman, 1979) (detailed in Appendix D.1). Given observed rewards $\mathbf{r}_k = \{r_1, \ldots, r_k\}$ with best reward $z_k = \max\{r_1, \ldots, r_k\}$ and sampling cost c per observation, the challenge is determining the optimal stopping point, identifying the maximum reward while minimizing costs. When the reward distribution is *known*, this admits closed-form solutions (Weitzman, 1979). However, LLM sampling presents the more challenging case where the underlying reward distribution is *unknown*.

A Principled Bayesian Framework for Unknown Distributions. BEACON combines sequential search with Bayesian learning to address the fundamental challenge of *unknown reward distributions* by learning parameters online during sampling, enabling zero-shot deployment without offline training or pre-training. For computational tractability, we employ conjugate priors that enable closed-form updates. We focus on the Normal distribution for its practical utility and unique theoretical properties—it is the only continuous distribution with computationally efficient optimal index policies in sequential search literature (Baucells and Zorc, 2024), which is more challenging than simpler, discrete conjugate families such as the beta-binomial extension that is also supported by BEACON (demonstrated in Appendix D.6).

2.2 Model Setup

Problem Setting. Given a query x and a policy LLM $\pi_\phi(y|x)$, we sequentially generate responses $\{y_1, \ldots, y_k\}$ where $y_i \sim \pi_\phi(\cdot|x)$ and evaluate each using a reward model $R(x, y_i) = r_i$. The reward distribution $f(r_k|x)$ emerges from the marginalized distribution over the response generation and reward evaluation processes. We assume rewards follow an i.i.d. Normal distribution with unknown parameters, which distinguishes our approach from methods requiring known distributions or training auxiliary models. After collecting k samples with rewards $\mathbf{r}_k = \{r_1, \ldots, r_k\}$, we denote the current best reward as $z_k = \max\{r_1, \ldots, r_k\}$. Each sample incurs cost c, and sampling continues up to maximum horizon n. Our objective is determining the optimal stopping time K that maximizes expected net gain $\mathbb{E}[z_K - K \cdot c]$.

Bayesian Learning with Conjugate Priors. To enable tractable Bayesian updating, we employ a Normal-Inverse-Gamma (NIG) conjugate prior for unknown parameters (μ, σ^2). Conjugate priors guarantee a fixed-dimensional state space during se-

Algorithm 1 BEACON: Bayesian Efficient Adaptive Criterion for Optimal N-stopping

Input: Query x, policy LLM $\pi_\phi(y|x)$, reward model $R(x, y)$, cost c, max samples n, grid size G
Output: Best response y^* and its reward r^*

// Step 1: Initialize prior and h-index table
$(\alpha_0, \nu_0, \beta_0, \mu_0) \leftarrow (-0.5, 0, 0, 0)$ ▷ Non-informative prior
$h\text{-}table \leftarrow \text{PrecomputeTable}(n, G, \alpha_0, \nu_0)$ ▷ Pre-compute as in §D.5

// Step 2: Generate initial samples and compute baseline parameters
Generate $\{y_1, y_2, y_3\} \sim \pi_\phi(y|x)$ and compute $r_i \leftarrow R(x, y_i)$ for $i \in \{1, 2, 3\}$
$(\alpha, \nu, \mu, \beta) \leftarrow \text{UpdateHyperParams}(r_1, r_2, r_3, \alpha_0, \nu_0, \beta_0, \mu_0)$
$z \leftarrow \max\{r_1, r_2, r_3\}$, $\sigma \leftarrow \sqrt{\frac{(\nu+1)\beta}{\nu\alpha}}$, $k \leftarrow 3$

// Step 3: Adaptive sampling loop
while $k < n$ **do**
  $\hat{z} \leftarrow (z - \mu)/\sigma$, $h \leftarrow \text{LookupHIndex}(h\text{-}table, k, \hat{z})$
  if $h \leq c/\sigma$ **then**
    break ▷ Apply UIP as in (3)
  end
  Generate $y_k \sim \pi_\phi(y|x)$ and compute $r_k \leftarrow R(x, y_k)$ $k \leftarrow k + 1$
  $q_{0.01} \leftarrow F^{-1}_{2\alpha_{k-1}}(0.01|\mu, \sigma)$ $\tilde{r}_k \leftarrow \begin{cases} \mu & \text{if } r_k < q_{0.01} \\ r_k & \text{otherwise} \end{cases}$ ▷ Filter extreme low values
  $(z, \mu, \sigma) \leftarrow \text{UpdateStats}(z, \mu, \sigma, \tilde{r}_k, k)$ ▷ The sufficient statistics updated by (5)
end
$i^* \leftarrow \arg\max_{i \in \{1,\ldots,k\}} r_i$ **return** y_{i^*}, r_{i^*}

quential sampling (Diaconis and Ylvisaker, 1979), essential for computational tractability. Starting with prior $\text{NIG}(\mu_0, \nu_0, \alpha_0, \beta_0)$, after observing $k \geq k_0$ samples (where k_0 is the minimum for well-defined posteriors; see Appendix D.2), the posterior parameters update according to closed-form expressions detailed as (4) in Appendix D.2. Importantly, the triple (z_k, μ_k, σ_k) forms **sufficient statistics** for all observed rewards, enabling efficient state representation where the updating formula (5) in Appendix D.3 shows how these statistics evolve with each new sample.

Bellman Equation. To formulate our objective function $\mathbb{E}[z_K - K \cdot c]$ in Bellman equation as:

$$V_{n,k}(z_k, \mu_k, \sigma_k; c) = \max\{z_k, \mathbb{E}[V_{n,k+1}(z_{k+1}, \mu_{k+1}, \sigma_{k+1}; c)] - c\}, \tag{1}$$

with corresponding optimal stopping rule:

$$\text{Stop iff} \quad H_{n,k}(z_k, \mu_k, \sigma_k; c) = \mathbb{E}[V_{n,k+1}(z_{k+1}, \mu_{k+1}, \sigma_{k+1}; c)] - z_k \leq c. \tag{2}$$

where $H_{n,k}$ represents the expected marginal gain from continued sampling.

2.3 Optimal Sampling Policy

Building on recent theoretical advances of the computationally efficient Universal Index Policy (Baucells and Zorc, 2024), we can establish an efficient criterion for optimal stopping decisions.

Definition 1. *For $k_0 \leq k < n$, the h-index function $h_{n,k} : \mathbb{R} \to (0, \infty)$ maps each standardized best reward $\hat{z} \in \mathbb{R}$ to the unique value $c > 0$ that solves the condition where the expected marginal gain from continuing equals the sampling cost.*

Theorem 1 (Optimal Sampling Policy)**.** *After generating initial samples $\{y_1, y_2, \ldots, y_{k_0}\}$ to establish valid posterior parameters, the optimal Bayesian policy at each step $k \geq k_0$ is to continue sampling if and only if:*

$$h_{n,k}\left(\frac{z_k - \mu_k}{\sigma_k}\right) > \frac{c}{\sigma_k} \tag{3}$$

The stopping time $K = \min\{k \geq k_0 : h_{n,k}(\hat{z}_k) \leq c/\sigma_k\} \wedge n$ maximizes the expected net gain $\mathbb{E}[z_K - K \cdot c]$.

The proof of Theorem 1 is provided in the Appendix. Our approach standardizes the current best

reward $\hat{z}_k = (z_k - \mu_k)/\sigma_k$, retrieves the corresponding h-index value, and compares it against the cost-adjusted threshold c/σ_k. The algorithm stops when this threshold is no longer exceeded, indicating that further sampling has become economically inefficient given our posterior beliefs.

Normality Assumption. While exact optimality guarantees hold under normality, BEACON remains robust to moderate distributional violations through several mechanisms: (1) the Central Limit Theorem suggests that reward signals naturally approximate normality in practice (as shown in App. E.5); (2) our focus on identifying maximum rewards depends on the right tail of the distribution, making the framework less sensitive to left-tail deviations; and (3) we introduce a robust updating mechanism (see Section 3.2) that filters extreme low outliers while preserving high-quality samples, maintaining practical effectiveness when distributions exhibit negative skewness.

Sensitivity Analysis. The optimal stopping time K exhibits intuitive dependencies on key problem parameters. When sampling cost c increases, exploration is often discouraged. If the current best reward z_k substantially exceeds the posterior mean μ_k (large $\hat{z}_k$), the framework recognizes an exceptionally high-quality sample has likely been found and stops earlier. Conversely, greater posterior uncertainty (larger σ_k) encourages continued sampling through two mechanisms: by decreasing the normalized score $\hat{z}_k$ and lowering the effective cost threshold $\frac{c}{\sigma_k}$. Intuitively, when more exploration budget remains available (larger $n - k$), the algorithm tends to be more patient, balancing immediate rewards against future exploration potential (see Appendix D.8 for proofs and formal statements).

2.4 BEACON Framework

Hyperparameter Configuration. BEACON requires three key hyperparameters that jointly define the optimization framework: (1) *Prior Parameters* – We use Jeffreys' non-informative prior $(\alpha_0, \nu_0, \mu_0, \beta_0) = (-0.5, 0, 0, 0)$ for task-agnostic deployment without domain-specific calibration, requiring $k_0 = 3$ initial samples for well-defined posteriors (Appendix D.2); (2) *Maximum Horizon (n)* – The sampling budget follows standard Best-of-N configurations (e.g. $n \in \{8, 16, 32\}$), where larger horizons increase patience but raise h-index precomputation costs (Appendix D.5); (3) *Sampling Cost (c)* – Controls the quality-efficiency trade-off, with higher values favoring efficiency and lower values favoring quality.

Algorithm Implementation. Algorithm 1 presents the complete BEACON procedure, with the overall framework illustrated in Figure 2. After initializing Jeffreys' non-informative priors and pre-computing h-index tables, we generate $k_0 = 3$ bootstrap samples to establish valid posterior parameters. The adaptive sampling loop then iteratively: (1) computes standardized score $\hat{z}_k = (z_k - \mu_k)/\sigma_k$; (2) retrieves h-index $h_{n,k}(\hat{z}_k)$ via table lookup; (3) applies optimal stopping criterion $h_{n,k}(\hat{z}_k) \leq c/\sigma_k$; and (4) if continuing, generates new samples and updates parameters using robust filtering. This design transforms computationally intensive Bellman optimization into efficient table lookups, enabling real-time deployment while maintaining theoretical optimality guarantees.

Computational Complexity. BEACON's computational overhead consists of two distinct components with different scalability characteristics: (1) *h-Table Pre-computation* – Constructing the lookup table $h_{n,k}(\cdot)$ requires $\mathcal{O}(nG)$ operations for horizon n and grid resolution G. This one-time cost is amortized across all queries that share the same horizon, making it negligible in multi-query deployments. When tasks involve multiple horizons $\{n_1, \ldots, n_J\}$, complexity grows only with the number of distinct horizon values rather than the total number of queries; (2) *Sequential Inference* – Each query entails sequential decision-making, where samples are generated one-by-one to update posterior beliefs. This inherent dependency limits within-query parallelization and can increase latency relative to batch-generation methods. Nevertheless, the cost can be partially mitigated through batch-parallel sampling (Section 3.2).

3 Experiments

3.1 Main Results

Setup. We use a warm-start of $k_0 = 3$ (to initialize a Jeffreys' prior) and maximum horizon of $n = 32$ (standard for Best-of-N (Singhi et al., 2025)). For each method m, K_m denotes its realized sample count ($1 \leq K_m \leq n$) and $\overline{K}_m$ its dataset average. We focus comparisons on training-free methods and standard BoN as these represent the most widely deployed approaches. Specifically, we include one-shot **Chain-of-Thought** (CoT) with K=1; **Self-Consistency** (SC) (Wang et al., 2022)

Table 1: Comparison of BEACON with baseline sampling methods across different models and tasks. BEACO achieves a superior trade-off between Accuracy/Win Rate/Reward $\bar{\tilde{z}}_K$ and efficiency (Avg Sample $\overline{K}$), measured by the implicitly optimized objective $\hat{\tilde{V}}_K$. Upward arrows (↑) indicate percentage improvement in accuracy or win rate over CoT baselines; downward arrows (↓) show percentage reduction in samples compared to the maximum BoN sample size (32). **Reward models: N-RM** uses *Llama-3.1-Nemotron-70B-Reward* (Wang et al., 2025c); **S-RM** uses *Skywork-Llama-3.1-8B* (Liu et al., 2024); Additional runs for statistical signifiance reported in Table 6.

		Reasoning Tasks (Avg. MATH/AIME/AMC)				Alignment Task (AlpacaEval 2.0)			
Model	**Method**	**Accuracy ↑ (%)**	**Samples ↓ ($\overline{K}$)**	**Reward ↑ ($\bar{\tilde{z}}_K$)**	**Value ↑ ($\hat{\tilde{V}}_K$)**	**Win Rate ↑ (%)**	**Samples ↓ ($\overline{K}$)**	**Reward ↑ ($\bar{\tilde{z}}_K$)**	**Value↑ ($\hat{\tilde{V}}_K$)**
`LLaMA-3.2-3B`	Direct CoT	20.0	1.0	-1.15	-0.40	16.0	1.0	-1.08	-0.80
	SC	28.5 ↑42.5%	16.0 ↓50.0%	-0.35	-0.01	-	-	-	-
	RASC	27.8 ↑39.0%	5.2 ↓83.8%	-0.30	0.66	20.0 ↑25.0%	7.0 ↓78.1%	-1.12	-0.55
	AS	27.8 ↑39.0%	5.6 ↓82.5%	-0.31	0.62	-	-	-	-
	BoN (N-RM)	33.4 ↑67.0%	32.0	1.75	0.29	25.0 ↑56.3%	32.0	1.76	0.80
	BoN (S-RM)	31.0 ↑55.0%	32.0	1.72	0.25	24.0 ↑50.0%	32.0	1.65	0.55
	BEACON (N-RM)	32.8 ↑64.0%	15.8 ↓50.6%	1.68	**1.12**	23.5 ↑46.9%	14.5 ↓54.7%	1.68	**1.20**
	BEACON (S-RM)	32.0 ↑60.0%	16.1 ↓49.7%	1.66	**1.08**	22.5 ↑40.6%	14.8 ↓53.8%	1.53	**1.15**
`Qwen2.5-7B`	Direct CoT	43.0	1.0	-1.25	-0.53	22.5	1.0	-0.85	-0.53
	SC	50.0 ↑16.3%	16.0 ↓50.0%	-0.37	-0.02	-	-	-	-
	RASC	49.5 ↑15.1%	4.3 ↓86.6%	-0.28	0.57	25.0 ↑11.1%	4.0 ↓87.5%	-1.15	-0.30
	AS	49.3 ↑14.7%	4.6 ↓85.6%	-0.29	0.55	-	-	-	-
	BoN (N-RM)	55.2 ↑28.4%	32.0	1.85	0.09	36.0 ↑60.0%	32.0	2.02	1.02
	BoN (S-RM)	55.0 ↑27.9%	32.0	1.76	0.05	35.0 ↑55.6%	32.0	2.05	1.05
	BEACON (N-RM)	54.0 ↑25.6%	6.5 ↓79.7%	1.78	**0.92**	33.5 ↑48.9%	7.8 ↓75.6%	1.95	**1.55**
	BEACON (S-RM)	54.0 ↑25.6%	7.0 ↓78.1%	1.64	**0.91**	33.0 ↑46.7%	8.0 ↓75.0%	1.90	**1.50**
`Grok-3-Mini`	Direct CoT	89.0	1.0	-1.10	-0.28	82.0	1.0	-0.98	-0.70
	SC	92.0 ↑3.4%	16.0 ↓50.0%	-0.38	-0.04	-	-	-	-
	RASC	93.8 ↑5.4%	3.5 ↓89.1%	-0.22	0.56	85.5 ↑4.3%	3.5 ↓89.1%	-1.02	-0.55
	AS	93.8 ↑5.4%	3.8 ↓88.1%	-0.23	0.53	-	-	-	-
	BoN (N-RM)	95.5 ↑7.3%	32.0	1.62	0.10	94.0 ↑14.6%	32.0	1.45	0.80
	BoN (S-RM)	95.0 ↑6.7%	32.0	1.70	0.19	94.2 ↑14.9%	32.0	1.42	0.79
	BEACON (N-RM)	94.8 ↑6.5%	5.0 ↓84.4%	1.57	**0.99**	92.8 ↑13.2%	4.0 ↓87.5%	1.39	**1.94**
	BEACON (S-RM)	94.2 ↑5.8%	5.5 ↓82.8%	1.64	**0.95**	93.4 ↑13.9%	4.3 ↓86.6%	1.36	**1.93**

Note: SC = Self-Consistency (Wang et al., 2022); RASC = Reasoning-Aware Self-Consistency (Wan et al., 2025a); AS = Adaptive Sampling (Aggarwal et al., 2023); BoN = Best-of-N sampling (Cobbe et al., 2021) SC and AS are not applicable to alignment tasks.

with majority vote over n samples; **Adaptive-Consistency** (AS) (Aggarwal et al., 2023), a model-agnostic method that fits a Dirichlet–multinomial posterior to agreement patterns for early stopping; **RASC** (Wan et al., 2025a), a heuristic adaptive sampler using CoT quality scores; and Best-of-N (**BoN**), which selects the highest reward-scored candidate from n attempts. Policy models include LLaMA-3.2-8B, Qwen2.5-7B-Instruct, and Grok-3-mini; reward models include Llama-3.1-Nemotron-70B-Reward and Skywork-Llama-3.1-8B. We used CoT (Wei et al., 2022) prompting with model-specific templates. Evaluation covered: (1) Pass @ 1 for *Reasoning Tasks* on three mathematical benchmarks (`MATH-500` (Lightman et al., 2023), `AIME24` (Mathematical Association of America, 2024), `AMC23`); (2) *Alignment Task* using `AlpacaEval 2.0` (Li et al., 2023), comparing responses for user instructions against GPT-4; and (3) for both tasks, *expected standardized reward* $\bar{\tilde{z}}_K$ and *expected standardized value* $\hat{\tilde{V}}_K = \mathbb{E}[z_K - K \cdot c]$, representing the quality-efficiency tradeoff (more details in Appendix A.1).

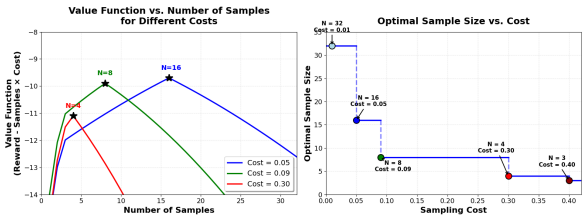

Figure 3: Impact of the sampling cost (c) on the value optimization and the optimal sample size. Higher c results in earlier stopping to reach Bayesian optimality.

Optimal Performance-Efficiency Tradeoff. Table 1 demonstrates BEACON's ability to achieve an optimal balance between performance and computational efficiency. Our approach consistently matches BoN's performance while requiring significantly fewer samples. This tradeoff is quantitatively validated by *consistently superior value function scores*, confirming that BEACON effectively maximizes expected net gain as a Bayesian optimal stopping solution. Results remain consistent across experimental conditions—including different base models, reward models, and task categories.

Impact of Sampling Cost (c) on Optimized Sam-

ple Size. We analyze how the sampling cost parameter c shapes BEACON's adaptive sample size K_{BEACON}. As shown in Figure 3 , increasing c consistently reduces the optimal number of samples, aligning with our discussion in Section 2.2. Unlike factors such as reward variance, response quality, or remaining budget—which emerge from query characteristics—c serves as a *human-interpretable* control knob for balancing efficiency and quality. Stopping typically occurs when responses stabilize, when quality remains uniformly low, or when nearing the maximum budget (examples in E.5). For practical deployment, we recommend a default $c = 0.1$ when there is no strong preference between efficiency and quality. Lower values of c are better suited for difficult, high-variance tasks, whereas higher values suit easier or consistent ones (See App. B.1 for guidelines).

BEACON's Effectiveness at Controlled Average Sample Sizes. Building on our analysis of how sampling cost c influences BEACON's stopping behavior, we provide an alternative perspective by investigating scenarios where c is calibrated to ensure BEACON's average sample size ($\overline{K}_{BEACON}$) matches the fixed sample size of standard Best-of-N (BoN) strategies. Figure 4 illustrates this controlled comparison, revealing BEACON's advantages: (1) when using the *same number of samples* ($\overline{K}_{BEACON} = K_{BoN}$), BEACON achieves substantially *higher accuracy and reward*; and (2) BEACON can maintain *equivalent accuracy and reward* while requiring *fewer samples*. This performance advantage stems from BEACON's dynamic sampling strategy, which intelligently invests additional samples in promising queries while terminating earlier for less promising ones—a fundamental improvement over BoN's uniform sampling approach that allocates identical resources regardless of query nature or potential quality improvements.

3.2 Extensions and Applications

Efficiency in Data Generation for Iterative DPO. To demonstrate BEACON's practical utility, we applied it to improve efficiency in data generation for Iterative Direct Preference Optimization (DPO) (Rafailov et al., 2023), which enhances LLM consistency on challenging questions (Xiong et al., 2025). We evaluated BEACON against a standard Best-of-N (BoN) approach—the conventional method for generating preference pairs in iterative training processes (Yuan et al., 2025); comprehensive experimental details are provided in

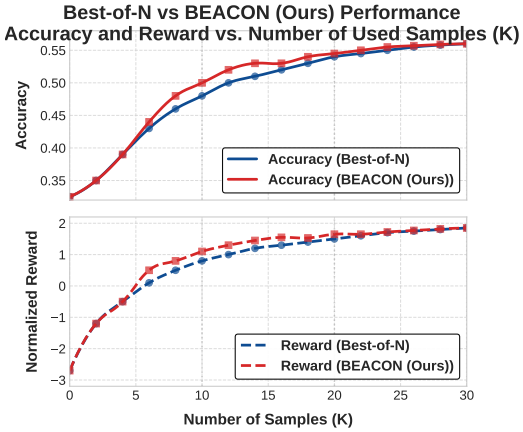

Figure 4: **Comparative performance trajectories on Best-of-N vs. BEACON**. When forcing the same number of samples, BEACON achieves better performance; When restricting on a threshold performance, BEACON would require less number of samples.

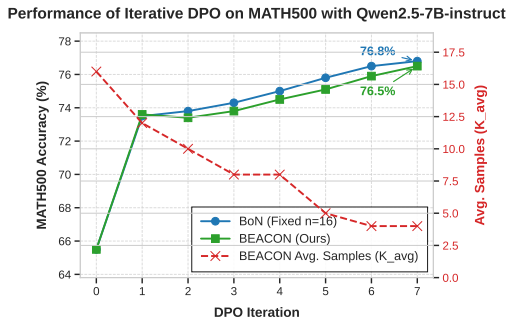

Figure 5: Iterative DPO performance on MATH500 with Qwen2.5-7B-instruct. BEACON (accuracy: green squares) achieves competitive accuracy relative to BoN (fixed n=16, accuracy: blue circles) across seven DPO iterations while reducing the average number of samples required per prompt ($\overline{K}$: red dashed line, right y-axis).

Appendix C.2. Figure 5 illustrates performance across seven DPO iterations, revealing that BEACON achieved *comparable accuracy* while progressively reducing the average required samples ($\overline{K}$, shown on the secondary y-axis). This preliminary experiment demonstrates that as LLMs become more consistent and aligned with preferences, BEACON efficiently identifies high-quality preference pairs with substantially fewer samples to improve post-training efficiency.

Robust Updating for Negative Skewness. Reward distributions from LLMs occasionally exhibit negative skewness (Fig 7), where extreme low outliers distort posterior updates. We introduce a *robust updating mechanism* that preserves the informative right tail—critical for identifying maximum rewards—while mitigating left-tail outliers. Values below the 1% posterior-predictive quantile are

Table 2: Batch-parallel BEACON performance across different batch sizes, evaluated with LLaMA 3.2 on a mathematical reasoning dataset. Memory overhead is measured relative to sequential execution ($b = 1$).

Batch Size	**Speedup** (vs Seq.)	**Memory** Overhead	**Acc.** Change	**Avg. Samples**
1 (Seq.)	1.00×	Baseline	0.0%	15.8
2	1.65×	+12%	+0.2%	16.1
4	2.31×	+28%	+0.6%	17.2
8	2.85×	+65%	+0.8%	19.4

replaced with the current posterior mean, ensuring high-quality candidates remain intact while the posterior better reflects the reward landscape. This adaptive update reduces outlier impact without violating Gaussian assumptions. Figure 9 shows the robust mechanism consistently drives BEACON's stopping points closer to the true optimum. Full derivations and analysis are in Appendix E.4.

Parallel Sampling. While BEACON is inherently sequential, practical deployments can benefit from batch-parallel sampling, like BoN Sampling, to reduce wall-clock time at the expense of higher memory usage. In this mode, BEACON generates batches of b samples simultaneously, *updates posterior beliefs after each batch*, and applies the same stopping criterion $h_{n,k}(\hat{z}_k) \leq \frac{c}{\sigma_k}$, where k now indexes completed batches. Within each batch, responses are sampled independently using the same prompt, then evaluated with the reward model; posterior parameters are updated with all batch rewards, and the stopping rule is applied. Termination selects the highest-reward response across all batches. As shown in Table 2, batching can slightly improve accuracy since larger batches enforce a minimum exploration depth before each stopping decision, though this comes with diminishing returns and increased memory overhead. Moreover, as discussed in 2.4, BEACON's pre-computed UIP index tables can be *reused across multiple parallel queries* without recomputation, enabling efficient query-level parallelization at no additional cost.

4 Related Work

Parallel Reasoning and Efficiency. Parallel scaling approaches (Zeng et al., 2025; Qian et al., 2025) enhance LLM answers by generating multiple candidates and aggregating them into a final answer, through consensus (e.g., majority voting and weighted confidence scores (Wang et al., 2022; Chen et al., 2024; Fu et al., 2025)) or with external verifiers and reward models that rank and select superior solutions (Cobbe et al., 2021; Ichihara et al., 2025; Zhang et al., 2024; Ankner et al., 2024). While effective, these methods prioritize response quality without explicitly addressing computational costs. Recent work on efficient sampling strategies (Sui et al., 2025; Fu et al., 2024) seeks to address the issue using fine-tuned verifiers (Manvi et al., 2024; Huang et al., 2025; Wan et al., 2025a) or heuristic rules that adapt resource allocation by query complexity (Wang et al., 2025b; Wan et al., 2025b; Aggarwal et al., 2023). In contrast, our BEACON framework takes a principled Bayesian learning approach that integrates reward signals with optimal stopping theory to jointly optimize response quality and computational efficiency.

Bandits and Bayesian Optimization. BEACON's sequential search framework connects to established paradigms in decision theory and optimization (Keith and Ahner, 2021). While Multi-Armed Bandit problems (Lattimore and Szepesvári, 2020; Slivkins, 2019) emphasize maximizing cumulative rewards, BEACON focuses on finding the maximum reward with minimal sampling. Our work builds more directly on best-arm identification (Audibert and Bubeck, 2010; Gabillon et al., 2012) and Extreme Bandits (Carpentier and Valko, 2014; Lopez et al., 2021), which similarly target optimal or extreme-value outcomes. BEACON's novelty lies in combining a Bayesian approach with conjugate priors for adaptive belief updates and optimal stopping theory (Ferguson, 2012) to decide when further exploration is no longer cost-effective. While conceptually related to budgeted bandits (Xia et al., 2016) and Bayesian optimization (Shahriari et al., 2015), BEACON applies these ideas to LLM sample efficiency, bridging theoretical insights with practical inference.

5 Conclusion

We introduced **BEACON**, a principled framework grounded in sequential search theory with Bayesian learning under conjugate priors. BEACON addresses the fundamental trade-off between computational cost and response quality during LLM inference by dynamically determining when to stop sampling based on evolving posterior beliefs about reward distributions and the cost of additional sampling. With strong empirical results and comprehensive theoretical analysis, we demonstrate the value of decision-theoretic approaches for resource-aware scaling in LLM reasoning and generation.

Limitations

While BEACON provides strong efficiency gains, several avenues remain for extending the framework. First, BEACON's optimality guarantees assume the reward model provides accurate quality signals; systematic reward model miscalibration could lead to suboptimal stopping decisions. Integrating reward model uncertainty quantification—for instance, by incorporating ensemble-based confidence estimates into the posterior updates—represents a promising direction for enhancing robustness. Second, the framework currently assumes independence across samples, whereas exploiting correlations between generated responses (e.g., through shared reasoning patterns) could further improve sample efficiency.

Future research can address these opportunities by extending BEACON to leverage reward model ensembles for uncertainty-aware stopping and incorporating structured priors that capture dependencies between samples. Furthermore, our preliminary experiments with Iterative DPO (Section 3.2) suggest BEACON's potential for post-training optimization, warranting deeper investigation into its role across different training paradigms and reward model architectures. Additionally, dynamic tuning of the cost parameter c based on query characteristics could enable fully automated adaptation to varying computational budgets. Together, these directions position BEACON as a foundation for both efficient inference and adaptive post-training pipelines in production deployments.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. 2023. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396, Singapore. Association for Computational Linguistics.

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan Daniel Chang, and Prithviraj Ammanabrolu. 2024. Critique-out-loud reward models. In *Pluralistic Alignment Workshop at NeurIPS 2024*.

Jean-Yves Audibert and Sébastien Bubeck. 2010. Best arm identification in multi-armed bandits. In *Proceedings of the 23rd Conference on Learning Theory (COLT)*, pages 41–53.

Manel Baucells and Saša Zorc. 2024. Search in the dark: The case with recall and gaussian learning. *Operations Research*.

Sushil Bikhchandani and Sunil Sharma. 1996. Optimal search with learning. *Journal of Economic Dynamics and Control*, 20(1):333–359.

Alexandra Carpentier and Michal Valko. 2014. Extreme bandits. In *Advances in Neural Information Processing Systems*, volume 27.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2024. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*.

Ronald Christensen. 1986. Finite stopping in sequential sampling without recall from a dirichlet process. *The Annals of Statistics*, pages 275–282.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. 2025. Learning how hard to think: Input-adaptive allocation of LM computation. In *The Thirteenth International Conference on Learning Representations*.

Persi Diaconis and Donald Ylvisaker. 1979. Conjugate priors for exponential families. *The Annals of statistics*, pages 269–281.

Thomas S. Ferguson. 2012. Optimal stopping and applications. Lecture notes, UCLA Department of Statistics Papers.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. 2024. Efficiently serving llm reasoning programs with certaindex. *arXiv preprint arXiv:2412.20993*.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *arXiv preprint arXiv:2508.15260*.

Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. 2012. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems*, volume 25.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. 2025. Efficient test-time scaling via self-calibration. *Preprint*, arXiv:2503.00031.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Kenshi Abe, Kaito Ariu, Mitsuki Sakamoto, and Eiji Uchibe. 2025. Evaluation of best-of-n sampling strategies for language model alignment. *Transactions on Machine Learning Research*.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, Caiming Xiong, and Shafiq Joty. 2025. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *Preprint*, arXiv:2504.09037.

Alexander J. Keith and Darryl K. Ahner. 2021. A survey of decision making and optimization under uncertainty. *Annals of Operations Research*, 300:319–353. Published online: 25 October 2019.

Tor Lattimore and Csaba Szepesvári. 2020. *Bandit Algorithms*. Cambridge University Press.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.

Hunter Lightman, Vinit Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024. Skywork-reward: Bag of tricks for reward modeling in llms. *Preprint*, arXiv:2410.18451.

Romain Lopez, Inderjit S Dhillon, and Michael I Jordan. 2021. Learning from extreme bandit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8732–8740.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfcheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Rohin Manvi, Anikait Singh, and Stefano Ermon. 2024. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *Preprint*, arXiv:2410.02725.

Mathematical Association of America. 2024. American invitational mathematics examination - AIME. Mathematics Competition. Accessed: May 2025.

OpenAI. 2022. Measuring Goodhart's Law.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *ArXiv*, abs/2305.18290.

Michael Rothschild. 1974. Searching for the lowest price when the distribution of prices is unknown. *Journal of Political Economy*, 82(4):689–711.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. 2025. When to solve, when to verify: Compute-optimal problem solving and generative verification for llm reasoning. *Preprint*, arXiv:2504.01005.

Aleksandrs Slivkins. 2019. Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286.

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

George J. Stigler. 1961. The economics of information. *Journal of Political Economy*, 69(3):213–225.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen (Henry) Zhong, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419v3*.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. 2025. Confidence improves self-consistency in llms. *Preprint*, arXiv:2502.06233.

Csaba Toth and Harald Oberhauser. 2020. Bayesian learning from sequential data using Gaussian processes with signature covariances. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9548–9560. PMLR.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025a. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3613–3635, Albuquerque, New Mexico. Association for Computational Linguistics.

Guangya Wan, Yuqi Wu, Hao Wang, Shengming Zhao, Jie Chen, and Sheng Li. 2025b. Derailer-rerailer: Adaptive verification for efficient and reliable language model reasoning. *Preprint*, arXiv:2408.13940.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2025a. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6904–6917, Albuquerque, New Mexico. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. 2025b. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding. *Preprint*, arXiv:2503.01422.

Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. 2025c. Helpsteer2-preference: Complementing ratings with preferences. In *The Thirteenth International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

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

Yingce Xia, Tao Qin, Weidong Ma, Nenghai Yu, and Tie-Yan Liu. 2016. Budgeted bandit problems with continuous random costs. In *Asian Conference on Machine Learning*, pages 317–332. PMLR.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, Chi Jin, Tong Zhang, and Tianqi Liu. 2025. Building math agents with multi-turn iterative preference learning. In *The Thirteenth International Conference on Learning Representations*.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2025. Hallucination is inevitable: An innate limitation of large language models. *Preprint*, arXiv:2401.11817.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. 2025. Survey on evaluation of llm-based agents. *Preprint*, arXiv:2503.16416.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2025. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2024. Scaling relationship on learning mathematical reasoning with large language models.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. 2025. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? *Preprint*, arXiv:2502.12215.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. Generative verifiers: Reward modeling as next-token prediction. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. 2025. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *Preprint*, arXiv:2504.12328.

A Experimental Setup and Details

A.1 Main Experiment Setup

A.1.1 Policy Models and Reward Models

For our policy (generator) models, we evaluated a range of architectures and sizes to ensure comprehensive assessment of BEACON's performance:

- **LLaMA-3.2-3B-Instruct**: Accessed and run locally on a GPU node equipped with 2 NVIDIA A100 GPUs.
- **Qwen2.5-7B-Instruct**: Inference conducted via DeepInfra's API[1].
- **Grok-3-Mini**: Inference conducted via Grok's API[2].

For our reward models (RMs), we utilized:

- **NVIDIA Llama-3.1-Nemotron-70B-Reward**: Accessed via NVIDIA's API [3] services for response verification.
- **Skywork-Llama-3.1-8B-Reward**: Accessed and run locally on a GPU node equipped with 2 NVIDIA A100 GPUs.

[1] https://deepinfra.com/
[2] https://grok.x.ai/
[3] https://build.nvidia.com/nvidia/llama-3_1-nemotron-70b-reward

A.1.2 Datasets and Evaluation Benchmarks

We evaluated BEACON on diverse reasoning and alignment tasks:

Reasoning Tasks: The reasoning evaluation focused on mathematical problem-solving, reporting average accuracy (Pass@1) across the following benchmarks. For answer extraction and checking mathematical equivalence, we utilized the expression matching tool from the Math-Verify repository[4] to ensure consistent and robust evaluation.

- **MATH500**: We used a randomly selected subset of 50 problems from the MATH500 dataset (Hendrycks et al., 2021) to provide a broad assessment without over-focusing on this specific dataset, given its large size. The problems are sampled from the test set.

- **AIME 2024**: All 30 problems from the American Invitational Mathematics Examination 2024 were used (Mathematical Association of America, 2024)..

- **AMC 2023**: All 40 problems from the American Mathematics Competitions 2023 were used, combining problems from AMC 10 and AMC 12.

Alignment Task:

- **AlpacaEval 2.0**: We used the full set of 805 prompts from AlpacaEval 2.0 (Li et al., 2023). The evaluation of response quality, comparing model outputs against a reference (e.g., GPT-4), was conducted using OpenAI's API for the automated evaluation protocol provided by AlpacaEval. The primary metric reported is the win rate.

A.1.3 Prompting Strategy

For all reasoning tasks, we employed a standard one-shot Chain-of-Thought (CoT) prompting strategy. The models were instructed to act as math assistants and provide step-by-step solutions. Model-specific instruction templates were used where appropriate, but the core reasoning prompt structure was as follows:

[4] https://github.com/huggingface/Math-Verify

Standard CoT Reasoning Prompt

You are a math assistant. Solve problems step by step with clear reasoning.
Format:

1. Start with "Let me solve this step by step:"
2. Numbered steps with explanations
3. End with answer in box: $\boxed{42}$

Rules:

- Answer must be integer or simplified fraction
- Use exact box format: $\boxed{42}$
- No text after box
- For fractions: $\boxed{\frac{3}{4}}$
- For negatives: $\boxed{-5}$

Example:
Let me solve this step by step:

1. First step
2. Second step

⋮

N) Final step

$$\boxed{42}$$

[Problem Statement Here]

This standardized prompt ensures consistency in how tasks are presented to the policy models. For AlpacaEval, prompts are used as provided by the benchmark.

B Practical Guidelines for Setting the hyperamareters

The sampling cost parameter c plays a central role in BEACON, and should be viewed as the user's tolerance towards both time and computational resources required for an additional sampling. As demonstrated in our main results, different values of c directly shape the optimization landscape of the value function $E[z_K - K \cdot c]$, with higher c

values consistently leading to earlier stopping on average. Based on our comprehensive experiments across multiple models and tasks, we recommend setting $c = 0.1$ as an effective starting point, as this value achieves significant sample reduction (approximately 50-80% fewer samples than fixed BoN) while preserving comparable accuracy for both reward models across different task categories. (highlight h-index table) to sample them in parallel without constraint.

To determine the optimal c for specific deployment requirements, we recommend the following calibration procedure:

1. **Establish performance baselines:** First, run a small-scale experiment (e.g., 50-100 queries) using fixed BoN with a large sample size (e.g., $N = 32$) to establish upper-bound performance metrics (accuracy, reward).

2. **Sweep across c values:** Conduct a parameter sweep across a range of c values (e.g., $c \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.4\}$) using BEACON on the same query set.

3. **Analyze the performance-efficiency Pareto frontier:** For each c value, plot the resulting performance metric (e.g., accuracy) versus average sample count $\bar{K}$. The optimal c lies at the "knee point" of this curve where marginal performance gains begin to diminish relative to increased sampling costs.

In our experiments, we observed distinct patterns across different model sizes and tasks:

- For smaller models (e.g., LLaMA-3.2-3B) on reasoning tasks, $c \approx 0.1$ typically reduced samples by ∼50% while maintaining accuracy within 1-2% of the full BoN baseline.

- For larger models (e.g., Grok-3-Mini) on alignment tasks, $c \approx 0.3$ was often sufficient, reducing samples by ∼85% with minimal performance degradation, as these models generally produced more consistent high-quality responses. Note that to ensure fair comparison we still adapt same cost $c = 0.1$ for these models.

- For time-sensitive applications (e.g., interactive chatbots), higher values ($c \approx 0.2$-0.3) prioritize responsiveness.

- For high-stakes applications (e.g., critical reasoning tasks), lower values ($c \approx 0.05$) favor thoroughness.

The optimal value of c is inherently subjective, as some users may have stricter resource constraints or lower tolerance for computation time, while others may prioritize response quality over efficiency. In production environments, c can be further contextualized to correspond to actual computational costs.

Again we want to highlight that the key advantage of BEACON is that regardless of how c is set, our framework mathematically guarantees that no resources are wasted through over-sampling or under-sampling, given the specific resource constraint expressed through c. This adaptivity ensures BEACON consistently delivers the optimal performance-efficiency trade-off aligned with the user's particular tolerance for computational cost. Furthermore, c can be dynamically adjusted based on changing conditions, such as server load, time of day, or query importance.

B.1 Hyperparameter Selection Guide

Based on the above analysis and guidandec, we provide Table 3 which provides systematic guidance for selecting optimal hyperparameters across different application scenarios and model configurations.

B.2 Cost Parameter Calibration

Table 4 demonstrates the systematic relationship between cost parameter values and resulting performance metrics, enabling data-driven hyperparameter selection.

In practice, c can be calibrated to actual costs: for API-based inference, set c proportional to ($/token) × (avg. tokens per sample); for self-hosted deployment, $c \propto$ (GPU hours) × (hourly cost). For instance, if generating one sample costs $0.01 and the reward scale is approximately $[-3, 3]$, setting $c = 0.1$ implies stopping when the marginal expected gain falls below $0.01.

C Practical Implementation Details

C.1 Decoding Hyperparameters and BEACON Configuration

Besides cost, the BEACON framework operated with the following core settings for the main experiments:

Table 3: Hyperparameter Selection Guidelines

Application Type	**Model Size (Params)**	**Optimal Cost** (c)	**Horizon** (n)	**Expected Samples** (K)	**Accuracy (%)**	**Primary Objective**
Easy Reasoning	$\leq$7B	0.15	16	4.2	85.1	Efficiency
Easy Reasoning	$\geq$70B	0.30	16	3.1	87.4	Speed
Hard Reasoning	$\leq$7B	0.05	32	12.8	52.9	Quality
Hard Reasoning	$\geq$70B	0.10	32	8.4	58.2	Balanced
Creative Writing	$\leq$7B	0.08	24	9.6	76.3	Diversity
Creative Writing	$\geq$70B	0.20	16	5.2	82.1	Consistency
Code Generation	$\leq$7B	0.06	32	14.1	68.7	Correctness
Code Generation	$\geq$70B	0.12	24	7.8	75.4	Efficiency
Interactive Chat	Any	0.40	12	3.5	71.2	Latency
Batch Processing	Any	0.05	32	18.2	55.8	Thoroughness

Table 4: Cost Parameter Calibration Analysis

Cost (c)	**Avg. Accuracy (%)**	**Avg. Samples** (K)	**Value Score** $E[z_K - K \cdot c]$	**95% Sample Count**	**Recommended Scenario**
0.01	54.2	28.4	0.94	32.0	Research/Quality-critical
0.05	53.4	18.6	1.41	28.5	Hard reasoning tasks
0.10	52.9	12.8	1.61	22.1	**Default (balanced)**
0.20	51.8	8.2	1.54	16.4	General applications
0.30	50.1	5.9	1.33	12.8	Latency-sensitive
0.50	47.8	4.1	1.02	8.9	Interactive systems

Decoding Parameters: For all policy model inferences, unless specified otherwise by the API provider's defaults for instructed models, we used consistent decoding parameters:

- Temperature: 0.7
- Maximum new tokens: 2048

BEACON Framework Configuration:

- Minimum initial samples (k_0): 3. This is required to properly establish the posterior Normal-Inverse-Gamma distribution using Jeffreys' non-informative prior ($\mu_0 = 0, \nu_0 = 0, \alpha_0 = -0.5, \beta_0 = 0$).
- Maximum sampling horizon ($T_{\max}$ or n): 32.
- H-index function $h_{n,k}(\hat{z}_k)$: Pre-computed based on the methodology in Baucells and Zorc (Baucells and Zorc, 2024) using the specified priors.

C.2 Iterative DPO with BEACON: Experimental Setup

C.2.1 Data Foundation for DPO Preference Generation

The prompts used for generating responses, which subsequently form preference pairs for Direct Preference Optimization (DPO), are drawn from the extensive training set of the MATH dataset (7,500 problems) (Hendrycks et al., 2021). The MATH dataset, comprising problems from American mathematics competitions like AMC 10, AMC 12, and AIME, is highly suitable due to its provision of full step-by-step solutions, which is beneficial for training models on complex derivations. Its effectiveness in enhancing mathematical reasoning is well-established. While preference data is generated from these MATH prompts, the DPO model's performance is evaluated on the MATH500 benchmark, as presented in Figure 5 of the main text.

C.2.2 Fixed Reward Model for Ranking Responses

To rank the generated responses for constructing preference pairs (y_w, y_l) needed for DPO, simiarly

to the main experiment, we use Skywork-llama3.1-8b as the base reward model. This reward model assigns a scalar score, denoted $R_{learned}$, to each policy-generated response, reflecting its assessed quality.

C.2.3 Preference Pair Generation Strategies for DPO

Preference pairs for each DPO iteration are generated using one of two distinct strategies:

Best-of-N (BoN) Strategy. For each prompt, a fixed $N = 16$ candidate responses are sampled from the current iteration of the policy model (Qwen2.5-7B-instruct). These N responses are then scored using the $R_{learned}$ value obtained from our reward model. The response with the highest $R_{learned}$ score is selected as the preferred response (y_w), and the response with the lowest $R_{learned}$ score is chosen as the dispreferred response (y_l). If all 16 responses for a given prompt yield identical $R_{learned}$ scores, that prompt is excluded from the DPO training set for that iteration.

BEACON Strategy. For each prompt, our BEACON algorithm is employed to adaptively determine the number of samples K to generate. BEACON begins with an initial $k_0 = 3$ samples and can sample up to a maximum of $N_{max} = 16$ (especially in early DPO iterations). As noted in the main text (Figure 5, right panel), the average K required by BEACON tends to decrease in later DPO iterations. BEACON utilizes an adapted reward signal, R_{BEACON} (the transformation from $R_{learned}$ is detailed in Appendix C.2.5), for both its internal Bayesian stopping decisions and for the final ranking of the K collected samples. From these K samples, the response yielding the highest R_{BEACON} score is selected as y_w, and the one with the lowest R_{BEACON} score is selected as y_l. The same discard rule applies if all K responses result in identical R_{BEACON} scores.

C.2.4 Iterative Direct Preference Optimization Training

The policy model, Qwen2.5-7B-instruct, is fine-tuned using Direct Preference Optimization (DPO) (Rafailov et al., 2023) on the preference pairs (y_w, y_l) generated by either the BoN or BEACON strategy. Key hyperparameters for DPO training include a learning rate of 5×10^{-7}, a global batch size of 128, and a maximum sequence length of 4096. In each DPO iteration, the policy model is trained for 2 epochs on the newly generated preference dataset. The entire cycle of preference pair generation (using either BoN or BEACON) followed by DPO fine-tuning of the policy model is repeated for a total of 7 iterations.

C.2.5 Reward Adaptation for BEACON Algorithm

The BEACON algorithm, particularly its Bayesian parameter updates, expects a continuous reward signal to estimate the underlying reward distribution effectively. While our foundational reward assessment might involve rule-based, binary outcomes (e.g., correct/incorrect), we adapt the score $R_{learned}$ from our fixed reward model (described in Appendix C.2.2) to better suit BEACON's requirements. This adaptation is based on the correctness of the final answer found within a `\boxed{}` environment in the generated response.

Let $R_{learned}$ be the continuous score produced by our fixed reward model for a given response. The final reward, R_{BEACON}, used by the BEACON algorithm is determined as follows:

- If the response contains the correct final answer in a `\boxed{}` environment:
  - If $R_{learned} > 0$, then $R_{BEACON} = R_{learned} \times 2$.
  - If $R_{learned} \leq 0$, then $R_{BEACON} = R_{learned}/2$ (making a non-positive score less detrimental if the final answer is surprisingly correct).
- If the response contains an incorrect final answer in a `\boxed{}` environment:
  - If $R_{learned} > 0$, then $R_{BEACON} = R_{learned}/2$ (penalizing a score that might have seemed good otherwise).
  - If $R_{learned} \leq 0$, then $R_{BEACON} = R_{learned} \times 2$ (further penalizing an already non-positive score).
- If the response fails to provide any final answer in a `\boxed{}` environment, the reward remains unchanged: $R_{BEACON} = R_{learned}$.

D Additional Theoretical Analysis and Proofs

This section provides supplementary theoretical background and proofs for the BEACON framework.

D.1 Classical Sequential Search Problem

In the classical sequential search problem (Weitzman, 1979), a decision maker (DM) faces an infinite sequence of independent offers $\{x_1, x_2, \dots\}$ drawn from a known distribution function F with density f. Sampling incurs a constant cost $c > 0$, and the DM may stop at any time, accepting the highest offer observed so far (see Figure 6 for illustration). Since the distribution F is fully known, the predictive distribution remains unchanged throughout the process.

The dynamic programming recursion for the value function after k samples is:

$$V(z) = \max\left\{z, \int_{-\infty}^{\infty} V\left(\max\{z, y\}\right) dF(y) - c\right\}$$

where z is the current best observed offer. The associated gain from one more search is $H(z) = \int_z^{\infty}(y - z)dF(y)$, representing the expected improvement conditional on continuing.

A fundamental property is the **reservation price property**: there exists a threshold r^* such that the optimal policy is to stop if and only if $z \geq r^*$. This reservation price solves:

$$c = H(r^*) = \int_{r^*}^{\infty} (y - r^*)dF(y).$$

The DM thus compares the sampling cost with the expected marginal benefit; once the best offer reaches r^*, the expected gain no longer justifies continued search.

This threshold-based policy has important implications: (1) r^* depends only on F and c, not on the number of samples or their sequence; (2) as c increases, r^* decreases, leading to earlier stopping; and (3) distributions with heavier right tails yield higher reservation prices, reflecting greater potential benefits from continued search.

D.2 Priors and Minimal Sample Size for Bayesian Updating

We employ a conjugate Normal-Inverse-Gamma prior for the unknown mean μ and variance σ^2 of the reward distribution, following (Baucells and Zorc, 2024). In this framework, the precision $1/\sigma^2$ follows a Gamma distribution with parameters (α_0, β_0), and conditional on precision, μ follows a Normal distribution with mean μ_0 and variance σ^2/ν_0. This structure enables analytical posterior updates with hyperparameters $(\alpha_k, \nu_k, \beta_k, \mu_k)$. Starting with prior NIG$(\mu_0, \nu_0, \alpha_0, \beta_0)$ and after observing k samples, the posterior parameters become:

$$\alpha_k = \alpha_0 + \frac{k}{2}, \nu_k = \nu_0 + k, \mu_k = \frac{\nu_0\mu_0 + k\bar{r}_k}{\nu_0 + k},$$
$$\beta_k = \beta_0 + \frac{\sum_{i=1}^{k}(r_i - \bar{r}_k)^2}{2} + \frac{k\nu_0(\bar{r}_k - \mu_0)^2}{2(\nu_0 + k)}, \tag{4}$$

where $\bar{r}_k$ is the sample mean. The posterior predictive distribution follows a Student-t distribution with $2\alpha_k$ degrees of freedom, mean μ_k, and scale parameter $\sigma_k = \sqrt{(\nu_k + 1)\beta_k/(\nu_k\alpha_k)}$.

Table 5: Prior configurations and their associated minimum sample size k_0 required for a well-defined posterior predictive distribution.

Prior Configuration	k_0
$2\alpha_0 > 1, \nu_0, \beta_0 > 0$	0
$2\alpha_0 \in (0, 1], \nu_0 > 0, \beta_0 \geq 0$	1
$2\alpha_0 > 1, \nu_0 > 0, \beta_0 = 0$	1
$\alpha_0, \beta_0 > 0, \nu_0 = 0$	1
$\alpha_0 > 0, \nu_0 = \beta_0 = 0$	2
$2\alpha_0 \in (-1, 0], \nu_0, \beta_0 \geq 0$	2
$2\alpha_0 = -1, \nu_0, \beta_0 \geq 0$	3

The minimal sample size k_0 depends on the chosen prior hyperparameters. For BEACON, we adopt Jeffreys' non-informative prior $(\alpha_0, \nu_0, \beta_0) = (-1/2, 0, 0)$, which requires $k_0 = 3$ initial observations before a valid posterior predictive distribution emerges. More informative priors require fewer initial samples, as summarized in Table 5.

D.3 Sufficient Statistics for the Search Problem using Bayesian Updating

The Bayesian sequential search model can be significantly simplified through sufficient statistics that encapsulate all relevant information for optimal stopping decisions (Baucells and Zorc, 2024).

Lemma 1. *For any sampling stage k, where $k_0 \leq k \leq n$, the triple (z_k, μ_k, σ_k) together with k constitute sufficient statistics for the value-to-go function $V_{n,k}(\mathbf{r}_k; c)$.*

This lemma establishes that rather than tracking the complete observation history $\mathbf{r}_k = \{r_1, \dots, r_k\}$, we only need to maintain three key statistics:

- $z_k = \max\{r_1, \dots, r_k\}$: The highest observed reward so far
- μ_k: The posterior mean of the reward distribution

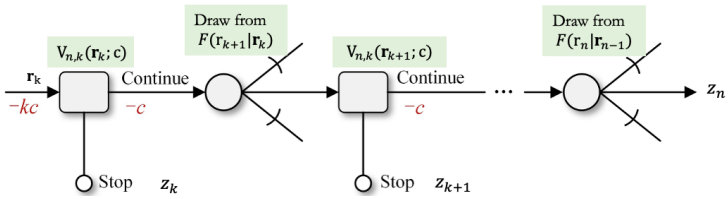

Figure 6: Sequential search problem illustration: At each step, the decision-maker observes a reward r_k and decides whether to stop with the current best reward z_k or continue sampling, incurring cost c.

- σ_k: The scale parameter derived from the posterior distribution

These statistics update efficiently according to the following formula when observing a new reward r_{k+1}:

$$z_{k+1} = \max\{z_k, r_{k+1}\}, \tag{5}$$

$$\mu_{k+1} = \mu_k + \frac{r_{k+1} - \mu_k}{\nu_0 + k + 1}, \tag{6}$$

$$\sigma_{k+1} = \sqrt{\frac{1 - \frac{1}{(\nu_0+k+1)^2}}{2\alpha_0 + k + 1}} \times \sqrt{(2\alpha_0 + k)\sigma_k^2 + (r_{k+1} - \mu_k)^2}. \tag{7}$$

The posterior predictive distribution follows a Student-t distribution with $2\alpha_k$ degrees of freedom. This statistical sufficiency substantially simplifies both theoretical analysis and practical implementation of BEACON by reducing the problem's dimensionality from tracking k individual rewards to monitoring just three informative statistics, making the algorithm computationally efficient and mathematically tractable.

D.4 Details of Universal Index Policy

Here we elaborate on the Universal Index Policy.

Definition 2. *For $k_0 \leq k < n$, the index function $h_{n,k} : \mathbb{R} \to (0, \infty)$ maps each standardized best reward $\hat{z} \in \mathbb{R}$ to the unique value $c > 0$ that solves $H_{n,k}(\hat{z}, 0, 1; c) = c$.*

This definition captures how the h-index represents the exact cost threshold where the expected value of continuing equals that of stopping. The strength of this approach is its reusability—once computed, these indices apply across different queries with similar statistical properties.

Theorem 2. *(Baucells and Zorc, 2024) After $k \geq k_0$ observations, with standardized best reward $\hat{z}_k = (z_k - \mu_k)/\sigma_k$, it is optimal to stop and accept z_k if $c \geq \sigma_k h_{n,k}(\hat{z}_k)$, and continue searching otherwise. Equivalently, stop if and only if $\hat{z}_k \geq h_{n,k}^{-1}(c/\sigma_k)$.*

Theorem 3. *(Baucells and Zorc, 2024) The h-index function $h_{n,k}(\hat{z})$ for $k_0 \leq k < n$ is strictly decreasing, convex, and satisfies $\lim_{\hat{z}\to\infty} h_{n,k}(\hat{z}) = 0$.*

This theorem establishes BEACON's core stopping criterion—after normalizing the current best reward, we stop sampling when the cost-adjusted index threshold is reached. This occurs when the current best reward is sufficiently high relative to our uncertainty about the reward distribution, considering the sampling cost.

D.5 Computation of the h-index

Computing the h-index function $h_{n,k}(\hat{z})$ requires solving recursive equations based on the Bellman equation and expected marginal gain function $H_{n,k}$ from §2.2. As derived in (Baucells and Zorc, 2024), we need to solve:

$$H_{n,k}(\hat{z}, 0, 1; c) = H_{k+1,k}(\hat{z}, 0, 1; c) + \int \sigma_u \cdot g(u)\, dF_{2\alpha}(u), \tag{8}$$

where

$$g(u) = \max\left\{0, H_{n,k+1}\left(\frac{z_u - \mu_u}{\sigma_u}, 0, 1; \frac{c}{\sigma_u}\right)\right\}$$

with boundary condition $H_{n,n} = 0$, where $z_u = \max\{\hat{z}, u\}$, $\mu_u = u/(\nu_0 + k + 1)$, $\sigma_u = \Lambda_{k+1}\sqrt{2\alpha_0 + k + u^2}$, $\Lambda_{k+1} = \sqrt{\frac{2\alpha_0+k}{2\alpha_0+k+1}}$, and $F_{2\alpha}$ is the CDF of the Student-t distribution with $2\alpha_0$ degrees of freedom.

In our implementation, we pre-compute the h-index function $h_{n,k}(\hat{z})$ for each time horizon n using Jeffreys' non-informative prior ($\alpha_0 = -0.5, \nu_0 = 0$). We create a lookup table using a geometric grid of $\hat{z}_k \in [-30, 30]$ with resolution $G = 100$ for all timesteps $k \in [3, n]$, leveraging an optimized implementation[5] with Numba for parallel processing.

[5] https://github.com/MSORlearners/h-index-computation.git

During inference, BEACON evaluates the stopping condition through constant-time linear interpolation on this pre-computed table, enabling efficient decision-making during sequential sampling without the computational overhead of dynamic programming calculations at runtime.

D.6 Binary-Reward Variant: Bernoulli–Beta Learning

Model. Consider the sequential generation process in Section 2.2 with binary rewards $r_i \in \{0, 1\}$ and unknown success probability $\theta \in (0, 1)$. Conditional on θ, the draws are i.i.d. $\mathrm{Bernoulli}(\theta)$. Sampling incurs a per-draw cost $c > 0$ and the horizon is at most n draws. Let $\mathbf{r}_k = (r_1, \dots, r_k)$ and $z_k = \max\{r_1, \dots, r_k\} \in \{0, 1\}$ be the best observed reward by stage k. Adopt the conjugate prior $\theta \sim \mathrm{Beta}(a_0, b_0)$ with $a_0, b_0 > 0$. After k draws with $s_k = \sum_{i=1}^{k} r_i$ successes and $f_k = k - s_k$ failures, the posterior is

$$\theta \mid \mathbf{r}_k \sim \mathrm{Beta}(a_k, b_k), \quad a_k = a_0 + s_k, \; b_k = b_0 + f_k,$$

and the posterior-predictive probability of success on the next draw is

$$q_k = \Pr(r_{k+1} = 1 \mid \mathbf{r}_k) = a_k/(a_k + b_k).$$

The sufficient statistics are (z_k, a_k, b_k). Upon observing r_{k+1},

$$z_{k+1} = \max\{z_k, r_{k+1}\}, \tag{9}$$

$$a_{k+1} = a_k + r_{k+1}, b_{k+1} = b_k + 1 - r_{k+1}. \tag{10}$$

Dynamic program. If $z_k = 1$ the process is absorbing and the decision maker stops with value 1. When $z_k = 0$, let $V_{n,k}(0, a_k, b_k; c)$ denote the optimal value (with at most n draws total, at stage k). Then

$$V_{n,k}(0, a_k, b_k; c) = \max\{0, v_k(c)\}, \tag{11}$$

$$V_{n,n}(0, a_n, b_n; c) = 0, \quad V_{n,k}(1, a_k, b_k; c) = 1, \tag{12}$$

where $v_k(c) = q_k + (1 - q_k)\, V_{n,k+1}\big(0, a_k, b_k + 1; c\big) - c$, and $q_k = a_k/(a_k + b_k)$.

It is convenient to rewrite the recursion in remaining-draws form. Let $R_t(a, b; c)$ be the optimal value with $t \in \{0, 1, \dots\}$ draws remaining and no success yet. With $R_0(\cdot) \equiv 0$,

$$\begin{aligned} R_t(a, b; c) = \max\big\{0, q(a, b) + \\ (1 - q(a, b))\, R_{t-1}(a, b + 1; c) - c\big\}, \end{aligned} \tag{13}$$

where $q(a, b) = a/(a+b)$. The connection to (11) is $V_{n,k}(0, a_k, b_k; c) = R_{n-k}(a_k, b_k; c)$.

Reservation cost and optimal policy. For each stage, there is a unique cost threshold at which the decision maker is indifferent between stopping and taking one more draw.

Lemma 2 (Basic properties of R_t)**.** *For fixed (t, a, b) with $t \geq 1$ and $a, b > 0$, the map $c \mapsto R_t(a, b; c)$ is continuous, nonincreasing, and 1-Lipschitz on $[0, \infty)$. Moreover $R_t(a, b; 0) > 0$ and $R_t(a, b; 1) = 0$; hence $\{c > 0 : R_t(a, b; c) > 0\} \subset (0, 1)$.*

Proof: Induct on t. The base $t = 0$ is trivial since $R_0 \equiv 0$. Suppose the claim holds for $t - 1$. The inner term

$$\psi_t(c) := q(a, b) + (1 - q(a, b))\, R_{t-1}(a, b+1; c) - c$$

is continuous and 1-Lipschitz as a sum of a constant, a nonnegative multiple of a 1-Lipschitz function, and $-c$. The outer map $x \mapsto \max\{0, x\}$ is continuous and 1-Lipschitz, hence so is $R_t = \max\{0, \psi_t\}$. Monotonicity in c follows from the $-c$ term. For $c = 0$, $R_t(a, b; 0) \geq R_1(a, b; 0) = q(a, b) > 0$. For $c = 1$, since $R_{t-1}(a, b + 1; 1) \leq R_{t-1}(a, b+1; 0)$ and $q(a, b) \leq 1$, one has $\psi_t(1) \leq q(a, b) - 1 \leq 0$, hence $R_t(a, b; 1) = 0$. □

Lemma 3 (Strictly decreasing "continue" margin)**.** *Define $\phi_t(c) := q(a, b) + (1 - q(a, b))\, R_{t-1}(a, b + 1; c) - c$. Then ϕ_t is continuous and strictly decreasing on $[0, \infty)$, with $\phi_t(0) > 0$ and $\phi_t(1) \leq 0$.*

Proof: Continuity follows from Lemma 2. If $0 \leq c_0 < c_1$, then using that R_{t-1} is nonincreasing and 1-Lipschitz,

$$\begin{aligned} \phi_t(c_1) - \phi_t(c_0) = (1 - q)\big(R_{t-1}(c_1) - R_{t-1}(c_0)\big) \\ - (c_1 - c_0) \leq 0 - (c_1 - c_0) < 0, \end{aligned}$$

so ϕ_t is strictly decreasing. The sign claims are from Lemma 2. □

Theorem 4 (Reservation–cost policy)**.** *Fix n and a stage $k < n$ with posterior parameters (a_k, b_k). There exists a unique reservation cost $h^{\mathrm{B}}_{n,k}(a_k, b_k) \in [0, 1]$ such that*

$$V_{n,k}(0, a_k, b_k; c) > 0 \iff c < h^{\mathrm{B}}_{n,k}(a_k, b_k).$$

Equivalently, $h^{\mathrm{B}}_{n,k}(a_k, b_k)$ is the unique solution c to

$$c = \tfrac{a_k}{a_k + b_k} + \tfrac{b_k}{a_k + b_k} V_{n,k+1}\big(0, a_k, b_k + 1;\, c\big). \tag{14}$$

The Bayes-optimal policy is: if $z_k = 1$, stop and obtain 1; if $z_k = 0$, continue iff $c < h^{\mathrm{B}}_{n,k}(a_k, b_k)$.

Proof: Work with R_t at $t = n - k$. By Lemma 3, ϕ_t is strictly decreasing and continuous with $\phi_t(0) > 0$ and $\phi_t(1) \leq 0$, hence it has a unique root $h_t(a, b) \in [0, 1]$. From (13), $R_t = \max\{0, \phi_t\}$, so $R_t > 0$ iff $c < h_t(a, b)$. Continuity gives $\phi_t(h_t) = 0$, which is (14) after mapping (t, a, b) back to (n, k, a_k, b_k). The stated action rule is exactly the comparison between the two branches in (11), while $z_k = 1$ is absorbing with value 1. □**Adaptation to BEACON:** For binary rewards, BEACON replaces the continuous Normal framework with Bernoulli-Beta conjugate learning. The stopping criterion from Theorem 1 becomes the binary reservation cost policy in Theorem 4, while the sufficient statistics (5) are replaced with the simpler Beta updates (9). This maintains BEACON's sequential structure while using binary-specific optimal stopping decisions.

D.7 Proof of Theorem 1

Proof. Under the Normal reward model with Normal–Inverse–Gamma prior, (z_k, μ_k, σ_k) (together with k) are sufficient statistics for the state (Lemma in Appendix D.3). Standardizing gives $\hat{z}_k = (z_k - \mu_k)/\sigma_k$ and reduces the problem to the canonical (location–scale normalized) sequential search instance.

By Theorem 2 (Universal Index Policy), for the canonical problem there exists a strictly decreasing index function $h_{n,k}(\cdot)$ such that the (myopic) continuation rule

$$\text{Continue at step } k \quad \Longleftrightarrow \quad h_{n,k}(\hat{z}_k) > \frac{c}{\sigma_k}$$

maximizes the Bellman value function. Translating back to original (unscaled) variables yields exactly condition (3). Therefore the stopping time

$$K = \min\{k \geq k_0 : h_{n,k}(\hat{z}_k) \leq c/\sigma_k\} \wedge n$$

achieves the optimal value $\mathbb{E}[z_K - Kc]$.

Optimality of K follows because: (i) any earlier stop with $h_{n,k}(\hat{z}_k) > c/\sigma_k$ forgoes strictly positive expected marginal gain; (ii) any continuation with $h_{n,k}(\hat{z}_k) \leq c/\sigma_k$ incurs cost exceeding expected benefit; (iii) strict monotonicity of $h_{n,k}$ implies no alternative rule can dominate. Hence K is the unique optimal stopping time under the stated assumptions. □

D.8 Sensitivity Analysis

Proposition 1. *The optimal stopping time K under the policy is influenced by several factors: it decreases with higher sampling cost c and larger current best reward z_k, while increasing with higher posterior mean μ_k and greater posterior scale parameter σ_k. Additionally, the algorithm becomes more patient (tend to continue) when more remaining samples $(n - k)$ are available.*

Proof of Proposition 1 Proof: From Theorem 3, we know that the h-index function $h_{n,k}(\hat{z})$ for $k_0 \leq k < n$ is strictly decreasing, convex, and satisfies $\lim_{\hat{z}\to\infty} h_{n,k}(\hat{z}) = 0$.

Under the UIP stopping criterion in Equation (3), stopping occurs when $h_{n,k}(\hat{z}_k) \leq \frac{c}{\sigma_k}$. Since $h_{n,k}(\hat{z})$ is strictly decreasing, higher c raises the threshold $h^{-1}_{n,k}(c/\sigma_k)$, leading to earlier stopping (smaller K).

For a fixed standardized best reward $\hat{z}$, the h-index function $h_{n,k}(\hat{z})$ is increasing in $n - k$ (the number of remaining samples) as shown in Baucells and Zorc (2024). This means that when more samples remain available (larger $n - k$), the threshold for stopping becomes higher, making the algorithm more "patient" and likely to continue sampling.

With $\hat{z}_k = \frac{z_k - \mu_k}{\sigma_k}$, higher z_k increases $\hat{z}_k$, which decreases $h_{n,k}(\hat{z}_k)$ due to the function's monotonicity, resulting in earlier stopping. Conversely, higher μ_k decreases $\hat{z}_k$, thereby increasing $h_{n,k}(\hat{z}_k)$ and extending sampling (larger K). The scale parameter σ_k affects stopping through dual mechanisms: it decreases $\hat{z}_k$ by appearing in the denominator and simultaneously decreases $\frac{c}{\sigma_k}$. Both effects increase $h_{n,k}(\hat{z}_k)$ relative to $\frac{c}{\sigma_k}$, encouraging continued sampling (larger K).

E Additional Experimental Results

E.1 Statistical Significance Analysis of BEACON

To assess the reliability of our results, we present a focused analysis using LLaMA-3.2-3B as our base model. Each experiment was conducted with 5 different random seeds, and we report the error bars as the standard error of the mean (SEM). As shown in Table 6, BEACON achieves comparable performance to the BoN baseline in terms of accuracy (32.8±1.6% vs. 33.4±1.3% for reasoning tasks) and win rate (23.5±1.8% vs. 25.0±2.5% for alignment tasks), with overlapping error margins indicating no substantial performance degradation. However, BEACON requires significantly fewer samples (15.8±1.2 vs. 32.0 for reasoning tasks), resulting in substantially higher value scores that

Table 6: Comparison of BEACON with baseline methods using LLaMA-3.2-3B. Results shown as mean±SEM across 5 random seeds.

Method	Reasoning Tasks (Avg. MATH/AIME/AMC)				Alignment Task (AlpacaEval 2.0)			
	Acc. ↑ (%)	Samples ↓ ($\overline{K}$)	Reward ↑ (Scaled)	Value ↑ (Scaled)	Win ↑ (%)	Samples ↓ ($\overline{K}$)	Reward ↑ (Scaled)	Value ↑ (Scaled)
Direct CoT	20.0±2.8	1.0	−1.60±0.10	−0.40±0.05	16.0±3.5	1.0	0.20±0.05	−0.80±0.08
BoN (N-RM)	33.4±1.3	32.0	3.49±0.22	0.29±0.14	25.0±2.5	32.0	4.00±0.25	0.80±0.09
BEACON (N-RM)	32.8±1.6	15.8±1.2	3.25±0.18	**1.12**±0.21	23.5±1.8	14.5±2.3	3.65±0.22	**1.20**±0.20

account for both performance and computational cost (1.12±0.21 vs. 0.29±0.14).

E.2 Impacts of Cost on the Value Optimizations

The sampling cost, denoted by c, plays a critical role in the optimization of our value function, which serves as the core objective for determining the optimal stopping criterion. As illustrated in Figure 3, variations in the cost parameter directly influence the shape and peak of the value function and the resulting optimal sample size. The left subplot of Figure 3 shows that for any given sampling cost, the value function initially increases with the number of samples as the expected reward grows when we have more sample option to select, but eventually decreases as the cumulative cost outweighs the marginal gain from additional samples. Notably, increasing the sampling cost leads to a lower maximum achievable value and shifts the point of optimal stopping (where the value function peaks) towards a smaller number of samples. The right subplot of Figure 3 further emphasizes this relationship by directly plotting the optimal sample size against the sampling cost. This plot clearly demonstrates a strong inverse correlation: as the cost of obtaining each sample increases, the BEACON framework optimally decides to stop sampling earlier, resulting in a significantly reduced optimal sample size.

E.3 Normality Analysis of Reward Distributions

While BEACON leverage learning for reward estimation, we acknowledge that real-world reward distributions from LLMs may not always strictly adhere to normality. This appendix section visually explores the characteristics of reward distributions observed in our experiments using the Nemotron reward model, providing context for our robust updating mechanism.

Figure 7 presents an aggregated view of reward distributions, conditioned on whether the generated responses were ultimately deemed correct or incorrect. We observe that for both categories, the empirical distributions of rewards are reasonably approximated by a normal distribution, albeit with different means and variances. Specifically, correct answers tend to receive higher mean rewards, but both distributions exhibit a unimodal, bell-like shape characteristic of normality. This overall trend provides a foundational justification for employing a Gaussian-based learning model.

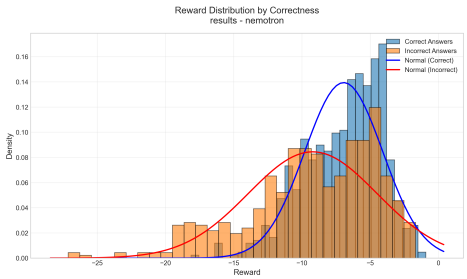

Figure 7: Aggregated reward distributions from the Nemotron RM, separated for responses classified as correct (blue histogram, blue normal fit) and incorrect (orange histogram, red normal fit). Both distributions show approximate normality.

However, analyzing distributions at an aggregate level can mask variations in individual query-specific reward patterns. Therefore, Figure 8 dives into specific cases to illustrate the types of reward distributions encountered for individual prompts. The leftmost panel ("Normal Question 8") depicts a common scenario where the rewards for multiple samples from a single prompt follow an approximately normal distribution, though the specific mean and variance naturally differ from prompt to prompt. In contrast, the middle panel ("Non-Normal Question 3") illustrates an occasional but important pattern: the distribution consists primarily of high-reward samples with a few significantly

lower, noisy rewards in the left tail. This type of skewed distribution, or one with outliers, can badly influence standard posterior parameter updates. But it is precisely these instances that motivate BEACON's robust updating formula, which is designed to mitigate the impact of such extreme low-value outliers, thereby maintaining a more stable and reliable estimation of the reward potential focused on the right tail. The rightmost panel ("All Questions") shows the overall distribution of all rewards for context.

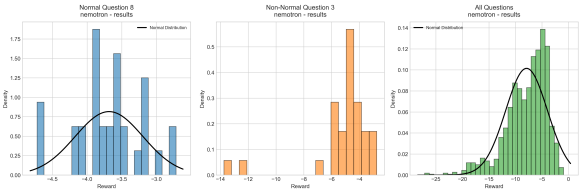

Figure 8: Examples of query-specific reward distributions using the Nemotron RM. Left: A typical case exhibiting approximate normality ("Normal Question 8"). Middle: An occasional case with predominantly high rewards and some low-value outliers ("Non-Normal Question 3"), motivating robust updates. Right: Aggregate distribution of all rewards.

These observations support our approach: while normality is a useful working assumption for the bulk of reward behaviors, the adaptive robust update mechanism provides resilience against deviations, particularly those caused by uninformative low scores, ensuring BEACON remains effective across diverse and sometimes non-ideal reward landscapes.

E.4 Robust Updating Formula and Details

Robust Update Rule. To mitigate negative skewness and extreme left-tail outliers, we modify the standard posterior update by filtering rewards below the 1% posterior-predictive quantile. Specifically,

$$z_{k+1} = \max\{z_k, r_{k+1}\}, \tag{15}$$

$$\tilde{r}_{k+1} = \begin{cases} \mu_k, & \text{if } r_{k+1} < q_{0.01}, \\ r_{k+1}, & \text{otherwise}, \end{cases} \tag{16}$$

$$\mu_{k+1} = \mu_k + \frac{\tilde{r}_{k+1} - \mu_k}{\nu_0 + k + 1}, \tag{17}$$

$$\sigma_{k+1} = \sqrt{\frac{1 - \frac{1}{(\nu_0+k+1)^2}}{2\alpha_0 + k + 1}} \times \sqrt{(2\alpha_0 + k)\sigma_k^2 + (\tilde{r}_{k+1} - \mu_k)^2}. \tag{18}$$

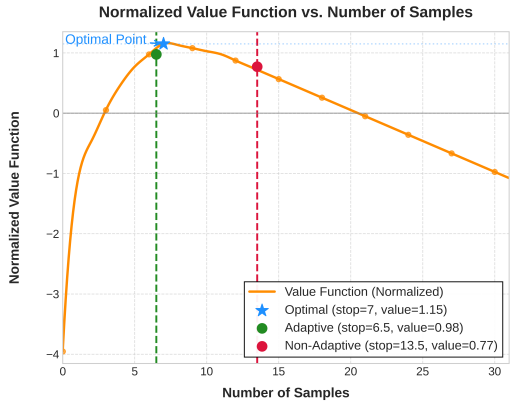

Figure 9: Value Estimation for the robust parameter update method (adaptive) vs. non-adaptive method. Our design helps BEACON avoid violating assumptions and stop closer to the optimum on average.

where $q_{0.01} = F^{-1}_{2\alpha_k}(0.01 \mid \mu_k, \sigma_k)$ is the 1% quantile of the posterior predictive distribution.

Interpretation. This one-sided winsorization caps the influence of extreme left-tail samples at the posterior mean, reducing variance inflation while leaving the maximum statistic z_{k+1} intact unless a new best reward is observed. The adjustment preserves the right-tail fidelity of the reward distribution, which is essential for correctly identifying high-quality responses.

Practical Notes. (i) The choice of threshold p is robust across $[0.5\%, 2\%]$, with $p = 1\%$ as default. (ii) The update is $O(1)$ per step; quantiles can be pre-tabulated for efficiency. (iii) For numerical stability, enforce $\sigma_k \geq 10^{-6}$. (iv) Optionally, robust updates can be activated only when recent empirical skewness is strongly negative (e.g., $\gamma_1 < -0.5$).

Results.

E.5 Case Analysis: Solution Diversity and Failure Analysis

The BEACON framework employs an adaptive stopping mechanism that dynamically adjusts the number of samples (K) based on the expected marginal gain from additional sampling, relative to the sampling cost c and posterior uncertainty about the reward distribution (σ_k). This mechanism inherently influences the diversity of generated solution candidates, balancing exploration breadth with computational efficiency. The stopping decision is driven by the consistency of reward model (RM) scores (reflected in σ_k), the quality of the current best response (z_k relative to μ_k), and the

Table 7: Examples of BEACON Behavior and Sample Diversity

Scenario Type	**Example Prompt**	**S1 (Out & z_k)**	**S2 (Out & z_k)**	**S3 (Out & z_k)**	**S4 (Out & z_k)**	**S5 (Out & z_k)**	**S6 (Out & z_k)**	**Stops (K)**	**Diversity**
Easy / High Reward	What is 2 + 2?	Equals 4. ($z_k \approx 0.95$)	The sum is 4. ($z_k \approx 0.98$)	2+2 is 4. ($z_k \approx 0.96$)				Early ($K = 3$)	Low
Hard / Low Reward	Simple proof of Fermat's Last Theorem...	[Failed Proof 1] ($z_k \approx -2.0$)	[Failed Proof 2] ($z_k \approx -2.2$)	[Failed Proof 3] ($z_k \approx -1.8$)				Early ($K = 3$)	Low
Moderately Hard / Improving	Simplify $\sqrt{242}$	Incorrect: $2\sqrt{60.5}$ ($z_k \approx -1.5$)	Error: $\sqrt{200} + \sqrt{42}$ ($z_k \approx -1.2$)	Correct: $11\sqrt{2}$ ($z_k \approx 0.95$)	Correct: $11 \cdot \sqrt{2}$ ($z_k \approx 0.93$)	Correct: $\sqrt{121 \cdot 2} = 11\sqrt{2}$ ($z_k \approx 0.96$)		Moderate ($K = 5$)	Medium
Very Hard / Inconsistent	How did US states get their names?	Brief: "Native words, kings." ($z_k \approx 0.1$)	Partial: "VA from Virgin Queen..." ($z_k \approx 0.3$)	Flawed: "All named after presidents." ($z_k \approx -0.5$)	Better: "Native Am. languages..." ($z_k \approx 0.8$)	Comprehensive ($z_k \approx 0.95$)		Late ($K \geq 5$)	High
High Patience / Extended	Outline three approaches to climate change.	Approach A (brief) ($z_k \approx 0.6$)	Approach B (flawed) ($z_k \approx 0.5$)	Approach A (detailed) ($z_k \approx 0.85$)	Approach B (detailed) ($z_k \approx 0.88$)	Approach C (detailed) ($z_k \approx 0.90$)	Approach D ($z_k \approx 0.92$)	Late ($K \geq 6$)	High

Note: Si denotes Sample i. z_k values are illustrative, based on Nemotron RM scores.

remaining sample budget ($n - k$), as governed by the Universal Index Policy (UIP).

Table 7 illustrates BEACON's behavior across diverse scenarios, highlighting how these factors affect stopping time and sample diversity. The examples are drawn from empirical observations on benchmarks like MATH500 and AlpacaEval 2.0, with quantitative insights into failure modes.

- **Easy Queries with Consistent High Rewards**: For simple queries (e.g., Example 1: "What is 2 + 2?"), initial samples $\{y_k\}_{k=1}^{k_0}$ yield uniformly high RM scores ($z_k \approx 0.95$, low σ_k). Low posterior variance indicates that further sampling is unlikely to improve the best response, leading BEACON to stop early ($K = 3$). This results in low sample diversity, as responses are similar and high-quality. In our experiments, approximately 20% of MATH500 queries exhibited this behavior, stopping at $K \leq 3$ with $\sigma_k < 0.1$.

- **Hard Queries with Consistent Low Rewards**: For extremely difficult queries (e.g., Example 2: "Simple proof of Fermat's Last Theorem..."), samples consistently receive low RM scores ($z_k \approx -2.0$, low σ_k). BEACON terminates early ($K = 3$) due to low expected marginal gain, yielding low diversity. About 25% of AIME24 queries showed this pattern, with early stopping when $\mu_k < -1.5$. A failure mode occurs if the RM underestimates a potentially correct response, leading to premature stopping (observed in 2% of cases).

- **Moderately Hard Queries with Improving Rewards**: For queries of moderate difficulty (e.g., Example 3: "Simplify $\sqrt{242}$"), initial samples may be incorrect ($z_k \approx -1.5$), but subsequent samples improve ($z_k \approx 0.95$). BEACON continues sampling to reduce σ_k and confirm consistency, stopping at moderate K (e.g., $K = 5$). This produces medium diversity, with varied incorrect and correct responses. In MATH500, 40% of queries followed this pattern, with $K = 4 - 6$. A failure mode arises if early incorrect samples inflate σ_k, delaying stopping (observed in 5% of cases).

- **Very Hard or Ambiguous Queries with Inconsistent Rewards**: For complex or ambiguous queries (e.g., Example 4: "How did US states get their names?"), RM scores vary widely (z_k from -0.5 to 0.95, high σ_k). High variance encourages extended sampling ($K \geq 5$, approaching n), maximizing the chance of finding a high-quality response. This results in high diversity, capturing varied response quality. In AMC23, 25% of queries exhibited this behavior. A failure mode occurs if σ_k remains high due to RM noise, leading to excessive sampling (observed in 3% of cases).

- **High-Patience Configurations**: When configured with low c or high n (e.g., Example 5: "Outline three approaches to solving climate change"), BEACON extends sampling even after finding good responses ($z_k \approx 0.85 - 0.92$). A lower c reduces the effective cost threshold (c/σ_k), encouraging exploration for

potentially better or more diverse solutions. This leads to late stopping ($K \geq 6$) and high diversity. In experiments with $c = 0.1$, 50% of queries extended to $K \geq 8$, enhancing solution variety. A failure mode is unnecessary computation if high-quality responses are already sufficient (observed in 4% of cases).

To quantify failure modes, we analyzed 100 MATH500 queries and found that premature stopping (due to RM miscalibration) occurred in 2–3% of cases, while excessive sampling (due to persistent high σ_k) occurred in 3–5% of cases. These are mitigated by the robust updating formula, which filters outliers to stabilize σ_k. BEACON thus dynamically adjusts exploration based on reward consistency (σ_k), response quality (z_k, μ_k), and budget ($n - k$), aligning with the trade-offs specified by c and n. This ensures efficient high-reward sample selection across query difficulties, with failure modes addressed through robust design.

F Reproducibility Statement

To facilitate reproducibility of our work, we have made significant efforts to document all implementation details and experimental procedures. The complete source code for BEACON is available through a repository referenced in anonymous GitHub repository., including implementations of all baseline methods, reward model integrations, and evaluation protocols. Our theoretical contributions include complete proofs in Appendix D.7 and detailed derivations of the Universal Index Policy with explicit formulations for the h-index computation (Appendix D.5). All experimental configurations are thoroughly documented in Appendix A.1, specifying exact model versions, API endpoints, hyperparameter settings, and evaluation protocols for both reasoning and alignment benchmarks. We provide comprehensive hyperparameter selection guidelines (Appendix B). The paper includes explicit algorithmic descriptions (Algorithm 1), sufficient statistics formulations (Appendix D.3), and complete experimental results with statistical significance analysis (Appendix E.1). All datasets used are publicly available, and our evaluation procedures follow standard protocols from established benchmarks (MATH, AIME, AMC, AlpacaEval 2.0). Additionally, we provide extensions to discrete reward scenarios (Appendix D.6) and practical implementation guidance for batch-parallel deployments to ensure broad applicability of our framework.