BEACON: BAYESIAN OPTIMAL STOPPING FOR EFFICIENT LLM SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sampling multiple responses is a common technique for improving the quality of LLM outputs, but it comes at the cost of additional computation resources. Determining when to stop generating further samples therefore requires balancing response quality against efficiency. Existing methods typically rely on heuristics rather than theoretically grounded optimization, leading to either inefficient under-exploration or wasted resources through oversampling. We introduce Bayesian Efficient Adaptive Criterion for Optimal N-stopping (BEACON), a principled adaptive sampling framework grounded in the theory of Sequential Search with Bayesian Learning. BEACON makes sampling decisions in a Bayesian-optimal manner by sequentially generating samples from the policy LLM and updating a posterior belief over reward distributions of the response given the query. The framework determines optimal stopping points by balancing reward consistency with computational cost, terminating when the expected marginal utility of further exploration no longer justifies the expense. We establish theoretical optimality guarantees for BEACON and computational complexity analysis showing the computational tractability. Empirical results on diverse reasoning and alignment benchmarks show that BEACON reduces average sampling requirements by up to 80% compared to baselines while matching or exceeding response quality. Beyond benchmark performance, we extend BEACON's applicability to cost-efficient preference data generation, provide principled guidance for hyperparameter selection, and present extensions to batch sampling—offering actionable insights for practitioners and laying a foundation for future work on adaptive sampling.

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 failing on queries they could solve correctly across different invocations (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 primarily rely 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., 2024a; 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., 2021b; 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.

While principled approaches based on **optimal stopping theory** exist for sequential decision problems (Ferguson, 2012), they have remained computationally intractable for real-time applications with unknown reward distributions. To bridge this theory-practice gap, we leverage recent computational advances in Bayesian optimal stopping (Baucells & Zorc, 2024) and reformulate LLM sampling as a sequential search problem to make principled adaptive sampling computationally feasible for practical application under the context of LLM settings. This framework ensures stopping decisions achieve **Bayesian optimality** given currently observed data, eliminating reliance on heuristic approximations (Rothschild, 1974). Within this paradigm, rather than learning reward distributions

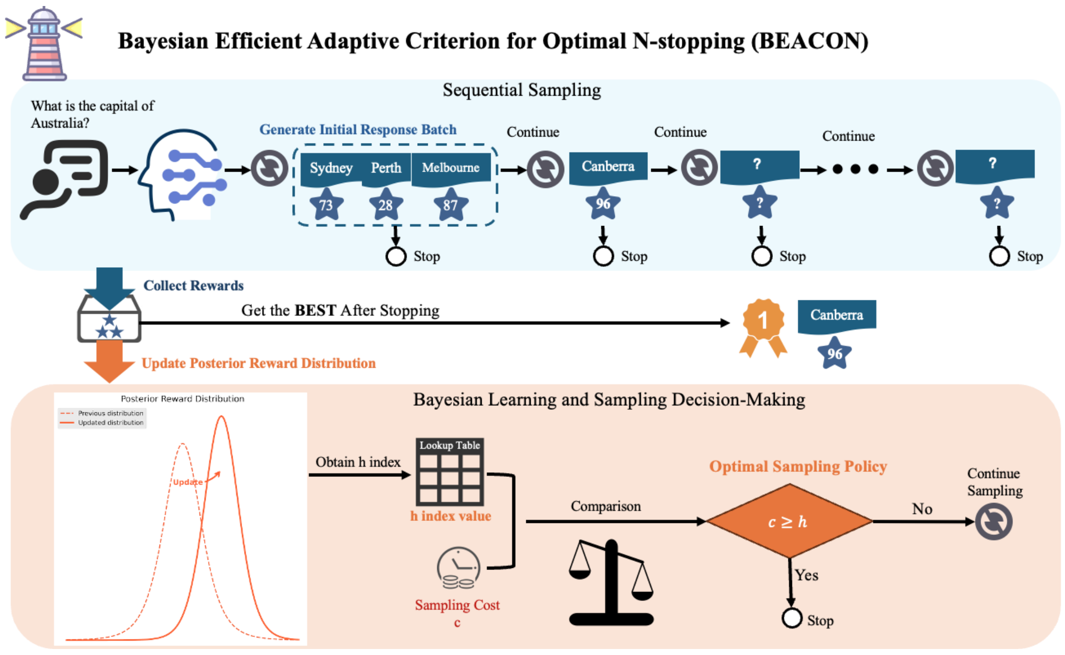
Bayesian Efficient Adaptive Criterion for Optimal N-stopping (BEACON)
Sequential Sampling
What is the capital of Australia?
Generate Initial Response Batch
Sydney
Perth
Melbourne
73
28
87
Continue
Canberra
96
Continue
?
Continue
Stop
Collect Rewards
Get the BEST After Stopping
Update Posterior Reward Distribution
Bayesian Learning and Sampling Decision-Making
Posterior Reward Distribution
Previous distribution
Updated distribution
Update
Obtain h index
Lookup Table
h index value
Sampling Cost c
Optimal Sampling Policy
Comparison
c ≥ h
No
Yes
Continue Sampling
Stop

Figure 1: **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.

offline, we conceptualize them as latent processes amenable to online updating: each generated response reveals information about the underlying reward distribution while incurring computational costs (Toth & 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 & Sharma, 1996).

We therefore introduce Bayesian Efficient Adaptive Criterion for Optimal N-stopping (**BEACON**), a novel adaptive sampling framework that makes theoretically 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**. As illustrated in Figure 1, BEACON sequentially collects responses and updates sufficient statistics of the posterior reward distribution, then 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 computational investment for additional samples exceeds the likelihood of discovering superior rewards. Our empirical evaluations on mathematical reasoning and alignment benchmarks demonstrate that BEACON substantially reduces average inference costs by up to 80% 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 summary, our main contributions are: (**1**) introducing **BEACON**, a principled adaptive sampling framework that achieves **theoretically optimal stopping decisions** by reformulating LLM sampling as a sequential search problem with Bayesian learning, enabling real-time optimization without pre-training requirements; (**2**) establishing theoretical optimality guarantees for our approach while making previously intractable Bayesian optimal stopping computationally feasible through efficient index-based policies; and (**3**) demonstrating substantial **efficiency gains** across diverse benchmarks—reducing sampling requirements by up to 80% while maintaining response quality—with comprehensive analysis of practical extensions including preference data generation, batch parallelization, and deployment guidelines for real-world applications.

2 Preliminaries: Sequential Search Problem with Learning

Sequential Search involves examining alternatives one by one, with the decision-maker choosing after each observation whether to accept the current best outcome or continue sampling (Stigler, 1961; Weitzman, 1979; Ferguson, 2006) (detailed in Appendix D.1). Given observed rewards $\mathbf{r}_k = \{r_1, \ldots, r_k\}$ with best reward so far $z_k = \max\{r_1, \ldots, r_k\}$ and a sampling cost c per observation, the decision-maker must determine whether to stop or continue. Unlike Multi-Armed Bandits that maximize cumulative rewards, sequential search aims to identify the maximum reward while minimizing sampling costs. When the reward distribution is *known*, this problem admits closed-form solutions based on reservation values (Weitzman, 1979). However, LLM sampling scenarios present a more challenging setting where the underlying reward distribution is *unknown*.

Sequential Search with Learning addresses cases where distribution parameters must be inferred during sampling. Here, Bayesian learning updates the posterior predictive distribution after each observation, with stopping decisions based on these evolving beliefs. The objective remains maximizing the expected net gain, $\mathbb{E}[z_K - K \cdot c]$, where K is the stopping time, while learning about the underlying distribution. The uncertainty about distribution parameters makes deriving tractable optimal policies difficult, which typically necessitates approximation methods.

Bayesian Learning provides the foundation for sequential search with unknown distributions. Computational tractability requires conjugate priors that enable closed-form posterior updates. For continuous distributions, the Normal distribution emerges as the natural choice due to its practical utility and theoretical properties—it represents the only continuous distribution case with computationally efficient optimal index policies in sequential search literature (Baucells & Zorc, 2024). Alternative approaches typically require simulation-based methods like MCMC, creating computational overhead that limits real-time deployment in time-sensitive applications.

3 Modeling LLM Sampling as Sequential Search Problem

3.1 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 frame our problem as a sequential search with optimal stopping to maximize net gain—balancing the highest reward against sampling cost. This enables principled decisions about when additional samples are no longer economically justified, replacing heuristics or learned approximations with theoretically grounded guarantees.

Handling Uncertainty with Bayesian Learning. Our Bayesian approach addresses the challenge of second-level uncertainty, the uncertainty about the reward distribution, by learning these parameters online during sampling. Unlike methods requiring offline training or domain-specific calibration, this enables immediate zero-shot deployment to new domains as the model adapts to each task's reward landscape through posterior updates.

A Principled Framework. By combining *sequential search* with *Bayesian learning*, BEACON enables adaptive sampling without pre-training requirements. While our approach supports various distributions with conjugate priors, we focus on the continuous Normal distribution for its mathematical properties and practical utility. The framework naturally extends to other conjugate families (e.g., Beta-Binomial and Dirichlet). We select the Normal distribution as it represents a more challenging theoretical framework than discrete cases while remaining widely practical for real-world reward modeling. See our extension to beta-binomial reward signals in Appendix D.6. Recent advances (Baucells & Zorc, 2024) have made optimal stopping with unknown Normal distributions computationally tractable, providing an implementable *index policy* with theoretical optimality guarantees.

3.2 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 is a critical assumption that distinguishes our approach from methods requiring known distributions or extensive pretraining data. 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 sequential sampling (Diaconis & Ylvisaker, 1979), essential for computational tractability. Starting with prior NIG$(\mu_0, \nu_0, \alpha_0, \beta_0)$ and after observing k samples, the posterior parameters become:

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

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 $\sigma_k = \sqrt{(\nu_k + 1)\beta_k/(\nu_k\alpha_k)}$.

Sufficient Statistics for Tractable Implementation. After observing $k \geq k_0$ samples (where k_0 is the minimum for well-defined posteriors; see Appendix D.2), the triple (z_k, μ_k, σ_k) forms sufficient statistics for all observed rewards (see Appendix D.3 for more details and the updating formula equation 6). This enables formulating our objective function $\mathbb{E}[z_K - K \cdot c]$ in Bellman equation as:

$$V_{n,k}(z_k, \mu_k, \sigma_k; c) = \max \{z_k, \mathbb{E}[V_{n,k+1}(z_{k+1}, \mu_{k+1}, \sigma_{k+1}; c) \mid z_k, \mu_k, \sigma_k] - c\}, \tag{2}$$

with corresponding optimal stopping rule:

$$\text{Stop iff} \quad H_{n,k}(z_k, \mu_k, \sigma_k; c) = \mathbb{E}[V_{n,k+1}(z_{k+1}, \mu_{k+1}, \sigma_{k+1}; c) \mid z_k, \mu_k, \sigma_k] - z_k \leq c. \tag{3}$$

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

3.3 Optimal Sampling Policy

Building on recent theoretical advances of the computationally efficient Universal Index Policy (Baucells & 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:*

$$h_{n,k}\left(\frac{z_k - \mu_k}{\sigma_k}\right) > \frac{c}{\sigma_k} \tag{4}$$

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.

3.4 Normality Assumption and Robust Parameter Updating

Normality Assumption. While BEACON builds on Gaussian learning principles, real-world reward distributions may deviate from normality. Our approach remains robust for several reasons: (1) reward models typically produce approximately normal distributions through preference aggregation; (2) by the central limit theorem, reward signals represent many additive uncertainties, naturally yielding normal-like distributions; (3) our focus on identifying maximum rewards depends primarily on the distribution's right tail; and (4) we implement a robust updating mechanism to handle outliers when distributions exhibit negative skewness.

Robust Updating for Negative Skewness. In practice, reward distributions may exhibit negative skewness with occasional extremely low outliers that distort posterior beliefs. We introduce a robust updating formula that preserves the distribution's right tail characteristics while mitigating outlier influence. Since our goal is identifying maximum rewards, we filter values below the 1% quantile of the current posterior by replacing them with the posterior mean. This maintains the integrity of high-quality candidates while ensuring the posterior better represents the reward distribution's right tail. Building on formula equation 6, our robust approach is:

$$\begin{aligned} z_{k+1} &= \max\{z_k, r_{k+1}\}, \quad \mu_{k+1} = \mu_k + \tfrac{\tilde{r}_{k+1} - \mu_k}{\nu_0 + k + 1}, \\ \sigma_{k+1} &= \sqrt{\tfrac{1 - 1/(\nu_0 + k + 1)^2}{2\alpha_0 + k + 1}} \cdot \sqrt{(2\alpha_0 + k)\sigma_k^2 + (\tilde{r}_{k+1} - \mu_k)^2}, \end{aligned} \tag{5}$$

where the filtered reward $\tilde{r}_{k+1} = \begin{cases} \mu_k & \text{if } r_{k+1} < q_{0.01} \\ r_{k+1} & \text{otherwise} \end{cases}$, with $q_{0.01} = F^{-1}_{2\alpha_k}(0.01|\mu_k, \sigma_k)$ representing the 1% quantile of the current posterior predictive distribution. Appendix E.3 provides empirical validation of the normality assumption, while §5.1 demonstrates this mechanism's effectiveness.

3.5 Sensitivity Analysis of Optimal Sample Size.

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.*

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
1  $(\alpha_0, \nu_0, \beta_0, \mu_0) \leftarrow (-0.5, 0, 0, 0)$                      ▷ Non-informative prior
2  $h\text{-}table \leftarrow$ PrecomputeTable$(n, G, \alpha_0, \nu_0)$                  ▷ Pre-compute as in §D.5
// Step 2: Generate initial samples and compute baseline parameters
3  Generate $\{y_1, y_2, y_3\} \sim \pi_\phi(y|x)$ and compute $r_i \leftarrow R(x, y_i)$ for $i \in \{1,2,3\}$
4  $(\alpha, \nu, \mu, \beta) \leftarrow$ UpdateHyperParams$(r_1, r_2, r_3, \alpha_0, \nu_0, \beta_0, \mu_0)$
5  $z \leftarrow \max\{r_1, r_2, r_3\}$, $\sigma \leftarrow \sqrt{\frac{(\nu+1)\beta}{\nu\alpha}}$, $k \leftarrow 3$
// Step 3: Adaptive sampling loop
6  while $k < n$ do
7      $\hat{z} \leftarrow (z - \mu)/\sigma$, $h \leftarrow$ LookupHIndex$(h\text{-}table, k, \hat{z})$
8      if $h \le c/\sigma$ then
9          break                                                    ▷ Apply UIP as in equation 4
10     end
11     Generate $y_k \sim \pi_\phi(y|x)$ and compute $r_k \leftarrow R(x, y_k)$ $k \leftarrow k+1$
12     $q_{0.01} \leftarrow F^{-1}_{2\alpha_{k-1}}(0.01|\mu,\sigma)$ $\tilde{r}_k \leftarrow \begin{cases} \mu & \text{if } r_k < q_{0.01} \\ r_k & \text{otherwise} \end{cases}$     ▷ Filter extreme low values
13     $(z, \mu, \sigma) \leftarrow$ UpdateStats$(z, \mu, \sigma, \tilde{r}_k, k)$              ▷ Update as in equation 5
14 end
15 $i^* \leftarrow \arg\max_{i \in \{1,\dots,k\}} r_i$ return $y_{i^*}$, $r_{i^*}$
```

When sampling cost c increases, exploration is naturally 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 proof.)

4 BEACON Framework Formulation

BEACON requires three key hyperparameters that jointly define the optimization framework:

Prior Parameters: We adopt Jeffreys' non-informative prior $(\alpha_0, \nu_0, \mu_0, \beta_0) = (-0.5, 0, 0, 0)$ to ensure task-agnostic deployment without requiring pre-training or domain-specific calibration. This requires minimal sample size $k_0 = 3$ for well-defined posterior predictive distributions (detailed in Appendix D.2). The non-informative prior ensures BEACON can adapt to diverse reward landscapes without strong distributional assumptions.

Maximum Horizon (n): The sampling budget determines maximum exploration potential, with values typically following standard Best-of-N configurations ($n \in \{4, 8, 16, 32\}$) for fair baseline comparison. As established in Proposition 1, larger horizons make BEACON more patient, continuing sampling longer when budget allows. However, larger n values increase h-index table pre-computation costs (Appendix D.5). The horizon should be selected based on task complexity, computational resources, and deployment requirements.

Sampling Cost (c): This parameter controls the quality-efficiency trade-off. Higher values prioritize computational efficiency by encouraging earlier stopping, while lower values favor response quality through extended exploration. We provide calibration guidelines in Appendix B, with $c = 0.1$ serving as an effective default.

4.1 Algorithm Implementation

Algorithm 1 presents the complete BEACON procedure. 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 $h_{n,k}(\hat{z}_k) \le c/\sigma_k$; 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.

Table 1: Comparison of BEACON with baseline sampling methods across different models and tasks. BEACON consistently achieves a superior trade-off between Accuracy/Win Rate/Reward $\bar{\hat{z}}_K$ and efficiency (Avg Sample $\overline{K}$), measured by the implicitly optimized objective $\bar{\hat{V}}_K$. 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).

		Reasoning Tasks (Avg. MATH/AIME/AMC)				**Alignment Task (AlpacaEval 2.0)**			
Model	**Method**	**Accuracy ↑ (%)**	**Samples ↓ ($\overline{K}$)**	**Reward ↑ ($\bar{\hat{z}}_K$)**	**Value ↑ ($\bar{\hat{V}}_K$)**	**Win Rate ↑ (%)**	**Samples ↓ ($\overline{K}$)**	**Reward ↑ ($\bar{\hat{z}}_K$)**	**Value↑ ($\bar{\hat{V}}_K$)**
`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	20.0 ↑25.0%	7.2 ↓77.5%	-1.13	-0.56
	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	25.0 ↑11.1%	4.2 ↓86.9%	-1.16	-0.31
	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	85.5 ↑4.3%	3.7 ↓88.4%	-1.03	-0.56
	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., 2021b). Naive SC unsupported for alignment tasks.

4.2 Computational Complexity

BEACON's computational overhead consists of two distinct components with different scalability characteristics:

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.

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 5.2), which generates batches while preserving adaptive stopping properties to reduce runtime.

5 Experiments

5.1 Main Results

Our framework (BEACON) operates with a minimum warm-start of $k_0 = 3$ (to initialize a Normal–Inverse–Gamma posterior under Jeffreys' non-informative prior) and a maximum horizon of $n = 32$ (consistent with standard Best-of-N settings (Singhi et al., 2025)). For each method m, we denote by K_m its realized sample count ($1 \leq K_m \leq n$) and by $\overline{K}_m$ its dataset average. While BEACON adaptively determines K via the precomputed h-index function $h_{n,k}(\hat{z}_k)$ (Baucells & Zorc, 2024), baselines follow fixed or heuristic policies: Direct one-shot Chain-of-Thought (CoT) with K=1; Self-Consistency (SC) (Wang et al., 2022) with majority vote over n samples; Adaptive-Consistency (AS) (Aggarwal et al., 2023), a cost-efficient, model-agnostic method that mathematically grounds early stopping by fitting a Dirichlet–multinomial posterior to agreement patterns and halting once sufficient consensus is reached; RASC (Wan et al., 2025a), a heuristic adaptive sampler guided by CoT quality scores; and fixed Best-of-N (BoN), which selects the highest reward-model–scored candidate from n attempts. The per-sample cost is c=, varied in our ablations, and further implementation details are provided in Appendix A.1.

We tested policy models (`LLaMA-3.2-8B-Instruct`, `Qwen2.5-7B-Instruct`, and `Grok-3-mini`) and reward models (`Llama-3.1-Nemotron-70B-Reward` (Wang et al., 2025c) and *Skywork-Llama-3.1-8B* (Liu et al., 2024)). We applied CoT (Wei et al., 2022) prompting with model-specific instruction templates. Our evaluation focused on (1) *Reasoning Tasks* using three mathematical benchmarks (`MATH-500` (Lightman et al., 2023), `AIME24` (Mathematical Association of America, 2024), and `AMC23`), reporting average accuracy (Pass@1) across these datasets; (2) *Alignment Task* using `AlpacaEval 2.0` (Li et al., 2023), which compares model responses against GPT-4 across diverse prompts; and (3) for both tasks we also reported the *expected standardized reward* $\bar{\bar{z}}_K$ and the *expected standardized value estimation objective* $\mathbb{E}[z_K - K \cdot c]$, denoted as $\hat{\bar{V}}_K$, representing the tradeoff between response alignment quality and computational efficiency when using K samples in expectation.

Optimal Performance-Efficiency Tradeoff. Table 1 comprehensively demonstrates BEACON's ability to achieve an optimal balance between performance and computational efficiency. Our approach consistently maintains similar performance to BoN while significantly reducing the needed samples. This optimal tradeoff is quantitatively validated by consistently superior value function scores, confirming that BEACON effectively maximizes the expected net gain as formulated in our objective function. Such results are consistent across varied experimental conditions—including different base models, reward model implementations, and distinct task categories.

Impact of Sampling Cost (c) on Optimized Sample Size. We examined how the sampling cost parameter c influences the adaptive sample size K_{BEACON}. As visualized in the left of Figure 6, different sampling costs reshape the objective value function landscape. This finding aligns with Proposition 1, which establishes that sample size K_{BEACON} decreases with higher costs c. While additional factors affecting K_{BEACON} (reward variance, remaining budget), these emerge from reward assessments and query characteristics rather than being directly controllable parameters. In contrast, cost c provides a *human-interpretable* control parameter for to balancing quality and efficiency. We also analyze how reward consistency (σ_k), response quality (μ_k and z_k), and remaining sample budget ($n - k$) implicitly affect stopping decisions on individual example in Appendix E.4, which demonstrates that stopping typically occurs when responses become more consistent (lower σ_k), when responses are uniformly low-quality (lower μ_k), or when approaching the maximum budget limit ($n - k \approx 0$).

Parameter Update Mechanism Improves Estimation Robustness and Practical Optimality. One novel design in BEACON is to adaptively update the parameter based on the distribution. Figure 2 shows that the robust update mechanism improves estimation robustness and overall performance over non-robust strategies. This advantage, detailed in Section 3.4, stems from the mechanism's ability to preserve important right-tail characteristics of the reward distribution by lessening the impact of noisy left-tail samples in order to *better uphold the normality assumption* without affecting the best estimate of reward. As a result, our method reaches stopping points (green marker in Figure 2) significantly closer to the true optimum. Appendix E.3 further supports this by illustrating the posterior's typical adherence to normality for both correct and incorrect responses, potential deviations from normality due to noisy left-tail samples, and how our adaptive updates effectively counteract such issues.

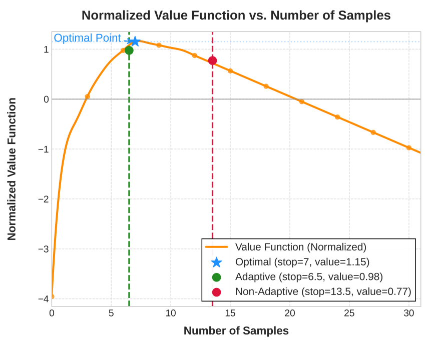

Figure 2: Value Estimation for the robust parameter update method (adaptive) with non-adaptive method; Our design helps BEACON less likely violate the assumption and thus stop closer to the optimal on average.

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 3 illustrates this controlled comparison, revealing BEACON's dual 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 resource allocation 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 difficulty or potential quality improvements.

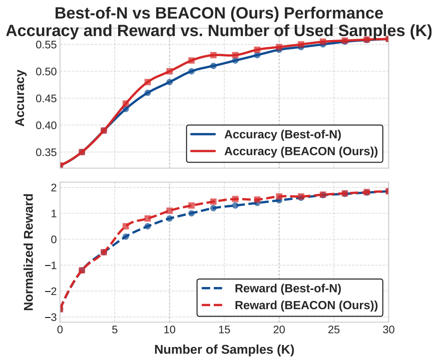

Figure 3: Comparative performance trajectories on Best-of-N vs. BEACON,showing that under same number of samples, our BEACON methods can select samples with a higher reward and better performance.

5.2 Practical Extensions and Analysis

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 Appendix C.1. Figure 4 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) from an initial 16 down to merely 4 by the final iteration. 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, highlighting its value for computational resource optimization.

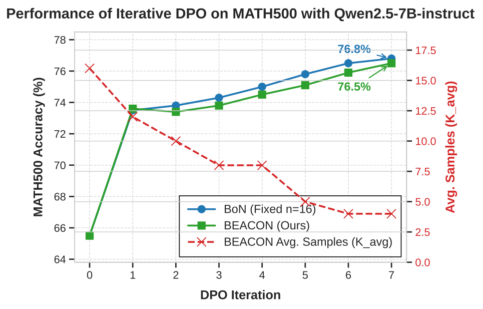

Figure 4: Iterative DPO performance on MATH500 with Qwen2.5-7B-instruct. BEACON (accuracy: green squares) achieves competitive MATH500 accuracy relative to BoN (Fixed n =16, accuracy: blue circles) across seven DPO iteration while reducing the average number of samples required per prompt ($\overline{K}$: red dashed line, right y-axis).

Batch-Parallel Extensions. While BEACON operates sequentially by design, practical deployments can benefit from batch-parallel sampling to reduce wall-clock time at the cost of increased memory usage. In batch-parallel mode, BEACON generates batches of b samples simultaneously and updates posterior beliefs after each complete batch. The stopping criterion remains unchanged: stop if $h_{n,k}(\hat{z}_k) \leq \frac{c}{\sigma_k}$, where k now counts completed batches rather than individual samples. Within each batch, samples are generated independently from the policy LLM using the same prompt. After each batch completes, we: (1) evaluate all b samples with the reward model, (2) update the posterior parameters using all batch rewards simultaneously, and (3) apply the stopping criterion. Upon termination, we select the response with the highest reward across all generated samples from all batches. Table 2 reports performance across different batch sizes using LLaMA 3.2 on mathematical reasoning tasks. The slight accuracy improvements at larger batch sizes occur because batching effectively increases the minimum number of samples generated before each stopping decision, providing more exploration opportunities. However, this comes with diminishing returns and increased memory overhead.

Practical Guidelines: For most applications, batch sizes of 2-4 provide the best speed-memory trade-off, achieving 1.6-2.3× speedup with minimal memory overhead (12-28%). Batch sizes above 4 show diminishing speedup returns while significantly increasing memory requirements. Additionally, BEACON's pre-computed UIP index tables can be reused across multiple parallel queries without recomputation, enabling efficient parallel processing at the query level with no additional cost.

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 ($b = 1$), accounting for the additional memory required to hold b forward passes in parallel during sampling.

Batch Size (b)	**Speedup** (vs Sequential)	**Memory** Overhead	**Accuracy** Change (%)	**Avg. Samples** (K)	**Value Score** $E[z_K - K \cdot c]$
1 (Sequential)	1.00×	Baseline	0.0	15.8	1.61
2	1.65×	+12%	+0.2	16.1	1.58
4	2.31×	+28%	+0.6	17.2	1.52
8	2.85×	+65%	+0.8	19.4	1.34

Hyperparameter Selection. The sampling cost parameter c represents the fundamental trade-off between computational resources and response quality in BEACON. We set a default $c = 0.1$, derived using Jeffreys' non-informative prior and empirical calibration, which typically reduces sampling by 50–80% while maintaining comparable performance to fixed Best-of-N. This calibration involves running fixed Best-of-N baselines on 50–100 validation queries, evaluating BEACON across $c \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.5\}$, and selecting the value at the **knee** of the accuracy–sample Pareto curve. When contextual information is available, c can be adjusted accordingly: lower values (∼0.05–0.08) for hard or ambiguous queries with high response variance (e.g., mathematical proofs), higher values (∼0.2–0.3) for easy queries with consistent answers (e.g., arithmetic), and model-dependent scaling where larger, more stable models justify higher c. The overall intuition is that c should decrease when exploration is valuable and increase when consistency makes additional sampling unnecessary, with detailed calibration procedures provided in Appendices B.1 and B.2.

6 Related Work

6.1 Parallel Inference Scaling as a Search Problem

Parallel scaling approaches (Zeng et al., 2025; Qian et al., 2025; Singhi et al., 2025) enhance LLM performance through multiple candidate generation and selection. These approaches operate at two distinct levels: (1) the solution level—through Best-of-N (BoN) search (Cobbe et al., 2021a; Ichihara et al., 2025) with reward models (Zhang et al., 2024; Ankner et al., 2024) or Majority Vote (Wang et al., 2023)—and (2) the token/step level—through tree-search algorithms such as Beam Search (Xie et al., 2023) and MCTS (Świechowski et al., 2022; Xie et al., 2024). While these methods primarily focus on finding optimal responses, our BEACON framework employs a Bayesian approach (Feng et al., 2025; Agarwal et al., 2025; Jinnai et al., 2025) to address what we term the "economy of sampling" challenge. Unlike prior work, BEACON explicitly optimizes the tradeoff between response quality and the computational cost of generating additional solutions.

6.2 Sampling Efficiency for Effective LLM Reasoning

The computational costs of scaling LLM reasoning have motivated development of efficient sampling strategies (Sui et al., 2025; Fu et al., 2024b). Existing approaches either employ fine-tuned external verifiers to predict generation utility (Manvi et al., 2024; Huang et al., 2025; Wan et al., 2025a) or rely on heuristic-based methods that analyze query complexity to adaptively allocate resources (Wang et al., 2025b; Wan et al., 2025b; Fu et al., 2024b; Aggarwal et al., 2023), which depend on manually-encoded knowledge.Our work diverges from these approaches by uniquely integrating reward signals with Bayesian Optimal Stopping theory to provide a principled framework that automatically optimizes the cost-effectiveness tradeoff at inference time in a theoretically-grounded manner rather than through heuristics, offering provable guarantees on the trade off between quality and efficiency.

6.3 Bandits and Bayesian Optimization

BEACON's sequential search framework connects to several established paradigms in decision theory and optimization (Keith & Ahner, 2021). While Multi-Armed Bandit (MAB) problems (Lattimore & Szepesvári, 2020; Slivkins, 2019) also involve sequential decision-making, they typically focus on maximizing cumulative rewards across trials, whereas BEACON aims to identify the maximum reward while minimizing sampling costs. Our work more directly builds upon best-arm identification (BAI) literature (Audibert & Bubeck, 2010; Gabillon et al., 2012) and Extreme Bandits (Carpentier & Valko, 2014; Lopez et al., 2021), which similarly target identifying optimal or extreme-value outcomes. BEACON's distinctive contribution lies in its Bayesian approach with conjugate priors that adaptively updates beliefs about unknown reward distribution parameters, combined with optimal stopping theory (Ferguson, 2012) to determine the precise moment when additional exploration no longer justifies its cost. While sharing conceptual foundations with budgeted bandits (Xia et al., 2016) and Bayesian optimization (Shahriari et al., 2015), BEACON's application to LLM sample efficiency represents a novel bridge between these theoretical frameworks and the practical challenges of language model sampling and inference.

7 Conclusion and Future Work

In this work, we introduced **Bayesian Efficient Adaptive Criterion for Optimal N-stopping (BEACON)**, a principled framework grounded in Sequential Search 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. Our experiments show that BEACON significantly improves the efficiency of sampling-based verification and reasoning methods, demonstrating the value of decision-theoretic approaches for building resource-aware LLM sampling.

Looking ahead, several promising research directions remain. First, our preliminary experiments with Iterative DPO (Section 5.2) suggest BEACON's potential role in post-training optimization: by adaptively determining when sufficient signal has been extracted from reward models, BEACON could enable more resource-efficient post-training pipelines while reducing the risks of over-optimization (Deng et al., 2025), which could align with recent findings that reinforcement learning methods can narrow the gap between pass@1 and pass@k performance by rewarding desirable yet inconsistent behaviors (Xiong et al., 2025; DeepSeek-AI et al., 2025; Yue et al., 2025). Also, BEACON's adaptability opens avenues for practical extensions, such as incorporating contextual information as informative priors to accelerate convergence and dynamically tuning both the cost parameter c and horizon n based on query complexity. Together, these directions highlight BEACON's potential as both a zero-shot deployable framework and a flexible foundation for advancing efficiency in LLM inference and training.

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) and detailed descriptions of our robust parameter updating mechanism (Section 3.4). 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.

REFERENCES

Dhruv Agarwal, Manoj Ghuhan Arivazhagan, Rajarshi Das, Sandesh Swamy, Sopan Khosla, and Rashmi Gangadharaiah. Searching for optimal solutions with LLMs via bayesian optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=aVfDrl7xDV.

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let‘s sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.761. URL https://aclanthology.org/2023.emnlp-main.761/.

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan Daniel Chang, and Prithviraj Ammanabrolu. Critique-out-loud reward models. In *Pluralistic Alignment Workshop at NeurIPS 2024*, 2024. URL https://openreview.net/forum?id=CljYUvIlRW.

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

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

Donald A Berry and Bert Fristedt. Bernoulli one-armed bandits–arbitrary discount sequences. *The Annals of Statistics*, 7(5):1086–1105, 1979.

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

Bradley P Carlin and Thomas A Louis. *Bayesian methods for data analysis*. CRC press, 2008.

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

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

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. Training verifiers to solve math word problems, 2021a.

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. Training verifiers to solve math word problems, 2021b. URL https://arxiv.org/abs/2110.14168.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of LM computation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6qUUgw9bAZ.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

Morris H DeGroot. *Optimal statistical decisions*. John Wiley & Sons, 2005.

Xun Deng, Han Zhong, Rui Ai, Fuli Feng, Zheng Wang, and Xiangnan He. Less is more: Improving llm alignment via preference data selection, 2025. URL `https://arxiv.org/abs/2502.14560`.

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

Yu Feng, Ben Zhou, Weidong Lin, and Dan Roth. BIRD: A trustworthy bayesian inference framework for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=fAAaT826Vv`.

Thomas S Ferguson. Optimal stopping and applications, 2006.

Thomas S. Ferguson. Optimal stopping and applications. Lecture notes, UCLA Department of Statistics Papers, 2012. URL `https://math.ucla.edu/~tom/Stopping/Contents.pdf`.

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

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. Efficiently serving llm reasoning programs with certaindex, 2024b. URL `https://arxiv.org/abs/2412.20993`.

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

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

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration, 2025. URL `https://arxiv.org/abs/2503.00031`.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Kenshi Abe, Kaito Ariu, Mitsuki Sakamoto, and Eiji Uchibe. Evaluation of best-of-n sampling strategies for language model alignment. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL `https://openreview.net/forum?id=H4S4ETc8c9`.

Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. Regularized best-of-n sampling with minimum Bayes risk objective for language model alignment. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *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)*, pp. 9321–9347, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL `https://aclanthology.org/2025.naacl-long.472/`.

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. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems, 2025. URL `https://arxiv.org/abs/2504.09037`.

Alexander J. Keith and Darryl K. Ahner. A survey of decision making and optimization under uncertainty. *Annals of Operations Research*, 300:319–353, May 2021. doi: 10.1007/s10479-019-03431-8. URL `https://doi.org/10.1007/s10479-019-03431-8`. Published online: 25 October 2019.

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

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`, 5 2023.

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

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms, 2024. URL `https://arxiv.org/abs/2410.18451`.

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

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfcheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL `https://openreview.net/forum?id=RwzFNbJ3Ez`.

Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation, 2024. URL `https://arxiv.org/abs/2410.02725`.

Mathematical Association of America. American invitational mathematics examination - AIME. Mathematics Competition, February 2024. URL `https://maa.org/math-competitions/american-invitational-mathematics-examination-aime`. Accessed: May 2025.

OpenAI. Measuring Goodhart's Law, 4 2022. URL `https://openai.com/index/measuring-goodharts-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. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=K3n5jPkrU6`.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *ArXiv*, abs/2305.18290, 2023. URL `https://api.semanticscholar.org/CorpusID:258959321`.

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

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

Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve, when to verify: Compute-optimal problem solving and generative verification for llm reasoning, 2025. URL `https://arxiv.org/abs/2504.01005`.

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

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=4FWAwZtd2n`.

George J. Stigler. The economics of information. *Journal of Political Economy*, 69(3):213–225, 1961. ISSN 00223808, 1537534X. URL `http://www.jstor.org/stable/1829263`.

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. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419v3*, Apr 2025.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in llms, 2025. URL `https://arxiv.org/abs/2502.06233`.

Csaba Toth and Harald Oberhauser. Bayesian learning from sequential data using Gaussian processes with signature covariances. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9548–9560. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/toth20a.html`.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *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)*, pp. 3613–3635, Albuquerque, New Mexico, April 2025a. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL `https://aclanthology.org/2025.naacl-long.184/`.

Guangya Wan, Yuqi Wu, Hao Wang, Shengming Zhao, Jie Chen, and Sheng Li. Derailer-rerailer: Adaptive verification for efficient and reliable language model reasoning, 2025b. URL `https://arxiv.org/abs/2408.13940`.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6904–6917, Albuquerque, New Mexico, April 2025a. Association for Computational Linguistics. ISBN 979-8-89176-195-7. URL `https://aclanthology.org/2025.findings-naacl.383/`.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=1PL1NIMMrw`.

Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding, 2025b. URL `https://arxiv.org/abs/2503.01422`.

Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. Helpsteer2-preference: Complementing ratings with preferences. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL `https://openreview.net/forum?id=MnfHxPP5gs`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=_VjQlMeSB_J`.

Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979. ISSN 00129682, 14680262. URL `http://www.jstor.org/stable/1910412`.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=NG7sS51zVF`.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Self-evaluation guided beam search for reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=Bw82hwg5Q3`.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

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. Building math agents with multi-turn iterative preference learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=WjKea8bGFF`.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is inevitable: An innate limitation of large language models, 2025. URL `https://arxiv.org/abs/2401.11817`.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of llm-based agents, 2025. URL `https://arxiv.org/abs/2503.16416`.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2025. URL `https://arxiv.org/abs/2401.10020`.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2024. URL `https://openreview.net/forum?id=cijO0f8u35`.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL `https://arxiv.org/abs/2504.13837`.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities?, 2025. URL `https://arxiv.org/abs/2502.12215`.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. URL `https://openreview.net/forum?id=CxHRoTLmPX`.

Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future, 2025. URL `https://arxiv.org/abs/2504.12328`.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: a review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, July 2022. ISSN 1573-7462. doi: 10.1007/s10462-022-10228-y. URL `http://dx.doi.org/10.1007/s10462-022-10228-y`.

THE USE OF LARGE LANGUAGE MODELS (LLMS)

Writing and Editing: LLMs were employed to improve the manuscript's clarity, grammar, and organization. This included: (1) Refining sentence structure and improving readability (2) Polishing grammar and fixing typographical errors (3) Correcting LaTeX code and formatting of tables, figures, and equations.

What LLMs Were Not Used For: LLMs were not involved in research ideation, experimental design, theoretical proof, data generation, result interpretation, or the development of the core BEACON framework. All experimental results, algorithmic contributions, and research insights are original work by the authors.

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.

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

- **Skywork-Llama-3.1-8B-Reward**: Accessed and run locally on a GPU node equipped with 2 NVIDIA A100 GPUs.

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:

```
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.

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

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 $\sim$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 $\sim$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.

C Practical Implementation Details

C.0.1 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 & Zorc, 2024) using the specified priors.

C.1 Iterative DPO with BEACON: Experimental Setup

C.1.1 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 4 of the main text.

C.1.2 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.1.3 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 4, 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.1.5), 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.1.4 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.1.5 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.1.2) 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 5 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.

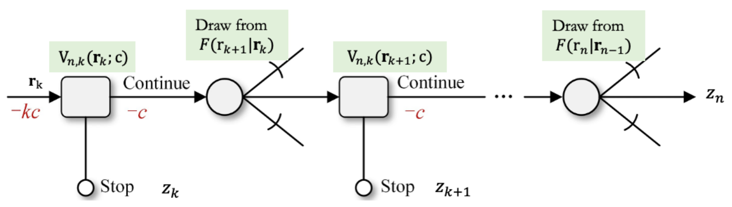
Draw from $F(\mathbf{r}_{k+1}|\mathbf{r}_k)$
$V_{n,k}(\mathbf{r}_k; c)$
$\mathbf{r}_k$
$-kc$
Continue
$-c$
$V_{n,k}(\mathbf{r}_{k+1}; c)$
Continue
$-c$
Draw from $F(\mathbf{r}_n|\mathbf{r}_{n-1})$
z_n
Stop
z_k
Stop
z_{k+1}

Figure 5: 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.

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 & 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 $(\alpha_k, \nu_k, \beta_k, \mu_k)$.

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 & 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 $\mathbf{r}_k = \{r_1, \ldots, r_k\}$, we only need to maintain three key statistics:

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

- 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}:

$$\begin{aligned} z_{k+1} &= \max\{z_k, r_{k+1}\}, \quad \mu_{k+1} = \mu_k + \tfrac{r_{k+1}-\mu_k}{\nu_0+k+1} \\ \sigma_{k+1} &= \sqrt{\tfrac{1-1/(\nu_0+k+1)^2}{2\alpha_0+k+1}} \cdot \sqrt{(2\alpha_0+k)\sigma_k^2 + (r_{k+1}-\mu_k)^2} \end{aligned} \tag{6}$$

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 & 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 & 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 §3.2. As derived in (Baucells & Zorc, 2024), we need to solve:

$$H_{n,k}(\hat{z}, 0, 1; c) = H_{k+1,k}(\hat{z}, 0, 1; c) + \int \sigma_u \max\left\{0, H_{n,k+1}\left(\tfrac{z_u-\mu_u}{\sigma_u}, 0, 1; \tfrac{c}{\sigma_u}\right) - \tfrac{c}{\sigma_u}\right\} dF_{2\alpha}(u), \tag{7}$$

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.

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 Setup. Consider the sequential generation process in Section 3.2, but suppose rewards are binary, $r_i \in \{0, 1\}$, with unknown success probability $\theta \in (0, 1)$. Conditional on θ, the rewards are i.i.d. $\mathrm{Bernoulli}(\theta)$. Sampling incurs a cost $c > 0$ per draw, with a maximum horizon n. Let $\mathbf{r}_k = (r_1, \ldots, r_k)$ and $z_k = \max\{r_1, \ldots, r_k\} \in \{0, 1\}$ denote the current best observed reward. We adopt a conjugate Beta prior $\theta \sim \mathrm{Beta}(a_0, b_0)$, where $a_0, b_0 > 0$. After k draws, with $s_k = \sum_{i=1}^{k} r_i$ successes and $f_k = k - s_k$ failures, the posterior becomes

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

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

and the posterior predictive for the next reward is Bernoulli with success probability

$$q_k = \mathbb{P}(r_{k+1} = 1 \mid \mathbf{r}_k) = \frac{a_k}{a_k + b_k}.$$

The sufficient statistics for the problem are (z_k, a_k, b_k), with state update

$$z_{k+1} = \max\{z_k, r_{k+1}\}, \quad a_{k+1} = a_k + r_{k+1}, \quad b_{k+1} = b_k + 1 - r_{k+1}.$$

When $z_k = 1$, the process is absorbing and the DM stops with value 1. For $z_k = 0$, the value-to-go function satisfies the recursion

$$V_{n,k}(0, a_k, b_k; c) = \max\Big\{0,\ q_k + (1 - q_k)V_{n,k+1}(0, a_k, b_k + 1; c) - c\Big\}, \qquad V_{n,n}(0, a_n, b_n; c) = 0,$$

with $V_{n,k}(1, a_k, b_k; c) = 1$.

Optimal Sampling Policy. For binary rewards with Beta-Binomial conjugate priors, we can leverage the optimal policy derived by Berry & Fristedt (1979) and DeGroot (2005), adapted to our sampling context:

Theorem 4 *For each stage $k < n$ with posterior parameters (a_k, b_k), define the* Bernoulli–Beta index

$$h^{\mathrm{B}}_{n,k}(a_k, b_k) := \sup\big\{c \geq 0 : V_{n,k}(0, a_k, b_k; c) > 0\big\}.$$

The Bayes-optimal policy is:

- *Stop if $z_k = 1$ (success already observed)*
- *When $z_k = 0$, continue sampling if and only if $c < h^{\mathrm{B}}_{n,k}(a_k, b_k)$*

To handle binary rewards with unknown probability, we simply replace the optimal sampling policy in Theorem 1 with the Bernoulli-Beta policy from Theorem 4. This adaptation maintains BEACON's principled Bayesian approach while accommodating discrete binary outcomes.

For categorical rewards with multiple outcomes, the Dirichlet-Multinomial conjugate pair provides an elegant solution, with the Dirichlet distribution serving as the conjugate prior for the multinomial distribution. Optimal sampling policies for this setting are also available in the literature (Carlin & Louis, 2008), further demonstrating the framework's adaptability beyond continuous distributions.

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 equation 4. 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 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 equation 4, 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 & 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

Table 6: Comparison of BEACON with baseline methods using LLaMA-3.2-3B

	Reasoning Tasks (Avg. MATH/AIME/AMC)				Alignment Task (AlpacaEval 2.0)			
Method	**Accuracy ↑ (%)**	**Samples ↓ ($\overline{N}$)**	**Reward ↑ (Scaled)**	**Value ↑ (Scaled)**	**Win Rate ↑ (%)**	**Samples ↓ ($\overline{N}$)**	**Reward ↑ (Scaled)**	**Value ↑ (Scaled)**
Direct CoT	20.02.8	1.0	-1.600.10	-0.400.05	16.03.5	1.0	0.200.05	-0.800.08
BoN (N-RM)	33.41.3	32.0	3.490.22	0.290.14	25.02.5	32.0	4.000.25	0.800.09
BEACON (N-RM)	32.81.6	15.81.2	3.250.18	**1.12**0.21	23.51.8	14.52.3	3.650.22	**1.20**0.20

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 (32.81.6% vs. 33.41.3% for reasoning tasks) and win rate (23.51.8% vs. 25.02.5% for alignment tasks), with overlapping error margins indicating no substantial performance degradation. However, BEACON requires significantly fewer samples (15.81.2 vs. 32.0 for reasoning tasks), resulting in substantially higher value scores that account for both performance and computational cost (1.120.21 vs. 0.290.14).

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 6, 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 6 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 6 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.

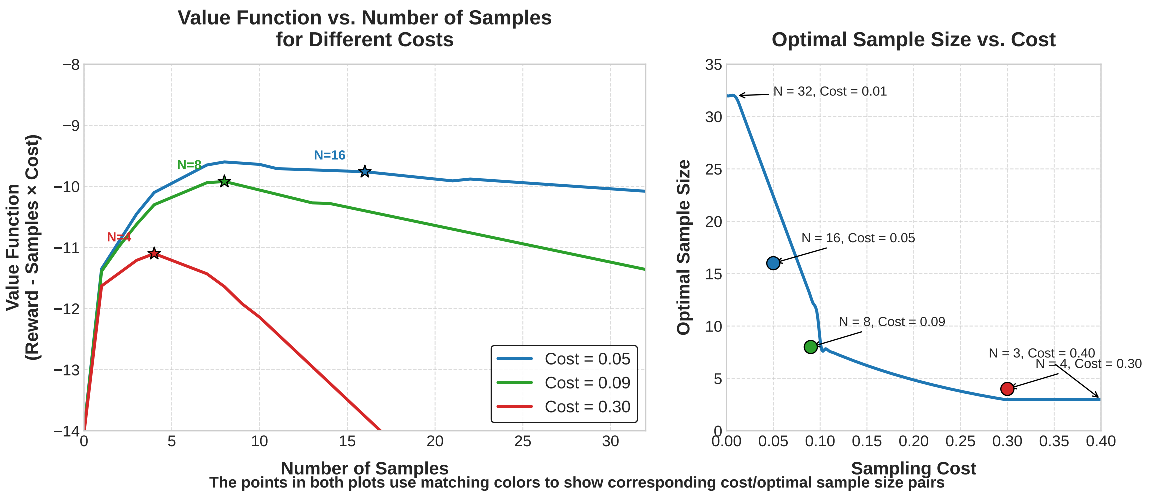

Figure 6: Impact of the sampling cost (c) on the value optimization and the resulting optimal sample size.

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.

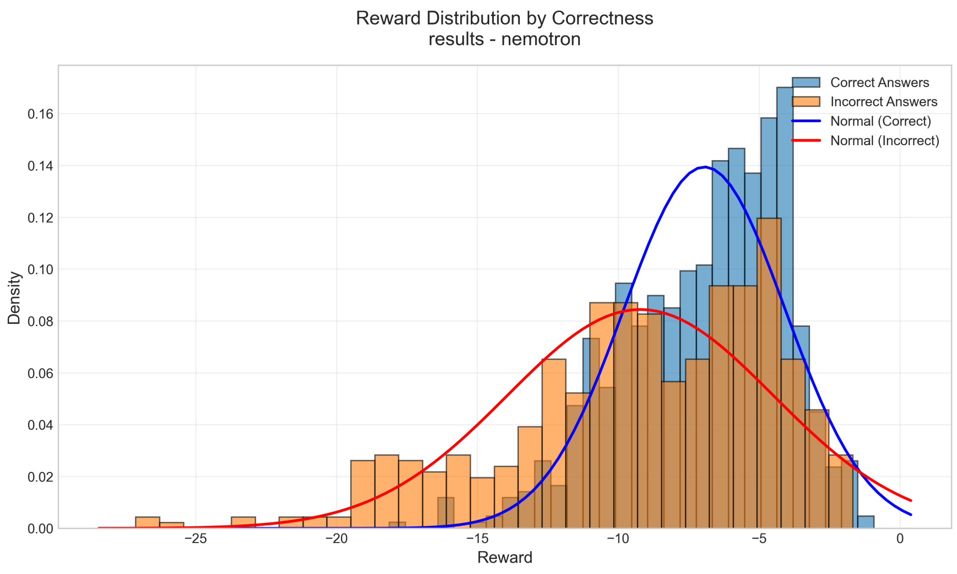

Figure 7: Aggregated reward distributions from the Nemotron RM, separated for responses classified as correct (blue histogram, blue normal fit) and incorrect (orange histogram, red normal fit). Both distributions show approximate normality.

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.

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

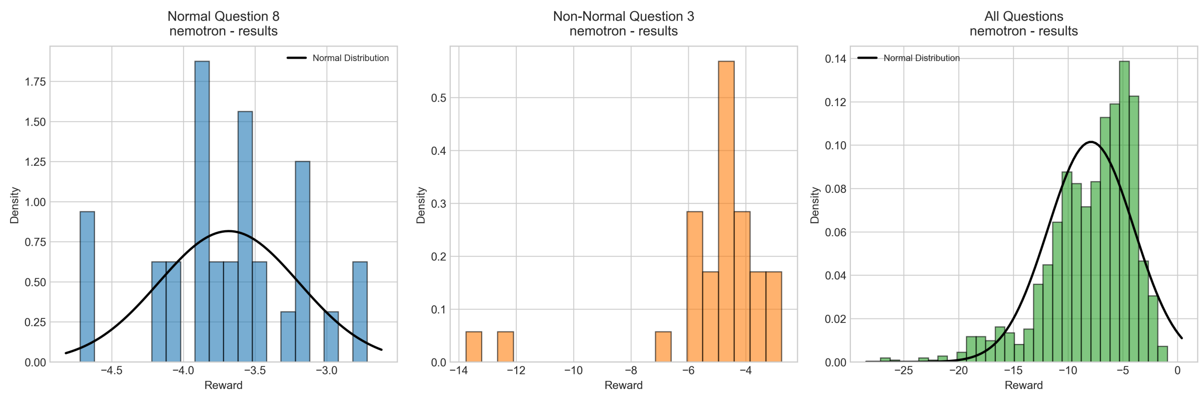

Figure 8: Examples of query-specific reward distributions using the Nemotron RM. Left: A typical case exhibiting approximate normality ("Normal Question 8"). Middle: An occasional case with predominantly high rewards and some low-value outliers ("Non-Normal Question 3"), motivating robust updates. Right: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

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.

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.