Efficient Cost-Aware LLM Evaluation via Bayesian Bandit Gittins Indices

Qian Xie [1] **Yueli He** [2] **Nairen Cao** [2]

Abstract

Selecting a high-performing LLM configuration increasingly requires comparing many candidate models, prompts, or decoding settings under limited evaluation budgets. Exhaustively evaluating every candidate on every benchmark example can be expensive, and each additional query incurs costs such as API pricing, token usage, latency, or human grading effort. We formulate this configuration-selection task as cost-aware Bayesian bandit evaluation and propose a Gittins index policy that treats each configuration's benchmark performance as latent, updates posterior uncertainty, allocates queries according to their value of information, and provides a stopping signal. The method combines sample efficiency with computational efficiency: after precomputing Gittins indices for the Gaussian posterior dynamics, online allocation only requires table lookup and posterior updates, and it naturally handles varying evaluation costs. Across GSM8K, PIQA, and MMLU response matrices of different sizes, our Gittins index policies usually reach lower simple regret with fewer evaluations or lower cumulative cost than UCB-E, a frequentist best-arm identification method, and UCB-E-LRF, its low-rank correlation-aware extension.

1. Introduction

Modern LLM evaluation is no longer a passive reporting exercise. A single evaluation study may compare dozens of model checkpoints, prompts, decoding rules, retrieval pipelines, or fine-tuning recipes across benchmarks such as GSM8K, PIQA, and MMLU (Cobbe et al., 2021; Bisk et al., 2020; Hendrycks et al., 2020). Running every candidate on every example is simple, but often wasteful: weak candidates can be identified early, strong candidates deserve more measurement, and different systems can have very different inference costs. As evaluation moves from one-off leaderboards toward recurring deployment decisions, the evaluator needs an adaptive rule for deciding which candidate to test next and when enough evidence has been collected.

This perspective naturally suggests a bandit formulation. Prior work has used bandit and racing ideas to accelerate language model evaluation, including frequentist best-arm identification methods and variance-reduction heuristics tailored to benchmark matrices (Zhou et al., 2025; Polo et al., 2024). UCB-E treats the problem as fixed-budget best-arm identification, while UCB-E-LRF augments this allocation rule with a low-rank model of correlations in the response matrix. These methods are compelling because they convert an offline response matrix or benchmark dataset into an online allocation problem: each additional query is an arm pull, and the goal is to find a high-performing arm with small evaluation cost. However, many LLM evaluation settings also contain useful prior information. Historical benchmark results, model family metadata, pilot runs, or domain expertise can all inform beliefs about likely performance before the current evaluation starts. A Bayesian formulation provides a principled way to encode such information and to update it as new benchmark observations arrive.

We introduce a Gittins index policy for cost-aware Bayesian bandit evaluation of LLMs. Each arm represents an LLM configuration, its latent mean is the benchmark score we wish to estimate, and each pull queries the configuration on benchmark data. The evaluator observes a noisy score and pays an arm-dependent cost. The final decision is the recommended arm, so performance is measured by the quality of this recommendation as a function of cumulative evaluation cost, rather than by cumulative reward during evaluation.

Our main methodological choice is to use Gittins indices for this Bayesian evaluation problem. Under a Gaussian observation approximation, the posterior mean of each arm evolves as a Gaussian random walk whose variance decreases deterministically with the number of observations. The resulting Gittins index is an interpretable priority score: it is high when an arm currently looks good, remains uncertain enough to justify more measurement, and is cheap enough to evaluate.

Our contributions are:

[1]Cornell University, Ithaca, New York, USA [2]New York University, Brooklyn, New York, USA. Correspondence to: Qian Xie and Nairen Cao <qx66@cornell.edu, nc1827@nyu.edu>.

Accepted to the ICML Workshop on Decision-Making from Offline Datasets to Online Adaptation.

- We formulate LLM evaluation from benchmark data as a cost-aware Bayesian bandit problem in which query costs can reflect API pricing, token usage, latency, or other evaluation costs.
- We derive a Gittins index policy under a Gaussian observation approximation and show how the posterior random-walk structure enables efficient offline index precomputation and online table lookup.
- We extend the policy to varying model costs through cost normalization, allowing adaptive allocation to trade off information value against evaluation expense.
- Across GSM8K, PIQA, and MMLU response matrices, the policy reaches lower simple regret than UCB-E and the correlation-aware UCB-E-LRF baseline under both unit-cost and cost-aware evaluation.

1.1. Related Work

Efficient LLM evaluation Efficient LLM evaluation has recently been studied as an adaptive sampling problem. Zhou et al. (2025) propose methods for speeding up language model evaluation, including UCB-E-LRF, which exploits low-rank structure in response matrices to share information across arms and examples. Shi et al. (2024) introduce TRIPLE, an elimination-based approach that combines successive halving with a linear model that predicts final arm performance from partial evaluations. Their response-matrix setup is closest to our fixed-cost-budget reporting view: adaptive evaluation proceeds under unit costs and is judged as best-arm identification, especially the probability of selecting the best arm. We instead report simple regret and allow varying evaluation costs. Polo et al. (2024) propose PromptEval for efficient multi-prompt evaluation and emphasize that evaluation effort can be saved by exploiting structure across prompts and tasks. Our work shares the same goal of reducing benchmark evaluation cost, but differs in its Bayesian treatment of uncertainty and its use of Gittins indices.

The Gittins Policy The Gittins index is a classical design principle for sequential decision making under uncertainty (Gittins et al., 2011; Scully & Terenin, 2025; Xie et al., 2024). It converts certain multi-armed allocation problems into independent single-arm stopping problems, producing an index for each arm and selecting the arm with the largest index. The terminal-reward formulation we consider is closer to finite-horizon Bayesian stopping problems than to the traditional infinite-horizon bandit objective that maximizes cumulative discounted rewards during play: evaluation scores are measurements, while utility is obtained through the final recommendation after paying query costs. Up to an additive best-arm value, this trades simple regret against cumulative evaluation cost. We use the Gittins principle as the basis for an adaptive LLM evaluation policy. The key technical convenience in our setting is that Gaussian observations make each arm's posterior mean follow a Gaussian random walk, which allows efficient precomputation of the index schedule.

2. Cost-Aware Bandit Evaluation

2.1. Arms, Examples, and Latent Performance

We consider an evaluation study with K candidate LLM systems. An arm $k \in \{1, \dots, K\}$ may correspond to a model, prompt, decoding configuration, retrieval pipeline, or any other deployable system whose benchmark performance is to be estimated.

A benchmark supplies a finite collection of examples, but the policy does not need to condition on the identity of a particular example. Let $Z_{k,j}$ denote the primitive score obtained by evaluating arm k on example j. We model these example-level scores as samples from an arm-specific distribution with latent mean θ_k. In a finite response matrix, this is the population-model approximation to randomized sampling without replacement from the corresponding row. Adaptive evaluation reveals only selected scores from these potential outcomes. The score may be a bounded accuracy score, a normalized human or LLM-judge score, a task-level utility, or another scalar evaluation outcome. Binary correctness is a common special case, where $Z_{k,j} \in \{0, 1\}$ indicates whether the model answered the example correctly.

The evaluator's goal is to identify an arm with large latent mean θ_k while spending as little evaluation cost as possible. Each pull incurs a cost $c_k > 0$, or an effective cost after rescaling raw monetary or computational costs into the same utility units as benchmark performance. In LLM evaluation, cost can be tied to API pricing through input-token and output-token charges, and may also include latency, local compute, or human grading effort. Prior work such as UCB-E (Audibert & Bubeck, 2010), TRIPLE (Shi et al., 2024), UCB-E-LRF (Zhou et al., 2025), and PromptEval (Polo et al., 2024) typically corresponds to the special case $c_k \equiv 1$.

2.2. Adaptive Evaluation and Recommendation

At global time t, an adaptive evaluation policy π can depend on the history

$$\mathcal{H}_t = \{(A_s, Y_s, c_{A_s}) : s \le t\}.$$

Here Y_s denotes the scalar observation returned by the sth allocation step; it may be a single example-level score $Z_{A_s,j}$ or a batch average of several such scores. We use the term policy broadly: π includes the rule for selecting the next arm A_t, the rule for deciding whether to stop at an endogenous stopping time T, and the final recommendation rule that

outputs $\hat{k}_T \in \{1, \dots, K\}$ from the observations collected so far. In fixed-budget experiments, the stopping rule is replaced by the budget constraint, but the same policy still produces an anytime recommendation after each step.

When a finite per-arm evaluation horizon is imposed, an arm is completed once it reaches that horizon. The stopping and recommendation rule then specifies whether evaluation should continue or terminate based on the current arm states and posterior beliefs.

2.3. Objectives

We consider two cost-aware formulations. The first is the fixed-cost-budget objective. Unlike standard stochastic bandits that optimize cumulative rewards from the arms played during evaluation, the observed scores here are measurements of benchmark performance. The payoff is the quality of the terminal recommendation, so this objective is finite-horizon and undiscounted. For a budget $\mathcal{C}$, the policy is evaluated by minimizing simple regret subject to the cumulative evaluation cost remaining within budget:

$$\min_{\pi} \mathbb{E}_{\pi}\left[\max_{k} \theta_k - \theta_{\hat{k}_T}\right] \quad \text{subject to} \quad \sum_{s=1}^{T} c_{A_s} \le \mathcal{C}.$$

Because θ_k is latent, experiments with completed response matrices report regret using the finite-matrix row mean

$$\theta_k^{\mathrm{mat}} = \frac{1}{N} \sum_{j=1}^{N} Z_{k,j} \tag{1}$$

as a plug-in proxy for θ_k. This lets us compare adaptive policies using only the entries they reveal online while still measuring regret against the best arm in the completed benchmark matrix.

The second formulation lets the terminal time be adaptive rather than fixed in advance. For a stopping time T, the objective is

$$\mathbb{E}\left[\theta_{\hat{k}_T} - \sum_{t=1}^{T} c_{A_t}\right].$$

If raw costs $\tilde{c}_k$ are measured in dollars, seconds, or tokens, we write $c_k = \lambda \tilde{c}_k$ for a cost scale λ that converts them into benchmark-score units. This objective trades final recommendation quality against the cumulative cost of the measurements used to make that recommendation.

3. Gittins Index Policy for Cost-Aware Bayesian Evaluation

Our goal is to adaptively estimate the mean benchmark performance of each candidate LLM configuration while accounting for varying evaluation costs. Each arm has an unknown mean score. In our implementation, one allocation to an arm returns the average score over a batch of randomized benchmark examples, and we treat this batch average as a noisy measurement of the arm's latent performance.

The algorithm has two components. First, it maintains a Bayesian posterior distribution over each arm mean and updates that posterior after every observed batch average. Second, it uses a cost-aware Gittins index to decide which arm is worth measuring next. The index is high when an arm currently looks promising, remains uncertain enough that new evidence could change the final recommendation, or is inexpensive to evaluate. The policy stops when an arm attaining the largest Gittins index has reached the finite evaluation horizon, and then recommends a completed arm with the largest posterior mean. Figure 1 summarizes how posterior beliefs, Gittins indices, benchmark batches, and the response matrix interact over one iteration of this loop.

3.1. Bayesian Bandit Model

We use the Bayesian bandit model as a way to update beliefs about the latent benchmark mean θ_k of each arm. This differs from frequentist allocation rules such as UCB-E, which build confidence bounds from empirical estimates. Here the evaluator starts with a prior belief, observes benchmark evidence, and converts that evidence into a posterior belief about each θ_k.

After n local pulls of arm k, let $\mathcal{D}_{k,n}$ denote the observations collected from that arm. The posterior distribution given $\mathcal{D}_{k,n}$ summarizes the current belief about θ_k. If evaluation stops, the posterior expected reward from recommending arm k is its posterior mean $\mu_{k,n} = \mathbb{E}[\theta_k \mid \mathcal{D}_{k,n}]$, so the recommended arm is $\arg\max_k \mu_{k,n_k}$.

A pull may evaluate one example or a batch of examples. Following the notation in Section 2, let $Z_{k,j}$ be the example-level score for arm k on example j. If the ith pull of arm k evaluates batch $\mathcal{B}_{k,i}$ of size B, the scalar observation used for posterior updating is the batch mean

$$\overline{Y}_{k,i} = \frac{1}{B} \sum_{j \in \mathcal{B}_{k,i}} Z_{k,j}.$$

The batch mean has conditional mean θ_k. Let τ_k^2 denote its conditional variance, so that, by the central limit theorem, $\overline{Y}_{k,i} \mid \theta_k$ is approximately $\mathcal{N}(\theta_k, \tau_k^2)$ as a Gaussian approximation. In the common binary-accuracy case, $\tau_k^2 = \theta_k(1-\theta_k)/B$. For simplicity and efficient index computation, we use a fixed approximation τ^2 to the batch-mean variance; Section 3.4 describes this choice.

We use the Gaussian prior-likelihood approximation

$$\theta_k \sim \mathcal{N}(\mu_0, v_0), \qquad \overline{Y}_{k,i} \mid \theta_k \sim \mathcal{N}(\theta_k, \tau^2),$$

where τ^2 is the fixed observation-noise variance in the like-

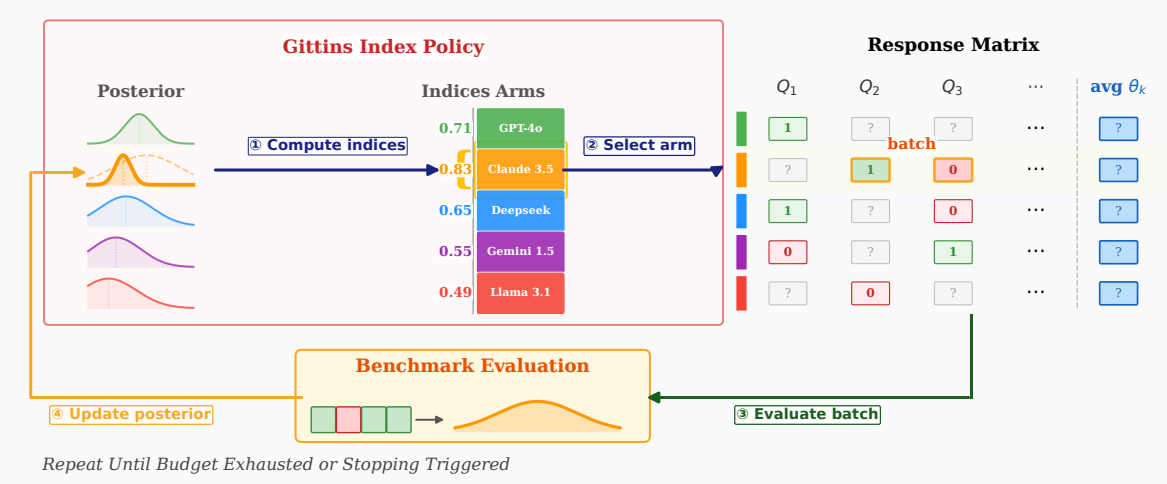


Figure 1. Overview of the proposed Gittins-index evaluation framework. The policy maintains posterior beliefs over candidate LLM configurations, computes Gittins indices to adaptively select the next arm batch to evaluate, observes benchmark responses from the response matrix, and updates the posterior iteratively until the evaluation budget is exhausted or a stopping condition is met.

lihood. To state the update compactly, fix a representative arm and suppress the arm index, writing θ, $\mathcal{D}_n$, μ_n, v_n, and $\overline{Y}_{n+1}$. The posterior after n local pulls is

$$\theta \mid \mathcal{D}_n \sim \mathcal{N}(\mu_n, v_n),$$

with update

$$\begin{aligned} v_{n+1} &= \left(v_n^{-1} + \tau^{-2}\right)^{-1}, \\ \mu_{n+1} &= \mu_n + \frac{v_n}{v_n + \tau^2}(\overline{Y}_{n+1} - \mu_n). \end{aligned} \tag{2}$$

When Eq. (2) is applied to arm k in Algorithm 1, the symbols are read as $\mu_{k,n}$, $v_{k,n}$, and $\overline{Y}_{k,n+1}$. Under fixed τ^2, the variance schedule for this representative arm is deterministic. In particular, the one-step variance reduction is

$$v_n - v_{n+1} = \frac{v_n^2}{v_n + \tau^2}.$$

Thus arms share the same variance schedule at a given local pull count only when they also share the same initial variance v_0 and fixed noise τ^2; with arm-specific prior variances or noise levels, the same update gives arm-specific deterministic schedules. The next step is to use these posterior updates not only to summarize current evidence, but also to decide which arm should be evaluated next.

3.2. Posterior Dynamics and the Gittins Policy

The posterior update also tells us what could happen to an arm's posterior mean if we evaluate it again. Let S_n denote the posterior mean of a representative arm after n local pulls, viewed as a random variable over future observations. Conditional on the current realized state $S_n = s$, selecting the arm gives

$$S_{n+1} \mid S_n = s \sim \mathcal{N}\left(s, \frac{v_n^2}{v_n + \tau^2}\right).$$

Fix a finite per-arm horizon H, measured in local pulls, so that an arm is completed after H batch evaluations. The arm state is therefore (S_n, n): the posterior mean follows a Gaussian random walk, while the posterior variance is determined by the local pull count. Selecting an arm changes only that arm's state and incurs its evaluation cost; an arm with $n = H$ is completed and becomes eligible for terminal recommendation. This separation lets us compare arms through single-arm index values.

To define the index, fix one arm with posterior-mean state $S_n = s$, query cost c, and remaining horizon $H - n$. Let α denote the outside terminal reward available from the other arms. Before the arm reaches H, the local actions are CONTINUE, which evaluates the arm once more, and STOP, which declines this unfinished arm in favor of the outside value α. Suppressing the fixed cost c in the value-function notation, the terminal value is

$$V_H(s; \alpha) = \max\{s, \alpha\},$$

and, for $n < H$,

$$\begin{aligned} Q_n^{\text{cont}}(s; \alpha) &= -c + \mathbb{E}[V_{n+1}(S_{n+1}; \alpha) \mid S_n = s], \\ Q_n^{\text{stop}}(s; \alpha) &= \alpha, \\ V_n(s; \alpha) &= \max\{Q_n^{\text{cont}}(s; \alpha), Q_n^{\text{stop}}(s; \alpha)\}. \end{aligned}$$

Here Q_n^{cont} is the Q-value function for evaluating the arm one more time, while Q_n^{stop} is the value of declining this unfinished arm in favor of the outside reward. For $n < H$ and cost c, the Gittins index is the smallest outside reward at which stopping is optimal in this local problem:

$$G_n^c(s) = \inf\{\alpha \in \mathbb{R} : Q_n^{\text{stop}}(s; \alpha) \geq Q_n^{\text{cont}}(s; \alpha)\}.$$

An arm receives a large index when its posterior mean is

high, its uncertainty creates value of information, or its evaluation cost is low.

At global time t, let $n_k(t)$ be the number of batches observed from arm k. For an unfinished arm, set

$$G_{k,t} = G^{c_k}_{n_k(t)}(\mu_{k,n_k(t)}).$$

A completed arm has terminal index $G_{k,t} = \mu_{k,H}$. Under required completion, if an arm attaining $\max_k G_{k,t}$ is completed, the policy stops and recommends a completed arm with the highest posterior mean among completed arms; otherwise, it evaluates an unfinished arm with largest index. This is the stopping rule used for the Gittins stopping markers in our experiments.

We now state the Bayesian optimality guarantee for the required-completion formulation.

Theorem 3.1 (Bayesian optimality under required completion)**.** *Consider the Gaussian Bayesian bandit model in Section 3.1 with independent arms, common prior $\mathcal{N}(\mu_0, v_0)$, fixed observation noise τ^2, finite per-arm horizon H, and nonnegative effective costs of arm pulls c_k for arms $k = 1, \dots, K$. Let Π^{req}_H be the class of policies that evaluate only unfinished arms and may terminate only by recommending an arm that has reached its finite horizon. For the required-completion objective*

$$\sup_{\pi \in \Pi^{\mathrm{req}}_H} \mathbb{E}_\pi \left[\theta_{\hat{k}_T} - \sum_{t=1}^{T} c_{A_t} \right],$$

an optimal policy is to assign terminal index $G_{k,t} = \mu_{k,H}$ to completed arms, assign $G_{k,t} = G^{c_k}_{n_k(t)}(\mu_{k,n_k(t)})$ to unfinished arms, and stop when an arm attaining $\max_k G_{k,t}$ is completed. Otherwise, the policy evaluates an unfinished arm with largest Gittins index $G_{k,t}$. At stopping, an optimal recommendation is any completed arm with largest posterior mean among completed arms, equivalently any completed arm with largest terminal index.

The proof is deferred to Appendix A.2.

3.3. Efficient Gittins Index Computation

We next describe how the index values used in the previous subsection are efficiently computed. The definition above suggests a dynamic program over both the posterior-mean state s and the outside reward α. Solving such a program at every online decision would be too expensive. We make the computation practical with three reductions. First, translational invariance converts the two-dimensional continuation value into a one-dimensional dynamic program over the advantage $s - \alpha$. Second, under fixed observation noise, the posterior-variance schedule is deterministic, so the resulting root schedule can be precomputed once for each cost and variance schedule. Third, the Gaussian expectations in the dynamic program are one-dimensional convolutions and can be evaluated efficiently offline.

Lemma 3.2 (Translational invariance of the continuation value)**.** *Fix the cost c and Gaussian random-walk transition. For any shift $\beta \in \mathbb{R}$ and any stage $n < H$,*

$$Q^{\mathrm{cont}}_n(s + \beta; \alpha + \beta) = Q^{\mathrm{cont}}_n(s; \alpha) + \beta.$$

Corollary 3.3 (Centered continuation value)**.** *Let $x = s - \alpha$ and define $q_n(x) := Q^{\mathrm{cont}}_n(x; 0)$. For any $n < H$,*

$$Q^{\mathrm{cont}}_n(s; \alpha) = q_n(s - \alpha) + \alpha.$$

Thus evaluating the unfinished arm is worthwhile exactly when $q_n(s - \alpha) > 0$.

The proof of Lemma 3.2 and Corollary 3.3 is included in Appendix A.3.

By Corollary 3.3, only the posterior-mean advantage $x = s - \alpha$ matters. Let $W_n(x) := V_n(x; 0)$, then the centered recursion is

$$\begin{aligned} W_H(x) &= \max\{x, 0\}, \\ W_n(x) &= \max\{0, q_n(x)\}, \\ q_n(x) &= -c + \mathbb{E}[W_{n+1}(X_{n+1}) \mid X_n = x], \end{aligned} \tag{3}$$

where $X_{n+1} \mid X_n = x \sim \mathcal{N}(x, v_n^2/(v_n + \tau^2))$ and the second line applies for $n < H$. Let r_n denote the root of the continuation value such that $q_n(r_n) = 0$. Then evaluating an unfinished arm is worthwhile exactly when $s - \alpha > r_n$. Equivalently, the index can be evaluated by

$$G^c_n(s) = s - r_n. \tag{4}$$

For each distinct tuple (c, v_0, τ^2, H), we solve the centered dynamic program once offline and store the root schedule

$$\{r_n(c, v_0, \tau^2, H)\}_{n=0}^{H-1}.$$

The fixed-noise approximation makes this lookup table reusable: arms with the same effective cost, prior variance, observation noise, and horizon share the same schedule. The conditional expectation in (3) is a Gaussian convolution on a one-dimensional grid, which we compute offline; our implementation accelerates this step with FFT-based convolution. Online allocation is then inexpensive: each decision requires only posterior updates, table lookups, and index comparisons. For a finite benchmark with N examples, $H = N$ for single-example pulls and $H = \lceil N/B \rceil$ for batches of size B.

Additional FFT derivation details are deferred to the full version of the paper.

3.4. Design Choices for LLM Benchmarks

Prior specification. For accuracy-style benchmarks, the latent mean lies in $[0, 1]$, so the prior can be chosen directly on the accuracy scale. A general choice is $\mathcal{N}(0.5, 0.04)$: its mean expresses no preference between low and high accuracy, and most prior mass lies in the plausible interval $[0.1, 0.9]$. When benchmark- or dataset-specific knowledge is available, we can instead use an informative common prior whose mean and variance are estimated from related tasks, historical evaluations, or pilot examples. Unless otherwise stated, all arms within a benchmark share the same prior so that the comparison focuses on adaptive allocation rather than arm-specific prior information.

Observation noise. For batched binary correctness observations, the empirical batch mean has variance $\theta_k(1-\theta_k)/B$. Batching also has a practical systems benefit: examples in a batch can often be evaluated in parallel, reducing wall-clock latency, a motivation also noted by Zhou et al. (2025). Since θ_k is unknown, we use the conservative fixed-noise approximation $\tau^2 = 1/(4B)$, the worst-case Bernoulli variance divided by the batch size. This choice is deliberately simple and can overestimate the noise for very easy or very hard tasks, but it has a computational advantage: the posterior variance schedule becomes deterministic, so the Gittins root schedule can be precomputed once and reused by all arms with the same cost, prior variance, noise level, and horizon. More adaptive alternatives include plug-in noise $\tau^2_{k,t} \approx \mu_{k,t}(1-\mu_{k,t})/B$ or estimates from pilot examples and historical response matrices; these require additional index computation because the variance schedule can depend on the arm's evolving state.

Cost scaling. Costs must be expressed in the same utility scale as benchmark performance. In API-based evaluation, raw cost may be computed from provider pricing, input-token counts, and output-token counts; for local models, latency or compute can be used as a proxy. If the scale λ is small, the policy explores more aggressively; if it is large, the stopping rule becomes conservative. We therefore study cost-scaling factors in addition to raw inference-cost ratios.

Recommendation and reporting. The Gittins index is used both for allocation and for required-completion early stopping. When early stopping is enabled, the policy stops once an arm attaining the largest current Gittins index has reached horizon H. At stopping, it recommends a completed arm with the largest posterior mean,

$$\hat{k}_t \in \operatorname*{arg\,max}_{k:n_k(t)=H} \mu_{k,H}.$$

If no completed arm has the largest index, the policy evaluates an unfinished arm with largest Gittins index. This matches the required-completion formulation in Theorem 3.1.

For fixed-budget curves, early stopping can be disabled so that the same allocation rule continues until the budget is exhausted. In that anytime reporting view, after each allocation we report the arm with the largest current posterior mean, which lets us evaluate simple regret as a function of cumulative cost.

Algorithm 1 summarizes the implementation. When early stopping is disabled, the algorithm ignores this stopping condition and continues until the budget is exhausted, unless all arms have reached the horizon.

Algorithm 1 Gittins index evaluation for a Bayesian bandit

Require: Prior (μ_0, v_0), raw costs $\{\tilde{c}_k\}_{k=1}^K$, cost scale λ.
Require: Observation noise τ^2, batch size B, horizon H, and budget.
Set $c_k = \lambda\tilde{c}_k$ for all arms k. {Index-scale costs.}
Precompute $\{r_n(c, v_0, \tau^2, H)\}_{n=0}^{H-1}$ for each needed cost c. {Offline table.}
Set $(\mu_{k,0}, v_{k,0}, n_k(0)) = (\mu_0, v_0, 0)$ for all k, and set $t = 0$. {Shared prior.}
while budget remains **do**
  if $n_k(t) = H$ for all arms k **then** {All rows complete.}
    return $\arg\max_k \mu_{k,H}$.
  end if
  Set $G_{k,t} = \mu_{k,H}$ for completed arms. {Terminal index.}
  Compute $G_{k,t} = G^{c_k}_{n_k(t)}(\mu_{k,n_k(t)})$ for each unfinished arm k. {Continuation index.}
  if early stopping is enabled **then**
    if an arm in $\arg\max_k G_{k,t}$ is completed **then**
      return $\arg\max_{k:n_k(t)=H} \mu_{k,H}$. {Best completed arm.}
    end if
  end if
  Select arm $A_t \in \arg\max_{k:n_k(t)<H} G_{k,t}$. {Max unfinished index.}
  Observe batch mean $\overline{Y}_{A_t,n_{A_t}(t)+1}$. {Batch query.}
  Update $(\mu_{A_t,\cdot}, v_{A_t,\cdot})$ using Eq. (2). {Bayesian update.}
  Set $n_{A_t}(t+1) = n_{A_t}(t) + 1$, set $n_j(t+1) = n_j(t)$ for all $j \neq A_t$, and set $t \leftarrow t+1$.
end while
return $\arg\max_k \mu_{k,n_k(t)}$. {Anytime output.}

4. Experiments

Benchmarks. We evaluate adaptive allocation policies using completed LLM response matrices built on three

standard benchmarks: GSM8K for grade-school mathematical reasoning (Cobbe et al., 2021), PIQA for physical commonsense reasoning (Bisk et al., 2020), and MMLU for multitask knowledge and reasoning (Hendrycks et al., 2020). The benchmark examples define the evaluation tasks, while the observed LLM correctness results come from existing response-matrix datasets. GSM8K and PIQA use the various-LLM and sampling-configuration response matrices from BanditEval (Zhou et al., 2025): GSM8K contains 122 arms and 1000 examples, while PIQA contains 103 arms and 1000 examples. MMLU uses response matrices from DOVE (Habba et al., 2025) across 57 subject datasets. For each subject, we evaluate 15 models with 100 prompting templates; each arm is a model-template pair, giving 1500 arms per subject.

For MMLU, we further group subjects by their empirical mean arm quality. This gives a natural notion of subject difficulty: if the average arm accuracy is high, then the subject is relatively easy for the collection of evaluated models and prompting templates; if the average arm accuracy is low, then the subject is relatively hard. We use this criterion to divide MMLU subjects into easy, medium, and hard groups. More details and analysis of the datasets can be found in Appendix B.

Baselines. We compare two versions of the proposed Gittins index policy with UCB-E, a classical fixed-budget best-arm identification method (Audibert & Bubeck, 2010), UCB-E-LRF, the low-rank-factorization variant used for faster language model evaluation by Zhou et al. (2025), and configuration-level BO baselines using PBGI, LogEI, and their cost-aware variants. Gittins-G denotes Gittins with the general prior, while Gittins-S denotes Gittins with a data-specific prior; the exact prior settings are listed in Table 4. UCB-E and UCB-E-LRF represent frequentist allocation rules that do not explicitly encode Bayesian priors or Gittins index stopping values. When a baseline includes a random initialization phase, the percentage refers to the sampling unit: for arm-level initialization, 5% means sampling 5% of arms uniformly at random and fully evaluating those arms on all benchmark examples. For BanditEval's UCB-E-LRF warm-up, the default is entry-level and samples $T_0 = 0.05KN$ arm–example entries from a K-arm, N-example response matrix before active selection.

Evaluation metric. The primary metric is simple regret as a function of the evaluation resources used. After allocation step t, each policy reports a recommendation $\hat{k}_t$, the arm with the highest current estimate of benchmark performance. Given the completed response matrix, we compute simple regret $\mathrm{SR}_t = \max_k \theta_k^{\mathrm{mat}} - \theta_{\hat{k}_t}^{\mathrm{mat}}$ using the finite-matrix target in Eq. (1). We plot this simple regret against cumulative example evaluations $M_t = \sum_{s=1}^{t} b_s$ in the unit-cost setting, where b_s is the number of examples evaluated at step s, and against cumulative evaluation cost $C_t = \sum_{s=1}^{t} c_{A_s,b_s}$ in the cost-aware setting, where c_{A_s,b_s} is the cost of evaluating the selected arm on that batch. For configuration-level BO baselines, one query evaluates a full configuration and consumes its corresponding full-evaluation cost.

Experimental conditions. For each benchmark, we consider both unit-cost and cost-aware settings. In the unit-cost setting, every batch has the same cost, isolating the effect of adaptive statistical allocation. In the cost-aware setting, batch costs reflect model-specific evaluation expense, so the policy must trade off information value against heterogeneous inference cost. Configurations that use the same base model share the same model-level token price, with task-specific input/output accounting described in Appendix C.

For the bandit methods, we use a fixed evaluation budget equal to 10% of the corresponding full-evaluation budget: 10% of the response-matrix entries in the unit-cost setting and 10% of the total full-evaluation cost in the cost-aware setting. UCB-E and UCB-E-LRF are run to this fixed budget, following the BanditEval protocol of Zhou et al. (2025); for the Gittins variants, we also report where the Gittins stopping rule from Section 3.4 would terminate. The BO baselines follow a configuration-level protocol, where evaluating one candidate reveals its aggregate score and consumes the full-evaluation cost of that configuration. They use a random-initialization phase capped at 5% of the configurations, followed by acquisition-driven selection under the same nominal 10% budget fraction; cost-aware BO runs stop before exceeding the 10% full-evaluation-cost budget. For the main GSM8K and PIQA plots, UCB-E and Gittins use batch size $B = 8$, while UCB-E-LRF uses $B = 32$. For the main MMLU aggregate plots, small subjects use $B = 2$ and large subjects use $B = 8$ for UCB-E and Gittins, while UCB-E-LRF uses $B = 32$. The medium-size bucket, full subject-level MMLU results, and batch-size ablations are included in the appendix.

Reporting. The main results report simple-regret curves for GSM8K and PIQA under unit-cost and cost-aware evaluation. Each curve compares the Gittins variants, UCB-E, UCB-E-LRF, and BO baselines on the same response matrices. Unless otherwise stated, shaded bands denote one standard error of the mean across repeated runs. Figure 2 presents these main curves. Figure 3 summarizes the corresponding MMLU trends. Dashed vertical markers indicate average stopping locations when available: for Gittins, these correspond to the Gittins stopping rule; for BO, they correspond to the BO budget or acquisition stopping point. These markers are distinct from the fixed 10% budget used for UCB-E and UCB-E-LRF. For the MMLU aggregate plots, we normalize the x-axis by each run's final cumula-

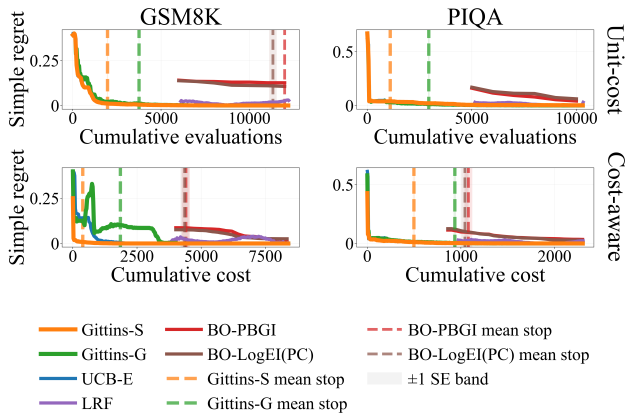


Figure 2. Main GSM8K and PIQA results. Simple regret is plotted against cumulative evaluations in the unit-cost setting and against cumulative cost in the cost-aware setting, using batch size $B = 8$ for UCB-E and Gittins and $B = 32$ for LRF. Each panel averages over 100 independent runs (5 ground-truth matrices × 20 seeds). Shaded bands denote ± 1 standard error. Dashed vertical lines show mean stopping locations across seeds: for Gittins, when the index-based rule first triggers; for BO, when the acquisition stopping criterion first triggers. Gittins-S reduces simple regret quickly across these panels and reaches low-regret stopping points well before the 10% evaluation-budget guideline used by UCB-E in BanditEval (Zhou et al., 2025).

tive evaluations in the unit-cost setting or final cumulative cost in the cost-aware setting, while keeping the y-axis as raw simple regret. Appendix D reports the MMLU unit-cost and cost-aware curves, common-prior, batch-size, and cost-scaling ablations, together with descriptive plots of the latent accuracy distribution for each MMLU prior bucket.

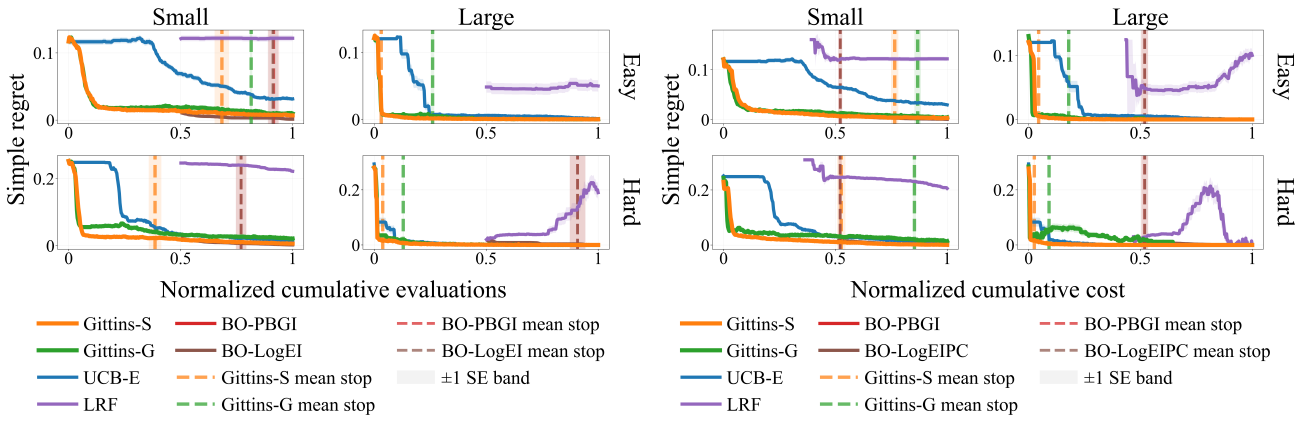
Implementation details. For the Gittins index policy, root schedules are precomputed for the common prior variance, noise, batch size, horizon, and cost scale used in each condition. During evaluation, the policy updates posterior means and variances using the Gaussian approximation in Section 3.1. Because online allocation only requires posterior updates and root-table lookup after this offline precomputation step, the per-batch decision overhead remains small relative to UCB-E-LRF; Figure 14 in Section D reports runtime comparisons. The BO baselines operate on converted configuration-level inputs: after the random-initialization phase, they fit a mixed Gaussian-process model to the observed configurations and select the next configuration using the corresponding acquisition function. For GSM8K and PIQA, we average over five response matrices, reflecting LLM output randomness, and run 20 randomized algorithm trials on each matrix. For MMLU, we run 20 randomized algorithm trials for each subject and aggregate over the subjects in each reported bucket.

Main findings. The results show that the Gittins-based policies reduce simple regret rapidly in the low-budget regime, with the clearest gains appearing when the response matrix contains many candidate arms. On GSM8K and PIQA, where the matrices have only 122 and 103 arms but 1000 examples per arm, the BO baselines are less competitive because each configuration-level BO query consumes a full row of the response matrix. Under the same 10% budget fraction, the 5% random-initialization phase already uses a substantial part of the BO budget, leaving relatively few acquisition-driven updates. The batched bandit methods can instead allocate partial evaluations across many configurations, and Gittins-S often reaches low regret early under both unit-cost and cost-aware evaluation.

The MMLU aggregate results show a clearer role for adaptive allocation. Each MMLU subject has 1500 model-template arms, so the policy must search over a much larger candidate set. After normalizing the x-axis by each run's final budget while keeping the y-axis as raw simple regret, Gittins-S consistently reduces simple regret early across the reported small and large subject buckets. The BO baselines are most competitive on subject matrices with fewer examples, where a full configuration evaluation is relatively cheap; on larger-example matrices, the same configuration-level protocol becomes less sample-efficient because each BO step observes an entire row rather than a small batch of entries. Thus, the BO baselines provide a useful comparison but do not consistently dominate the Gittins policies under the same budgeted evaluation setting.

The Gittins stopping locations usually appear after the Gittins regret curves have flattened, suggesting that the rule stops once a completed arm is also the most valuable arm under the index comparison. These stopping points are generally earlier than the BO budget stop and earlier than the 10% fixed-budget endpoint used for UCB-E and UCB-E-LRF in BanditEval (Zhou et al., 2025). On larger response matrices, the stopping point can occur around 1% of the full response-matrix budget, giving a large reduction in evaluation effort while preserving the low-regret recommendation reached by the curve.

The comparison between Gittins-G and Gittins-S highlights the role of prior calibration. Gittins-S tends to benefit from data-specific priors, especially on harder MMLU groups where the prior mean is better aligned with the latent accuracy range and early allocation decisions are more consequential. At the same time, Gittins-G remains competitive in several easier or smaller settings, suggesting that a general prior can be sufficient when many arms have similar high accuracies or when regret becomes small quickly. Overall, the results suggest that data-specific priors are useful, but their mean and variance should be calibrated carefully rather than assumed to be uniformly beneficial across all



(a) Unit-cost setting.

(b) Cost-aware setting.

Figure 3. Aggregate MMLU results by subject size and difficulty. Simple regret is plotted against cumulative evaluations in a and cumulative cost in b; the x-axis is rescaled to $[0, 1]$ by each run's final budget, and the y-axis shows raw simple regret. For UCB-E and Gittins, small subjects use batch size $B = 2$ and large subjects use $B = 8$; LRF uses batch size $B = 32$ where completed runs are available. Each panel averages over all subjects in the size–difficulty bucket and 20 independent runs per subject. Shaded bands denote ± 1 standard error. Dashed vertical lines show mean stopping locations across seeds: for Gittins, when the index-based rule first triggers; for BO, when the acquisition stopping criterion first triggers. Across subject-size and difficulty buckets, Gittins-S remains competitive and generally reduces simple regret earlier than UCB-E, while LRF is less effective where completed runs are available.

task difficulties.

5. Conclusion

We formulated LLM evaluation as a cost-aware Bayesian bandit problem and derived a practical Gittins index policy from the Gaussian random-walk dynamics of posterior means under Gaussian benchmark observations. The formulation provides a natural way to use posterior uncertainty when deciding which candidate to evaluate next, while also accounting for heterogeneous model costs such as API pricing and token usage. Empirically, the policy improves simple regret relative to UCB-E, UCB-E-LRF, and BO across response-matrix benchmarks, while the MMLU prior ablations show that data-specific priors are useful when well calibrated but can be less effective on easy datasets.

Future work. Future work will broaden the comparison set to include additional efficient evaluation methods such as PromptEval (Polo et al., 2024) and TRIPLE (Shi et al., 2024). We also plan to develop a more principled procedure for choosing priors, especially for easy datasets where the current data-specific prior appears miscalibrated. On the modeling side, extending the index computation beyond constant Gaussian noise and toward richer priors that share information across model families, prompts, and benchmark tasks remains an important direction.

Acknowledgments

QX thanks Theodore Brown, Ziv Scully, and Alexander Terenin for a prior collaboration from which the translational-invariance reduction and FFT-based approach to efficient Gittins-index computation originated. QX also thanks Ziv Scully for helpful discussions on the optional-completion setting, Kyuseong Choi for introducing QX to the area of efficient LLM evaluation and sharing related work, Tianyi Peng for inviting QX to a related project and discussions connecting LLM evaluation with multi-armed bandits, and Jin Peng Zhou and Ruihan Wu for sharing the BanditEval data and answering questions about the datasets.

References

Audibert, J.-Y. and Bubeck, S. Best arm identification in multi-armed bandits. In *COLT-23th Conference on learning theory-2010*, pp. 13–p, 2010.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dumitriu, I., Tetali, P., and Winkler, P. On playing golf with two balls. *SIAM Journal on Discrete Mathematics*, 16(4): 604–615, 2003.

Gittins, J., Glazebrook, K., and Weber, R. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

Habba, E., Arviv, O., Itzhak, I., Perlitz, Y., Bandel, E., Choshen, L., Shmueli-Scheuer, M., and Stanovsky, G. DOVE: A large-scale multi-dimensional predictions dataset towards meaningful LLM evaluation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11744–11763. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.611. URL HTTPS://ACLANTHOLOGY.ORG/2025.FINDINGS-ACL.611/.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Polo, F. M., Xu, R., Weber, L., Silva, M., Bhardwaj, O., Choshen, L., de Oliveira, A. F., Sun, Y., and Yurochkin, M. Efficient multi-prompt evaluation of llms. *Advances in Neural Information Processing Systems*, 37:22483–22512, 2024.

Scully, Z. and Terenin, A. The gittins index: A design principle for decision making under uncertainty. In *Tutorials in Operations Research: Advances in Analytics and Operations Research: Improving Decisions to Secure the Future*, pp. 28–70. INFORMS, 2025.

Shi, C., Yang, K., Chen, Z., Li, J., Yang, J., and Shen, C. Efficient prompt optimization through the lens of best arm identification. *Advances in Neural Information Processing Systems*, 37:99646–99685, 2024.

Xie, Q., Astudillo, R., Frazier, P. I., Scully, Z., and Terenin, A. Cost-aware bayesian optimization via the pandora's box gittins index. *Advances in Neural Information Processing Systems*, 37:115523–115562, 2024.

Zhou, J. P., Belardi, C. K., Wu, R., Zhang, T., Gomes, C. P., Sun, W., and Weinberger, K. Q. On speeding up language model evaluation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL HTTPS://OPENREVIEW.NET/FORUM?ID=3CVWO5DBZN.

A. Theoretical Details

A.1. Gaussian Random-Walk Details

We derive the posterior update and the induced random-walk transition used in Section 3. Fix an arm and suppress the arm index. Let $\overline{Y}_{n+1}$ denote the scalar observation from the next pull, which may be either a single-example score or a batch mean. Suppose that after n local pulls the current posterior distribution of the arm mean is

$$\theta \mid \mathcal{D}_n \sim \mathcal{N}(\mu_n, v_n),$$

and that the next observation follows the fixed-noise Gaussian likelihood

$$\overline{Y}_{n+1} \mid \theta \sim \mathcal{N}(\theta, \tau^2).$$

By Bayes' rule, the posterior density after observing $\overline{Y}_{n+1}$ is proportional to the likelihood times the current posterior:

$$p(\theta \mid \mathcal{D}_{n+1}) \propto p(\overline{Y}_{n+1} \mid \theta)\, p(\theta \mid \mathcal{D}_n).$$

Substituting the Gaussian likelihood and the current Gaussian posterior gives

$$p(\theta \mid \mathcal{D}_{n+1}) \propto \exp\left(-\frac{1}{2}\left[\frac{(\theta-\mu_n)^2}{v_n} + \frac{(\overline{Y}_{n+1}-\theta)^2}{\tau^2}\right]\right).$$

Expanding the terms that depend on θ,

$$\frac{(\theta-\mu_n)^2}{v_n} + \frac{(\overline{Y}_{n+1}-\theta)^2}{\tau^2} = \left(v_n^{-1}+\tau^{-2}\right)\theta^2 - 2\left(\frac{\mu_n}{v_n} + \frac{\overline{Y}_{n+1}}{\tau^2}\right)\theta + \text{const.}$$

Thus the exponent is quadratic in θ, so the posterior is Gaussian. Matching the quadratic and linear coefficients with the Gaussian form

$$\exp\left(-\frac{(\theta-\mu_{n+1})^2}{2v_{n+1}}\right)$$

gives

$$v_{n+1}^{-1} = v_n^{-1} + \tau^{-2}, \qquad \frac{\mu_{n+1}}{v_{n+1}} = \frac{\mu_n}{v_n} + \frac{\overline{Y}_{n+1}}{\tau^2}.$$

Therefore,

$$v_{n+1} = \left(v_n^{-1}+\tau^{-2}\right)^{-1}, \qquad \mu_{n+1} = v_{n+1}\left(\frac{\mu_n}{v_n} + \frac{\overline{Y}_{n+1}}{\tau^2}\right).$$

Equivalently, the posterior mean update can be written in the incremental form

$$\mu_{n+1} = \mu_n + \frac{v_n}{v_n+\tau^2}\left(\overline{Y}_{n+1} - \mu_n\right).$$

This form shows that the new posterior mean moves from the old posterior mean toward the new observation, with gain

$$\frac{v_n}{v_n+\tau^2}.$$

To describe the posterior mean as a state process, we first compute the predictive distribution of the next observation. Conditioned on the current data $\mathcal{D}_n$, we may write

$$\overline{Y}_{n+1} = \theta + \varepsilon_{n+1}, \qquad \theta \mid \mathcal{D}_n \sim \mathcal{N}(\mu_n, v_n), \qquad \varepsilon_{n+1} \sim \mathcal{N}(0, \tau^2),$$

with θ and ε_{n+1} conditionally independent given $\mathcal{D}_n$. Hence

$$\overline{Y}_{n+1} \mid \mathcal{D}_n \sim \mathcal{N}(\mu_n, v_n + \tau^2).$$

We now view the posterior mean as the Markov state used below. Let S_n denote the posterior mean after n local pulls, viewed as a random variable over future observations. If the current realized state is $S_n = s$, then the posterior update gives

$$S_{n+1} = s + \frac{v_n}{v_n + \tau^2} \left(\overline{Y}_{n+1} - s \right).$$

Since

$$\overline{Y}_{n+1} - s \mid S_n = s \sim \mathcal{N}(0, v_n + \tau^2),$$

we have

$$S_{n+1} \mid S_n = s \sim \mathcal{N}\left(s, \left(\frac{v_n}{v_n + \tau^2} \right)^2 (v_n + \tau^2) \right).$$

Therefore,

$$S_{n+1} \mid S_n = s \sim \mathcal{N}\left(s, \frac{v_n^2}{v_n + \tau^2} \right).$$

Finally, under the fixed-noise likelihood, the variance update

$$v_{n+1} = \left(v_n^{-1} + \tau^{-2} \right)^{-1}$$

does not depend on the realized observation $\overline{Y}_{n+1}$. Therefore the posterior variance follows a deterministic schedule indexed only by the local pull count n, while the posterior mean follows the Gaussian random-walk transition above whenever the arm is selected. Across arms, this schedule is identical only for arms with the same initial variance and fixed noise level; otherwise each arm has its own deterministic schedule.

A.2. Proof of Bayesian Optimality

Theorem A.1 (Restatement of Theorem 3.1)**.** *Consider the Gaussian Bayesian bandit model in Section 3.1 with independent arms, common prior $\mathcal{N}(\mu_0, v_0)$, fixed observation noise τ^2, finite per-arm horizon H, and nonnegative effective costs of arm pulls c_k for arms $k = 1, \ldots, K$. Let Π_H^{req} be the class of policies that evaluate only unfinished arms and may terminate only by recommending an arm that has reached its finite horizon. For the required-completion objective*

$$\sup_{\pi \in \Pi_H^{\mathrm{req}}} \mathbb{E}_\pi \left[\theta_{\hat{k}_T} - \sum_{t=1}^{T} c_{A_t} \right],$$

an optimal policy is to assign terminal index $G_{k,t} = \mu_{k,H}$ to completed arms, assign $G_{k,t} = G^{c_k}_{n_k(t)}(\mu_{k,n_k(t)})$ to unfinished arms, and stop when an arm attaining $\max_k G_{k,t}$ is completed. Otherwise, the policy evaluates an unfinished arm with largest Gittins index $G_{k,t}$. At stopping, an optimal recommendation is any completed arm with largest posterior mean among completed arms, equivalently any completed arm with largest terminal index.

Proof. We prove the theorem by reducing the required-completion problem to a Markov chain selection problem. For each arm k, define one local chain:

State. The local state is $x_k = (s_k, n_k)$, where s_k is the posterior mean and $n_k \in \{0, \ldots, H\}$ is the number of observed batches. The terminal states are those with $n_k = H$.

Action. At each decision time, choose one arm k. Only the chosen arm moves; every other arm keeps its current state.

Reward. If the chosen arm is unfinished, the immediate reward is $-c_k$. If the chosen arm is terminal, the process stops and the terminal reward is s_k.

Transition. If the chosen arm is in state (s_k, n_k) with $n_k < H$, then its next state is $(S', n_k + 1)$, where

$$S' \mid s_k, n_k \sim \mathcal{N}\left(s_k, \frac{v_{n_k}^2}{v_{n_k} + \tau^2} \right).$$

The global state is the collection of local states, initialized at $((\mu_0, 0), \ldots, (\mu_0, 0))$.

By Eq. (2), the state of arm k can be represented by (μ_{k,n_k}, n_k). Under fixed τ^2, Appendix A.1 shows that the posterior mean follows the Gaussian random-walk transition, so this state is Markov. Conditional on this state, the next state distribution of arm k is independent of the states and histories of all other arms, and pulling arm k incurs only its own effective cost c_k. Since $s_k = \mathbb{E}[\theta_k \mid x_k]$, the terminal reward is the posterior expected reward from recommending completed arm k. Thus maximizing expected total reward in the constructed selection problem is exactly the required-completion objective.

The Gittins-index optimality theorem for Markov chain selection implies that an optimal policy selects a chain with largest index until a terminal chain has largest index (Dumitriu et al., 2003; Scully & Terenin, 2025). Applying this result to the single-arm value function used to define $G_n^c(s)$ gives the stated rule. Terminal chains have index equal to their posterior mean, so when a completed arm attains the largest index, recommending any completed arm with largest posterior mean among completed arms gives an optimal terminal reward. □

A.3. One-Dimensional Gittins Index Computation

We now spell out the reduction used by the precomputation step in Algorithm 1. Fix a single arm, suppress the arm index, and let α be the outside terminal reward available from the other arms. With the cost c fixed and suppressed in the value-function notation, the terminal value is

$$V_H(s; \alpha) = \max\{s, \alpha\},$$

and, for $n < H$,

$$\begin{aligned} Q_n^{\text{cont}}(s; \alpha) &= -c + \mathbb{E}[V_{n+1}(S_{n+1}; \alpha) \mid S_n = s], \\ Q_n^{\text{stop}}(s; \alpha) &= \alpha, \\ V_n(s; \alpha) &= \max\{Q_n^{\text{cont}}(s; \alpha), Q_n^{\text{stop}}(s; \alpha)\}. \end{aligned}$$

The transition kernel depends only on differences in posterior means:

$$S_{n+1} \mid S_n = s \sim \mathcal{N}(s, \sigma_n^2), \qquad \sigma_n^2 = \frac{v_n^2}{v_n + \tau^2}.$$

Proof of Lemma 3.2. The terminal value satisfies

$$V_H(s + \beta; \alpha + \beta) = \max\{s + \beta, \alpha + \beta\} = V_H(s; \alpha) + \beta.$$

Assume the corresponding shift identity holds for the next-stage value. Since the transition kernel is Gaussian with additive mean, the next state from $s + \beta$ has the same distribution as $S_{n+1} + \beta$ when the current state is s. Therefore

$$\begin{aligned} Q_n^{\text{cont}}(s + \beta; \alpha + \beta) &= -c + \mathbb{E}[V_{n+1}(S_{n+1} + \beta; \alpha + \beta) \mid S_n = s] \\ &= -c + \mathbb{E}[V_{n+1}(S_{n+1}; \alpha) + \beta \mid S_n = s] \\ &= Q_n^{\text{cont}}(s; \alpha) + \beta. \end{aligned}$$

Taking the maximum with the shifted outside reward gives the next-stage value identity needed for the induction. The result follows by backward induction. □

Corollary 3.3 follows by applying Lemma 3.2 with shift $\beta = \alpha$ to the centered state $x = s - \alpha$ and outside reward 0. Define the centered value

$$W_n(x) = V_n(\alpha + x; \alpha) - \alpha.$$

Then the recursion no longer depends on α:

$$W_H(x) = \max\{x, 0\},$$

and

$$\begin{aligned} W_n(x) &= \max\{0, q_n(x)\}, \\ q_n(x) &= -c + \mathbb{E}[W_{n+1}(X_{n+1}) \mid X_n = x], \end{aligned}$$

where $X_{n+1} \mid X_n = x \sim \mathcal{N}(x, \sigma_n^2)$ and the recursion for W_n applies for $n < H$. This is the one-dimensional dynamic program: for each stage n, the numerical grid is only over the centered posterior-mean advantage x.

For comparison with an outside reward α, evaluating an unfinished arm is worthwhile exactly when $q_n(s-\alpha) > 0$. Let r_n denote the crossing point satisfying

$$q_n(r_n) = 0.$$

The Gittins index for cost c is therefore

$$G_n^c(s) = s - r_n.$$

The online policy does not need to store the full two-argument value function. During precomputation we represent the one-dimensional functions W_n and q_n on a grid in order to propagate the recursion backward, but the objects used online are the roots $\{r_n\}_{n=0}^{H-1}$ for each distinct cost and variance schedule.

The conditional expectation $\mathbb{E}[W_{n+1}(X_{n+1}) \mid X_n = x]$ in the recursion for q_n is a convolution of the next-stage value grid with a Gaussian transition kernel. On an evenly spaced grid this convolution can be computed efficiently using FFT-based convolution, which is why the index schedule can be computed once offline and then reused cheaply during adaptive evaluation.

B. Dataset Details

B.1. Dataset Description

GSM8K and PIQA. For GSM8K and PIQA, we use the response matrices released with BanditEval (Zhou et al., 2025). The underlying benchmarks are Grade School Math 8K (GSM8K) (Cobbe et al., 2021) and Physical Interaction: Question Answering (PIQA) (Bisk et al., 2020). GSM8K consists of grade-school math word problems, while PIQA evaluates physical commonsense reasoning through question answering.

The BanditEval response matrices are constructed from a collection of publicly available language models and sampling configurations. In particular, BanditEval considers 11 models: GPT2, GPT2-Large, CodeLLaMA, Tulu-7B, Tulu-2-7B, Gemma-7B, Phi2, Llema-7B, LLaMA-2-7B, Mistral-7B, and StarCoder-7B. For each model, responses are generated using three temperature choices $\{0, 0.5, 1\}$, two maximum decoding lengths $\{128, 512\}$, and two zero-shot prompting strategies: directly asking for the answer, and using the chain-of-thought prompt "Let's think step by step." The Cartesian product of these choices yields $11 \times 3 \times 2 \times 2 = 132$ possible model–configuration arms.

However, the released response matrices contain some missing configurations. As a result, the final matrices used in our experiments contain 122 arms for GSM8K and 103 arms for PIQA. Both GSM8K and PIQA contain 1,000 examples in the response matrices. Each benchmark is represented by five independently generated response matrices, corresponding to five random seeds used when querying the LLMs. Since the benchmark examples and arm set are fixed, the differences across these matrices mainly reflect randomness in LLM response generation.

MMLU. For MMLU, we use the DOVE response matrices (Habba et al., 2025). MMLU is a multiple-choice question-answering benchmark consisting of 57 subjects and approximately 14,000 examples in total. We treat each MMLU subject as a separate problem instance. Each subject is evaluated across 15 LLM configurations and 100 prompting techniques, resulting in 1500 arms per subject.

The 15 LLM configurations used in the MMLU response matrices are Llama-3-8B, Llama-3-8B-Instruct, Llama-3-70B-Instruct, CodeLlama-34B-Instruct, FLAN-T5-XL, FLAN-T5-XXL, FLAN-UL2, Merlinite-7B, Mixtral-8x7B-Instruct-v0.1, Mistral-7B-Instruct-v0.2, Gemma-7B, Gemma-7B-IT, Falcon-40B, Mistral-7B-v0.1, and Falcon-180B. The number of examples varies across subjects; the full list of subjects used in our evaluation is provided in Table 1.

B.2. Dataset Statistics

We provide descriptive statistics for the response matrices used in our evaluation. For each dataset or subject, we compute the empirical quality of each arm as its mean accuracy over benchmark examples. The following figures visualize the resulting distribution of arm qualities. These plots help illustrate the spread of arm performance, the location of the empirical mean, the best arm, and the prior mean used in informative-prior experiments.

GSM8K and PIQA. Figure 4 shows the per-arm quality distributions for GSM8K and PIQA across the five independently generated response matrices. The distributions are highly stable across seeds for both datasets. This indicates that, although the response matrices are generated independently, the randomness from LLM response generation introduces only limited

variation at the aggregate level. GSM8K has a noticeably lower overall accuracy distribution than PIQA, while PIQA exhibits a tighter concentration of high-quality arms.

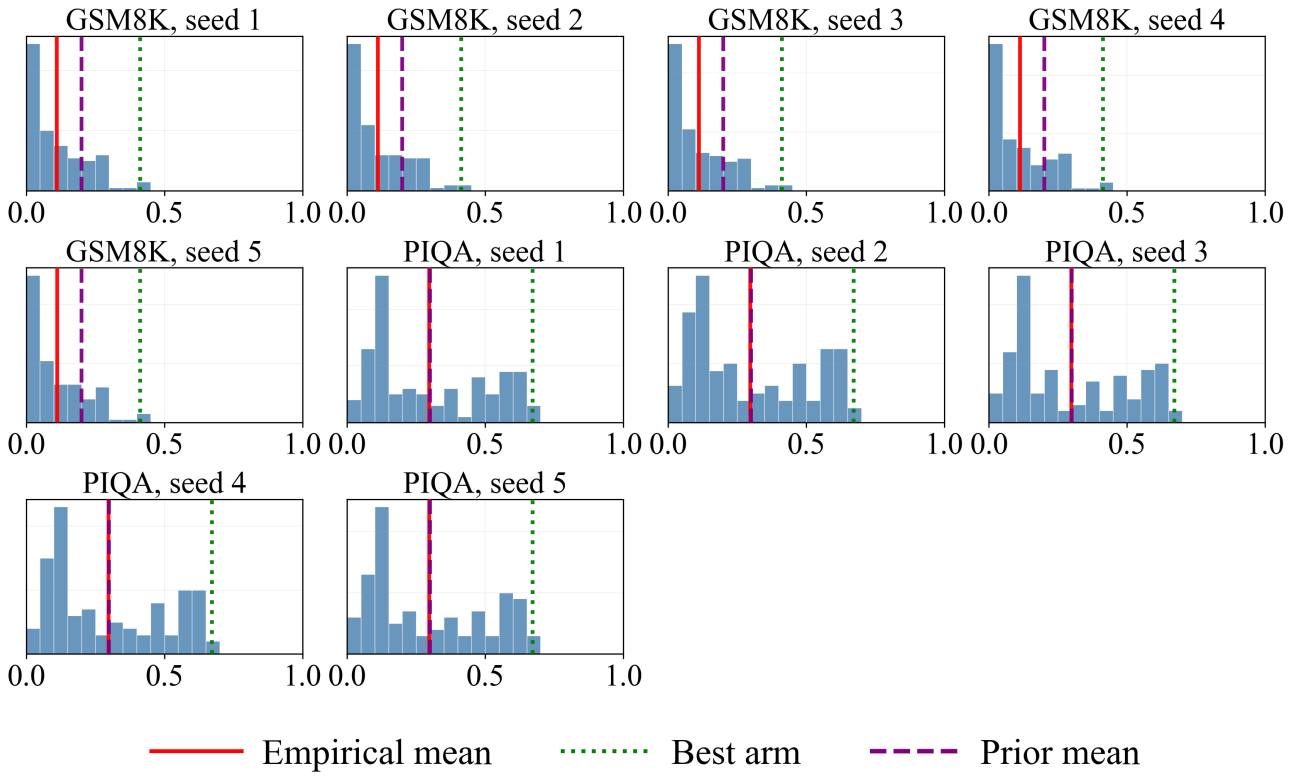


Figure 4. Distribution of per-arm quality for GSM8K and PIQA across five random seeds. GSM8K has 122 arms and 1,000 examples, while PIQA has 103 arms and 1,000 examples. Each subplot corresponds to one dataset–seed pair; bars show the histogram of arm quality, defined as mean accuracy over queried examples. Vertical lines mark the empirical mean (red), best arm (green), and prior mean (purple). The distributions are highly similar across seeds, suggesting that the observed variation is small when the only source of randomness is the LLM response generation.

MMLU. For MMLU, we treat each subject as a separate problem instance. We divide the subjects into three difficulty buckets according to the empirical mean arm quality of the subject. The intuition is that if the mean arm quality is high, then the subject is easier for the collection of LLM configurations and prompting techniques; conversely, a lower mean arm quality indicates a harder subject. We refer to these buckets as high-, medium-, and low-prior buckets, with prior means $\mu_0 = 0.75$, $\mu_0 = 0.6$, and $\mu_0 = 0.4$, respectively. Figures 5–7 show the per-arm quality distributions for the three buckets.

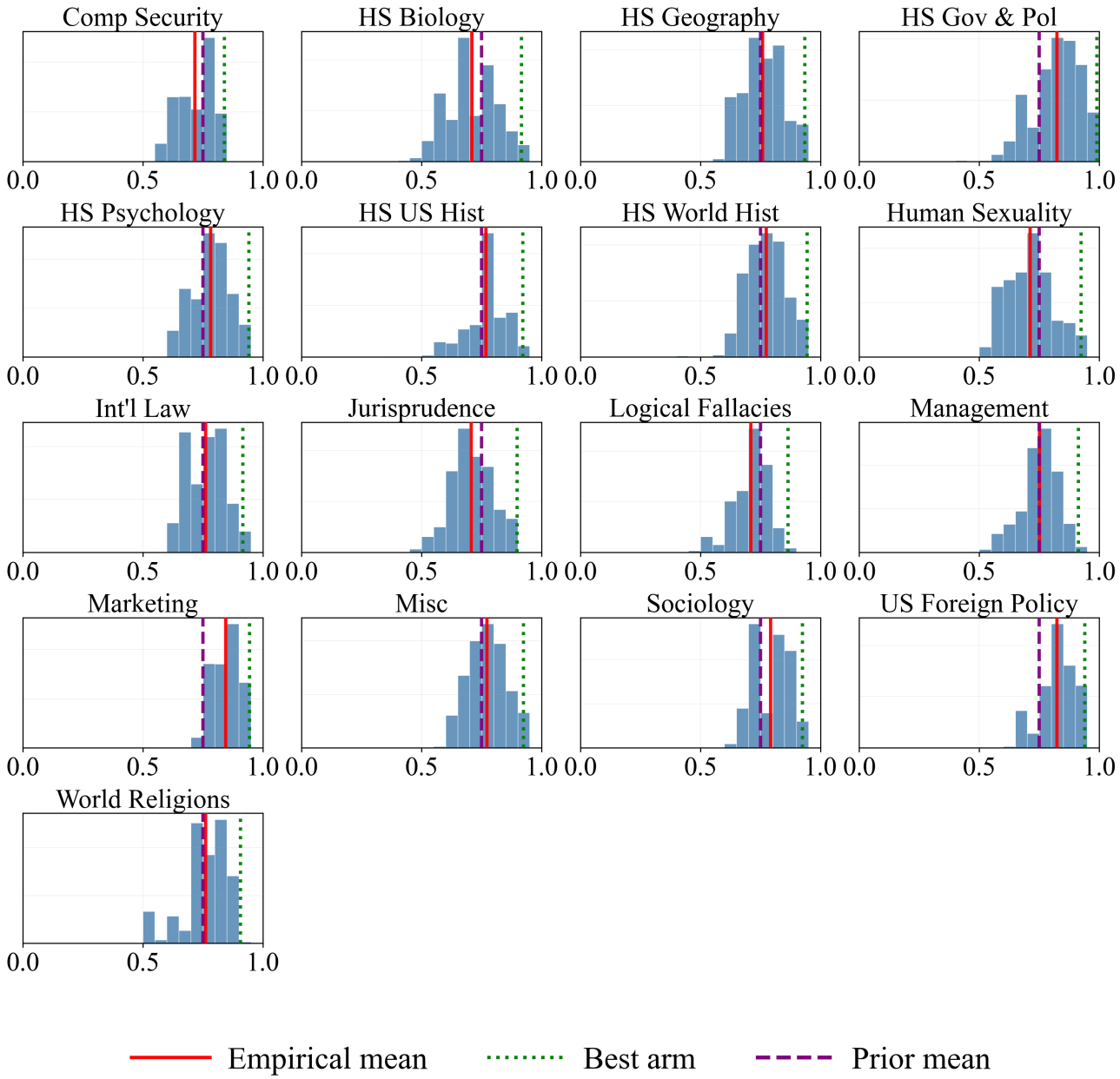


Figure 5. Easy subjects: Distribution of per-arm quality for MMLU subjects assigned to the high informative-prior bucket ($\mu_0 = 0.75$). Each subplot corresponds to one subject; bars show the histogram of arm quality, defined as mean accuracy over questions. Vertical lines mark the empirical mean (red), best arm (green), and prior mean (purple).

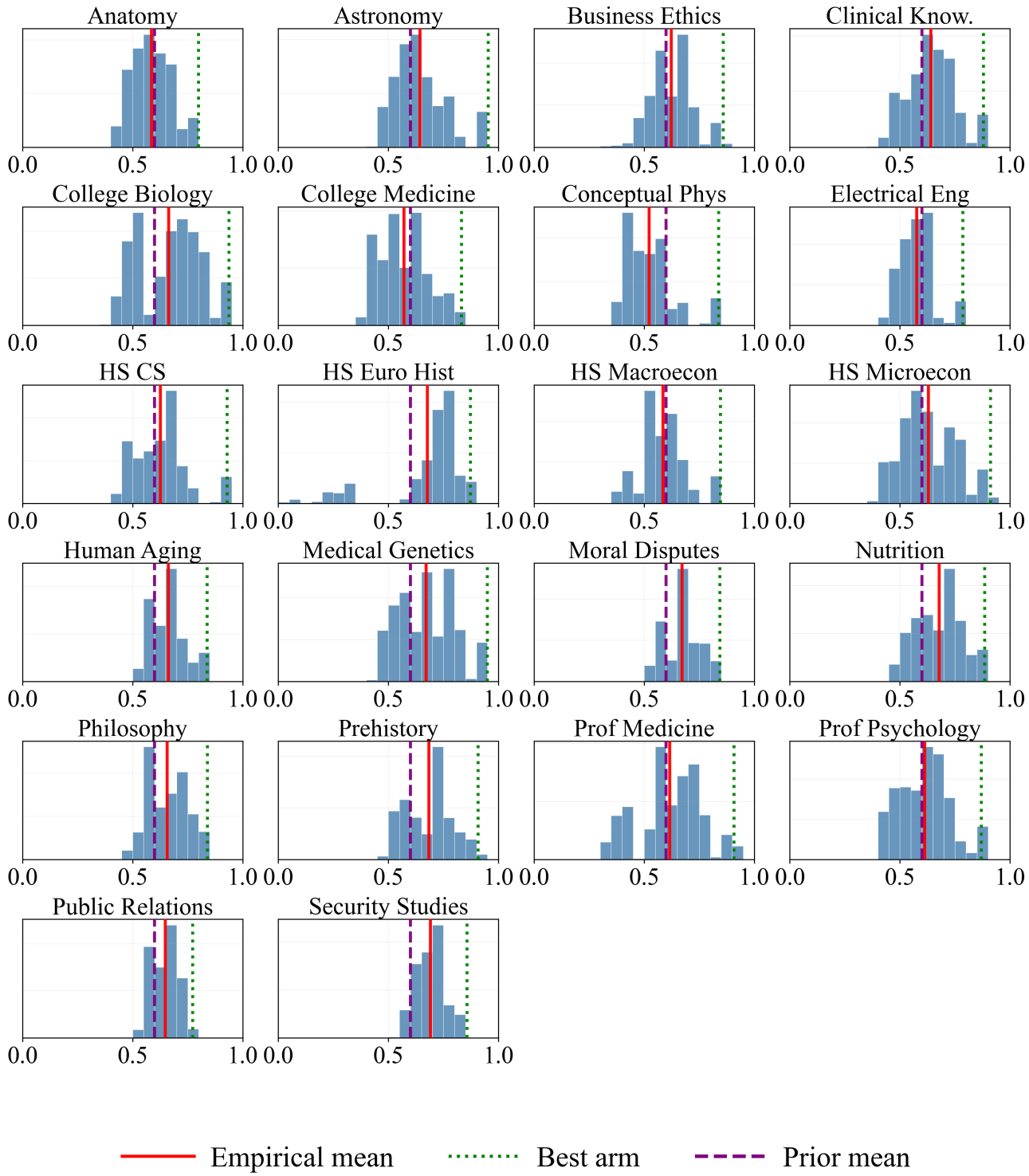


Figure 6. Medium subjects: Distribution of per-arm quality for MMLU subjects assigned to the medium informative-prior bucket ($\mu_0 = 0.6$). Each subplot corresponds to one subject; bars show the histogram of arm quality, defined as mean accuracy over questions. Vertical lines mark the empirical mean (red), best arm (green), and prior mean (purple).

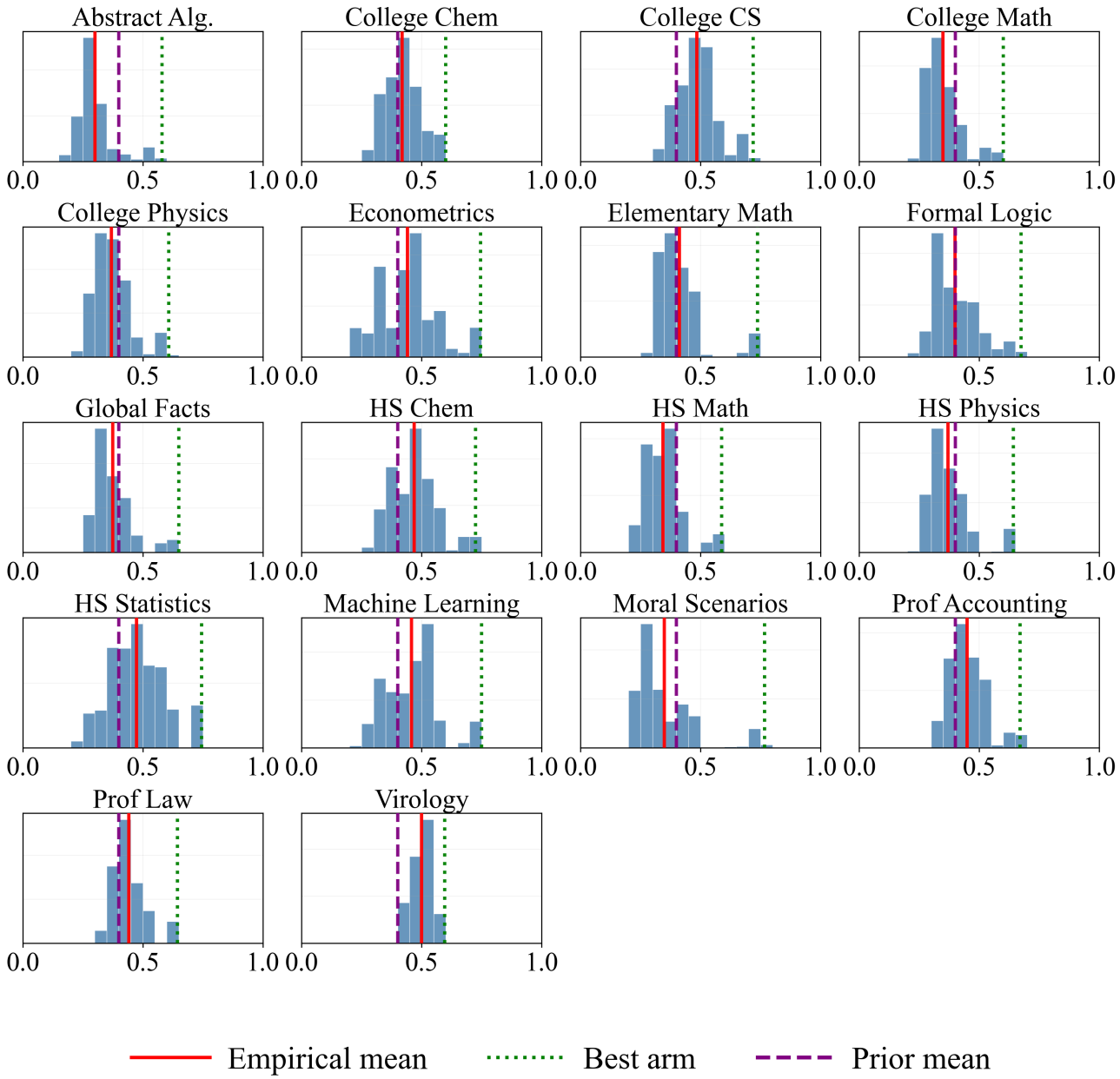


Figure 7. Hard subjects: Distribution of per-arm quality for MMLU subjects assigned to the low informative-prior bucket ($\mu_0 = 0.4$). Each subplot corresponds to one subject; bars show the histogram of arm quality, defined as mean accuracy over questions. Vertical lines mark the empirical mean (red), best arm (green), and prior mean (purple).

Table 1. MMLU datasets used in our evaluation. Each dataset is evaluated across 15 LLM configurations and 100 prompting techniques, so the number of rows is 1500 times the number of benchmark examples. Difficulty is based on prior accuracy: datasets with high prior accuracy are labeled easy, those with medium prior accuracy are labeled medium, and those with low prior accuracy are labeled hard. The size category groups datasets by the number of benchmark examples.

Dataset	Rows	Difficulty	Size	Full Name
Abstract Alg.	1500×100	hard	small	Abstract Algebra
Anatomy	1500×135	medium	small	Anatomy
Astronomy	1500×152	medium	medium	Astronomy
Business Ethics	1500×100	medium	small	Business Ethics
Clinical Know.	1500×265	medium	medium	Clinical Knowledge
College Biology	1500×144	medium	small	College Biology
College Chem	1500×100	hard	small	College Chemistry
College CS	1500×100	hard	small	College Computer Science
College Math	1500×100	hard	small	College Mathematics
College Medicine	1500×173	medium	medium	College Medicine
College Physics	1500×102	hard	small	College Physics
Comp Security	1500×100	easy	small	Computer Security
Conceptual Phys	1500×235	medium	medium	Conceptual Physics
Econometrics	1500×114	hard	small	Econometrics
Electrical Eng	1500×145	medium	small	Electrical Engineering
Elementary Math	1500×378	hard	medium	Elementary Mathematics
Formal Logic	1500×126	hard	small	Formal Logic
Global Facts	1500×100	hard	small	Global Facts
HS Biology	1500×310	easy	medium	High School Biology
HS Chem	1500×203	hard	medium	High School Chemistry
HS CS	1500×100	medium	small	High School Computer Science
HS Euro Hist	1500×165	medium	medium	High School European History
HS Geography	1500×198	easy	medium	High School Geography
HS Gov & Pol	1500×193	easy	medium	High School Government and Politics
HS Macroecon	1500×390	medium	medium	High School Macroeconomics
HS Math	1500×270	hard	medium	High School Mathematics
HS Microecon	1500×238	medium	medium	High School Microeconomics
HS Physics	1500×151	hard	medium	High School Physics
HS Psychology	1500×545	easy	large	High School Psychology
HS Statistics	1500×216	hard	medium	High School Statistics
HS US Hist	1500×204	easy	medium	High School US History
HS World Hist	1500×237	easy	medium	High School World History
Human Aging	1500×223	medium	medium	Human Aging
Human Sexuality	1500×131	easy	small	Human Sexuality
Int'l Law	1500×121	easy	small	International Law
Jurisprudence	1500×108	easy	small	Jurisprudence
Logical Fallacies	1500×163	easy	medium	Logical Fallacies
Machine Learning	1500×112	hard	small	Machine Learning
Management	1500×103	easy	small	Management
Marketing	1500×234	easy	medium	Marketing
Medical Genetics	1500×100	medium	small	Medical Genetics
Misc	1500×783	easy	large	Miscellaneous
Moral Disputes	1500×346	medium	medium	Moral Disputes
Moral Scenarios	1500×895	hard	large	Moral Scenarios
Nutrition	1500×306	medium	medium	Nutrition
Philosophy	1500×311	medium	medium	Philosophy
Prehistory	1500×324	medium	medium	Prehistory
Prof Accounting	1500×282	hard	medium	Professional Accounting
Prof Law	1500×1534	hard	large	Professional Law
Prof Medicine	1500×272	medium	medium	Professional Medicine
Prof Psychology	1500×612	medium	large	Professional Psychology
Public Relations	1500×110	medium	small	Public Relations
Security Studies	1500×245	medium	medium	Security Studies
Sociology	1500×201	easy	medium	Sociology
US Foreign Policy	1500×100	easy	small	US Foreign Policy
Virology	1500×166	hard	medium	Virology
World Religions	1500×171	easy	medium	World Religions

C. Experiment Setup and Implementation Details

Computing environment. Experiments were run on CPU-only computing resources using precomputed response matrices rather than online LLM inference. The response matrix serves as the offline ground-truth oracle: during bandit simulation, each policy observes only the selected $(\text{arm}, \text{question})$ entries, while per-arm matrix means are used to compute simple regret. For the BO baselines, the response matrices are converted into configuration-level inputs, where evaluating one candidate reveals its aggregate score. Reported per-batch runtime therefore measures policy-side allocation overhead rather than model-inference time.

Experimental repetitions. For GSM8K and PIQA, we run each adaptive policy on each of the five independently generated response matrices. For every response matrix, we perform 20 independent algorithm trials, each with independently randomized example orders and algorithmic randomness. We report averages over all matrix–trial pairs. For MMLU, we run 20 randomized algorithm trials for each subject and aggregate results across the corresponding set of subjects. The BO baselines use the same randomized-trial structure.

Baseline initialization and warm-up. For a response matrix with K arms and N examples, the BO baselines use a configuration-level random-initialization phase. In our main BO runs, a 5% random initialization means drawing approximately $0.05K$ arms uniformly without replacement and observing the full rows for those arms, i.e., evaluating the selected configurations on all N examples before the acquisition-driven phase begins. This corresponds to 6 initial configurations for GSM8K ($K = 122$), 5 for PIQA ($K = 103$), and 75 for each MMLU subject ($K = 1500$). Under the nominal 10% BO budget, the corresponding total configuration budgets are 12, 10, and 150 configurations, respectively. This convention is distinct from the low-rank-factorization warm-up in BanditEval (Zhou et al., 2025). There, UCB-E-LRF first draws T_0 individual arm–example pairs uniformly from the $K \times N$ response matrix, and the default experimental setting uses $T_0 = 0.05KN$. The standalone LRF baseline in that paper is also written with a T_0 entry-level warm-up parameter, so any such warm-up should be interpreted as a percentage of response-matrix entries rather than a percentage of fully evaluated arms.

Evaluation costs. The cost-aware bandit experiments assign each arm a batch cost proportional to the estimated cost of running that model configuration on B benchmark examples. An arm is a model plus sampling or prompting configuration; when several arms share a base model, they share the model-level price, while the batch multiplier accounts for how many examples are queried at that allocation step. Depending on the available metadata, this can be computed from API pricing, input/output token counts, generated-token counts, wall-clock latency, or a normalized proxy such as model size. For the BO baselines, each converted input row corresponds to a complete configuration, and its cost is the estimated full-evaluation cost of that configuration on the corresponding benchmark or MMLU subject. The Gittins dynamic program uses costs rescaled to the same utility units as the index computation, while the BO baselines use configuration-level costs for cost-aware acquisition and budget accounting.

Table 2. Model-level evaluation costs used for GSM8K and PIQA. Costs are reported in USD per 1M tokens. GSM8K uses an assumed input-to-output token ratio of $1:2$, so the evaluation cost is computed as input cost plus twice the output cost. PIQA uses input-only pricing. These model-level costs are used to construct the per-arm cost vectors for the cost-aware experiments.

Model	Input cost	Output cost	GSM8K cost	PIQA cost
CodeLlama	0.30	0.30	0.90	0.30
Gemma-7B	0.20	0.20	0.60	0.20
GPT-2	0.10	0.10	0.30	0.10
GPT-2 Large	0.10	0.10	0.30	0.10
LLaMA2-7B	0.20	0.20	0.60	0.20
Llemma-7B	0.80	1.20	3.20	0.80
Mistral-7B	0.05	0.20	0.45	0.05
Phi-2	0.05	0.10	0.25	0.05
StarCoder2-7B	0.20	0.20	0.60	0.20
Tulu	0.20	0.20	0.60	0.20
Tulu2	0.20	0.20	0.60	0.20

Gaussian approximation. For benchmark scores grouped in batches, a batch of B responses produces an empirical mean score. The default binary-accuracy observation variance is $\tau^2 = 1/(4B)$, the worst-case binary variance divided by the

Table 3. Model-level input costs used for MMLU. Costs are reported in USD per 1M input tokens. MMLU uses input-only pricing. Each MMLU subject contains 1500 arms, corresponding to 15 models paired with 100 prompt configurations; therefore, each model-level input cost is repeated across the corresponding prompt arms.

MMLU model	Input cost
Llama-3-8B	0.05
Llama-3-8B-Instruct	0.03
Llama-3-70B-Instruct	0.51
CodeLlama-34B-Instruct	0.776
FLAN-T5-XL	0.60
FLAN-T5-XXL	1.80
FLAN-UL2	5.00
Merlinite-7B	0.60
Mixtral-8x7B-Instruct-v0.1	0.54
Mistral-7B-Instruct-v0.2	0.14
Gemma-7B	0.20
Gemma-7B-IT	0.07
Falcon-40B	0.84
Mistral-7B-v0.1	0.11
Falcon-180B	1.25

batch size. For other score types, we estimate τ^2 from pilot batches or historical response matrices, and we also consider plug-in variances in which τ^2 is updated from the current posterior mean.

Prior settings. All arms use the same prior mean and variance, representing a shared belief before benchmark-specific evidence is collected. Table 4 lists the general default prior and the data-specific common priors used by Gittins-G and Gittins-S. Figures 5, 6, and 7 show the MMLU latent-accuracy distributions used to motivate the informative prior buckets.

Table 4. Prior settings used in the Gittins-index experiments. We report Gaussian priors as $\mathcal{N}(\mu_0, v_0)$. The default prior is a general accuracy-scale choice. Dataset-specific priors provide shared benchmark-level information without using arm-specific prior means.

Benchmark / group	Prior	Description
All benchmarks	$\mathcal{N}(0.5, 0.04)$	General default prior
GSM8K	$\mathcal{N}(0.2, 0.01)$	Dataset-specific common prior
PIQA	$\mathcal{N}(0.3, 0.02)$	Dataset-specific common prior
MMLU low-accuracy bucket	$\mathcal{N}(0.4, 0.02)$	Harder subjects
MMLU medium-accuracy bucket	$\mathcal{N}(0.6, 0.02)$	Medium-difficulty subjects
MMLU high-accuracy bucket	$\mathcal{N}(0.75, 0.01)$	Easier subjects

D. Additional Experiment Results

D.1. Ablation Studies

We include three ablations to separate the statistical and cost-sensitive components of the Gittins index policy.

Choice of common prior. We compare the common priors listed in Table 4. This ablation tests whether Bayesian allocation gains come from useful shared prior information or from the index rule alone, without giving different arms different prior means.

Batch size. For UCB-E and the Gittins variants, we vary the batch size B used to construct empirical batch-mean observations. Smaller batches provide more frequent adaptation but noisier observations; larger batches reduce Gaussian approximation error and posterior noise but make each allocation decision coarser. UCB-E-LRF uses $B = 32$ in the reported settings, while BO baselines operate at the configuration level and therefore do not use a bandit batch size. Table 5 summarizes the batch-size grids used for GSM8K, PIQA, and the MMLU subject-size buckets.

Cost-scaling factor. The Gittins index policy uses a cost-scaling factor to convert evaluation costs into the continuation penalty used by the index computation. In the cost-aware setting, this factor multiplies the original per-evaluation costs; in

Table 5. Batch-size settings used in the simple-regret experiments. For UCB-E and both Gittins variants, batch size B denotes the number of benchmark examples queried for the selected arm at each allocation step. UCB-E-LRF uses $B = 32$ in all reported settings. BO baselines operate at the configuration level, so evaluating one candidate reveals its aggregate score and does not use a bandit batch size. For MMLU, subject matrices are grouped by the number of benchmark examples, and the batch-size grid for UCB-E and Gittins is chosen according to this subject-size bucket.

Benchmark / group	Matrices	Example-count rule	Batch sizes
GSM8K	5 response matrices	1000 examples each	$B \in \{8, 16, 32\}$
PIQA	5 response matrices	1000 examples each	$B \in \{8, 16, 32\}$
MMLU small	22 subject matrices	$n_{\text{examples}} \leq 150$	$B \in \{2, 4, 8\}$
MMLU medium	30 subject matrices	$151 \leq n_{\text{examples}} \leq 400$	$B \in \{4, 8, 16\}$
MMLU large	5 subject matrices	$n_{\text{examples}} > 400$	$B \in \{8, 16, 32\}$

the unit-cost setting, it multiplies a common unit cost. Varying this factor changes how aggressively the policy stops: larger values make evaluations effectively more expensive and encourage earlier stopping, while smaller values encourage more exploration. This exposes the tradeoff between simple regret at stopping and total evaluation cost.

D.2. Per-Subject MMLU Results

We provide additional MMLU results under informative priors in Figures 8–13. The subjects are grouped into easy, medium, and hard groups according to their empirical mean arm quality. The corresponding prior means are $\mu_0 = 0.75$, $\mu_0 = 0.6$, and $\mu_0 = 0.4$, respectively. For each bucket, we report both unit-cost results, where the horizontal axis is the number of evaluations, and cost-aware results, where the horizontal axis is cumulative evaluation cost. Across the MMLU subjects, the Gittins-based policies generally improve over UCB-E, but the stronger variant depends on the difficulty bucket. Gittins-G performs particularly well on easy subjects, while Gittins-S is more consistently effective on medium and hard subjects, suggesting that a data-specific prior is more helpful when the average task difficulty is lower. The cost-aware setting further amplifies the advantage of the Gittins-based methods, especially in the medium and hard buckets.

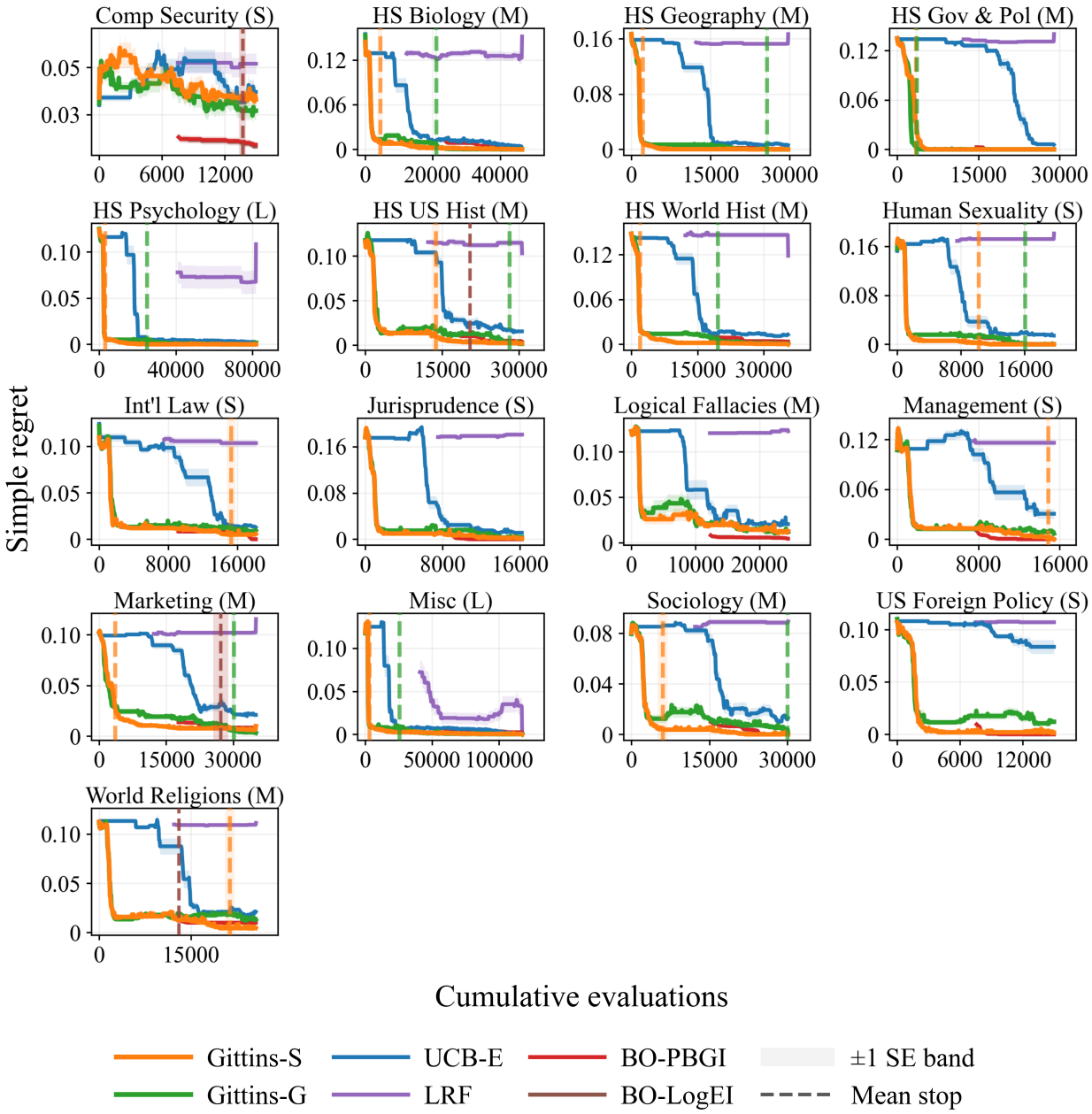


Figure 8. MMLU unit-cost results for easy subjects (high prior bucket). Simple regret is plotted against cumulative evaluations. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ throughout. Shaded regions denote ± 1 standard errors across seeds. Dashed vertical lines mark mean stopping times, with faint vertical bands showing ± 1 standard errors. On easier subjects, the Gittins variants usually reach low simple regret rapidly and often outperform UCB-E. BO is competitive on several tasks but is less uniformly strong, while LRF is generally less effective and less stable.

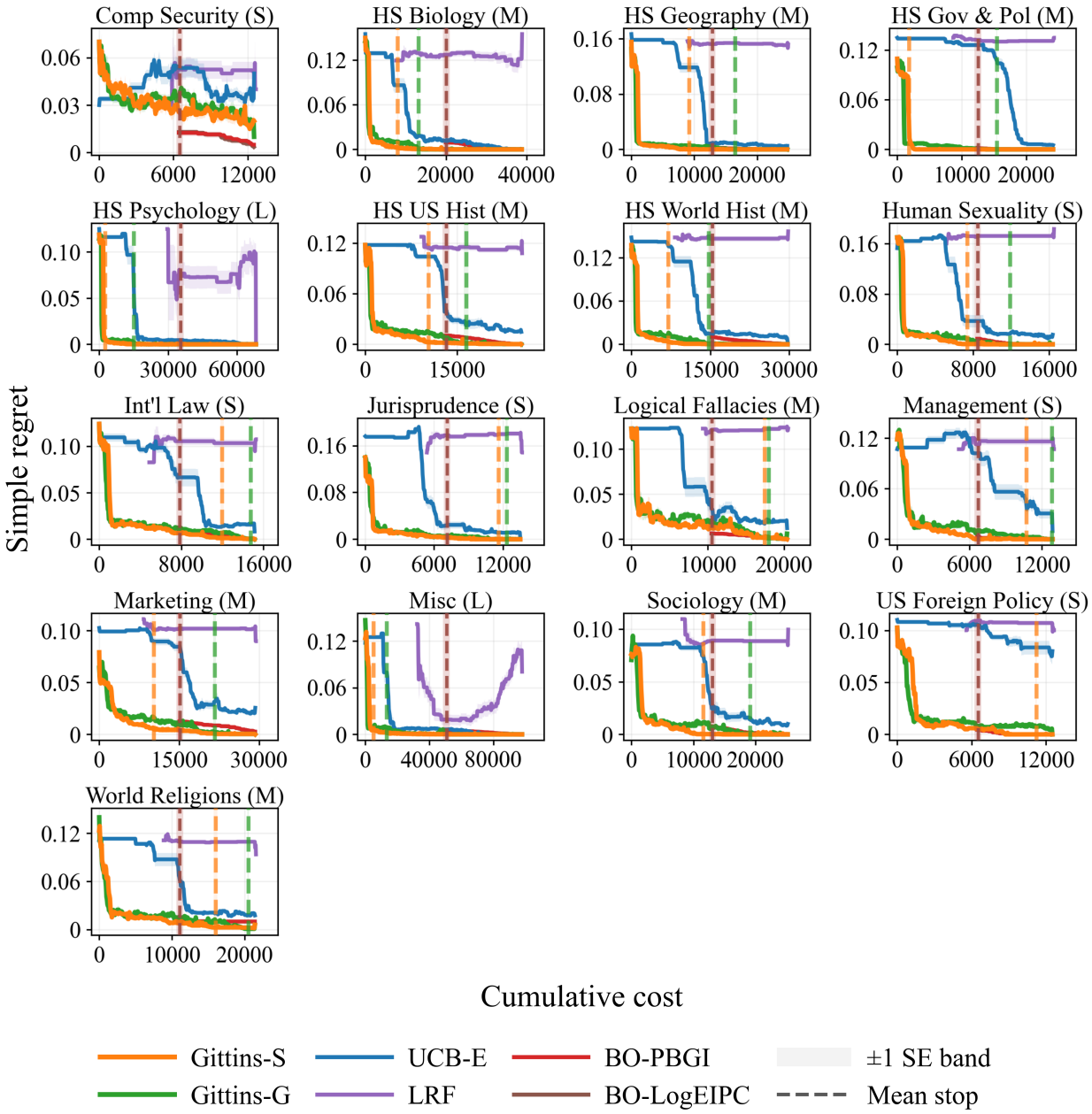


Figure 9. MMLU cost-aware results for easy subjects (high prior bucket). Simple regret is plotted against cumulative cost. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ throughout. Shaded regions denote ± 1 standard errors across seeds. Dashed vertical lines mark mean stopping times, with faint vertical bands showing ± 1 standard errors. In this cost-aware setting, Gittins variants typically reach low simple regret at relatively small cumulative cost. BO is strong on some subjects but is not consistently better than Gittins, and LRF remains comparatively weak and unstable.

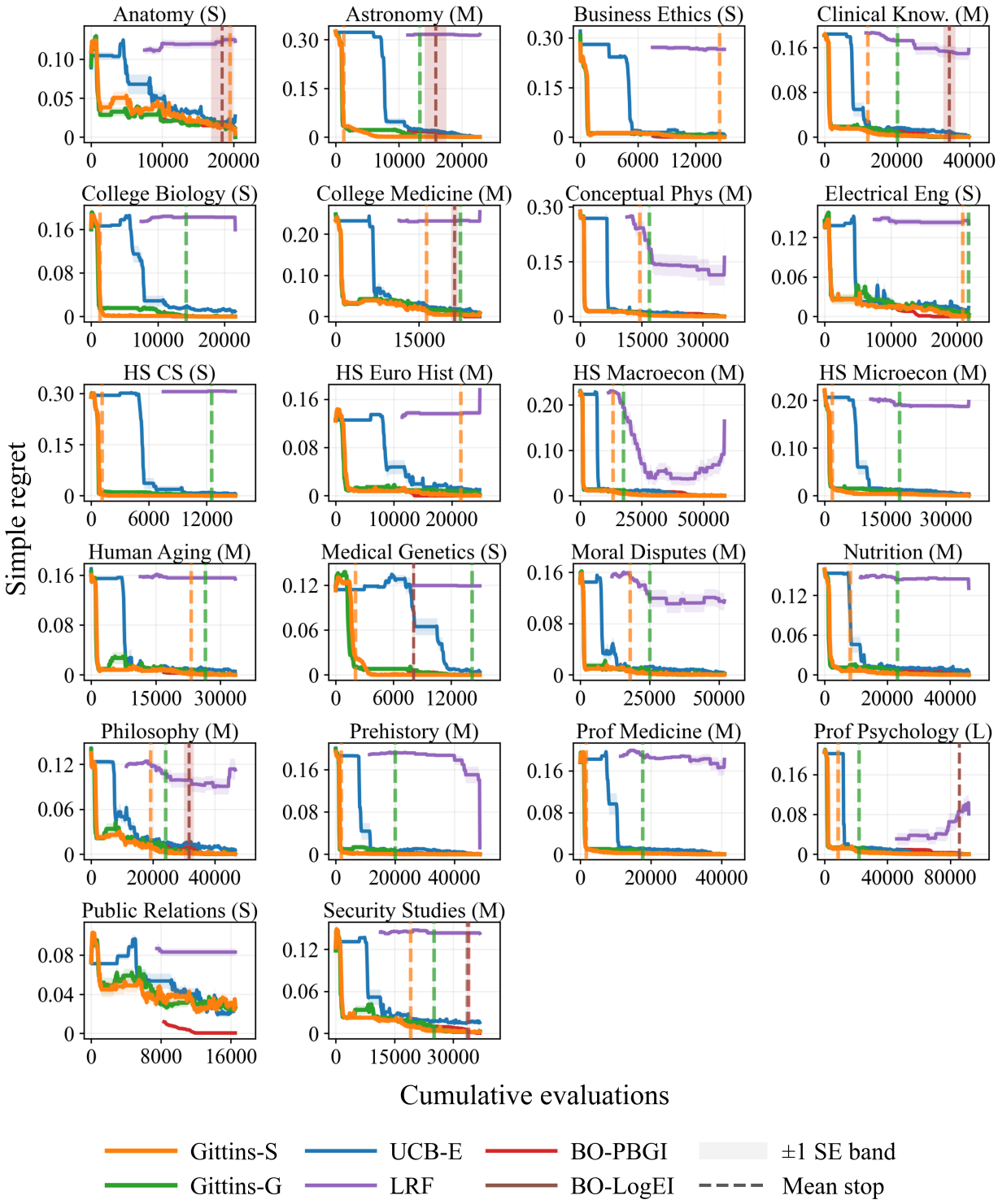


Figure 10. MMLU unit-cost results for medium-difficulty subjects (medium prior bucket). Simple regret is plotted against cumulative evaluations. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ throughout. Shaded regions denote ± 1 standard errors across seeds. Dashed vertical lines mark mean stopping times, with faint vertical bands showing ± 1 standard errors. Across these mixed-difficulty subjects, Gittins-S is often the most reliable variant, reaching low simple regret earlier than UCB-E on many tasks. Gittins-G remains competitive on several subjects, and BO performs well when the arm-level configuration structure is informative. LRF is generally less effective.

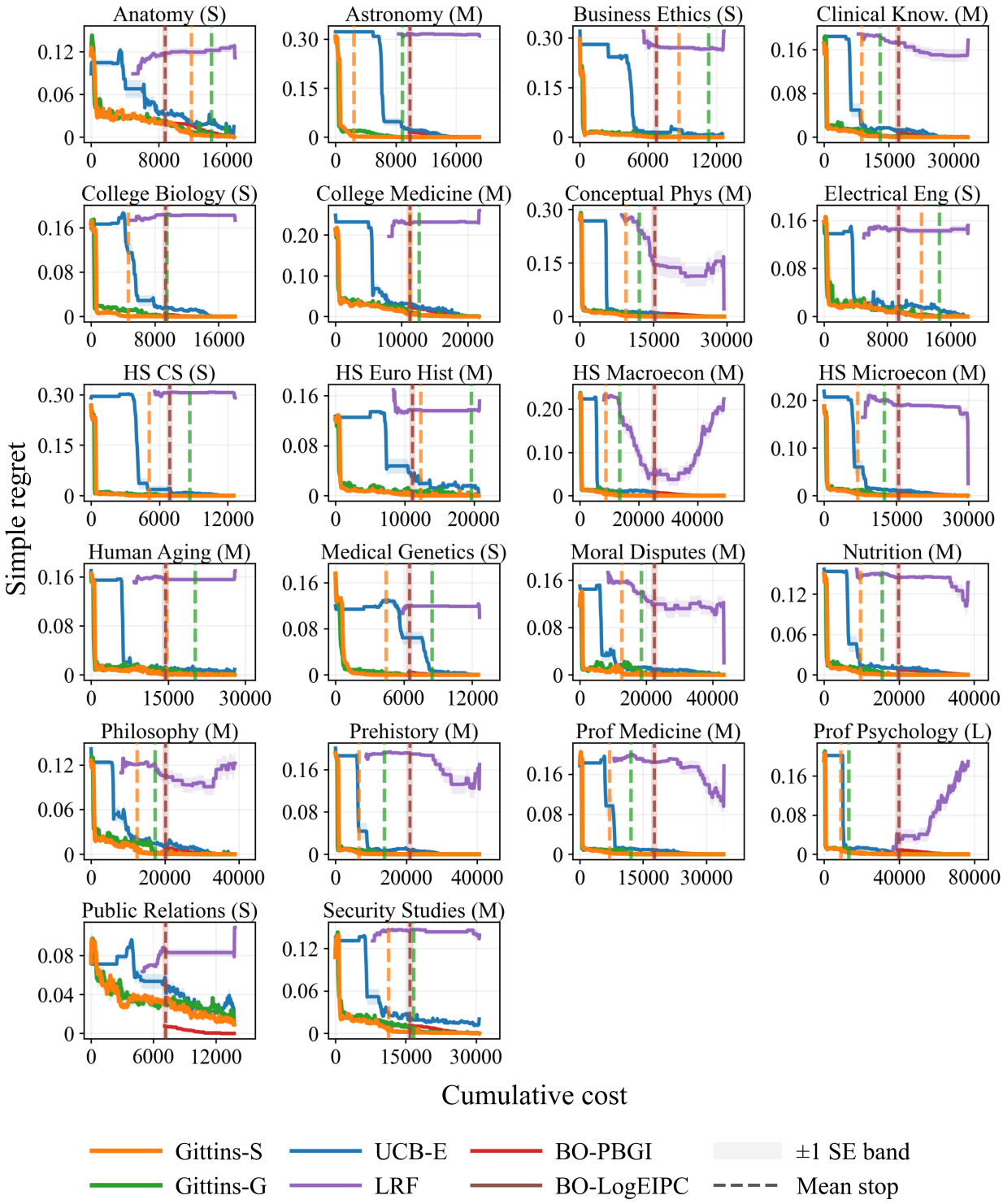


Figure 11. MMLU cost-aware results for medium-difficulty subjects (medium prior bucket). Simple regret is plotted against cumulative cost. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ throughout. Shaded regions denote ± 1 standard errors across seeds. Dashed vertical lines mark mean stopping times, with faint vertical bands showing ± 1 standard errors. The Gittins variants usually achieve low simple regret with substantially smaller cumulative cost than UCB-E. Gittins-S is often the strongest and most stable method in this bucket, while BO is competitive on selected subjects. LRF remains less stable and typically underperforms.

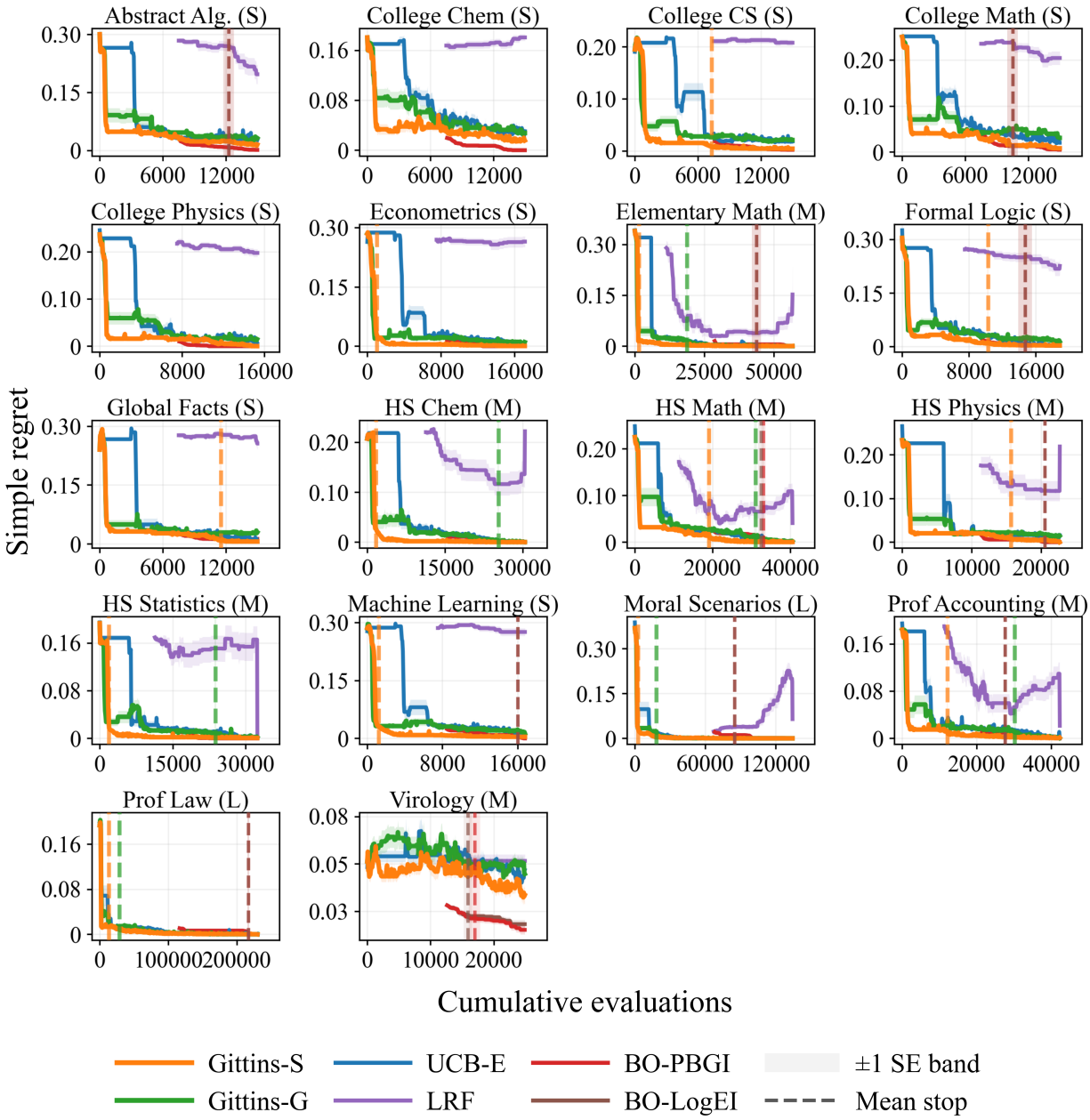


Figure 12. MMLU unit-cost results for hard subjects (low prior bucket). Simple regret is plotted against cumulative evaluations. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ throughout. Shaded regions denote ± 1 standard errors across seeds. Dashed vertical lines mark mean stopping times, with faint vertical bands showing ± 1 standard errors. On hard subjects, Gittins-S is usually the most effective and stable method, often reaching low simple regret earlier than UCB-E and more consistently than Gittins-G. BO is strong on a few subjects but is not uniformly dominant, while LRF is substantially less reliable.

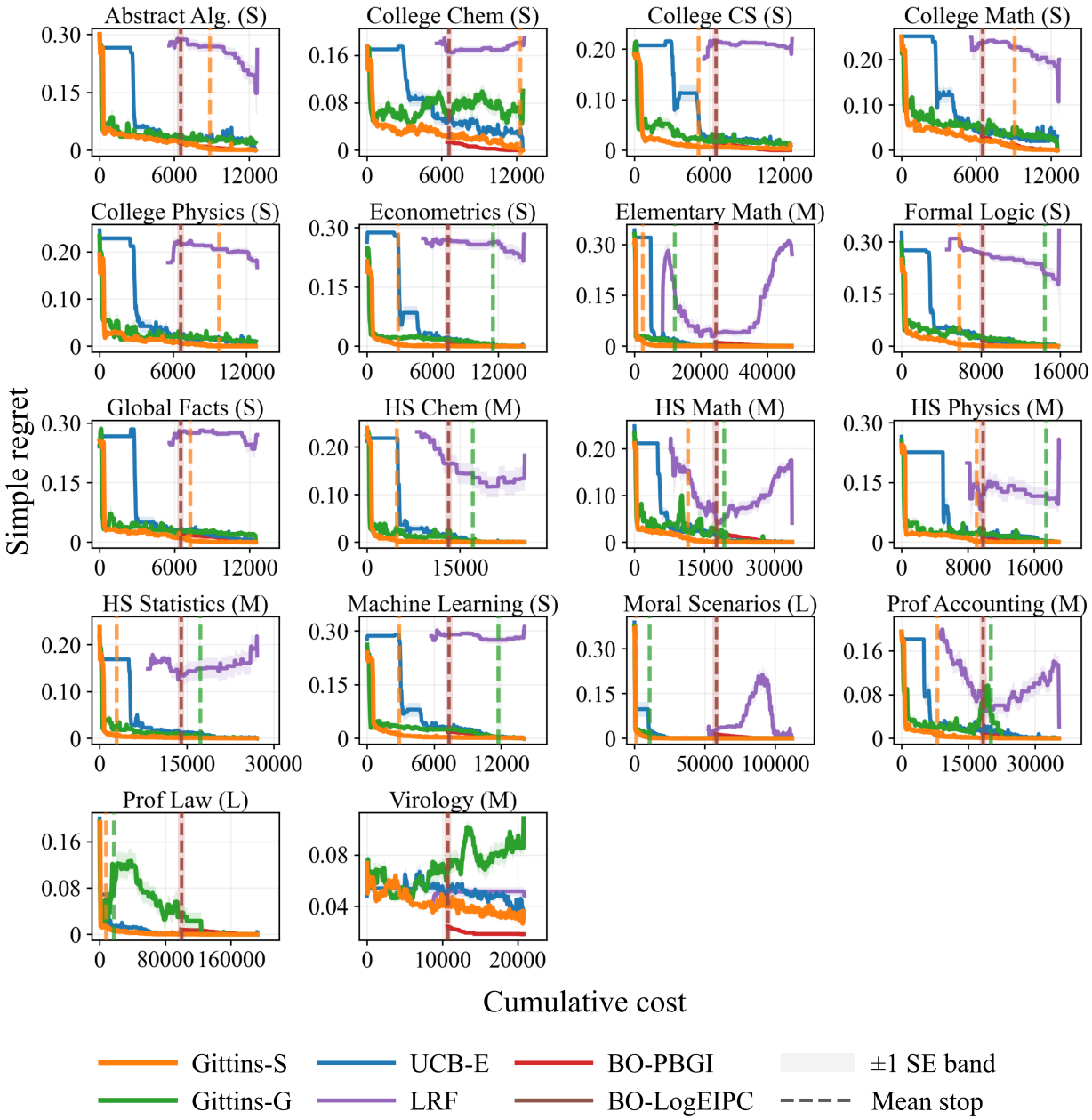


Figure 13. MMLU cost-aware results for hard subjects (low prior bucket). Simple regret is plotted against cumulative cost. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ throughout. Shaded regions denote ± 1 standard errors across seeds. Dashed vertical lines mark mean stopping times, with faint vertical bands showing ± 1 standard errors. In this harder cost-aware setting, Gittins-S generally provides the strongest overall performance, reaching low simple regret with smaller cumulative cost than UCB-E on most subjects. Gittins-G can be competitive on some tasks but is less stable under heterogeneous costs. BO performs well on selected tasks, while LRF remains the least reliable baseline.

D.3. Computational Overhead Comparison

Timing protocol. Wall-clock timing measurements report the per-batch decision time of each allocation policy, measured inside the simulation loop after the response matrix and prior parameters have been loaded. The reported time covers the policy's score computation and batch selection, including UCB-E confidence bonuses, the low-rank-factorization step in UCB-E-LRF, and Gittins-index scoring using the precomputed root table. It excludes LLM inference, ground-truth loading, response-matrix construction, and the one-time Gittins root-table precomputation, which is recorded separately. Because the response matrix is precomputed, a "pull" is implemented as an array read and contributes negligible time relative to the policy decision step.

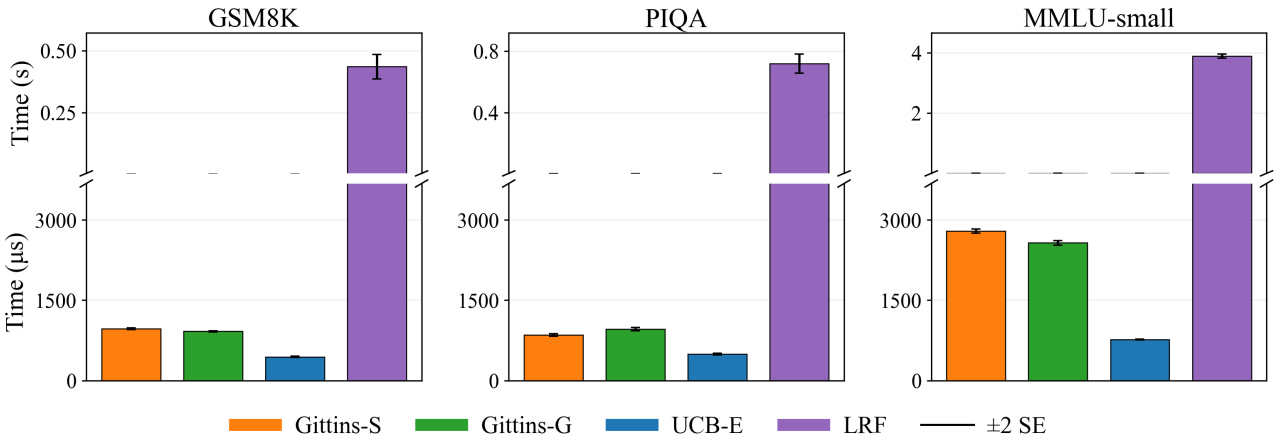


Figure 14. Timing comparison under the unit-cost setting. Each panel reports the per-batch decision time for GSM8K, PIQA, and MMLU-small. Bars show the median across completed runs, and black error bars indicate ± 2 standard errors computed across runs. GSM8K and PIQA use batch size $B = 16$, while MMLU-small aggregates subjects with 100–150 examples and uses batch size $B = 4$. The timing measures policy-side batch-selection overhead rather than model inference time; one-time Gittins root-table precomputation is excluded and recorded separately. We see that the online per-batch overhead of the Gittins variants is close to UCB-E and much smaller than UCB-E-LRF, whose low-rank factorization step dominates runtime, especially on MMLU-small.