

THE GITTINS INDEX DESIGN PRINCIPLE FOR COST-AWARE BAYESIAN DECISION-MAKING UNDER UNCERTAINTY

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Qian Xie

August 2026

© 2026 Qian Xie
ALL RIGHTS RESERVED

THE GITTINS INDEX DESIGN PRINCIPLE FOR COST-AWARE BAYESIAN DECISION-MAKING UNDER UNCERTAINTY

Qian Xie, Ph.D.

Cornell University 2026

Selecting high-performing designs through costly evaluations is a crucial problem in automated machine learning and adaptive experimentation, with applications including hyperparameter tuning, control design, scientific discovery, and LLM evaluation. In these settings, decision makers adaptively gather information while balancing potential performance improvements against cumulative evaluation costs under uncertainty.

This dissertation studies cost-aware Bayesian decision-making based on the Gittins index. The first part of the dissertation connects cost-aware Bayesian optimization to the Pandora’s Box problem from economics, where the Gittins index policy is Bayesian-optimal. This perspective leads to a novel and principled policy class for selecting the next costly evaluation and deciding when further evaluation is no longer worthwhile, supported by theoretical guarantees and validated through competitive empirical performance on black-box optimization tasks.

The dissertation then considers a broader class of Bayesian decision problems in which information about each option is revealed through richer partial observations collected over multiple costly stages. I first study an application-driven special case, best model configuration identification in efficient LLM evaluation, where evaluating a configuration on benchmark examples produces noisy samples informative about its latent mean performance. I then generalize to settings where evaluations can be paused and resumed, and partial observations such as neural network training

curves drift toward and gradually reveal an option's latent mean. Building on connections to Bayesian bandits and Markov chain selection problems, where Gittins index policies are also known to be Bayesian-optimal, the dissertation develops computationally efficient Gittins index methods for adaptively allocating evaluations across options with varying costs.

Together, these works illustrate how Gittins indices can serve as a cost-aware design principle for Bayesian decision-making under uncertainty, including Bayesian optimization and bandit-like Bayesian decision problems with multi-stage feedback.

BIOGRAPHICAL SKETCH

Qian Xie was born and raised in Guangzhou, China. Her interest in mathematics began through her participation in mathematical olympiads during primary and middle school. She later majored in computer science in the Yao Class at Tsinghua University, where she had the opportunity to learn about the frontiers of computer science and its intersections with other fields. During college, she also became interested in mathematical modeling, which led her to pursue graduate study in transportation planning and engineering at New York University, where her research focused on queueing control. She later entered the Ph.D. program in operations research at Cornell University. At Cornell, her research has focused on Bayesian decision-making under uncertainty, using probabilistic methods from operations research to develop principled, cost-aware, and computationally efficient approaches to (automated) machine learning.

This dissertation is dedicated to my family, particularly my partner Nairen, whose support makes this work possible.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all those whose guidance, support, and companionship made this degree possible.

First and foremost, I would like to thank my advisor, Professor Ziv Scully, for his guidance and support. Although I had the privilege of working with Ziv for only three and a half years, his influence on me has been profound. He is the most passionate teacher and presenter I have ever encountered. When I first began working with him, his approach to advising often felt like teaching: during our first few meetings, he patiently walked me through the interesting Pandora’s box problem and the cool tool of Gittins indices. I was so captivated that I “abandoned” queueing—even though I had initially planned to continue working in that area—and the wheel of fate began to turn. Throughout our work together, Ziv has shown a remarkable ability to make key observations about a problem, helping me understand it both more clearly and deeply while inspiring me to approach it from a new perspective. Beyond his guidance in research, his example and advice have greatly improved my presentation skills.

My sincere thanks also go to my committee member, Professor Peter Frazier. Although Peter is not officially my advisor, he has generously mentored me on several projects. He is one of the most dedicated and engaged senior professors I have encountered. I still remember the times when he stayed up late to help us with paper submissions and rebuttals. During meetings, he was always willing to work through ideas with us, even when the topic was new to him, as with our work on LLM inference. I learned a great deal from our brainstorming discussions and the insightful examples he offered. His interests span Bayesian optimization, bandits, AI for science, and online platforms, and our discussions gave me the opportunity to explore many research topics.

I also benefited from many other mentors. My committee member, Professor David Bindel, asked insightful questions at my A-exam and B-exam that encouraged me to examine my research more carefully. I am grateful to Alexander Terenin, who introduced me to Bayesian optimization and mentored me on several projects, including my first machine learning paper as a first author. When I had little experience with software engineering or research poster design, he patiently guided me through both. My internship mentor, Dr. Masood Mortazavi, encouraged me to explore connections between reinforcement learning and Bayesian optimization. His enduring passion for research, reflected in both his fresh insights and his enthusiasm for writing code himself, especially inspired me. I thank Raul Astudillo for answering my questions about BoTorch and Bayesian optimization benchmarks. My collaboration with Su Jia led to my first machine learning paper and gave me valuable early research experience. Finally, Professor Li Jin, my advisor at NYU, gave me a strong foundation in applied probability and my first experience conducting research and submitting a paper.

I am especially grateful to my junior collaborators, Linda Cai, Theodore Brown, and Alon Jacobson. I have “battled” alongside each of them on projects over the years. These projects did not always progress smoothly, but we worked through their challenges together, and I deeply appreciate their persistence, hard work, and companionship throughout the process. Wenye Hua at Microsoft and Tianyi Peng at Columbia introduced me to the world of AI agents through our collaboration. I also enjoyed mentoring Yueli He and Yu Yu. Being part of their research journeys was rewarding, and working with them helped me grow as a mentor.

Special thanks go to Poornima Buathong, Linda Cai, Yu Tang, Yue Hu, Shunhua Jiang, Wenzhe Fan, and Zhiyuan Tang for participating in my mock defense and practice job talks. I also thank my groupmates, George Yu, Shefali Ramakrishna,

and Amit Harlev, for attending rehearsals of my research talks. Their thoughtful questions and constructive feedback helped me refine my delivery and prepare for these important moments. I am also grateful to Professors Katya Scheinberg and Jamol Pender, as well as Mingming Wang, for helping me navigate logistical matters throughout my Ph.D.

Finally, I thank my family for being with me throughout this journey. In particular, my deepest gratitude goes to my partner, Nairen Cao, who stood by me during the most difficult period of my Ph.D. and was a constant source of support. He patiently listened to my practice talks and to the ideas and experiences I wanted to share. Although his background is in theoretical computer science, a field different from my own, he eventually became my collaborator as well. I feel incredibly fortunate to have shared both life and research with him.

TABLE OF CONTENTS

Biographical Sketch	iii
Dedication	iv
Acknowledgements	v
Table of Contents	viii
List of Tables	xii
List of Figures	xiv
1 Introduction	1
1.1 Cost-Aware Bayesian Decision-Making	2
1.1.1 From One-Shot to Multi-Stage Feedback	3
1.2 The Gittins Index Design Principle	4
1.2.1 Pandora’s Box Indices for Bayesian Optimization	5
1.2.2 Markov Chain Indices for Multi-Stage Feedback	6
1.2.3 From Analytical Models to Computable Policies	7
1.3 Organization of the Dissertation	8
I Cost-aware Bayesian Optimization with Adaptive Stop- ping via the Pandora’s Box Gittins Index	10
2 Cost-aware Bayesian Optimization via the Pandora’s Box Gittins Index	11
2.1 Introduction	11
2.2 Cost-aware Bayesian optimization	14
2.2.1 Probabilistic models and acquisition functions	15
2.2.2 Expected improvement per unit cost	16
2.3 The Pandora’s Box Gittins index for Bayesian optimization	17
2.3.1 The Pandora’s Box problem	18
2.3.2 Optimally solving Pandora’s Box	19
2.3.3 An acquisition function class for cost-aware Bayesian opti- mization	23
2.4 Experiments	27
2.4.1 Bayesian regret	29
2.4.2 Synthetic benchmarks	30
2.4.3 Empirical objectives	31
2.5 Conclusion	33
3 Cost-aware Stopping for Bayesian Optimization	34
3.1 Introduction	35
3.2 Bayesian Optimization and Adaptive Stopping	38
3.2.1 Cost-aware Bayesian Optimization	38
3.2.2 Adaptive Stopping Rules	40
3.3 A Stopping Rule Based on the Pandora’s Box Gittins Index	42

3.3.1	Theoretical Guarantees on Cost-Adjusted Regret	45
3.3.2	Practical Implementation Considerations for the PBGI/LogEIPC Stopping Rule	48
3.4	Experiments	50
3.4.1	Bayesian Regret	52
3.4.2	Automated Machine Learning Benchmarks	53
3.5	Conclusion	57

II Bandit-Like Bayesian Decision Problems with Multi-Stage Feedback via the Markov Chain Gittins Index 59

4	Efficient Cost-Aware LLM Evaluation via Bayesian Bandit Gittins Indices	60
4.1	Introduction	61
4.1.1	Related Work	63
4.2	Cost-Aware LLM Configuration Recommendation	65
4.2.1	Candidate Configurations, Examples, and Latent Performance	65
4.2.2	Adaptive Evaluation and Recommendation	66
4.2.3	Objectives	67
4.3	Gittins Index Policy for Cost-Aware Bayesian Evaluation	68
4.3.1	Bayesian Bandit Model	68
4.3.2	Posterior Dynamics and the Gittins Policy	71
4.3.3	Efficient Gittins Index Computation	73
4.3.4	Design Choices for LLM Benchmarks	75
4.4	Experiments	77
4.5	Conclusion	86
5	Bayesian-optimal Decision-making under Cost-aware Multi-stage Feedback via the Gittins Index	87
5.1	Introduction	88
5.2	Bayesian-optimal Decision-making via Gittins Indices	90
5.2.1	A Review of Pandora’s Box	91
5.2.2	From Pandora’s Box to Markov Chain Selection	92
5.2.3	Linear Gaussian State-space Models	94
5.3	Gittins Indices for Gaussian State Space Models	95
5.3.1	Reduction to One-Dimensional Gaussian Random Walks	96
5.3.2	Local Value Functions of Gaussian Random Walks	99
5.3.3	Efficient Approximate Dynamic Programming via Fast Fourier Transforms	102
5.3.4	Extension to LGSSMs with hidden states	103
5.4	Synthetic Bayesian Regret Experiments	105
5.5	Conclusion	107

A	Supplement to Chapter 2	108
A.1	Illustrations	108
A.1.1	Visualization of synthetic benchmark functions	108
A.1.2	Comparison of behaviors between EI and PBGI	109
A.1.3	An example where expected improvement underperforms	109
A.2	Theory and calculations	110
A.2.1	Additional intuition on Pandora’s Box	110
A.2.2	Gradient of the PBGI acquisition function	111
A.2.3	Small-cost limit of PBGI acquisition function	113
A.2.4	Closed-form expression for the PBGI unknown-cost variant	114
A.2.5	Relationship between expected budget-constrained and cost-per-sample problems	115
A.2.6	Auxillary results from optimization theory	127
A.3	Experimental setup	132
A.4	Additional experimental results	138
A.4.1	Impact of log-scaling and numerical stability on EI and PBGI	138
A.4.2	Runtime comparison	140
A.4.3	Hyperparameter choice for PBGI-D	140
A.4.4	Effect of unknown costs	141
A.4.5	Alternative visualization of experimental results: means and standard errors	141
A.4.6	Kernel and problem hyperparameters	142
B	Supplement to Chapter 3	154
B.1	Theoretical Analysis and Calculations	154
B.2	Experimental Setup	160
B.3	Additional Experiment Results	167
B.3.1	Runtime Comparison	168
B.3.2	Order of Stopping and Posterior Updates	169
B.3.3	Additional Experiment Results: Empirical	169
B.3.4	Additional Experiment Results: Bayesian Regret	184
C	Supplement to Chapter 4	198
C.1	Theoretical Details	198
C.1.1	Gaussian Random-Walk Details	198
C.1.2	Proof of Bayesian Optimality	200
C.1.3	One-Dimensional Gittins Index Computation	202
C.2	Dataset Details	204
C.2.1	Dataset Description	204
C.2.2	Dataset Statistics	207
C.3	Experiment Setup and Implementation Details	215
C.4	Additional Experiment Results	228
C.4.1	Ablation Studies	228
C.4.2	Per-Subject MMLU Results	230

C.4.3	Computational Overhead Comparison	237
D	Supplement to Chapter 5	240
D.1	Theory	240
D.2	Fast Fourier Transform Details	242
D.3	Experimental Details	243
	Bibliography	247

LIST OF TABLES

3.1	Percentage of LCBench tasks ranked in the Top- k ($k = 1, 2, 3$) by cost-adjusted simple regret for among acquisition–stopping rule pairs (excluding the hindsight stopping rule), for our PBGI/LogEIPC stopping rule paired with LogEIPC or PBGI, evaluated on large datasets (with $> 10,000$ instances) and on all datasets, for cost-scaling factor $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}\}$	57
B.1	Number of trials (out of 50) where each stopping rule failed to trigger within 200 iterations, for each dataset in the LCBench (first three) and NATS (last three) benchmarks and each acquisition function. Results are identical across acquisition functions.	172
C.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.	212
C.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.	219
C.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.	220
C.4	AlpacaEval model-level pricing table for the 153-model response matrix. Input and output prices are in USD per 1M tokens. The combined cost is input cost + $8 \times$ output cost, matching the AlpacaEval input-to-output token ratio used for cost-aware runs. . . .	221
C.5	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.	228

C.6	Batch-size settings used in the simple-regret experiments. For UCB-E and both Gittins variants, batch size B denotes the number of examples queried for the selected arm at each allocation step. For AlpacaEval, these examples correspond to instruction-level pairwise preference scores rather than binary correctness labels. UCB-E-LRF uses $B = 32$ in all reported settings. Bayesian optimization 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.	229
-----	---	-----

LIST OF FIGURES

2.1	An illustration of this work’s key idea. We view cost-aware Bayesian optimization as an extension of the Pandora’s Box problem, and derive the cost-aware acquisition function α_t^{PBGI} by incorporating the posterior into the Bayesian-optimal Pandora’s Box acquisition function α^*	19
2.2	A Bayesian optimization problem with varying costs on which LogEIPC—a numerically stable implementation of EIPC, see Section 2.4 and Appendix A.4.1—has poor performance, inspired by Astudillo et al. (2021, Section A). The domain is $X = [-500, 500]$, which we visualize on the subinterval $[-5, 5]$. Left: illustration of the non-uniform prior variance, which is given by a Matérn-5/2 kernel scaled by a narrow bump function. Center: the cost function, which is a narrow bump-shaped function. Right: median regret curves and quartiles for LogEIPC and PBGI. Legend refers only to regret curves.	23
2.3	Left: contour plots showing how EI (left) and PBGI (center-left, center-right) depend on the posterior mean and standard deviation at a given point (lighter colors indicate higher values). We see that PBGI values high standard deviation more than EI. Right: PBGI performance across values of λ , under the setup of the Bayesian regret experiment of Section 2.4 with $d = 8$. We plot the median of a set of samples using $n = 256$ random seeds, along with quartiles to show variability. We see that large λ -values decrease regret sooner, but eventually lose out to smaller λ -values.	26
2.4	Regret curves for objective functions sampled from the prior, shown using medians, as well as quartiles to indicate experiment variability. We see in the cost-aware setting that both PBGI variants usually exhibit comparable performance to LogEIPC and LogEICC, with PBGI-D decisively outperforming other baselines. In the uniform-cost setting, the story is similar for $d = 8$ and $d = 16$: PBGI and PBGI-D perform comparably to the best baselines, which are LogEI and UCB, as well as MSEI for $d = 8$. However, for $d = 32$, all methods perform comparably to random search, with PBGI, PBGI-D, LogEI, and UCB having near-identical median performance. See Figure A.8 for an alternative visualization using mean and standard error.	28

2.5	Synthetic benchmark regret curves, shown using medians, as well as quartiles to assess variability. All objective functions are defined with dimension $d = 16$. We see in the cost-aware setting that PBGI, PBGI-D, LogEIPC, and LogEICC all perform similarly on the heavily-multimodal Ackley function, matching or outperforming the non-myopic BMSEI baseline. On the Levy and Rosenbrock functions, PBGI-D matches—and for some cost budgets outperforms—all baselines, including the non-myopic BMSEI. Under uniform costs, PBGI performs well on Ackley and Levy, but is outperformed by PBGI-D and most baselines on Rosenbrock. See Figure A.8 for an alternative visualization using mean and standard error.	30
2.6	Empirical benchmark regret curves, shown using medians, as well as quartiles to show variability. We see in both the varying-cost and uniform-cost settings that PBGI exhibits strong performance on the Pest Control and Lunar Lander problems. On the Robot Pushing problem, LogEI variants perform strongest, with PBGI-D performing second-strongest and matching or outperforming all other baselines. See Figure A.8 for an alternative visualization using mean and standard error.	32
3.1	Illustration of the PBGI/LogEIPC stopping rule under a uniform-cost setting. When the cost-per-sample is large ($c(x) \equiv 0.1$), the maximum LogEIPC acquisition value falls below the threshold 0.0 and the minimum PBGI acquisition value exceeds the threshold (current best observed value), indicating stopping; when the cost-per-sample is small ($c(x) \equiv 0.0001$), the maximum LogEIPC acquisition value remains above the threshold 0.0 and the minimum PBGI acquisition value is smaller than the threshold (current best observed value), indicating no stopping.	45
3.2	Cost-adjusted simple regret across acquisition-stopping rule pairs in 1D and 8D Bayesian regret setting. In 1D, objective functions are sampled from a GP with a Matérn-5/2 kernel and a linear cost function scaled by $\lambda = 0.1, 0.01, 0.001$. The Immediate baseline is omitted at $\lambda = 0.001$ due to its much higher regret (mean 0.6942, error bar [0.5314, 0.8570]). In 8D, objective functions are also drawn from a GP with a Matérn-5/2 kernel, using three cost functions scaled by $\lambda = 0.01$	54

3.3	Cost-adjusted simple regret across acquisition function–stopping rule pairs on LCBench and NATS-Bench. The objective is to minimize validation error on classification tasks, with scaled proxy runtime as evaluation cost, scaled by representative values of λ (10^{-3} , 10^{-4} , 10^{-5} for LCBench and 10^{-5} for NATS-Bench). For LCBench, results are aggregated across 35 datasets using min–max normalization. Our PBGI/LogEIPC stopping rule, when paired with either LogEIPC or PBGI, typically ranks among the top 3 pairs and closely approaches the hindsight optimal on LCBench and on cifar10-valid in NATS-Bench, but slightly worse on the other two NATS datasets, likely due to model misspecification.	58
4.1	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.	69
4.2	Main GSM8K, PIQA, and AlpacaEval results. The top row shows unit-cost evaluation and the bottom row shows cost-aware evaluation; simple regret is plotted against the percentage of the corresponding exhaustive evaluation budget consumed. UCB-E and Gittins use batch size $B = 8$, while LRF uses $B = 32$. Each GSM8K and PIQA panel averages over 100 independent runs (5 ground-truth matrices $\times$ 20 randomized trials); AlpacaEval panels average over the available response matrix and randomized trials. Shaded bands denote ± 1 standard error. Bayesian optimization curves are shown after the 5% configuration-level random-initialization phase. Dashed vertical lines show mean stopping locations when available: for Gittins, when the index-based stopping rule first triggers; for Bayesian optimization, the recorded Bayesian optimization stopping point. These large-example benchmarks show the main advantage over configuration-level Bayesian optimization: Gittins-S enters the low-regret regime early under both cost settings, while Bayesian optimization lags because all-example configuration evaluations leave little room for acquisition-driven improvement under the same budget fraction. Gittins-S also reduces simple regret faster than SySRs and enters the low-regret regime earlier across these panels. Gittins-G is also competitive, though its cost-aware GSM8K curve shows larger early variation before converging.	82

4.3	Aggregate MMLU results by subject size and difficulty. Columns show the easy–small, easy–large, hard–small, and hard–large buckets; the top row shows unit-cost evaluation and the bottom row shows cost-aware evaluation. Simple regret is plotted against the percentage of the corresponding exhaustive evaluation budget consumed. For UCB-E and Gittins, the small size bucket uses batch size $B = 2$ and the large size bucket uses $B = 8$; LRF uses batch size $B = 32$. Each panel averages over all subjects in the size–difficulty bucket and 20 independent runs per subject. Shaded bands denote ± 1 standard error. Bayesian optimization curves are shown after the 5% configuration-level random-initialization phase. Dashed vertical lines show mean stopping locations when available: for Gittins, when the index-based stopping rule first triggers; for Bayesian optimization, the recorded Bayesian optimization stopping point. These 1500-arm tasks show the main advantage over cost-unaware frequentist bandits: Gittins-G and Gittins-S usually enter the low-regret regime earlier than UCB-E and LRF, especially in the cost-aware and harder buckets. Across both cost settings and all four size–difficulty buckets, Gittins-S also reduces simple regret faster than SySRs and enters the low-regret regime earlier.	83
5.1	Backward solution of the dynamic program. At the terminal stage, $Q_{T,\text{loc}}^*(x) = x$; at each earlier stage, the next-stage Q -function is thresholded at zero, convolved with the stage-specific Gaussian increment density, and shifted by the evaluation cost.	96
5.2	Translational equivariance of $Q_{t,\text{loc}}^*$: shifting the state and threshold by β shifts the Q -function by β	101
5.3	Synthetic model-matched Bayesian-regret experiments. Each panel shows the tradeoff between evaluation cost and Bayesian regret for a fixed number of arms and horizon. Columns correspond to horizons $T \in \{10, 50, 100\}$; rows correspond to arm counts $N \in \{100, 500, 1000\}$. The Gittins policy is evaluated across cost-scaling parameters λ and compared with random search and successive halving baselines. Relative to random search, Gittins reaches the Pareto frontier; the successive-halving results are dominated and therefore do not reach this frontier.	106
A.1	An illustration of the two-dimensional Ackley, Levy, and Rosenbrock functions. ¹ From the visual, one can see that these functions differ in terms of multimodality and ridge-regions within the optimization landscape.	108

A.2	Comparison between three acquisition functions, namely Expected Improvement (EI), Pandora’s Box Gittins Index (PBGI) with a large λ -value of $\lambda = 10^0$, and Pandora’s Box Gittins Index with a small λ -value of $\lambda = 10^{-5}$. All three are computed using the same posterior distribution with four data points. The maximum is shown as a large dot.	144
A.3	A Bayesian optimization problem with varying costs on which LogEI, a numerically stable implementation of EI—which by construction ignores costs—has poor performance. See Section 2.4 and Appendix A.4.1 for additional details. Like Figure 2.2, this construction also mirrors Astudillo et al. (2021, Section A). The domain is $X = [-500, 500]$, which we visualize on the subinterval $[-5, 5]$. Left: illustration of the non-uniform prior variance, which is given by a Matérn-5/2 kernel scaled by a narrow bump function, plus a constant. Center: the cost function, which is a narrow bump-shaped function. Right: regret curves, shown as medians and quartiles.	144
A.4	Experimental results comparing the impact of log-scaling and numerical stability on the performance of EI and PBGI, tested on Bayesian regret with $d = 16$ and shown using quartiles. We see that $\text{Exp}(\text{LogEI})$ is close to Unst. EI, and similarly for S-PBGI vs PBGI. However, $\text{Log}(\text{Unst. EI})$ improves significantly over EI, and LogEI improves even more.	145
A.5	Runtime comparison of PBGI against baselines for computing the acquisition function on the Ackley benchmark across different dimensions ($d = 4, 8, 16$). We see that runtime of PBGI is slower than LogEI and TS, but faster than KG and MSEI.	145
A.6	Behavior of PBGI-D under different choices of the initial value λ_0 , and the coefficient of decay β , on the Bayesian regret experiment with $d = 8$ and other parameters set the same way as in Section 2.4. We show medians under $n = 256$ samples, along with quartiles to assess variability. We see that smaller λ_0 -values act similarly to the behavior seen in Figure 2.3 for PBGI, trading off risk-seeking vs. risk-averse behavior, but with a substantially smaller gap between the two variants. Larger β -values lead to more-abrupt decay curves for λ , with an effect qualitatively similar to λ_0 . In both cases, though our relatively-large sample size allows us to see small differences between methods, their overall impact on performance is close-to-nonexistent on the scale of variability.	146
A.7	Experimental results for the Robot Pushing empirical benchmark, with the known-cost variant (left) and unknown-cost variant (right). We see that LogEIPC, LogEICC, and PBGI-D performing best in both cases.	146

A.8	Regret curves for the Bayesian regret and synthetic benchmark experiments, as well as best observed values for the empirical experiments, showing using means (solid lines) and two times standard errors (shaded regions). See Figures 2.4 to 2.6 for details.	147
A.9	Comparison of Bayesian regret across Gaussian process priors with different kernels over different dimensions, in the uniform-cost setting. All length scales are $\kappa = 10^{-1}$. We see that overall behavior is similar, but the precise thresholds at which each example switches between the easy, medium-hard, and very hard regimes differ. . . .	148
A.10	Comparison of Bayesian regret across Gaussian process priors with different kernels over different dimensions, in the cost-aware setting. All length scales are $\kappa = 10^{-1}$. We see that overall behavior is similar, but the precise thresholds at which each example switches between the easy, medium-hard, and very hard regimes differ. . . .	149
A.11	Comparison of Bayesian regret across different length scales and dimensions, with a Matérn-5/2 kernel, in the uniform-cost setting. We see similar overall behavior, but each example switches between the easy, medium-hard, and very hard regimes at different precise thresholds.	150
A.12	Comparison of Bayesian regret across different length scales and dimensions, with a Matérn-5/2 kernel, in the cost-aware setting. We see similar overall behavior, but each example switches between the easy, medium-hard, and very hard regimes at different precise thresholds.	151
A.13	Comparison of regret for synthetic benchmark functions under different dimensions, in the uniform-cost setting. We see that most methods perform similarly for $d = 4$, with differences between the most-competitive methods emerging as dimension increases to $d = 8$ and $d = 16$	152
A.14	Comparison of regret for synthetic benchmark functions under different dimensions, in the cost-aware setting. We see that most methods perform similarly for $d = 4$, with differences between the most-competitive methods emerging as dimension increases to $d = 8$ and $d = 16$	153
B.1	Comparison of the raw and moving-averaged PBGI/LogEIPC stopping rule signals (i.e., the LogEIPC acquisition values) in Bayesian regret 8D experiments for multiple acquisition functions, with linear cost and stopping thresholds at $\log(0.1)$, $\log(0.01)$ and $\log(0.001)$. <i>Left:</i> The unaveraged signals exhibits large, high-frequency fluctuations due to the difficulty of acquisition optimization in high dimensions. <i>Right:</i> Applying moving average (window=20) smooths these wiggles, yielding more stable stopping signals.	163

B.2	Surface plots of cost functions over $[0, 1]^2$: (Left) Uniform cost of 1 across domain. (Middle) Normalized linear cost increasing with the mean of x_1 and x_2 . (Right) Periodic cost with $\alpha = 2$, $\beta = 2$, normalized by Bessel-based factor.	164
B.3	Empirical relationship between the number of model parameters and runtime for three LCBench datasets. Each subplot shows a scatter plot of actual runtime (y -axis) against number of model parameters (x -axis), along with a fitted linear regression line. The observed linear trend supports using 0.001 times the number of model parameters as a proxy for runtime. For <i>Fashion-MNIST</i> and <i>adult</i> , the fitted slopes are close to 0.001. The slope for higgs is slightly higher, possibly due to a few outliers.	165
B.4	Empirical relationship between the number of FLOPs and runtime for the three NATS-Bench datasets. Each subplot shows a heatmap of actual runtime (y -axis) against number of FLOPs (x -axis), along with a fitted linear regression line. The observed linear trend supports using $\alpha \times \text{\#FLOPs} + \beta$ as a proxy for runtime.	167
B.5	Evolution of per-iteration computation time (in log scale) for different stopping stopping rules when paired with six acquisition policies on the 8-dimensional Bayesian regret benchmark. Each subplot shows the average runtime (in seconds) over 50 iterations under one acquisition function—LCB, Thompson Sampling, LogEIPC, PBGI($\lambda = 10^{-1}$), PBGI($\lambda = 10^{-2}$), and PBGI($\lambda = 10^{-3}$). Curves correspond to four stopping criteria: our PBGI/LogEIPC stopping rule, SRGap-med, UCB–LCB, and the probabilistic regret bound (PRB). Convergence and GSS can be applied using only the best observed value and thus require no additional computation time, thus they are omitted here. LogEIPC-med relies on the same underlying statistical computations as the LogEIPC rule, and therefore its runtime is not measured separately. The results should that PRB incurs significant computational overhead compared to other stopping rules.	168

B.6	Illustration of a single draw from a Matérn-5/2 Gaussian process on $[0, 1]$ with lengthscale 0.1, optimized using PBGI acquisition function under uniform cost and cost-scaling factor $\lambda = 0.01$. We compare two variants of PBGI stopping rules: the <i>before-posterior-update</i> (this-round) stopping rule and the <i>after-posterior-update</i> (next-round) stopping rule. Left: The latent objective function (solid gray) and evaluation sequences for <i>before-posterior-update</i> stopping (blue circles) and <i>after-posterior-update</i> stopping (orange crosses). The dotted blue line and the dashed orange line mark the best observed value under each respective rule. Right: Cost-adjusted regret for each stopping rule. In this example, <i>after-posterior-update</i> stopping achieves strictly lower cost-adjusted regret despite performing more evaluations.	170
B.7	Comparison of simple regret of five acquisition functions: LogEIPC, PBGI($\lambda = 10^{-4}$), LCB, TS, and PBGI-D on LCBench datasets, using scaled proxy runtime as evaluation cost. We can see that indeed the cost-aware LogEIPC and PBGI outperforms the cost-unaware UCB and TS. PBGI-D with out recommended λ_0 is also competitive.	171
B.8	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the <i>Fashion-MNIST</i> dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-3}, 10^{-4}, 10^{-5}$. We can see that the PBGI/LogEIPC stopping rule consistently achieves cost-adjusted regret close to the hindsight optimal adaptive stopping (shown by the red marker) as well as the hindsight optimal fixed-iteration stopping (given by the minimum point along the fixed-budget curve) when paired with the PBGI acquisition function, though not always the best.	173
B.9	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the <i>adult</i> dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-3}, 10^{-4}, 10^{-5}$. We can see that the PBGI/LogEIPC stopping rule consistently achieves the cost-adjusted regret close to hindsight optimal adaptive stopping (shown by the red marker) as well as the hindsight optimal fixed-iteration stopping (given by the minimum point along the fixed-budget curve) when paired with the PBGI or TS acquisition function, particularly with PBGI.	174

B.10	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the <i>higgs</i> dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-3}, 10^{-4}, 10^{-5}$. We can see that the PBGI/LogEIPC stopping rule consistently achieves the best cost-adjusted regret when paired with the LogEIPC, PBGI, or TS acquisition function, particularly with PBGI. These pairs not only approach the hindsight optimal adaptive stopping (shown by the red marker) but also perform comparably to the hindsight optimal fixed-iteration stopping (the minimum point along the fixed-budget curve).	175
B.11	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the <i>cifar10-valid</i> dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-4}, 10^{-5}, 10^{-6}$. We can see that the PBGI/LogEIPC stopping rule consistently achieves the best cost-adjusted regret when paired with the LogEIPC, PBGI, or TS acquisition function, particularly with PBGI. These pairs not only approach the hindsight optimal adaptive stopping but also perform comparably to the hindsight optimal fixed-iteration stopping.	176
B.12	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the <i>cifar100</i> dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-4}, 10^{-5}, 10^{-6}$. The PBGI/LogEIPC stopping rule remains competitive at $\lambda = 10^{-4}$ and 10^{-6} , though not always the best. At $\lambda = 10^{-5}$, unlike in most experiments, it stops noticeably late (though still outperforming several other rules), even when paired with its matching acquisition function. By Lemma 3, under model match, the PBGI/LogEIPC rule with the corresponding acquisition function should never incur negative expected cost-adjusted regret before stopping. The suboptimal behavior observed here points to significant model misspecification on the <i>cifar100</i> dataset.	177

B.13	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the <i>ImageNet16-120</i> dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-4}, 10^{-5}, 10^{-6}$. The PBGI/LogEIPC stopping rule remains competitive at $\lambda = 10^{-4}$ and 10^{-6} , though not always the best. At $\lambda = 10^{-5}$, unlike in most experiments, it stops noticeably late (though still outperforming several other rules), even when paired with its matching acquisition function. By Lemma 3, under model match, the PBGI/LogEIPC rule with the corresponding acquisition function should never incur negative expected cost-adjusted regret before stopping. The suboptimal behavior observed here points to significant model misspecification on the <i>ImageNet16-120</i> dataset.	178
B.14	Comparison of cost-adjusted simple regret on three LCBench datasets with $\lambda = 10^{-4}$, using scaled proxy runtime vs. scaled actual runtime as evaluation cost. While our PBGI/LogEIPC stopping rule performs slightly worse under actual runtime (e.g., dropping from best to second-best on the <i>higgs</i> dataset), likely due to cost model misspecification, it remains close to the hindsight optimal in all cases.	179
B.15	Cost-adjusted simple regret across acquisition function and stopping rule pairs under the unknown-cost setting on LCBench with $\lambda = 10^{-4}$. Our PBGI/LogEIPC stopping rule remains close to the hindsight optimal when paired with the LogEIPC-exp or PBGI acquisition function, sometimes slightly worse than heuristics such as GSS and Convergence. It is also slightly worse than Immediate on <i>Adult</i> , likely due to a cost-model misspecification in the unknown-cost setting.	180
B.16	Cost-adjusted simple regret of all 35 LCBench datasets when $\lambda = 10^{-3}$. Datasets where the matched PBGI pair underperforms, meaning that it does not rank among the top three acquisition function–stopping rule pairs by cost-adjusted simple regret, are highlighted in light grey.	181
B.17	Cost-adjusted simple regret of all 35 LCBench datasets when $\lambda = 10^{-4}$. Datasets where the matched PBGI pair underperforms, meaning that it does not rank among the top three acquisition function–stopping rule pairs by cost-adjusted simple regret, are highlighted in light grey.	182
B.18	Cost-adjusted simple regret of all 35 LCBench datasets when $\lambda = 10^{-5}$. Datasets where the matched PBGI pair underperforms, meaning that it does not rank among the top three acquisition function–stopping rule pairs by cost-adjusted simple regret, are highlighted in light grey.	183

B.19	Top-20 validation–test rank preservation for every LCBench dataset. Datasets are split into $\mathcal{D}_{\text{under}}$ (red) and $\mathcal{D}_{\text{top}}$ (green) based on whether the matched PBGI pair ranks among the top three PBGI-based acquisition function–stopping rule pairs formed by the seven stopping rules in $\mathcal{S}$ at $\lambda = 10^{-4}$. The dashed line marks Top-20 = 0.5. Datasets where the matched PBGI pair underperforms also have markedly lower validation-to-test rank preservation, indicating that their weaker cost-adjusted regret reflects a val/test mismatch in the benchmark rather than a failure mode of the stopping rule itself.	184
B.20	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 1D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under uniform cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).	185
B.21	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 1D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under linear cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).	186
B.22	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 1D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under periodic cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).	187
B.23	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 8D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under uniform cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).	188
B.24	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 8D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under linear cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).	189

B.25	Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 8D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under periodic cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).	190
B.26	Single-trajectory illustration of how the moving-average window size W affects the PBGI/LogEIPC stopping signal on the 8D Bayesian regret benchmark under periodic costs (seed 0, $\lambda = 10^{-2}$), with PBGI ($\lambda = 10^{-2}$) as the acquisition function. The light grey curve is the raw $\alpha_t^{\text{LogEIPC}}$, the blue curve its W -iteration trailing average, the red dashed line the stopping threshold $\log \lambda$, and the green star the resulting stopping iteration. Without smoothing ($W = 1$) the rule fires early (iter. 24, regret 4.04) on a noisy dip; smoothing defers the trigger and recovers low cost-adjusted regret (iter. 119, regret 2.78 at $W = 20$).	191
B.27	Single-trajectory illustration analogous to Figure B.26, with the LogEIPC acquisition function (same seed, same cost regime, $\lambda = 10^{-2}$). Without smoothing the rule fires at iter. 34 with regret 4.01; with $W = 20$ it fires at iter. 93 with regret 2.66.	192
B.28	Single-trajectory illustration analogous to Figure B.26, with Thompson sampling (TS) as the acquisition function. Unlike the matched-acquisition cases, smoothing here pushes the stopping iteration later (iter. 71 at $W = 1$ vs. iter. 215 at $W = 20$) without a corresponding drop in cost-adjusted regret. TS does not satisfy the expected-improvement-exceeds-cost property of Lemma 3, so the PBGI/LogEIPC stopping rule has no matched guarantee for it.	193
B.29	Single-trajectory illustration analogous to Figure B.26, with LCB as the acquisition function. The raw LogEIPC signal has a clean transient: smoothing changes the stopping iteration only slightly (iter. 2 at $W = 1$ vs. iter. 5 at $W \geq 5$) and the cost-adjusted regret stabilizes at 2.73 for all $W \geq 5$	194
B.30	Effect of the moving-average window size $W \in \{1, 5, 10, 20\}$ on the average stopping iteration (blue, left axis) and the average cost-adjusted simple regret (orange, right axis) for the PBGI/LogEIPC stopping rule on the 8D Bayesian regret benchmark under <i>periodic</i> costs ($\lambda = 10^{-2}$, 50 seeds, error bars are ± 2 standard errors). Each subplot fixes one acquisition function. Smoothing yields the largest cost-adjusted regret reductions for the matched PBGI and LogEIPC pairs, a smaller reduction for LCB, and no benefit for TS.	195
B.31	Aggregate ablation analogous to Figure B.30 under <i>uniform</i> costs. Smoothing slightly reduces cost-adjusted regret for the matched PBGI and LogEIPC pairs and is roughly neutral for LCB and TS.	196

B.32	Aggregate ablation analogous to Figure B.30 under <i>linear</i> costs. Smoothing reduces cost-adjusted regret for the matched PBGI and LogEIPC pairs and for LCB, with no benefit for TS.	197
C.1	Distribution of per-arm quality for GSM8K, PIQA, and AlpacaEval. GSM8K has 122 arms and 1,000 examples, PIQA has 103 arms and 1,000 examples, and AlpacaEval has 153 arms and 805 examples. For GSM8K and PIQA, each subplot corresponds to one dataset–seed pair across five random seeds; for AlpacaEval, the subplot uses the no-rounding-debias matrix used in our experiments. Bars show the histogram of arm quality, defined as mean accuracy or win rate over queried examples. Vertical lines mark the empirical mean (red), best arm (green), and prior mean (purple). The GSM8K and PIQA distributions are highly similar across seeds, suggesting that the observed variation is small when the only source of randomness is LLM response generation.	208
C.2	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).	209
C.3	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).	210
C.4	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).	211

C.5	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. Bayesian optimization is competitive on several tasks but is less uniformly strong, while LRF is generally less effective and less stable. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.	231
C.6	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. Bayesian optimization is strong on some subjects but is not consistently better than Gittins, and LRF remains comparatively weak and unstable. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.	232
C.7	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 Bayesian optimization performs well when the arm-level configuration structure is informative. LRF is generally less effective. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.	233

C.8	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 Bayesian optimization is competitive on selected subjects. LRF remains less stable and typically underperforms. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.	234
C.9	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. Bayesian optimization is strong on a few subjects but is not uniformly dominant, while LRF is substantially less reliable. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.	235
C.10	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. Bayesian optimization performs well on selected tasks, while LRF remains the least reliable baseline. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.	236

- C.11 Total wall-clock runtime under the unit-cost setting. Each panel reports the total runtime for a completed simulated allocation run on GSM8K, PIQA, AlpacaEval, MMLU-small, MMLU-medium, and MMLU-large. Bars show medians across completed runs, with black error bars indicating ± 1 standard error. Hollow diamonds mark median Gittins stopping times, and hollow circles mark median Bayesian optimization stopping times, estimated by scaling each run’s total runtime by the fraction of the evaluation budget used at stopping. UCB-E and Gittins use $B = 8$ on GSM8K/PIQA/AlpacaEval and $B = 2, 4, 8$ on MMLU-small/medium/large, respectively; UCB-E-LRF uses $B = 32$ throughout. Bayesian optimization baselines correspond to PBGI run with unit costs and LogEI. The broken vertical axis is used because UCB-E-LRF is much slower on larger MMLU tasks. Overall, Gittins remains close to UCB-E except on the larger MMLU settings, while UCB-E-LRF incurs substantially larger overhead. SySRs remains computationally light across the benchmark groups. The Bayesian optimization baselines are also computationally light in these total-runtime measurements, partly because their configuration-level protocol requires fewer sequential allocation decisions than the batch-level bandit methods. 238
- C.12 Total wall-clock runtime under the cost-aware setting. Each panel reports the total runtime for a completed simulated allocation run on GSM8K, PIQA, AlpacaEval, MMLU-small, MMLU-medium, and MMLU-large. Bars show medians across completed runs, with black error bars indicating ± 1 standard error. Hollow diamonds mark median Gittins stopping times, and hollow circles mark median Bayesian optimization stopping times, estimated by scaling each run’s total runtime by the fraction of the cost budget used at stopping. UCB-E and Gittins use $B = 8$ on GSM8K/PIQA/AlpacaEval and $B = 2, 4, 8$ on MMLU-small/medium/large, respectively; UCB-E-LRF uses $B = 32$ throughout. Bayesian optimization baselines correspond to PBGI and LogEIPC. The broken vertical axis is used because UCB-E-LRF is much slower on larger MMLU tasks. Overall, Gittins remains close to UCB-E except on the larger MMLU settings, while UCB-E-LRF incurs substantially larger overhead. SySRs remains computationally light across the benchmark groups. The Bayesian optimization baselines are also computationally light in these total-runtime measurements, partly because their configuration-level protocol requires fewer sequential allocation decisions than the batch-level bandit methods. 239

D.1	Policy rollouts for the Gittins policy applied to a synthetic Markov chain selection task. The cost scaling parameter λ reflects the explore/exploit tradeoff.	244
-----	--	-----

CHAPTER 1

INTRODUCTION

Selecting a good design often requires learning its value through costly experiments. A machine learning practitioner may train and validate candidate models, a scientist may run simulations or laboratory experiments, and an evaluator may test language model configurations on benchmark examples. In each case, the decision maker begins uncertain about which option is best, gathers information sequentially, and ultimately recommends or deploys one option. The central difficulty is that information is valuable but not free: each additional evaluation can improve the final decision, but consumes time, computation, money, or experimental resources.

This dissertation studies how to make such decisions in a cost-aware Bayesian manner. A Bayesian model represents uncertainty about the value of each option and updates that uncertainty as observations arrive. A policy must then answer three coupled questions:

- (i) which option should be evaluated next,
- (ii) how much information should be gathered about it, and
- (iii) when should evaluation stop and a final recommendation be made?

Although these questions arise in many forms, the chapters of this dissertation share a common design principle: compare options using *Gittins indices*. A Gittins index assigns each option a scalar priority that balances its potential value, the uncertainty that further evaluation may resolve, and the cost of obtaining that information. Under suitable independence and Bayesian modeling assumptions, selecting the option with the largest index is not merely a heuristic decomposition of a difficult sequential problem; it is a Bayesian-optimal policy.

The dissertation develops this principle through two complementary perspectives. The first connects cost-aware Bayesian optimization to Pandora’s Box, whose Bayesian-optimal policy ranks boxes using Weitzman’s reservation values (Weitzman, 1979). The second connects problems with multi-stage feedback to Markov chain selection, whose Bayesian-optimal policy ranks chains using Gittins indices (Gittins et al., 2011a). These two index rules were discovered independently, but Weitzman’s reservation values are mathematically equivalent to the Gittins indices for the corresponding one-shot Markov chains. For consistency across the dissertation, we therefore refer to Weitzman’s reservation values as *Pandora’s Box Gittins indices*. Together, these perspectives show how the same index design principle can guide decisions across different feedback models and application domains.

1.1 Cost-Aware Bayesian Decision-Making

Let θ_k denote the unknown terminal value of option $k \in \{1, \dots, K\}$. Depending on the application, an option may be a point in a black-box search space, a model configuration, or a candidate whose performance is revealed over time. A policy sequentially chooses evaluations based on the history observed so far. Evaluation of option k incurs a cost and produces information about θ_k . At a stopping time T , the policy recommends an option $\hat{k}_T$.

One natural goal is to minimize the simple regret of the final recommendation,

$$\max_k \theta_k - \theta_{\hat{k}_T}, \tag{1.1}$$

subject to a limit on cumulative evaluation cost. Another is to combine the two quantities directly, minimizing expected simple regret plus cumulative cost. The first formulation is appropriate when an evaluation budget is imposed externally.

The second makes stopping endogenous: continued evaluation is worthwhile only when its expected benefit exceeds its cost.

Throughout this dissertation, the evaluations themselves are treated as information-gathering actions, and performance is assessed by the quality of the final choice together with the cost of reaching it. This differs from standard cumulative-reward bandit objectives, which also value rewards collected while learning. Under the terminal-decision objectives considered here, a useful policy should direct evaluations toward options for which additional information is most useful for making the final choice, account for heterogeneous evaluation costs, and determine when further evaluation is no longer worthwhile.

1.1.1 From One-Shot to Multi-Stage Feedback

The first part of this dissertation considers settings in which each evaluation produces one-shot feedback about the performance of a selected design. Problems with this structure are commonly approached using Bayesian optimization: a probabilistic model captures uncertainty about an expensive black-box objective, and an acquisition function selects the next design to evaluate. Chapters 2 and 3 study the cost-aware version of this problem and connect it to Pandora’s Box, a well-studied problem from economics with a discrete set of independent alternatives. In Pandora’s Box, opening a box incurs a cost and reveals the reward inside, providing an analytically solvable model from which to design practical acquisition and stopping rules for Bayesian optimization.

The second part further considers settings in which richer information is available through multi-stage feedback. An option can be evaluated over multiple stages, with each costly stage producing a partial observation that can inform subsequent

allocation decisions. Evaluation may be paused and resumed, and the option’s posterior state evolves as evidence accumulates. LLM benchmark evaluation has this form when each model configuration is tested on only a subset of examples. Training-curve and simulation settings have an even richer version of this structure: observations arrive over stages and may drift toward an option’s latent terminal value. By using this intermediate information, a policy can adapt its evaluation decisions before paying the cost required to complete every option. We connect such problems to the selection of independent Markov chains.

The one-shot and multi-stage settings thus connect to two classical sequential decision problems: Pandora’s Box and Markov chain selection, respectively. Both admit Bayesian-optimal policies based on Gittins indices, although the indices take different forms to reflect their different feedback structures. This shared optimality motivates the Gittins index design principle developed throughout the dissertation.

1.2 The Gittins Index Design Principle

The Gittins index converts a coupled sequential decision problem into a collection of single-option stopping problems (Gittins et al., 2011a). Informally, consider focusing on one option while comparing it with a fixed outside alternative of value g . The option’s index is the critical value of g at which the decision maker is indifferent between continuing to evaluate that option and taking the alternative. A high index can result from a high current posterior value, substantial upside that remains unresolved, or low evaluation cost. Once an index is computed for every option, the policy proceeds with the option having the largest index and stops when no unopened or unfinished option is preferable to the best available terminal choice.

This decomposition is attractive for both conceptual and computational reasons. Conceptually, an index makes the exploration–exploitation–cost trade-off explicit in a single priority score. Computationally, the single-option problems can often be solved offline or reused across many options, avoiding direct solution of a joint dynamic program whose dimension grows with the number of candidates. The central challenges addressed in this dissertation are how to adapt Gittins index policies to problems in which the distributions of option values are unknown and must be learned from observations, and how to efficiently compute approximations of the resulting indices that are sufficiently accurate for practical use.

1.2.1 Pandora’s Box Indices for Bayesian Optimization

In Pandora’s Box, each unopened box has an independent reward distribution and a known opening cost. The optimal reservation value for a box is determined by equating its expected improvement over a fixed alternative with its opening cost (Weitzman, 1979). Opening boxes in decreasing order of these reservation values, and stopping when the best revealed reward exceeds every remaining reservation value, is Bayesian-optimal.

In the original Pandora’s Box terminology, these indices are called *Weitzman’s reservation values*. Following the unified terminology introduced above, we call them Pandora’s Box Gittins indices.

Chapters 2 and 3 use this result as a design principle for cost-aware Bayesian optimization. Chapter 2 interprets the Pandora’s Box reservation value as an acquisition function, the Pandora’s Box Gittins Index (PBGI). This yields an alternative to expected improvement per unit cost (EIPC), which divides expected improvement by evaluation cost. PBGI treats cost more directly and in a more

principled manner: cost enters the value-of-information equation that defines the index, rather than being applied as an external normalization. The method is exact for the idealized independent discrete problem and serves as a tractable acquisition rule when transferred to correlated Gaussian process models.

Chapter 3 makes the stopping implication explicit. If the best value already observed is greater than the index of every unevaluated alternative, then further evaluation is not justified in the Pandora’s Box model. This observation motivates a cost-aware stopping rule for Bayesian optimization and connects the decision of where to sample with the decision of whether to sample at all. The resulting acquisition–stopping pairs are evaluated using cost-adjusted simple regret, which combines the simple regret of the final solution with cumulative evaluation cost.

1.2.2 Markov Chain Indices for Multi-Stage Feedback

The Pandora’s Box model reveals an option’s value all at once. Chapters 4 and 5 move to settings in which information is revealed gradually. Each option is represented by a Markov state summarizing the observations collected so far, and evaluating that option advances its state by one costly stage. The associated Gittins index again arises from a single-option optimal stopping problem, but its value now depends on the option’s evolving posterior state.

Chapter 4 studies an application-driven instance: efficient LLM evaluation. Each arm represents an LLM configuration, and each pull evaluates that configuration on benchmark examples. The observations are noisy measurements of latent mean performance. Under a Gaussian approximation, posterior means evolve according to Gaussian random walks with decreasing uncertainty. This structure permits index values to be precomputed and retrieved efficiently during evaluation. The resulting

policy allocates examples adaptively, incorporates configuration-dependent costs, and produces an anytime final recommendation.

Chapter 5 generalizes from repeated noisy observations of a fixed latent mean to broader multi-stage feedback, in which observations can drift toward and gradually reveal an option’s latent mean. It considers Gaussian state-space models, which can represent partially observed trajectories in problems such as model training and other multi-stage experiments. Although the relevant Gittins policy is characterized by Bayesian optimality theory, its indices are generally unavailable in closed form. The chapter reduces the index computation to structured one-dimensional dynamic programs and accelerates their solution using fast Fourier transforms. This turns a theoretical optimality characterization into a practical policy for a substantially richer class of continuous-state models.

1.2.3 From Analytical Models to Computable Policies

Across both parts, the dissertation progresses from analytically tractable models with Bayesian-optimal index policies to computable policies for practical decision problems. Pandora’s Box and Markov chain selection provide the analytic foundations, but applying their index principles requires adapting them to settings with learned and correlated values or richer feedback, and determining how the resulting indices can be computed efficiently.

The common contribution is therefore not a single index formula. It is a way of designing sequential information-gathering policies: formulate each candidate’s evolution as a stopping problem, use evaluation cost to define the value of continued information, compute the resulting option-level index, and rank options by that index. The chapters study where this recipe is exact, how it can motivate effective

approximations outside the exact model, and how its computational bottlenecks can be overcome.

1.3 Organization of the Dissertation

The remainder of the dissertation consists of two parts, each containing two chapters. In Part I, Chapter 3 complements Chapter 2 by adding stopping to acquisition. Part II extends Part I from one-shot to multi-stage feedback, and within Part II, Chapter 5 generalizes the feedback model considered in Chapter 4.

Chapter 2: Cost-aware Bayesian optimization via the Pandora’s Box Gittins index. This chapter connects cost-aware Bayesian optimization to Pandora’s Box, derives PBGI as an acquisition function, and evaluates it on synthetic and empirical black-box optimization problems.

Chapter 3: Cost-aware stopping for Bayesian optimization. This chapter complements Chapter 2 by developing and evaluating a cost-aware stopping rule for Bayesian optimization, with guarantees for the expected cost-adjusted simple regret of the resulting acquisition–stopping policies.

Chapter 4: Efficient cost-aware LLM evaluation via Bayesian bandit Gittins indices. This chapter formulates LLM benchmark evaluation as cost-aware Bayesian best-arm identification and develops an efficiently computable Gittins index policy for allocating partial evaluations across configurations.

Chapter 5: Bayesian-optimal decision-making under cost-aware multi-stage feedback via the Gittins index. This chapter generalizes Chapter 4 to Gaussian state-space models with richer multi-stage feedback and uses Gaussian structure and fast Fourier transforms to compute accurate index approximations efficiently.

Part I

Cost-aware Bayesian Optimization with Adaptive Stopping via the Pandora's Box Gittins Index

CHAPTER 2

COST-AWARE BAYESIAN OPTIMIZATION VIA THE PANDORA’S BOX GITTINS INDEX

This chapter is based on joint work with Raul Astudillo, Peter I. Frazier, Ziv Scully, and Alexander Terenin (Xie et al., 2024).

Abstract. Bayesian optimization is a technique for efficiently optimizing unknown functions in a black-box manner. To handle practical settings where gathering data requires use of finite resources, it is desirable to explicitly incorporate function evaluation costs into Bayesian optimization policies. To understand how to do so, we develop a previously-unexplored connection between cost-aware Bayesian optimization and the Pandora’s Box problem, a decision problem from economics. The Pandora’s Box problem admits a Bayesian-optimal solution based on an expression called the Gittins index, which can be reinterpreted as an acquisition function. We study the use of this acquisition function for cost-aware Bayesian optimization, and demonstrate empirically that it performs well, particularly in medium-high dimensions. We further show that this performance carries over to classical Bayesian optimization without explicit evaluation costs. Our work constitutes a first step towards integrating techniques from Gittins index theory into Bayesian optimization. Code available at: [HTTPS://GITHUB.COM/QIANJAN
EXIE/PANDORABAYESOPT](https://github.com/QianJanXie/PandoraBayesOpt).

2.1 Introduction

Bayesian optimization is a framework for optimizing functions whose evaluation is time-consuming or expensive. It is widely used for hyperparameter tuning of

machine learning algorithms (Snoek et al., 2012), robot control (Martinez-Cantin, 2017), material design (Zhang et al., 2020), and other areas. Bayesian optimization works by forming a probabilistic model for the objective function, and then chooses where to sample via an acquisition function that balances the explore-exploit trade-offs arising from uncertainty in this model.

We study *cost-aware* Bayesian optimization, where one must pay a cost to acquire another sample and this cost may vary with where the function is evaluated. Costs are an important factor in practical scenarios. For instance, in hyperparameter tuning using GPUs rented from a cloud provider, training a neural network for twice as many epochs may carry twice the financial cost.

Despite its practical relevance, cost-aware Bayesian optimization is studied less than standard Bayesian optimization, where budgets are framed in terms of the number of function evaluations and costs are not explicitly considered. Many existing theoretically-principled cost-aware approaches (Yue and Kontar, 2020; Jiang et al., 2020; Lee et al., 2021; Astudillo et al., 2021; Belakaria et al., 2023) rely on multi-step lookahead computations that are computationally expensive and can be numerically brittle, limiting their applicability. Other approaches lack a theoretical foundation and risk having poor performance on certain problems. For example, one of the most popular cost-aware acquisition functions used in practice, expected improvement per unit cost (Snoek et al., 2012), has recently been theoretically shown by Astudillo et al. (2021) to perform arbitrarily worse than the optimal policy. Thus, in the cost-aware setting, there is a need for theoretically-principled and computationally-straightforward acquisition functions with good empirical performance.

In this work, we develop such an approach. To do so, we introduce a novel link

between cost-aware Bayesian optimization and a discrete-space decision problem from economics called the *Pandora’s Box* problem (Weitzman, 1979; Doval, 2018; Singla, 2018). The Pandora’s Box problem admits an explicit Bayesian-optimal solution. We show how this solution can be used to develop a novel acquisition function class for two cost-aware Bayesian optimization settings: (i) *expected budget-constrained* Bayesian optimization, where there is a constraint on the expected cost of the samples taken, and (ii) *cost-per-sample* Bayesian optimization where the total costs incurred are subtracted from the final objective function value. The resulting acquisition functions are closely connected to expected improvement variants, but incorporate costs in a different, non-multiplicative way.

We evaluate the proposed acquisition function, termed the *Pandora’s Box Gittins Index (PBGI)*, on a comprehensive set of experiments to understand its strengths and weaknesses. On both sufficiently-easy low-dimensional problems and too-difficult high-dimensional ones, performance is comparable to baselines. On most medium-hard problems of moderate dimension, the proposed acquisition function either matches or outperforms the strongest baselines. Surprisingly, we find this performance carries over to the classical setting with uniform costs. We also discuss limitations, including behavior on problems where baselines are stronger.

The Pandora’s Box Gittins Index is a version of the *Gittins index* (Gittins et al., 2011b), a general framework for deriving optimal policies for a variety of bandit-like decision problems (Weber, 1992; Dumitriu et al., 2003a; Gupta et al., 2019) which is widely-used in queueing theory and related areas (Glazebrook and Niño-Mora, 2001; Aalto et al., 2009; Scully et al., 2020). Other versions of the Gittins index have recently been proposed for Bayesian optimization with an infinite-horizon discounted cumulative regret objective (Persky, 2021). Our Pandora’s-Box-based

formulation is instead designed for the simple regret objective more common in the literature (Garnett, 2023a). Our work thus opens a novel angle of attack for designing acquisition functions specialized to specific practical settings of interest.

Contributions. In this work, we (i) connect the Pandora’s Box problem with a variant of cost-aware Bayesian optimization over a discrete search space. Using this connection, we (ii) explore the use of Gittins indices, which are Bayesian-optimal for the Pandora’s Box problem, as an acquisition function for general cost-aware Bayesian optimization where data is incorporated via the posterior distribution. We (iii) demonstrate the resulting acquisition function has strong empirical performance on a variety of problems of moderate-to-high dimension, including the varying-cost problems it was designed for, as well as classical cost-unaware problems.

2.2 Cost-aware Bayesian optimization

In black-box optimization, we are interested in finding the global optimum of an unknown (potentially stochastic) function $f : X \rightarrow \mathbb{R}$ defined on some compact domain, using pointwise function evaluations of f at locations $x_1, \dots, x_T \in X$ that we select sequentially. We are interested in policies that achieve low *simple regret*—see Garnett (2023a, Sec. 10.1)—namely

$$\mathbb{E} \sup_{x \in X} f(x) - \mathbb{E} \max_{1 \leq t \leq T} f(x_t) \tag{2.1}$$

where the expectation is taken over all sources of randomness in the function f and the policy, including the stopping time T and sequential selections $x_1, \dots, x_T$. In our setup, obtaining a new function evaluation at a point x carries a non-zero *cost* $c(x) \in \mathbb{R}_+$, assumed automatically-differentiable unless discussed otherwise. We consider settings that integrate costs into the problem in different ways:

- (a) In the *expected budget-constrained* setting, there is a budget $B \in \mathbb{R}_+$, and the algorithm is not allowed to exceed this budget in expectation.
- (b) In the *cost-per-sample* setting, at each time the algorithm must choose whether to pay a cost and obtain a new function evaluation, or to stop and return some previously-observed point. In this setting, we add the total sum of costs at termination time to the regret.

Note that the cost function $c : X \rightarrow \mathbb{R}_+$ can be constant, which we term *uniform costs*. In this case, (a) reduces to standard black-box optimization with a finite time horizon, and (b) reduces to a variant of stopping-aware Bayesian optimization. These are not the only possible settings: one can also consider other variants including the almost-sure budget-constrained setting where the algorithm is not allowed to exceed the budget in a strict manner. Since we are interested primarily in the role of costs rather than stopping times in this work, we mostly work with budget constraints throughout this paper, especially almost-sure budget constraints for our empirical results. We will use the cost-per-sample setting as a conceptual framework with which to study budget-constrained settings.

2.2.1 Probabilistic models and acquisition functions

Bayesian optimization algorithms for solving various black-box optimization problems work by (i) building a *probabilistic model* of f —that is, a probability distribution which quantifies what is known about f given the data points $(x_t, y_t)_{t=1}^T$ seen so far—where $y_t = f(x_t)$ are previous function evaluations—then (ii) using the model and its uncertainty to decide where to evaluate the unknown function next. For an introduction, see Frazier (2018) and Garnett (2023a). Following standard practice, we work with Gaussian process models (Rasmussen and Williams, 2006a).

Let $f \mid x_{1:t}, y_{1:t}$ be the respective posterior distribution.

To decide where to evaluate f next, one uses the model to define a (potentially random) *acquisition function* $\alpha_t : X \rightarrow \mathbb{R}$, which quantifies how promising a particular location is given what is known so far. We then evaluate f at

$$x_{t+1} = \arg \max_{x \in X} \alpha_t(x), \quad (2.2)$$

obtaining an additional data point that is used to reduce uncertainty and further improve the model.

2.2.2 Expected improvement per unit cost

The most popular cost-aware acquisition function is *expected improvement per unit cost* (EIPC) (Snoek et al., 2012), defined via

$$\alpha_t^{\text{EIPC}}(x) = \frac{\text{EI}_{f \mid x_{1:t}, y_{1:t}}(x; \max_{1 \leq \tau \leq t} y_\tau)}{c(x)} \quad \text{EI}_\psi(x; y) = \mathbb{E} \max(0, \psi(x) - y) \quad (2.3)$$

where we have written $\alpha_t^{\text{EIPC}}(\cdot)$ in terms of the general *expected improvement function* EI_ψ , defined with respect to some random function $\psi : X \rightarrow \mathbb{R}$, and a comparator point y . With this notation, EIPC can be interpreted as the ratio of the expected improvement, with respect to the current posterior and using the best point seen so far as the comparator, to the cost.

In the uniform-cost case, where the costs $c(x) = C \in \mathbb{R}_+$ are constant for all x , this acquisition function reduces to the classical *expected improvement* (EI) acquisition function $\alpha_t^{\text{EI}}(x) = \text{EI}_{f \mid x_{1:t}, y_{1:t}}(x; \max_{1 \leq \tau \leq t} y_\tau)$. In turn, expected improvement can be derived by considering the setup where the unknown function f is randomly sampled from the model's prior. If we imagine that the optimization process continues for only one more time step before stopping, maximizing expected improvement is the optimal strategy in this one-step-lookahead scenario.

Since EIPC reduces to EI in the uniform-cost case where $c(x) = C$, it follows that it chooses the same points whether $C = 0.0001$ or $C = 1\,000\,000$. This is somewhat peculiar: one might expect that a cost-aware acquisition function should be more risk-averse if costs are high, and vice versa if they are low. Thus, EIPC is perhaps best suited to settings where heterogeneity of costs is the main factor at play: without heterogeneity, one can simply apply standard acquisition function variants (Chowdhury and Gopalan, 2017). However, even with heterogeneous costs, Astudillo et al. (2021) show there exist reasonable problems where EIPC performs arbitrarily worse than the optimal policy in an approximation-ratio sense, due to over-sampling low-value low-cost points. Analogous behavior can occur in multi-fidelity settings, which Wu et al. (2020) argue can be especially undesirable in the presence of model-misspecification.

In spite of this rather negative outlook, EIPC has been shown to work well on many practical problems, is computationally efficient and reliable, and is the standard cost-aware choice in BoTorch (Balandat et al., 2020). We therefore ask: *can one develop a technically-principled and computationally-straightforward alternative with at-least-comparable empirical performance?*

2.3 The Pandora’s Box Gittins index for Bayesian optimization

To develop a cost-aware acquisition function, we study a simplified decision problem that captures key difficulties of the main problem but is tractable enough to yield analytic insights. An analogous strategy is used classically to derive expected improvement, by exactly solving a simplified one-step decision problem. We study a different simplified decision problem, which can also be solved exactly, but where

the simplification is spatial rather than temporal in nature. Specifically, we connect Bayesian optimization with the *Pandora’s Box* problem from economics. To do so, we describe Pandora’s Box in Section 2.3.1 and its solution in Section 2.3.2, showing along the way how these ideas can be reinterpreted from the view of Bayesian optimization. We illustrate this in Figure 2.1. Then, in Section 2.3.3, we use Pandora’s Box to derive a novel class of cost-aware acquisition functions.

2.3.1 The Pandora’s Box problem

The *Pandora’s Box* problem (Weitzman, 1979; Gittins et al., 2011b) is a sequential decision-making problem. It begins with a finite set of boxes, which we collect into a set and label $X = \{1, \dots, N\}$. Each box has a *hidden reward*, denoted by $f(x)$, and an *inspection cost*, denoted by $c(x)$. The rewards are given by mutually independent random variables whose distributions are known and vary between different boxes.

The decision-making process starts with a set of closed boxes, and proceeds in discrete time steps. At time t , one can choose to do one of two things:

1. *Open a box $x_t \in X$.* This incurs cost $c(x_t)$, but reveals the exact value $f(x_t)$ of the reward inside the box, which is drawn using the box’s respective reward distribution.
2. *Stop opening new boxes, and take the reward from the best opened box.* This ends the decision-making process, and yields a terminal reward equal to the maximum value among the boxes opened so far, with the convention that at least one box must be opened.

The policy’s goal is to maximize the expected net utility, which is the reward of

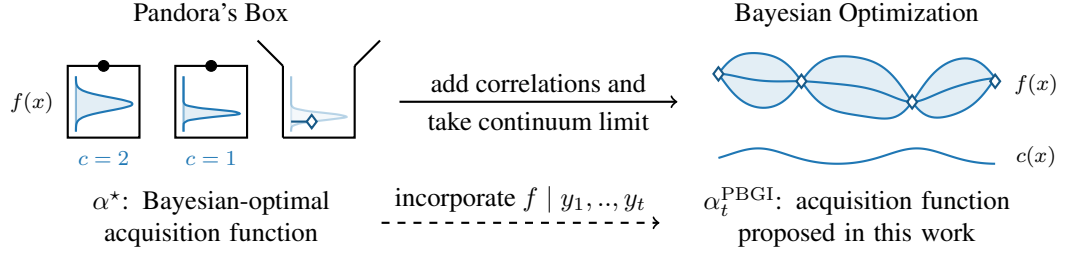


Figure 2.1: An illustration of this work’s key idea. We view cost-aware Bayesian optimization as an extension of the Pandora’s Box problem, and derive the cost-aware acquisition function α_t^{PBG} by incorporating the posterior into the Bayesian-optimal Pandora’s Box acquisition function α^* .

the best open box minus the expected total costs of all boxes opened so far, and is written

$$\mathbb{E} \max_{1 \leq t \leq T} f(x_t) - \mathbb{E} \sum_{t=1}^T c(x_t) \quad (2.4)$$

where T is a random variable that denotes the number of opened boxes, indicating that the policy terminates at time $T + 1$.

If we subtract the objective (2.4) from $\mathbb{E} \sup_{x \in X} f(x)$, which is constant with respect to the policy, we obtain the sum of the simple regret objective (2.1) defined in Section 2.2 and expected total costs. *The Pandora’s Box problem is therefore equivalent to a special case of cost-aware black-box optimization*, specifically the cost-per-sample variant of Section 2.2, where (a) the domain X is a finite set, and (b) the objective function f is random, with independent $f(x)$ and $f(x')$ for $x \neq x'$. We will return to this point in the sequel, but first study the Pandora’s Box problem in more detail.

2.3.2 Optimally solving Pandora’s Box

The Pandora’s Box problem gives rise to an explore-exploit tradeoff: a policy must balance the opportunity gained from learning the value of the reward contained

inside the box with the cost of opening it. Since the reward distributions are known, this tradeoff is captured within a Markov decision process (MDP). By general MDP theory, there exists an optimal policy describing which box, if any, one should open for a given configuration—we call such a policy *Bayesian-optimal*.

This MDP can be solved explicitly, with a remarkably simple solution, first derived in an economics setting by Weitzman (1979). We start by associating with each box $x \in X$ a number $\alpha^*(x)$ known as the *Gittins index* (Gittins et al., 2011b). Define

$$\alpha^*(x) = g \quad \text{where } g \text{ solves} \quad \text{EI}_f(x; g) = c(x) \quad (2.5)$$

where $\text{EI}_f(x; y)$, previously defined in (2.3) of Section 2.2, is the *expected improvement of x relative to y* —the same expression which appeared in the expected improvement acquisition function variants α_t^{EI} and α_t^{EIPC} . Note that, unlike in those cases, α^* is *not time-dependent* due to the lack of correlations or conditioning. Since $\text{EI}_f(x; g)$ is strictly decreasing in g , the root-finding problem (2.5) admits a unique solution for every value of $c(x)$.

To understand what $\alpha^*(x)$ represents, consider a single closed box x , and suppose there is a second, open box with reward f^* . Is opening box x better than taking the reward f^* from the open box? This amounts to whether the expected improvement from opening x balances out the opening cost $c(x)$: one can show that opening x is better if and only if $\text{EI}_f(x; f^*) > c(x)$. The value $\alpha^*(x)$ tells us *how large does the alternative reward f^* need to be, for stopping and taking it to be at least as good as opening box x* —a kind of *fair value* which makes different boxes directly comparable to one another.

If we decide which box to open via the aforementioned fair values, we obtain the *Gittins index policy*, which proceeds as follows. At each time t , let $f_t^* =$

$\max_{1 \leq \tau \leq t} f(x_\tau)$ be the maximum reward among all open boxes, and let x_t^* be the box of maximum Gittins index value $\alpha^*(x)$ among unopened boxes, with ties broken according to a given ordering. With this notation, using a tie-breaking rule that stops as early as possible—but noting that other tie-breaking rules, including stopping as late as possible, or stopping with some probability, are also valid—we get:

- If $f_t^* < \alpha^*(x_t^*)$, the policy opens box x^* .
- If $f_t^* \geq \alpha^*(x_t^*)$, the policy stops and receives terminal reward f_t^* .

It turns out that opening boxes according to the order determined by their fair values, in the sense above, is not only a good idea, but is outright Bayesian-optimal. We state this formally as follows.

Theorem 1 (Weitzman (1979)). *Let X be a finite set, let $f : X \rightarrow \mathbb{R}$ be a finite-mean random function for which $f(x)$ is independent of $f(x')$ for $x \neq x'$, and let $c : X \rightarrow \mathbb{R}_+$, without loss of generality, be deterministic. Then, for the cost-per-sample problem, the policy defined by maximizing the Gittins index acquisition function α^* with its associated stopping rule is Bayesian-optimal.*

In the language of Bayesian optimization, this means that not only is there an explicit Bayesian-optimal policy for the Pandora’s Box setting, but this policy also *takes the form of maximizing an acquisition function*. This gives an explicit solution for the cost-per-sample setting, thereby showing Pandora’s Box fits our original goal of finding a simplified decision problem that sheds insights on cost-aware Bayesian optimization. For an alternative proof, see Kleinberg et al. (2016, Theorem 1). Using Lagrangian relaxation, we show that the obtained solution carries over to the expected budget-constrained setting.

Theorem 2. *Consider the expected budget-constrained problem, with the assumptions of Theorem 1. Assume the problem is feasible and the constraint is active, namely $\min_{x \in X} c(x) < B < \sum_{x \in X} c(x)$. Then there exists a $\lambda > 0$ and a tie-breaking rule such that the policy defined by maximizing the Gittins index acquisition function $\alpha^*(\cdot)$, defined using costs $\lambda c(x)$, is Bayesian-optimal.*

A proof is given in Appendix A.2. This result extends a special case of Aminian et al. (2024, Theorem 1). Compared to that work, we consider only Pandora’s Box, but allow general reward distributions—including those with infinite support, such as Gaussian rewards. The optimal λ depends on the budget constraint B implicitly via a convex optimization problem given in Appendix A.2. In budget-constrained problems, we therefore view λ as a hyperparameter, which controls the degree to which the algorithm is risk-averse vs. risk-seeking—precisely what we argued was missing from EIPC in Section 2.2.

One can intuitively understand λ using a needle-in-haystack metaphor. Suppose one wants to find the best needle, but can only search a fraction of the haystack in expectation, represented by the budget B . The key phrase is *in expectation*: if the search is not promising, one can stop early and avoid wasting budget, otherwise one can search more of the haystack. This prompts the question: in what situations should one continue searching? The answer depends on the interplay between the budget B , costs c , and best value f_t^* seen so far, and is encoded by λ in Theorem 2. Larger λ -values correspond to smaller budgets B , incentivizing one to search less of the haystack. Crucially, the optimal acquisition function α^* behaves differently for different λ -values, and therefore for different budgets: roughly, smaller budgets cause α^* to explore less. We will return to this in Section 2.3.3 and Figure 2.3.

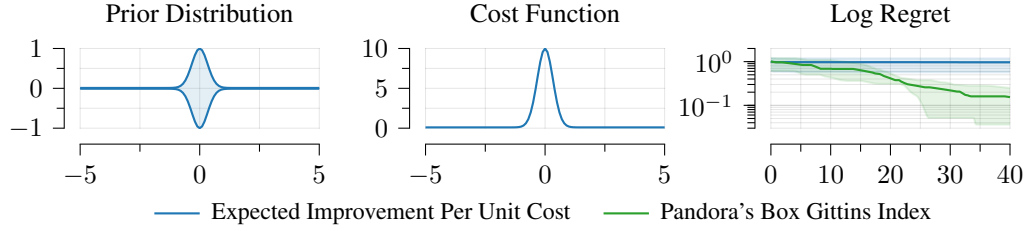


Figure 2.2: A Bayesian optimization problem with varying costs on which LogEIPC—a numerically stable implementation of EIPC, see Section 2.4 and Appendix A.4.1—has poor performance, inspired by Astudillo et al. (2021, Section A). The domain is $X = [-500, 500]$, which we visualize on the subinterval $[-5, 5]$. Left: illustration of the non-uniform prior variance, which is given by a Matérn-5/2 kernel scaled by a narrow bump function. Center: the cost function, which is a narrow bump-shaped function. Right: median regret curves and quartiles for LogEIPC and PBGI. Legend refers only to regret curves.

2.3.3 An acquisition function class for cost-aware Bayesian optimization

To adapt α^* to the Bayesian optimization setting, we need to handle two differences: (i) X does not need to be discrete, and (ii) a general probabilistic model is used for f . Since Theorem 1 ostensibly requires $f(x)$ to be independent of $f(x')$ for all $x \neq x'$, the key question is *how to incorporate data and spatial correlations* into α^* . We propose to do so in the simplest and most obvious way: namely, at each time t , we plug the posterior distribution $f \mid x_{1:t}, y_{1:t}$ in place of f . This yields three variants, depending on the precise cost-aware setting one is interested in:

1. *Budget-constrained*: define *Pandora's Box Gittins index (PBGI)* acquisition function

$$\alpha_t^{\text{PBGI}}(x) = g \quad \text{where } g \text{ solves} \quad \text{EI}_{f \mid x_{1:t}, y_{1:t}}(x; g) = \lambda c(x) \quad (2.6)$$

and λ is a hyperparameter that should be tuned to match the evaluation budget B .

2. *Cost-per-sample*: we can directly apply α_t^{PBGI} in this setting as well, but now λ is instead interpreted as unit-conversion factor which ensures costs and rewards have the same units, and the Pandora’s Box stopping rule is used for deciding when to terminate the optimization procedure and return the best observed value.
3. *Adaptive decay*: here, we do not have a pre-defined budget or cost-based stopping rule. Define the *Pandora’s Box Gittins index with adaptive decay (PBGI-D)* acquisition function $\alpha^{\text{PBGI-D}}(x)$ analogously to $\alpha^{\text{PBGI}}(x)$, but with a time-dependent λ_t schedule, set according to the Pandora’s Box stopping rule. Specifically, λ_0 is initialized to a given value, then set

$$\lambda_{\tau+1} = \begin{cases} \frac{\lambda_{\tau}}{\beta} & \text{if } f_{\tau}^* \geq \alpha_{\tau}^{\text{PBGI}}(x_{\tau}^*; \lambda_{\tau}) \\ \lambda_{\tau} & \text{otherwise} \end{cases} \quad (2.7)$$

where $\beta > 1$ is a decay factor that is used to decrease λ_{τ} at any time τ when the Pandora’s Box stopping rule triggers. The advantage of this variant is that it can be more robust to different ranges of the policy hyperparameters: we discuss this further in Appendix A.4.3.

To understand this acquisition function class, one can think of it via the following approximation: for the general cost-aware Bayesian optimization problem, we (a) correctly incorporate observed data into the prior to obtain the posterior, but then (b) pick new samples according to the rule that would have been Bayesian-optimal if the posterior had no correlations. Said differently, α^{PBGI} arises from exactly solving a simplified dynamic program, where the simplification is of a spatial nature, rather than the usual temporal lookahead. One can therefore expect this acquisition function to work best in situations where correlations are not the decisive factor for determining performance.

In what problems does this happen? In stationary kernels, correlations encode local dependence. Therefore, one can expect α^{PBGI} to be approximately-optimal in settings where the key decisions involve choosing between different far-away data points. One can intuitively expect this to occur more often in high-dimensional problems, where the volume of the search space is large and most points are far away from each other. We will examine this point empirically in the sequel.

Computation. To compute α^{PBGI} efficiently, note that $y \mapsto \text{EI}_\psi(x; y)$ is monotone. As a result, the value g in (2.6) can be computed efficiently via bisection search. In Appendix A.2.2, we show that (i) the gradient of α^{PBGI} can be computed straightforwardly via an explicit analytical expression without any additional optimization, and (ii) the resulting computational costs are closer to those of expected improvement than those of expensive multi-step-lookahead-based approaches.

Extension to stochastic and non-automatically-differentiable costs. One can show that Theorem 1 holds in the more general case where the cost function is random, as long as one substitutes $c(x)$ with its mean. The same holds for Theorem 2, as shown in Appendix A.2.5. In this setting, therefore, stochasticity of $c(x)$ affects performance, but does not change the optimal strategy. Building on these observations, we propose extensions of α_t^{PBGI} to the more general setting where costs are modeled using a log-normal process in Appendix A.2.4.

Qualitative behavior and comparisons. Compared to cost-unaware acquisition functions such as EI, PBGI can be more risk-averse if costs are large or more risk-seeking if costs are small. In varying-cost budget-constrained settings, this tradeoff is mediated by λ , and the obtained decisions can differ significantly

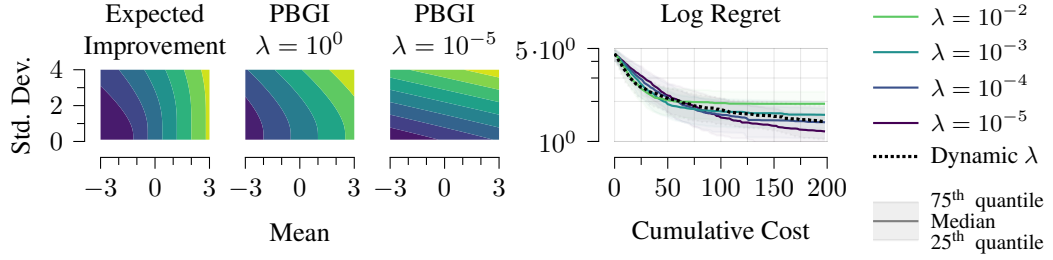


Figure 2.3: Left: contour plots showing how EI (left) and PBGI (center-left, center-right) depend on the posterior mean and standard deviation at a given point (lighter colors indicate higher values). We see that PBGI values high standard deviation more than EI. Right: PBGI performance across values of λ , under the setup of the Bayesian regret experiment of Section 2.4 with $d = 8$. We plot the median of a set of samples using $n = 256$ random seeds, along with quartiles to show variability. We see that large λ -values decrease regret sooner, but eventually lose out to smaller λ -values.

from those of widely-used baselines such as EIPC. In particular, PBGI can make qualitatively different decisions on problems where there is a high-variance point with a large cost, among a set of many low-variance low-cost points. In Figure 2.2, we adapt the construction of Astudillo et al. (2021, Section A) into a one-dimensional Bayesian optimization problem with a non-stationary prior, and observe that EIPC—even in its numerically stable logarithmic form described in Section 2.4—has substantially worse performance than PBGI.

The PBGI acquisition functions depends on $f \mid x_{1:t}, y_{1:t}$ through its mean and standard deviation at each point. We plot this in Figure 2.3. This shows for large λ that PBGI can resemble EI, whereas for small λ it is nearly linear. Specifically, in the $\lambda \rightarrow 0$ limit, we have

$$\alpha_t^{\text{PBGI}}(\cdot) \approx \mu_t(\cdot) + \sigma_t(\cdot) \sqrt{2 \log \frac{\sigma_t(\cdot)}{\lambda c(\cdot)}} \quad (2.8)$$

where μ_t and σ_t are the mean and standard deviation of $f \mid x_{1:t}, y_{1:t}$. This expression is similar to the upper confidence bound (UCB) acquisition function whose dependence is exactly linear, with heterogeneous learning rate parameter set to

$\eta_t = \sqrt{2 \log \frac{\sigma_t(x)}{\lambda c(x)}}$. A derivation is given in Appendix A.2.3. For small λ , one can thus view PBGI as giving a way to automatically tune UCB’s confidence parameter in a careful way depending on $c(x)$.

2.4 Experiments

We now empirically evaluate the Gittins-index-based acquisition function on cost-aware problems. We also evaluate on the same problems with a spatially-constant cost function, a setting we term *uniform costs*—this facilitates comparisons with classical, cost-unaware baselines. In both cases, mirroring practical settings, we work with a deterministic, algorithm-independent evaluation budget.

We implement all methods in BoTorch (Balandat et al., 2020) using Matérn Gaussian processes with smoothness $\nu = 5/2$. For Bayesian regret experiments where objectives are sampled from the prior, we fix the length scale to be $\kappa = 10^{-1}$ for both the prior and the posterior. For synthetic and empirical experiments, we apply maximum marginal likelihood optimization to dynamically adjust the length scale every iteration. To ensure that our results are not sensitive to these and other hyperparameter choices, all experiments were repeated with alternatives given in Appendix A.4.6. Each experiment was repeated for 16 seeds to assess variability, unless stated otherwise. Experimental details are in Appendix A.3.

We evaluate both PBGI variants from Section 2.3.3, namely α^{PBGI} with $\lambda = 10^{-4}$, and $\alpha^{\text{PBGI-D}}$ with $\lambda_0 = 10^{-1}$ and $\beta = 2$. To assure ourselves that performance differences are not primarily due to tuning, we deliberately use the same λ -values for PBGI and the same (λ_0, β) -values for PBGI-D on all problems. Comparisons with other (λ_0, β) -values can be found in Appendix A.4.3.

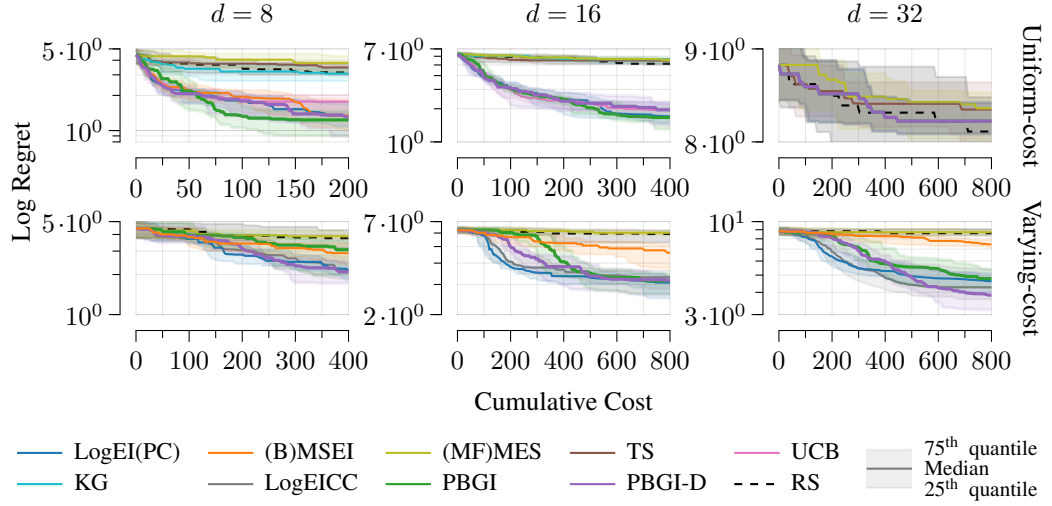


Figure 2.4: Regret curves for objective functions sampled from the prior, shown using medians, as well as quartiles to indicate experiment variability. We see in the cost-aware setting that both PBGI variants usually exhibit comparable performance to LogEIPC and LogEICC, with PBGI-D decisively outperforming other baselines. In the uniform-cost setting, the story is similar for $d = 8$ and $d = 16$: PBGI and PBGI-D perform comparably to the best baselines, which are LogEI and UCB, as well as MSEI for $d = 8$. However, for $d = 32$, all methods perform comparably to random search, with PBGI, PBGI-D, LogEI, and UCB having near-identical median performance. See Figure A.8 for an alternative visualization using mean and standard error.

For varying-cost problems—that is, those with spatially non-constant cost—we compare with *(log) expected improvement per unit cost (LogEIPC)* (Snelson et al., 2003; Snoek et al., 2012), *(log) expected improvement with cost cooling (LogEICC)* (Lee et al., 2020), *multi-fidelity max-value entropy search (MFMES)* (Takeno et al., 2020) and *budgeted multi-step expected improvement (BMSEI)*, (Astudillo et al., 2021), which was shown in that work to have state-of-the-art cost-aware performance. For uniform-cost problems—that is, those with constant costs—we compare with *log expected improvement (LogEI)* (Ament et al., 2023), *Thompson sampling (TS)*, *upper confidence bound (UCB)* (Srinivas et al., 2009), *max-value entropy search (MES)* (Wang and Jegelka, 2017), *knowledge gradient (KG)* (Frazier et al., 2009), and *multi-step expected improvement (MSEI)* (Jiang et al., 2020). We choose these

because (i) they are standard, and (ii) acquisition function optimization succeeds for them on our problems, reducing confounding. We work with logarithmic expected improvement variants (LogEI, LogEIPC, LogEICC), as they have recently been shown to have substantially better numerical and optimization properties than their traditional counterparts (Ament et al., 2023)—see Appendix A.4.1 for details.

2.4.1 Bayesian regret

For our first experiment, we examine how well the proposed acquisition functions perform on random functions sampled from the prior. To quantify the effect of problem difficulty, we vary the dimension of the domain $X = [0, 1]^d$, and consider $d \in \{8, 16, 32\}$. Results, in terms of empirical regret curves and their associated quartiles, are shown in Figure 2.4. Additional results for $d = 4$, demonstrating comparable performance between both PBGI variants and all baselines, as well as sensitivity comparisons involving the Gaussian process prior with varying kernel hyperparameters such as different smoothness values and length scales, are included in Appendix A.3.

In the low-dimensional case of $d = 8$, PBGI and LogEI variants achieve similar or better performance to the non-myopic (B)MSEI baseline. Once we increase dimension to $d = 16$, we see bigger differences, with PBGI variants, LogEI variants, and UCB now decisively outperforming (B)MSEI. Notably, the PBGI variants are also competitive in the uniform-cost setting—in spite of being designed for cost-aware problems. This can be explained via the curse of dimensionality: as dimension increases, the problem begins to look more like the uncorrelated Pandora’s Box problem where using Gittins index is Bayesian-optimal. Eventually, however, the problem becomes too difficult for meaningful progress to be made within our

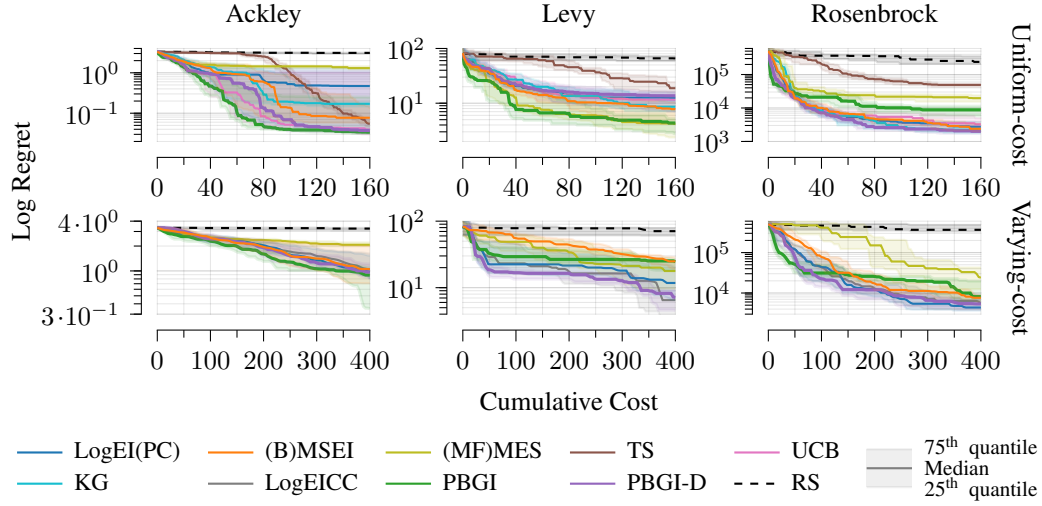


Figure 2.5: Synthetic benchmark regret curves, shown using medians, as well as quartiles to assess variability. All objective functions are defined with dimension $d = 16$. We see in the cost-aware setting that PBGI, PBGI-D, LogEIPC, and LogEICC all perform similarly on the heavily-multimodal Ackley function, matching or outperforming the non-myopic BMSEI baseline. On the Levy and Rosenbrock functions, PBGI-D matches—and for some cost budgets outperforms—all baselines, including the non-myopic BMSEI. Under uniform costs, PBGI performs well on Ackley and Levy, but is outperformed by PBGI-D and most baselines on Rosenbrock. See Figure A.8 for an alternative visualization using mean and standard error.

computational budget, as seen for the uniform-cost problem with $d = 32$, where all deterministic methods perform near-identically and no method outperforms random search.

2.4.2 Synthetic benchmarks

Next, we consider standard synthetic global optimization benchmark functions. To represent a variety of geometric properties, we examine the *Ackley*, *Levy*, and *Rosenbrock* functions. A visualization of the two-dimensional versions of these functions is given in Appendix A.1.

Figure 2.5 presents results for $d = 16$. Additional results for $d \in \{4, 8\}$ showing

that PBGI and all baselines perform similar, are given in Appendix A.3. We see that the behavior of different acquisition functions varies according to the function. On the Ackley function, PBGI outperforms PBGI-D and the baselines. In contrast, on the Levy function, PBGI is best in the uniform-cost setting, but PBGI-D, LogEIPC, and LogEICC are best in the varying-cost setting. We conclude that PBGI can in principle offer stronger performance than PBGI-D, as long as λ is not-too-suboptimal, while PBGI-D tends to be less-performant but is more robust to this hyperparameter choice.

We also examine performance on the banana-shaped Rosenbrock-function, which has a small number of local optima (Shang and Qiu, 2006). In this case, PBGI-D performs the strongest, matching the LogEI variants, while outperforming PBGI and (B)MSEI. This can intuitively be explained by the one-step optimality of expected improvement, which better-exploits the less-multimodal objective, while PBGI and multi-step-based acquisition functions are more conservative and may therefore require different tuning to perform best. We conclude that PBGI-D may be a better choice in settings where there is a potential mismatch between the objective and the prior in terms of degree of multimodality, as this can be counteracted in part by its decay behavior, particularly compared to PBGI.

2.4.3 Empirical objectives

Finally, we benchmark PBGI policies on three empirical global optimization problems motivated by applied challenges: *Pest Control* where $d = 25$ (Oh et al., 2019), *Lunar Lander* where $d = 12$ (Eriksson et al., 2019), and *Robot Pushing* where $d = 14$ (Wang and Jegelka, 2017). Detailed descriptions of these problems and associated cost functions are in Appendix A.3. Note that, for Lunar Lander and

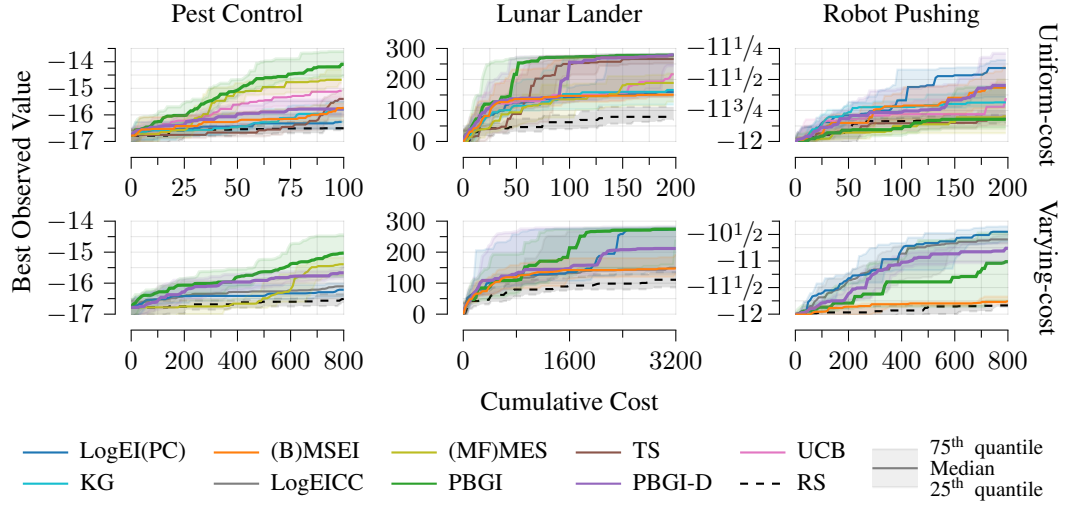


Figure 2.6: Empirical benchmark regret curves, shown using medians, as well as quartiles to show variability. We see in both the varying-cost and uniform-cost settings that PBGI exhibits strong performance on the Pest Control and Lunar Lander problems. On the Robot Pushing problem, LogEI variants perform strongest, with PBGI-D performing second-strongest and matching or outperforming all other baselines. See Figure A.8 for an alternative visualization using mean and standard error.

Robot Pushing, the cost functions used are not automatically-differentiable. To handle this challenge and illustrate how our acquisition function can be used when the cost function is unknown, we apply unknown-cost PBGI and baseline variants, where the costs are modeled using a second independent log-Gaussian process: details on this unknown-cost PBGI variant, including its analytic form, are given in Appendix A.2.4.

From Figure 2.6, we see that the PBGI matches or outperforms baselines on Pest Control and Lunar Lander, in both the varying-cost and uniform-cost settings. On the other hand, PBGI performs poorly on Robot Pushing, where instead LogEI variants perform best and PBGI-D performs second-best; the non-myopic BMSEI baseline also performs poorly. This mirrors behavior previously seen on the Rosenbrock function, from which we suspect that a mismatch between prior and objective multimodality may be at play here as well. Note also that unlike in

the Bayesian regret experiments, UCB’s performance is far from strongest. This may be in part because we tune UCB using the schedule of Srinivas et al. (2009), which is derived specifically for Bayesian regret, and may be less-ideal for other settings. In comparison, PBGI-D works reasonably well on five of the six cases.

2.5 Conclusion

In this paper, we introduced a new acquisition function class for cost-aware Bayesian optimization, the *Pandora’s Box Gittins index*, based on an unexplored connection between Bayesian optimization and the *Pandora’s Box* problem from economics. We observed promising performance from two variants of this acquisition function class on both cost-aware problems which are the focus of this work, and, additionally, on classical uniform-cost problems. Performance gains tended to be largest on higher-dimensional and multi-modal problems. Our work constitutes a first step toward integrating ideas from Gittins index theory, including insights from generalizations of Pandora’s Box, and related areas such as queueing theory, into Bayesian optimization.

CHAPTER 3

COST-AWARE STOPPING FOR BAYESIAN OPTIMIZATION

This chapter is based on joint work with Linda Cai, Alexander Terenin, Peter I. Frazier, and Ziv Scully (Xie et al., 2025).

Abstract. In automated machine learning, scientific discovery, and other applications of Bayesian optimization, deciding when to stop evaluating expensive black-box functions in a cost-aware manner is an important but underexplored practical consideration. A natural performance metric for this purpose is the *cost-adjusted simple regret*, which explicitly captures the trade-off between solution quality and cumulative evaluation cost. Existing stopping rules for Bayesian optimization are either heuristic, or are theoretically grounded but designed to optimize simple regret without accounting for evaluation costs; as a result, they provide no guarantees against unnecessary evaluations when costs are high. We propose a *principled cost-aware stopping rule* for Bayesian optimization that adapts to varying evaluation costs without heuristic tuning. Our rule is grounded in a theoretical connection to state-of-the-art cost-aware acquisition functions, namely the Pandora’s Box Gittins Index (PBGI) and log expected improvement per cost (LogEIPC). When paired with either acquisition function, we prove that the resulting policy satisfies a theoretical guarantee bounding the expected cost-adjusted simple regret. Across synthetic tasks and empirical benchmarks including hyperparameter optimization and neural architecture size search, pairing our stopping rule with PBGI or LogEIPC usually matches or outperforms other acquisition-function–stopping-rule pairs in terms of cost-adjusted simple regret. Code available at: [HTTPS://GITHUB.COM/QIANJANEXIE/COSTAWARESTOPPINGBAYESOPT](https://github.com/QIANJANEXIE/COSTAWARESTOPPINGBAYESOPT).

3.1 Introduction

Bayesian optimization is a framework designed to efficiently find approximate solutions to optimization problems involving expensive-to-evaluate black-box functions, where derivatives are unavailable. Such problems arise in applications like hyperparameter tuning (Snoek et al., 2012), robot control optimization (Martinez-Cantin, 2017), and material design (Zhang et al., 2020). It works by iteratively, (a) forming a probabilistic model of the black-box objective function based on data collected thus far, then (b) optimizing an acquisition function, which balances exploration-exploitation tradeoffs, to carefully choose a new point at which to observe the unknown function in the next iteration.

In this work, we consider the *cost-aware* setting, where one must pay a cost to collect each data point, and study *adaptive stopping rules* that choose when to stop the optimization process. After stopping at some terminal time, we measure performance in terms of simple regret, which is the difference in value between the best solution found so far and the global optimum. Collecting a data point can reduce simple regret, but incurs cost in order to do so.

As an example, consider using a cloud computing environment to tune the hyperparameters of a classifier in order to optimize a performance metric on a given test set. Training and evaluating test error takes some number of CPU or GPU hours, that may depend on the hyperparameters used. These come with a financial cost, billed by the cloud computing provider, which define our cost function. The objective value is the business value of deploying the trained model under the given hyperparameters—a given function of the model’s accuracy. From this perspective, simple regret can be understood as the opportunity cost for deploying a suboptimal model instead of the optimal one. Motivated by the need to balance

cost-performance tradeoff in examples such as above, we aim to design stopping rules that optimizes *expected cost-adjusted simple regret*, defined as the sum of simple regret and the cumulative cost of data collection.

Several stopping rules have been proposed for Bayesian optimization. Simple heuristics—such as fixing the number of iterations or stopping when the best value remains unchanged for a certain number of iterations—are widely used, but can either stop too early or lead to unnecessary evaluations. Other approaches include acquisition-function-based rules that stop when metrics such as probability of improvement (Lorenz et al., 2015) fall below a preset threshold, or when expected improvement (EI) (Nguyen et al., 2017; Zhou et al., 2024) or knowledge gradient (KG) (Frazier and Powell, 2008) fall below the cost per sample. Regret-bound-based rules instead aim to optimize simple regret (Makarova et al., 2022; Ishibashi et al., 2023; Wilson, 2024; Li et al., 2023a). These approaches are primarily developed for the classical uniform-cost setting and do not explicitly account for varying evaluation costs. Our work instead provides a principled stopping rule with theoretical guarantees that applies both to the uniform-cost setting and more generally to varying evaluation costs.

In this work, we study how to design cost-aware stopping rules, motivated by two primary factors. First, state-of-the-art cost-aware acquisition functions such as the Pandora’s Box Gittins Index (PBGi) (Xie et al., 2024) and log expected improvement per cost (LogEIPC) (Ament et al., 2023) have not yet been studied in the adaptive stopping setting. As our experiments in Section 3.4 show, introducing adaptive stopping can further improve the performance of these acquisition functions compared to fixed-budget evaluations. Second, while regret-based stopping rules such as UCB–LCB (Makarova et al., 2022) guarantee low simple regret, they often

incur excessive evaluation costs, resulting in high cost-adjusted regret, as reflected in our experiments.

Our work builds upon the Bayesian-optimal Pandora’s Box decision principle underlying the PBGI acquisition function and extends it to the adaptive stopping setting in correlated Bayesian optimization. Furthermore, we show that an existing stopping rule proposed for the EI acquisition function in the classical cost-unaware setting (Nguyen et al., 2017; Zhou et al., 2024), admits a principled choice of the stopping threshold as the evaluation cost per sample, rather than a heuristic tuning parameter. Under this choice, the resulting stopping condition is equivalent to the PBGI stopping rule, and naturally extends to the cost-aware setting. This stopping rule can therefore naturally be paired with acquisition functions derived from either the PBGI or the EI design principles. Acquisition functions such as KG or MES, which are derived from alternative principles such as value-of-information or entropy search, inherently require their own matched stopping rules.

Our contributions are summarized as follows:

1. *A Novel Cost-Aware Stopping Rule.* We present an adaptive stopping rule derived from Pandora’s box theory, which also naturally extends to the expected improvement design principle, establishing a unified and principled framework applicable to both classic and cost-aware Bayesian optimization.
2. *Theoretical Guarantees.* We prove in Theorem 4 that our stopping rule, when paired with the PBGI or LogEIPC acquisition function, satisfies a theoretical upper bound on the expected cost-adjusted regret, which constitutes the first theoretical guarantee of this type for any adaptive stopping rule for Bayesian optimization.
3. *Empirical Validation.* We conduct a systematic empirical evaluation across

multiple acquisition-function—stopping-rule pairs in cost-aware Bayesian optimization. Our results on synthetic tasks and empirical benchmarks show that pairing our proposed stopping rule with the PBGI or LogEIPC acquisition function usually matches or outperforms other pairs.

3.2 Bayesian Optimization and Adaptive Stopping

In black-box optimization, the goal is to find an approximate optimum of an unknown objective function $f : X \rightarrow \mathbb{R}$ using a limited number of function evaluations at points $x_1, \dots, x_T \in X$ where X is the search space and T is a given search budget, potentially chosen adaptively. The convention measures performance in terms of *expected simple regret* (Garnett, 2023b, Sec. 10.1), given by

$$\mathcal{R} = \mathbb{E} \left[\min_{1 \leq t \leq T} f(x_t) - \inf_{x \in X} f(x) \right], \quad (3.1)$$

where the expectation is taken over all sources of randomness, including the sequence of points $x_1, \dots, x_T$ selected by the algorithm. Bayesian optimization approaches this problem by building a probabilistic model of f —typically a Gaussian process (GP) (Rasmussen and Williams, 2006b), conditioned on the observed data points $(x_t, y_t)_{t=1}^T$, where $y_t = f(x_t)$. For each iteration $t = 1, \dots, T$, an acquisition function $\alpha_t : X \rightarrow \mathbb{R}$ then guides the selection of new samples by carefully balancing the exploration-exploitation tradeoff arising from uncertainty about f .

3.2.1 Cost-aware Bayesian Optimization

Cost-aware Bayesian optimization (Lee et al., 2020; Astudillo et al., 2021) extends the above setup to account for the fact that evaluation costs can vary across the search space. For instance, in the hyperparameter tuning example of Section 3.1,

costs can vary according to the time needed to train a machine learning model under a given set of hyperparameters x . Cost-aware Bayesian optimization handles this by introducing a cost function $c : X \rightarrow \mathbb{R}^+$, which may be known or unknown ahead. The cost function, or observed costs, are then used to construct the acquisition function α_t .

In this work, we focus primarily on the *cost-per-sample* formulation (Chick and Frazier, 2012; Cashore et al., 2016; Xie et al., 2024) of cost-aware Bayesian optimization, which seeks methods with stopping time τ , not necessarily fixed, that achieve a low *expected cost-adjusted simple regret*

$$\mathcal{R}_c = \mathbb{E} \left[\underbrace{\min_{1 \leq t \leq \tau} f(x_t) - \inf_{x \in X} f(x)}_{\text{simple regret}} + \underbrace{\sum_{t=1}^{\tau} c(x_t)}_{\text{cumulative cost}} \right]. \quad (3.2)$$

One can also work with the *expected budget-constrained* formulation (Xie et al., 2024), which incorporates budget constraints explicitly, and seeks algorithms which achieve a low expected simple regret under an expected evaluation budget. Here, performance is evaluated in terms of

$$\mathcal{R} = \mathbb{E} \left[\min_{1 \leq t \leq \tau} f(x_t) - \inf_{x \in X} f(x) \right] \quad \text{subject to} \quad \mathbb{E} \left[\sum_{t=1}^{\tau} c(x_t) \right] \leq B. \quad (3.3)$$

The stopping rules we study can be applied in this setting as well: we discuss this in Section 3.3.1.

For both settings, we work with a few acquisition functions—chiefly, *log expected improvement per cost* (*LogEIPC*) (Xie et al., 2024) (the log variant (Ament et al., 2023) of the expected improvement per cost (Snoek et al., 2012)) and the *Pandora’s Box Gittins Index* (*PBGI*) (Xie et al., 2024). At iteration t , let $x_{1:t} := \{x_1, \dots, x_t\}$ be the evaluated points, $y_{1:t} := \{y_1, \dots, y_t\}$ be their observed values, $y_{1:t}^* = \min_{1 \leq s \leq t} y_s$ be the current best observed value, and $f \mid x_{1:t}, y_{1:t}$ be the posterior distribution of

f . Then the LogEIPC acquisition function is defined as

$$\alpha_t^{\text{LogEIPC}}(x) := \log \frac{\text{EI}_{f|x_{1:t}, y_{1:t}}(x; y_{1:t}^*)}{c(x)}, \quad (3.4)$$

i.e., the logarithm of the expected improvement divided by the cost; and the PBGI acquisition function is defined as

$$\alpha_t^{\text{PBGI}}(x) := \begin{cases} g & \text{s.t. } \text{EI}_{f|x_{1:t}, y_{1:t}}(x; g) = c(x), \quad x \notin x_{1:t}, \\ f(x), & x \in x_{1:t}, \end{cases} \quad (3.5)$$

where the value g is the threshold such that the expected improvement over g equals the cost¹ and $\text{EI}_\psi(x; y) = \mathbb{E}[\max(y - \psi(x), 0)]$ is the expected improvement at point x with respect to some random function $\psi : X \rightarrow \mathbb{R}$ and a baseline value y . We also consider classical cost-unaware acquisition functions—*lower confidence bound (LCB)* and *Thompson sampling (TS)*; see Garnett (2023b) for details.

3.2.2 Adaptive Stopping Rules

To the best of our knowledge, stopping rules for Bayesian optimization typically do not incorporate cost explicitly into the stopping criterion. We broadly categorize existing methods as follows:

Criteria based on convergence or significance of improvement. This includes empirical convergence, namely stopping when the best value remains unchanged for a fixed number of iterations, or the *global stopping strategy (GSS)* (Bakshy et al., 2018), which stops when the improvement is no longer statistically significant relative to the inter-quartile range (i.e., the range between the 25th percentile and the 75th percentile) of prior observations. In the multi-fidelity

¹For an evaluated point x , we set $g = f(x)$ since the posterior at x collapses to a point mass at the observed value $f(x)$ and its evaluation cost can be treated as 0.

setting, Foumani and Bostanabad (2025) proposed stopping when the high-fidelity surrogate’s estimated optimum has stabilized over a window of iterations, which is related to cost-awareness but differs from our setting with explicitly varying evaluation costs.

Acquisition-based criteria. This includes stopping rules based on acquisition functions such as PI, EI, and KG. These approaches typically stop when the acquisition value falls below a threshold—such as a predetermined constant (Lorenz et al., 2015; Locatelli, 1997), the median of the initial acquisition values (Ishibashi et al., 2023), or the cost of sampling (Nguyen et al., 2017; Zhou et al., 2024; Frazier and Powell, 2008). They are typically defined via fixed thresholds and are primarily designed for the uniform-cost setting, without explicitly accounting for varying evaluation costs.

Regret-based criteria. This includes stopping rules based on confidence bounds such as *UCB-LCB* (Makarova et al., 2022) and the *gap of expected minimum simple regrets* (Ishibashi et al., 2023). These stop when certain estimated regret bounds fall below a preset or data-driven threshold. The related *probabilistic regret bound (PRB)* stopping rule (Wilson, 2024) stops when estimated simple regret is below a small threshold ϵ with confidence $1 - \delta$. In contrast to these forward-looking rules, Li et al. (2023a) propose a backward-looking stopping rule that detects when the search has entered a locally convex region and stops once an estimate of local regret falls below a predetermined threshold.

In settings beyond Bayesian optimization, Chick and Frazier (2012) have proposed a cost-aware stopping rules for finite-domain sequential sampling problems with independent values. Although this formulation allows for varying costs, it does not extend to general Bayesian optimization settings which use correlated GP

models.

3.3 A Stopping Rule Based on the Pandora’s Box Gittins Index

In this work, we propose a new stopping rule tailored for two state-of-the-art acquisition functions used in cost-aware Bayesian optimization: LogEIPC and PBGI, introduced in Section 3.2. As discussed by Xie et al. (2024), these acquisition functions are closely connected, arising from two different approximations of the intractable dynamic program which defines the Bayesian-optimal policy for cost-aware Bayesian optimization. We now show that this connection can be used to obtain a principled stopping criterion to be paired with both acquisition functions.

A stopping rule for PBGI. To derive a stopping rule for PBGI, consider first the Pandora’s Box problem, from which it is derived. Pandora’s Box can be seen as a special case of cost-per-sample Bayesian optimization, as introduced in Section 3.2, where $X = \{1, \dots, N\}$ is a discrete space and f is assumed uncorrelated. In this setting, the observed values do not affect the posterior distribution—meaning, we have $f(x') \mid x_{1:t}, y_{1:t} = f(x')$ at all unobserved points x' —and thus the acquisition function value at point x is time-invariant and can be written simply as $\alpha^{\text{PBGI}}(x)$, where $\text{EI}_f(x; \alpha^{\text{PBGI}}(x)) = c(x)$. Following Weitzman (1979), one can show using a Gittins index argument that selecting x_t to minimize α^{PBGI} is Bayesian-optimal under minimization—the algorithm which does so achieves the smallest expected cost-adjusted regret, among all adaptive algorithms.

One critical detail applied in the optimality argument of Weitzman (1979) is that the policy defined by α^{PBGI} is *not* Bayesian-optimal for any fixed deterministic

T . Instead, it is optimal only when the stopping time T is chosen according to the condition

$$\min_{x \in X \setminus \{x_1, \dots, x_t\}} \alpha^{\text{PBG I}}(x) \geq y_{1:t}^*, \quad (3.6)$$

where $\geq$ can be replaced by $>$ ², and as before, $y_{1:t}^*$ is the best value observed so far.

In the correlated setting we study, Xie et al. (2024) extend $\alpha^{\text{PBG I}}$ to define an acquisition function by recomputing it at each iteration t based on the posterior mean and variance, which defines $\alpha_t^{\text{PBG I}}$ as in Equation (3.5). In order to also extend the associated stopping rule, a subtle design choice arises: should we use $\alpha_{t-1}^{\text{PBG I}}$ (before posterior update) or $\alpha_t^{\text{PBG I}}$ (after posterior update) in Equation (3.6)? While prior theoretical work (Gergatsouli and Tzamos, 2023) adopts the former choice, we instead use the latter (after posterior update), resulting in the stopping rule

$$\min_{x \in X \setminus \{x_1, \dots, x_t\}} \alpha_t^{\text{PBG I}}(x) \geq y_{1:t}^*. \quad (3.7)$$

This choice is motivated by two reasons. First, we argue it more faithfully reflects the intuition behind Weitzman’s original stopping rule. In the independent setting, $\alpha^{\text{PBG I}}(x)$ represents a kind of *fair value* for point x (Kleinberg et al., 2016). For an evaluated point $x \in \{x_1, \dots, x_t\}$, this is simply the observed function value. For an unevaluated point $x \notin \{x_1, \dots, x_t\}$, the fair value reflects uncertainty in $f(x)$ and the cost $c(x)$. The stopping rule Equation (3.7) then says to *stop when the best fair value is among the already-evaluated points*—namely, when no point provides positive expected gain relative to its cost if evaluated in the next round, conditioned on current observations. In the correlated setting, the fair value naturally extends to $\alpha_t^{\text{PBG I}}$, because this incorporates all known information at a given time. Second,

²Following Xie et al. (2024, Appendix B), optimality holds under any tie-breaking rule in the cost-per-sample setting, but only under a carefully-chosen stochastic tie-breaking rule in the expected budget-constrained setting. For simplicity, we use the stopping-as-early-as-possible tie-breaking rule in this paper.

we show in Appendix B.3.2 that using α_t^{PBGI} yields tangible empirical gains in cost-adjusted regret.

Connection with LogEIPC. The above reasoning may at first seem to be fundamentally tied to the Pandora’s Box problem and its Gittins-index-theoretic properties. We now show that it admits a second interpretation in terms of log expected improvement per cost, which arises from a completely different one-time-step approximation to the Bayesian-optimal dynamic program for cost-aware Bayesian optimization in the general correlated setting.

To show this, we start with definitions above and apply a sequence of transformations. Recall that for any unevaluated point $x \notin \{x_1, \dots, x_t\}$, $\alpha_t^{\text{PBGI}}(x)$ is defined in Equation (3.5) to be the solution to

$$\text{EI}_{f|x_{1:t}, y_{1:t}}(x; \alpha_t^{\text{PBGI}}(x)) = c(x), \quad (3.8)$$

and since $\text{EI}_\psi(x; y)$ is strictly increasing in y , any inequality involving $\alpha_t^{\text{PBGI}}(x; c)$ can be lifted through the EI function without changing its direction, which means $\alpha_t^{\text{PBGI}}(x) \geq y_{1:t}^*$ holds if and only if $\text{EI}_{f|x_{1:t}, y_{1:t}}(x; y_{1:t}^*) \leq c(x)$. This implies that the PBGI stopping rule from Equation (3.7) is equivalent to stopping when

$$\text{EI}_{f|x_{1:t}, y_{1:t}}(x; y_{1:t}^*) \leq c(x), \quad \forall x \in X \setminus \{x_1, \dots, x_t\}, \quad (3.9)$$

meaning *stop when no point’s expected improvement is worth its evaluation cost*.

Rearranging the inequality and taking logs, we can rewrite this condition using the LogEIPC acquisition function³ :

$$\max_{x \in X \setminus \{x_1, \dots, x_t\}} \alpha_t^{\text{LogEIPC}}(x; y_{1:t}^*) \leq 0. \quad (3.10)$$

³Excluding evaluated points is not theoretically required but often beneficial in practice, as numerical stability adjustments can cause their expected improvement to remain positive.

When $c(x) \equiv c_0$, Equation (8) reduces to the EI thresholding stopping rule (Nguyen et al., 2017; Zhou et al., 2024)

$$\max_{x \in X \setminus \{x_1, \dots, x_t\}} \alpha_t^{\text{EI}}(x; y_{1:t}^*) \leq c_0. \quad (3.11)$$

We call the stopping rule given by the equivalent conditions (Equations (3.7), (3.9) and (3.10)) the *PBGI/LogEIPC stopping rule*. Figure 3.1 gives an illustration of how the rule behaves in a simple setting, demonstrating that it is more conservative—preferring to stop earlier—when the cost is high.

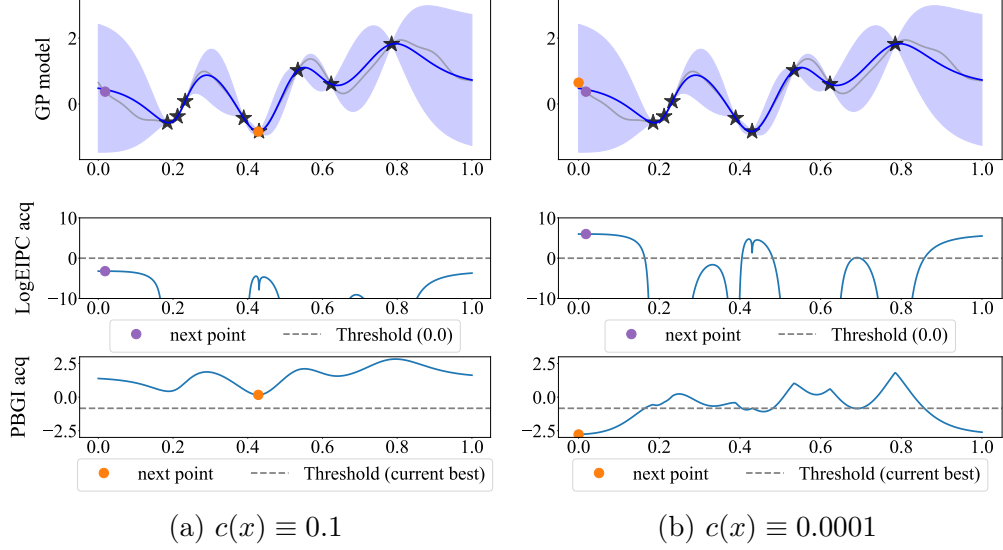


Figure 3.1: Illustration of the PBGI/LogEIPC stopping rule under a uniform-cost setting. When the cost-per-sample is large ($c(x) \equiv 0.1$), the maximum LogEIPC acquisition value falls below the threshold 0.0 and the minimum PBGI acquisition value exceeds the threshold (current best observed value), indicating stopping; when the cost-per-sample is small ($c(x) \equiv 0.0001$), the maximum LogEIPC acquisition value remains above the threshold 0.0 and the minimum PBGI acquisition value is smaller than the threshold (current best observed value), indicating no stopping.

3.3.1 Theoretical Guarantees on Cost-Adjusted Regret

The connection between PBGI and LogEIPC in fact goes beyond a shared stopping rule. In Lemma 3, we prove that when paired with this stopping rule, both

acquisition functions guarantee that at each iteration before stopping, the expected improvement at the selected point is at least its evaluation cost.

Lemma 3. *Let X be compact, and let $f : X \rightarrow \mathbb{R}$ be a random function with prior mean $\mu(\cdot)$. Consider a Bayesian optimization algorithm that begins at some initial point $x_1 \in X$ with cost $C = c(x_1)$, acquires subsequent points using either the PBGI or LogEIPC acquisition function, and terminates according to the PBGI/LogEIPC stopping rule. Let $\tau = \min_{t \geq 1} \{\sup_{x \in X} \alpha_t^{\text{LogEIPC}}(x) \leq 0\}$ be the algorithm's stopping time, and denote the posterior expected improvement function by $\alpha_t^{\text{EI}}(x) = \text{EI}_{f|x_{1:t}, y_{1:t}}(x; y_{1:t}^*)$. Then, for all $t < \tau$, $\alpha_t^{\text{EI}}(x_{t+1}) \geq c(x_{t+1})$.*

As a consequence of this claim, we show in Theorem 4 that the PBGI/LogEIPC stopping rule, when paired with PBGI or LogEIPC, achieves cost-adjusted simple regret no worse than stopping immediately after the initial evaluation. Notably, many acquisition function–stopping rule pairs fail to satisfy this *no-worse-than-immediate* property in practice (see Figures 3.2 and 3.3), largely because most stopping rules are designed with simple regret alone in mind and do not explicitly account for evaluation costs. Moreover, this guarantee is tight in the worst case: when evaluation costs are sufficiently large relative to the maximum achievable improvement, immediate stopping is optimal (see Proposition 36). To our knowledge, this is the first formal guarantee on cost-adjusted simple regret for Bayesian optimization, generalizing beyond simpler settings such as Pandora's Box, which assumes independent and discrete evaluations.

Theorem 4. *Consider the setting and algorithm specified in Lemma 3. Let $U := \mu(x_1) - \mathbb{E}[\min_{x \in X} f(x)] < \infty$, then the algorithm's expected cost-adjusted simple*

regret is bounded by

$$\mathbb{E} \left[y_{1:\tau}^* - \min_{x \in X} f(x) + \sum_{t=1}^{\tau} c(x_t) \right] \leq \mathbb{E} \left[\underbrace{y_1 - \min_{x \in X} f(x) + c(x_1)}_{\text{cost-adjusted regret of immediate stopping}} \right] = U + C. \quad (3.12)$$

This result yields a bound on expected cumulative cost and, under the natural assumption that practical evaluations have a positive minimum cost, a high-probability finite-time termination guarantee.

Corollary 5. *Consider the setting and algorithm specified in Lemma 3. Then the expected cumulative cost of the algorithm is bounded by $\mathbb{E} [\sum_{t=1}^{\tau} c(x_t)] \leq U + C$. Further, if evaluation costs are uniformly bounded below by a constant $c_0 > 0$, i.e., $c(x) \geq c_0, \forall x \in X$, then for any $\delta \in (0, 1)$, the algorithm terminates in at most $\frac{U+C}{\delta \cdot c_0}$ iterations with probability $1 - \delta$.*

Remark 6. *Lemma 3 and Theorem 4 are stated for simplicity under initialization from a single point x_1 . The results extend directly to arbitrary initial datasets $(x_{1:t_0}, y_{1:t_0})$ by conditioning on the observations and treating the resulting posterior as the effective prior. The same guarantees then hold for subsequent iterations $t \geq t_0$, with all quantities defined with respect to this updated posterior.*

Implication in the Budgeted Setting We first note that in the discrete Pandora’s Box setting, under an expected budget constraint B , minimizing $\alpha_t^{\text{PBG}}(x)$ is Bayesian-optimal: it achieves the lowest *expected simple regret*, among all algorithms satisfying the expected cumulative cost within the budget (i.e., $\mathbb{E} [\sum_{t=1}^{\tau} c(x_t)] \leq B$), and the cost function used in defining $\alpha_t^{\text{PBG}}(x)$ is $\lambda c(x)$ for some *cost-scaling factor* λ which depends on B (Xie et al., 2024, Theorem 2).

This result, at first, appear to be completely Pandora’s-Box-theoretic: it requires

X to be discrete and f to be independent. In the more general correlated setting of cost-aware Bayesian optimization, however, Bayesian optimality of PBGI may no longer hold, and the relationship between B and the choice of λ is not immediately clear. Theorem 4 helps bridge this gap: it provides an upper bound on the expected cumulative cost up to the stopping time, which in turn yields the following principled choice of λ that ensures compliance with the budget constraint.

Corollary 7. *Consider the setting, algorithm, and notation specified in Lemma 3 and Theorem 4, but with costs rescaled by a factor $\lambda > 0$: both the acquisition values and the stopping conditions are computed using $\lambda c(\cdot)$. For some budget $B > C$, if the cost-scaling factor is set to $\lambda = \frac{U}{B-C}$, then the algorithm’s expected cumulative unscaled cost satisfies $\mathbb{E} [\sum_{t=1}^{\tau} c(x_t)] \leq B$.*

If this choice of λ proves overly conservative—leading to underspending within the budget—it can be paired with the PBGI-D variant of Xie et al. (2024), which starts with $\lambda = \lambda_0$ and halves it each time stopping is triggered. Choosing $\lambda_0 = U/(B - C)$ aligns the initial fixed- λ phase with the budget in expectation, while ensuring that the adaptive decay is activated as designed. In Figure B.7 in Appendix B.3, we show PBGI-D with this choice of λ_0 is competitive.

All proofs in this section are deferred to Appendix B.1.

3.3.2 Practical Implementation Considerations for the PBGI/LogEIPC Stopping Rule

Expressing objective and cost in common units. Evaluation costs are often measured in different units than the objective, such as time versus accuracy. To compare them directly, we rescale costs by a constant $\lambda > 0$, the conversion rate

between objective improvement and cost. For instance, if one is willing to spend 1000 seconds to improve test accuracy by 0.01, then $\lambda = 10^{-5}$. Importantly, in the cost-per-sample setting we mainly study, λ is a fixed constant set by the problem provider, rather than a parameter tuned heuristically by the user. The budgeted setting without a natural unit conversion is discussed in Section 3.3.1.

Unknown costs. In practice, evaluation costs are often not known in advance. They can be modeled either (i) deterministically, using domain knowledge (e.g., proportional to model size in hyperparameter tuning), or (ii) stochastically, via assumptions on the distribution of $c(x)$ (our theoretical guarantees still hold after replacing $c(x)$ with $\mathbb{E}[c(x)]$). In Section 3.4, we present empirical results under both modeling approaches. For the latter, we follow Astudillo et al. (2021, Proposition 2) to model $\ln c(x)$ as a GP with posterior mean $\mu_{\ln c}$ and variance $\sigma_{\ln c}^2$. To compute LogEIPC or PBGI, as well as their related stopping rules (including those in the prior work and ours), we replace $c(x)$ in Equations (3.4), (3.5) and (3.9) with $\mathbb{E}[c(x)] = \exp(\mu_{\ln c}(x) + (\sigma_{\ln c}(x))^2/2)$. The empirical performance is preserved. See Appendix B.3.3 for further discussion.

Preventing spurious stops. Although our stopping rule has theoretical guarantees, in practice, it—like other adaptive stopping rules—can still suffer from spurious stops caused by two main sources. First, early in the optimization process, the GP model parameters often fluctuate significantly as new data points are collected, causing unstable stopping signals. To mitigate this, we enforce a stabilization period consisting of the first few evaluations, during which we do not allow any stopping rule to trigger. Second, imperfect optimization of the acquisition function—which is especially common in continuous higher-dimensional search

spaces—can lead to misleading stopping signals. To handle this, we smooth the stopping signal in this setting by applying a moving average over a window of W iterations, and require the stopping condition to hold consistently under this smoothed signal before stopping. See Figure B.1 for an illustration. We use the same window size $W = 20$ for both the stabilization period and smoothing to minimize the number of parameters in our main experiments. An ablation study on the effect of window size can be found in Appendix B.3.4.1.

3.4 Experiments

To evaluate our proposed PBGI/LogEIPC stopping rule, we design three complementary sets of experiments that progressively evaluate its performance. First, we consider an idealized, low-dimensional Bayesian regret setting in which the GP model exactly matches the true objective. This controlled setting allows us to isolate the effect of different stopping rules without interference from model misspecification and acquisition function optimization errors. Then, we move to a higher-dimensional Bayesian regret setting where acquisition-function optimization is imperfect. Finally, we evaluate in practical settings with objective model misspecification, using the LCBench hyperparameter tuning benchmark and the NATS neural architecture size search benchmark. In each case, we compare pairs of acquisition functions and stopping rules, which we describe next.

Acquisition functions. We consider four common acquisition functions that were discussed in Section 3.2: PBGI, LogEIPC, LCB, and TS—chosen for their competitive performance, computational efficiency, and close connections to existing stopping rules.

Baselines. We compare the proposed PBGI/LogEIPC stopping rule against several stopping rules from prior work: *UCB-LCB* (Makarova et al., 2022), *LogEIPC-med* (Ishibashi et al., 2023), *SRGap-med* (Ishibashi et al., 2023), and *PRB* (Wilson, 2024). We also include two simple heuristics used in practice: *Convergence* and *GSS*. *UCB-LCB* stops once the gap between upper and lower confidence bounds falls below a configurable threshold θ . *LogEIPC-med* stops when the log expected improvement per cost drops beneath $\log(\eta)$ plus the median of its initial I values. *SRGap-med* stops when the gap of the expected minimum simple regret falls below χ times the median of its initial I values. *PRB* triggers once a probabilistic regret bound satisfies the regret tolerance ϵ and confidence δ parameters. *Convergence* stops as soon as the best observed value remains unchanged for k iterations. *GSS* stops if the recent improvement is less than $\phi \times \text{IQR}$ over the past k trials where IQR denotes the inter-quartile range of current observations. Finally, we include two reference baselines: *Immediate*, which stops right after the initial evaluation (see Section 3.3.1), and *Hindsight*, which, for each trial, selects the stopping time that yields the lowest cost-adjusted simple regret in hindsight, thus providing a lower bound on achievable performance.

In our experiments, unless specified otherwise, we follow parameter values recommended in the literature, and set $\theta = 0.01$, $\eta = 0.01$, $\chi = 0.01$, $I = 20$, $\epsilon = 0.1$ for Bayesian regret experiments, $\epsilon = 0.5\%$ of the best test error for empirical experiments, $\delta = 0.05$, $k = 5$, and $\phi = 0.01$. For the stabilization and smoothing window, we use $W = 20$, aligned with the initial window size $I = 20$ used in EI-med (Ishibashi et al., 2023).

Each experiment is repeated with 50 random seeds to assess variability, and we report the mean with error bars (2 times the standard error) for each stopping

rule. Each trial, in the sense of a run with a distinct random seed, is capped at a fixed number of iterations; if a stopping rule is not triggered within this limit, the stopping time is set to the cap. Details are provided in Appendix B.2.

3.4.1 Bayesian Regret

We first evaluate our PBGI/LogEIPC stopping rule on random functions sampled from prior. Specifically, objective functions are sampled from Gaussian processes with Matérn 5/2 kernels with length scale 0.1, defined over spaces of dimension $d = 1$ and $d = 8$. We consider a variety of evaluation cost function types, including uniform costs, linearly increasing costs in terms of parameter magnitude, and periodic costs that vary non-monotonically over the domain.

In the 1-dimensional experiments, we perform an exhaustive grid search over $[0, 1]$ to isolate the effect of stopping behavior from numerical optimization. In the 8-dimensional setting, due to instability from higher dimensionality, we apply moving average with a window size of 20 when computing the PBGI/LogEIPC stopping condition. See Figure B.1 in Appendix B.2 for an illustration. More details on our experiment setup, and computational considerations are provided in Appendices B.2 and B.3.

Figure 3.2 shows a comparison of cost-adjusted regret for pairings of acquisition functions and stopping rules under different cost-scaling factors $\lambda = 0.1, 0.01, 0.001$ in the 1-dimensional setting, and under different cost regimes (uniform, linear, periodic) in 8-dimensional setting.

In the 1-dimensional setting, the optimization problem is relatively straightforward and all acquisition functions have nearly identical hindsight optima, inde-

pendent of cost scale or cost type. From the plots, our PBGI/LogEIPC stopping rule, when paired separately with the PBGI and LogEIPC acquisition functions, delivers competitive performance for each λ , and is particularly strong in handling high-cost scenarios—those with large λ .

In the 8-dimensional experiments, however, clear gaps emerge. Under uniform and linear costs, PBGI, LogEIPC, and LCB exhibit similar hindsight optima, substantially better than TS, while in the periodic case, PBGI and LogEIPC achieve much better hindsight optima than LCB and TS. In every cost regime, combining our stopping rule with the PBGI or LogEIPC acquisition function yields cost-adjusted regret that nearly matches the hindsight optimal.

This demonstrates that our PBGI/LogEIPC stopping rule is relatively robust to the increased optimization difficulty. Further discussions of our experiments across multiple cost types and cost-scaling factors can be found in Appendix B.3.

3.4.2 Automated Machine Learning Benchmarks

We then evaluate on two empirical automated machine learning benchmarks based on use cases from hyperparameter optimization and neural architecture search: LCBench (Zimmer et al., 2021) and NATS-Bench (Dong et al., 2021). Unless otherwise specified, results in this section assume a known cost function; results under the unknown-cost setting are provided in Appendix B.3.

LCBench (Zimmer et al., 2021) provides training data of 2,000 configurations over a 7-dimensional hyperparameter space, evaluated on 35 OpenML (Vanschoren et al., 2014) datasets. In the *unknown-cost* experiments, the cost is the full 51-epoch training time. For the known-cost setting, we observe that training time scales

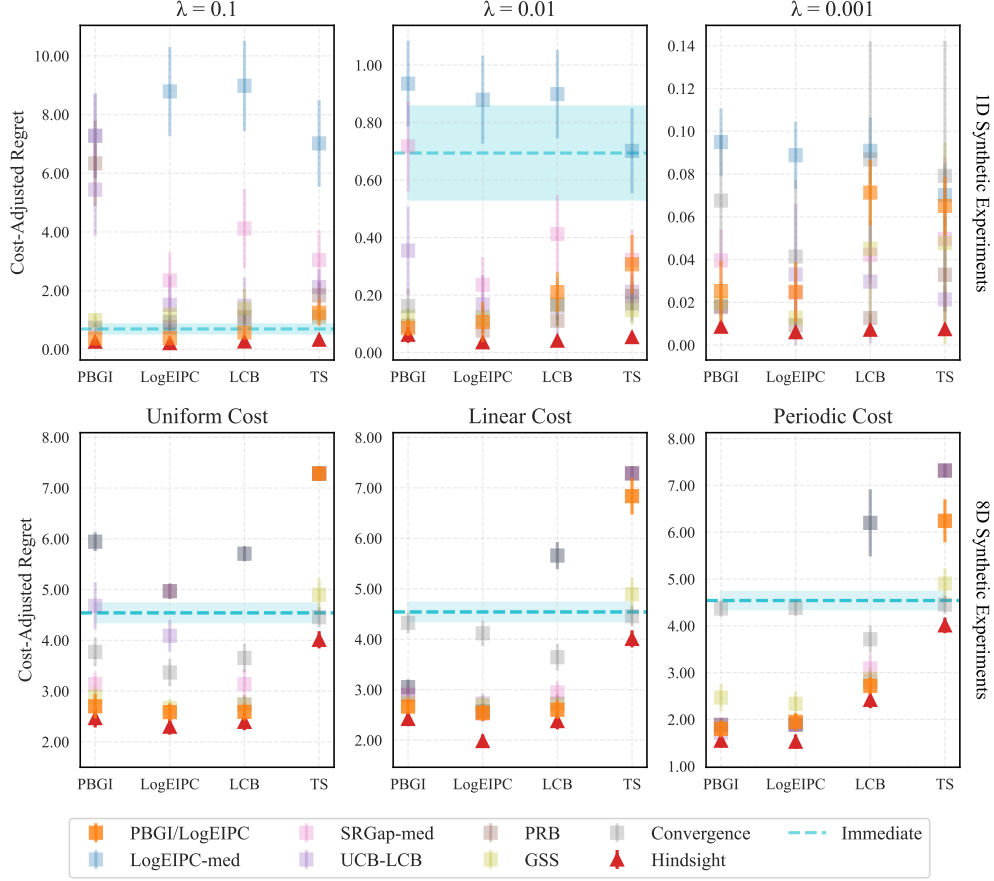


Figure 3.2: Cost-adjusted simple regret across acquisition-stopping rule pairs in 1D and 8D Bayesian regret setting. In 1D, objective functions are sampled from a GP with a Matérn-5/2 kernel and a linear cost function scaled by $\lambda = 0.1, 0.01, 0.001$. The Immediate baseline is omitted at $\lambda = 0.001$ due to its much higher regret (mean 0.6942, error bar [0.5314, 0.8570]). In 8D, objective functions are also drawn from a GP with a Matérn-5/2 kernel, using three cost functions scaled by $\lambda = 0.01$.

approximately linearly with the number of model parameters. Based on a linear regression fit (see Appendix B.2), we adopt αp as a proxy for training time, where p is the number of model parameters and α is a dataset-specific estimated coefficient.

NATS-Bench (Dong et al., 2021) provides a search space of 32,768 neural architectures varying in channel sizes across layers, evaluated on three datasets. As in LCBench, we similarly approximate the full 90-epoch (training and validation) runtime using a linear function $\alpha F + \beta$ of the number of floating point operations

F ; see Appendix B.2 for details.

We report the mean and error bars (2 times standard error) of the *cost-adjusted simple regret*, where we consider the evaluation costs to be some representative cost-scaling factor λ (see Section 3.3.2) multiplied with cumulative runtime. Simple regret is computed as the difference between (a) test error of the configuration with best validation error at the stopping time and (b) the best test error over all configurations. We present the results using proxy runtime here, and defer to Appendix B.3 the additional results under (i) the *cost model misspecification* scenario (proxy runtime used during Bayesian optimization but actual runtime used for final performance evaluation), and (ii) the *unknown-cost* scenario (actual runtime used during Bayesian optimization).

Figure 3.3 presents performance of acquisition function–stopping rule pairs on LCBench and NATS-Bench. For LCBench, we report min–max normalized cost-adjusted simple regret (see definition in Appendix B.3) aggregated across 35 datasets in Figure 3.3 evaluated under three representative⁴ cost-scaling values. For NATS-Bench, we report cost-adjusted simple regret under $\lambda = 10^{-5}$, with additional results under two other representative cost-scaling factors shown in Figures B.11 to B.13 in Appendix B.3.

Per-dataset LCBench results are provided in Figures B.16 to B.18 in the appendix. Across the 35 LCBench tasks, around 75% show our PBGI/LogEIPC stopping rule performing competitively when paired with either PBGI or LogEIPC. The remaining outliers are mostly (aside from two exceptions) very small datasets with < 10000 instances (i.e., data points) that might lead to severe model mis-

⁴These λ values are chosen to avoid degenerate cases—neither so large that the policy stops after only a few evaluations (e.g., $\lambda = 10^{-4}$ on NATS-Bench, see Figure B.14) nor so small that evaluations are effectively free.

specification. We also report the rank frequency of our PBGI/LogEIPC stopping rule when paired separately with the PBGI and LogEIPC acquisition functions, evaluated on medium-to-large datasets (with $> 10,000$ instances) and all datasets; see Table 3.1. A diagnosis of the underperforming datasets can be found in Appendix B.3.3.1, where we find that the apparent underperformance is likely driven by benchmark-intrinsic val/test mismatch rather than by the stopping rule itself.

Additional results under the cost model misspecification setting and the unknown-cost setting are provided in Figures B.14 and B.15 in Appendix B.3, along with comparisons between our adaptive stopping rule and fixed-iteration baselines. Overall, our PBGI/LogEIPC stopping rule consistently performs strongly in terms of cost-adjusted simple regret, when paired with either PBGI or LogEIPC. We also report, in Table B.1 of Appendix B.3, how often each stopping rule fails to trigger within the 200-iteration cap: baselines such as SRGap-med and UCB-LCB often fail to stop early, particularly on NATS-Bench, whereas ours reliably stops before the cap.

In empirical benchmarks such as LCBench and NATS-Bench, where objective misspecification relative to the GP prior is unavoidable, our stopping rule remains robust. When paired with PBGI, it delivers performance close to the hindsight optimal, except on two NATS tasks with potential severe misspecification. Pairing with LogEIPC is slightly less competitive, likely reflecting PBGI’s stronger robustness under misspecification. Indeed, Figure B.7 in Appendix B.3 compares acquisition functions under fixed-budget evaluations (without stopping) and shows that PBGI consistently attains lower simple regret than LogEIPC. Thus, while our stopping rule applies to both acquisition functions, the matched PBGI combination is a particularly strong default under objective misspecification.

λ	Acq	Top 1		Top 2		Top 3	
		Large	All	Large	All	Large	All
10^{-3}	PBGI	20.0%	20.0%	70.0%	48.6%	80.0%	60.0%
	LogEIPC	40.0%	31.4%	65.0%	45.7%	70.0%	62.8%
10^{-4}	PBGI	40.0%	28.6%	70.0%	51.5%	80.0%	62.9%
	LogEIPC	30.0%	22.9%	50.0%	45.8%	85.0%	68.7%
10^{-5}	PBGI	50.0%	28.6%	70.0%	51.5%	90.0%	74.4%
	LogEIPC	20.0%	20.0%	50.0%	48.6%	85.0%	71.5%

Table 3.1: Percentage of LCBench tasks ranked in the Top- k ($k = 1, 2, 3$) by cost-adjusted simple regret for among acquisition-stopping rule pairs (excluding the hindsight stopping rule), for our PBGI/LogEIPC stopping rule paired with LogEIPC or PBGI, evaluated on large datasets (with $> 10,000$ instances) and on all datasets, for cost-scaling factor $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}\}$.

3.5 Conclusion

We develop the *PBGI/LogEIPC stopping rule* for Bayesian optimization with varying evaluation costs. Paired with either the PBGI or LogEIPC acquisition function, it (a) satisfies a theoretical guarantee bounding the expected cost-adjusted simple regret (Section 3.3.1), and (b) shows strong empirical performance in terms of cost-adjusted simple regret (Section 3.4). We believe our framework can be extended to settings involving noisy, multi-fidelity or batched evaluations, as well as alternative objective formulations—for instance, applying a sigmoid transformation to test error rather than a linear one, to reflect real-world user preferences that shift sharply once error falls below a threshold.

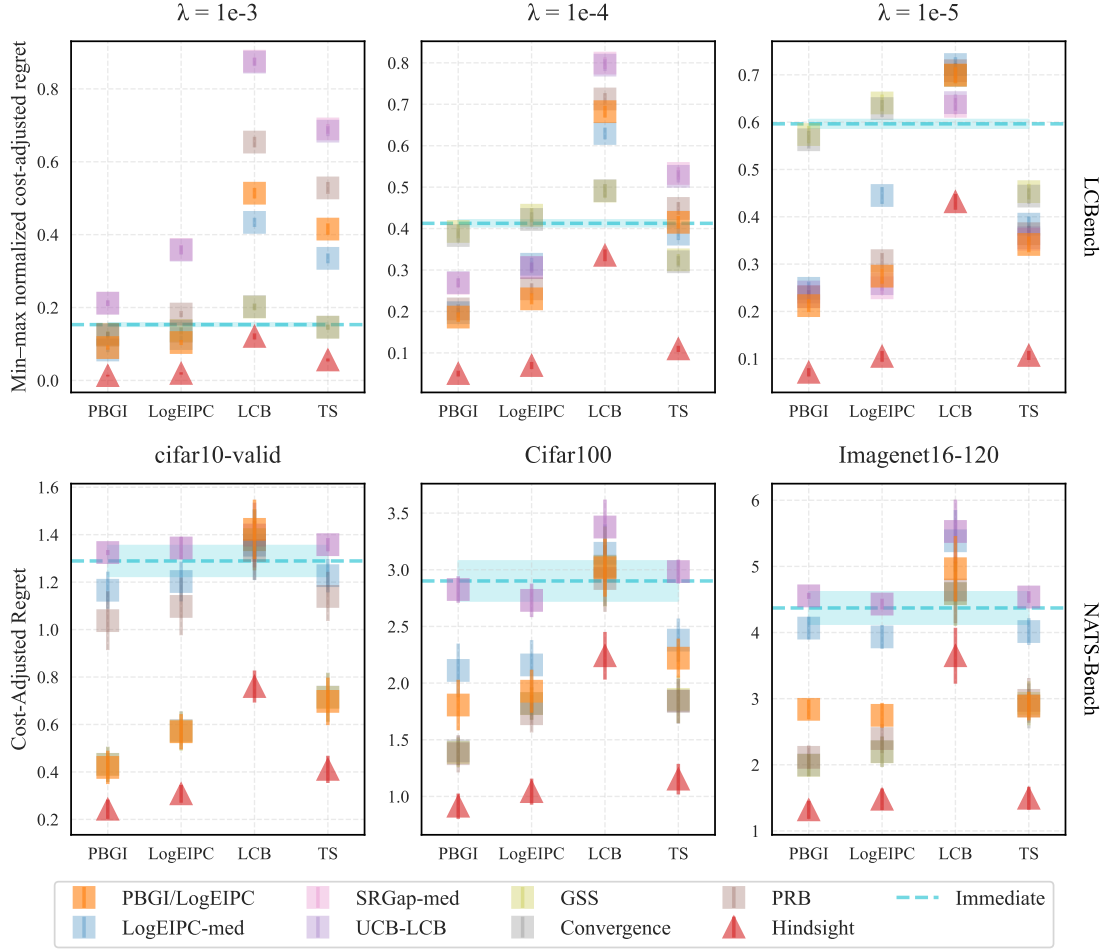


Figure 3.3: Cost-adjusted simple regret across acquisition function-stopping rule pairs on LCBench and NATS-Bench. The objective is to minimize validation error on classification tasks, with scaled proxy runtime as evaluation cost, scaled by representative values of λ (10^{-3} , 10^{-4} , 10^{-5} for LCBench and 10^{-5} for NATS-Bench). For LCBench, results are aggregated across 35 datasets using min-max normalization. Our PBGI/LogEIPC stopping rule, when paired with either LogEIPC or PBGI, typically ranks among the top 3 pairs and closely approaches the hindsight optimal on LCBench and on cifar10-valid in NATS-Bench, but slightly worse on the other two NATS datasets, likely due to model misspecification.

Part II

Bandit-Like Bayesian Decision

Problems with Multi-Stage Feedback

via the Markov Chain Gittins Index

CHAPTER 4

EFFICIENT COST-AWARE LLM EVALUATION VIA BAYESIAN BANDIT GITTINS INDICES

This chapter is based on joint work with Yueli He and Nairen Cao (Xie et al., 2026).

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 induced by batch-mean approximations, online allocation only requires table lookup and posterior updates, and it naturally handles varying evaluation costs. Across GSM8K, PIQA, AlpacaEval, and MMLU response matrices, Gittins shows two clear gains: on large-example benchmarks it outperforms configuration-level Bayesian optimization, and on large-candidate MMLU tasks it markedly outperforms cost-unaware frequentist bandit baselines. At its stopping point, the recommended arm often attains near-zero simple regret after using only about 1–10% of the exhaustive evaluation cost.

4.1 Introduction

Modern LLM evaluation is no longer a passive reporting exercise. A single evaluation study may compare many LLM configurations, each defined by choices such as the model, prompting technique, temperature, decoding strategy, or other inference settings, 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 is a frequentist bandit method for best-arm identification, while UCB-E-LRF augments this allocation rule with a low-rank model of correlations in the response matrix. We also compare with configuration-level Bayesian optimization (Snoek et al., 2012), which evaluates each selected configuration on all benchmark examples. These methods are compelling because they convert an offline response matrix or benchmark dataset into an online allocation problem, where the goal is to find a high-performing configuration under a small evaluation budget. 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.

Our experiments show two distinct regimes. Across many settings, the arm recommended at the Gittins stopping point attains near-zero simple regret using only about 1–10% of the exhaustive evaluation cost. On large-example benchmarks such as GSM8K, PIQA, and AlpacaEval, configuration-level Bayesian optimization spends each query on an all-example candidate evaluation, while Gittins can distribute partial batches and reaches low regret much earlier. On MMLU, where each subject contains 1500 candidate model–template arms, Gittins is much stronger than cost-unaware frequentist bandit rules because it combines posterior uncertainty, prior information, and evaluation cost in one allocation index.

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 approximation to batch-mean benchmark observations, 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:

- 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 approximation to batch-mean observations 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.
- We compare against UCB-E, correlation-aware UCB-E-LRF, and configuration-level Bayesian optimization, finding gains over Bayesian optimization on large-example benchmarks and marked gains over cost-unaware frequentist bandits on large-candidate benchmarks, with the stopping-time recommendation often attaining near-zero simple regret using about 1–10% of the exhaustive evaluation cost.

4.1.1 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. Lyu et al. (2026) propose SySRs, which exploits similarities among model response vectors, while 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. Earlier response-matrix work emphasizes best-arm identification through the probability of correct selection at a terminal budget. We instead evaluate recommendation quality throughout allocation via 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 bandit 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., 2011a; Scully and 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 the Gaussian batch-mean approximation makes each arm’s posterior mean follow a Gaussian random walk, which allows efficient precomputation of the index schedule.

4.2 Cost-Aware LLM Configuration Recommendation

4.2.1 Candidate Configurations, Examples, and Latent Performance

We consider an evaluation study with K candidate LLM configurations, indexed by $k \in \{1, \dots, K\}$. A candidate configuration 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 configuration k on example j . We model these example-level scores as samples from a configuration-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 recommend a configuration with large latent mean θ_k while spending as little evaluation cost as possible. Each allocation to configuration k 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

efficient-evaluation work is often cost-unaware in this sense: BanditEval (Zhou et al., 2025) considers UCB-E (Audibert and Bubeck, 2010), a frequentist bandit method for best-arm identification, together with UCB-E-LRF, its low-rank response-matrix variant, and both treat evaluations as unit-cost allocations. TRIPLE (Shi et al., 2024) and PromptEval (Polo et al., 2024) similarly focus on the special case $c_k \equiv 1$.

4.2.2 Adaptive Evaluation and Recommendation

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

$$\mathcal{H}_t = \{(I_s, Y_s, c_{I_s}) : s \leq t\}. \quad (4.1)$$

Here $I_s \in \{1, \dots, K\}$ is the configuration selected at step s , and Y_s denotes the scalar observation returned by that allocation step; it may be a single example-level score $Z_{I_s, j}$ or a batch average of several such scores. We use the term policy broadly: π includes the rule for selecting the next configuration I_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-configuration evaluation horizon is imposed, a configuration is completed once it reaches that horizon. The stopping and recommendation rule then specifies whether evaluation should continue or terminate based on the current configuration states and posterior beliefs.

4.2.3 Objectives

We consider two cost-aware formulations. The first is the fixed-cost-budget objective. Unlike reward-maximization problems that optimize cumulative rewards from the configurations selected 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:

$$\min_{\pi} \mathbb{E}_{\pi} \left[\max_k \theta_k - \theta_{\hat{k}_T} \right] \quad \text{subject to} \quad \sum_{s=1}^T c_{I_s} \leq \mathcal{C}. \quad (4.2)$$

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

$$\theta_k^{\text{mat}} = \frac{1}{N} \sum_{j=1}^N Z_{k,j} \quad (4.3)$$

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 configuration 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

$$\mathbb{E} \left[\theta_{\hat{k}_T} - \sum_{t=1}^T c_{I_t} \right]. \quad (4.4)$$

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.

4.3 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. From this section onward, we use the standard Bayesian bandit term *arm* for a candidate configuration: arm k is configuration k , A_t denotes the arm selected at global allocation step t , one pull evaluates that configuration on one example or a batch of examples, and θ_k is its latent 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 4.1 summarizes how posterior beliefs, Gittins indices, benchmark batches, and the response matrix interact over one iteration of this loop.

4.3.1 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

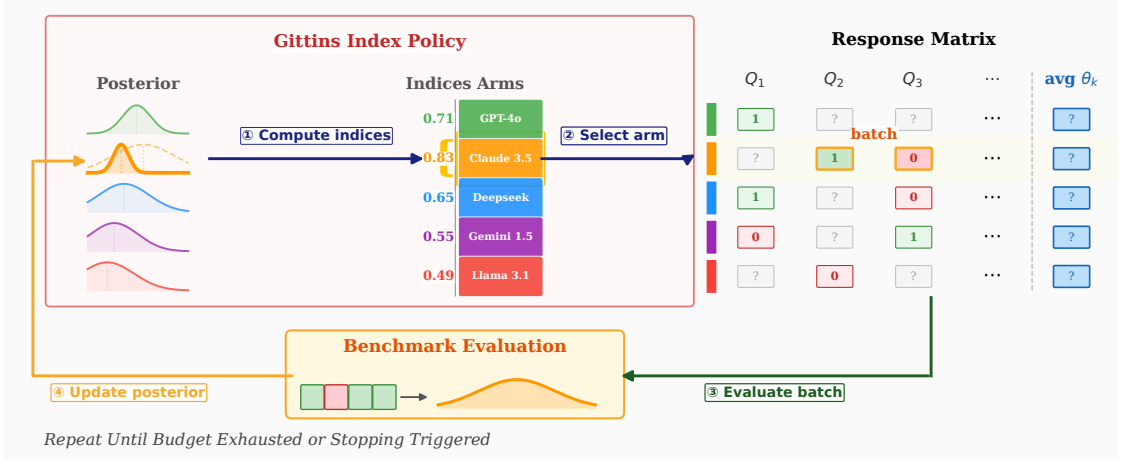


Figure 4.1: Overview of the proposed Gittins-index evaluation framework. The policy maintains posterior beliefs over candidate LLM configurations, computes Gittins indices to adaptively select the next arm batch to evaluate, observes benchmark responses from the response matrix, and updates the posterior iteratively until the evaluation budget is exhausted or a stopping condition is met.

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 4.2, let $Z_{k,j}$ be the example-level score for arm k on example j . If the i th pull of arm k evaluates batch $\mathcal{B}_{k,i}$ of size B , the scalar observation used for posterior updating is the batch mean

$$\bar{Y}_{k,i} = \frac{1}{B} \sum_{j \in \mathcal{B}_{k,i}} Z_{k,j}. \quad (4.5)$$

The batch mean has conditional mean θ_k . Let τ_k^2 denote its conditional variance,

so that, by the central limit theorem, $\bar{Y}_{k,i} \mid \theta_k$ 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 4.3.4 describes this choice.

We use the Gaussian prior-likelihood approximation

$$\theta_k \sim \mathcal{N}(\mu_0, v_0), \quad \bar{Y}_{k,i} \mid \theta_k \sim \mathcal{N}(\theta_k, \tau^2), \quad (4.6)$$

where τ^2 is the fixed observation-noise variance in the likelihood. To state the update compactly, fix a representative arm and suppress the arm index, writing θ , $\mathcal{D}_n$, μ_n , v_n , and $\bar{Y}_{n+1}$. The posterior after n local pulls is

$$\theta \mid \mathcal{D}_n \sim \mathcal{N}(\mu_n, v_n), \quad (4.7)$$

with update

$$\begin{aligned} v_{n+1} &= (v_n^{-1} + \tau^{-2})^{-1}, \\ \mu_{n+1} &= \mu_n + \frac{v_n}{v_n + \tau^2}(\bar{Y}_{n+1} - \mu_n). \end{aligned} \quad (4.8)$$

When Eq. (4.8) is applied to arm k in Algorithm 1, the symbols are read as $\mu_{k,n}$, $v_{k,n}$, and $\bar{Y}_{k,n+1}$. Under fixed τ^2 , the variance schedule for this representative arm is deterministic. In particular, the one-step variance reduction is

$$v_n - v_{n+1} = \frac{v_n^2}{v_n + \tau^2}. \quad (4.9)$$

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.

4.3.2 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

$$S_{n+1} \mid S_n = s \sim \mathcal{N}\left(s, \frac{v_n^2}{v_n + \tau^2}\right). \quad (4.10)$$

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

$$V_H(s; \alpha) = \max\{s, \alpha\}, \quad (4.11)$$

and, for $n < H$,

$$\begin{aligned} Q_n^{\text{cont}}(s; \alpha) &= -c + \mathbb{E}[V_{n+1}(S_{n+1}; \alpha) \mid S_n = s], \\ Q_n^{\text{stop}}(s; \alpha) &= \alpha, \\ V_n(s; \alpha) &= \max\{Q_n^{\text{cont}}(s; \alpha), Q_n^{\text{stop}}(s; \alpha)\}. \end{aligned} \quad (4.12)$$

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:

$$G_n^c(s) = \inf\{\alpha \in \mathbb{R} : Q_n^{\text{stop}}(s; \alpha) \geq Q_n^{\text{cont}}(s; \alpha)\}. \quad (4.13)$$

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

$$G_{k,t} = G_{n_k(t)}^{c_k}(\mu_{k,n_k(t)}). \quad (4.14)$$

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.

Markov-chain selection applies because the Gaussian random walk is a Markov chain (Dumitriu et al., 2003b; Scully and Terenin, 2025).

Proposition 8 (Bayesian optimality under required completion). *Consider the Gaussian Bayesian bandit model in Section 4.3.1 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_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

$$\sup_{\pi \in \Pi_H^{\text{req}}} \mathbb{E}_\pi \left[\theta_{\hat{k}_T} - \sum_{t=1}^T c_{A_t} \right], \quad (4.15)$$

an optimal policy is to assign terminal index $G_{k,t} = \mu_{k,H}$ to completed arms, assign $G_{k,t} = G_{n_k(t)}^{\text{ck}}(\mu_{k,n_k(t)})$ 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 C.1.2.

4.3.3 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 9 (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 $n < H$,*

$$Q_n^{\text{cont}}(s + \beta; \alpha + \beta) = Q_n^{\text{cont}}(s; \alpha) + \beta. \quad (4.16)$$

Corollary 10 (Centered continuation value). *Let $x = s - \alpha$ and define $q_n(x) := Q_n^{\text{cont}}(x; 0)$. For any $n < H$,*

$$Q_n^{\text{cont}}(s; \alpha) = q_n(s - \alpha) + \alpha. \quad (4.17)$$

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

The proof of Lemma 9 and Corollary 10 is included in Appendix C.1.3.

By Corollary 10, only the posterior-mean advantage $x = s - \alpha$ matters. Let $W_n(x) := V_n(x; 0)$, then the centered recursion is

$$\begin{aligned} W_H(x) &= \max\{x, 0\}, \\ W_n(x) &= \max\{0, q_n(x)\}, \\ q_n(x) &= -c + \mathbb{E}[W_{n+1}(X_{n+1}) \mid X_n = x], \end{aligned} \quad (4.18)$$

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

$$G_n^c(s) = s - r_n. \quad (4.19)$$

For each distinct tuple (c, v_0, τ^2, H) , offline computation stores the root schedule

$$\{r_n(c, v_0, \tau^2, H)\}_{n=0}^{H-1}. \quad (4.20)$$

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 (4.18) 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 .

See the FFT-based dynamic program in Chapter 5, Section 5.3.3, particularly (5.23), and the detailed derivation in Appendix D, Section D.2.

4.3.4 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 $\tau_{k,t}^2 \approx \mu_{k,t}(1 - \mu_{k,t})/B$ 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 for allocation and for the optional required-completion stopping rule. Under that stopping rule, the policy stops once an arm attaining the largest current Gittins index has reached horizon H . Its terminal output is restricted to completed arms: it chooses the completed arm with the largest posterior mean. 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 Proposition 8.

The simple-regret curves use a different anytime reporting convention. At every allocation step, including steps before any arm is completed, we report the arm with the largest current posterior mean among all arms,

$$\hat{k}_t \in \arg \max_{k=1,\dots,K} \mu_{k,n_k(t)}. \quad (4.21)$$

Thus an unfinished arm can be the recommendation used to compute the plotted simple regret. The completion restriction above applies only to the required-completion stopping rule and its terminal recommendation; in the experiments, the dashed Gittins stopping markers indicate where that rule would stop, while the regret curve itself is computed using the all-arm posterior-mean recommendation at each time.

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.

4.4 Experiments

Benchmarks. We evaluate adaptive allocation policies using completed LLM response matrices built on four standard benchmarks: GSM8K for grade-school mathematical reasoning (Cobbe et al., 2021), PIQA for physical commonsense reasoning (Bisk et al., 2020), AlpacaEval for instruction-following quality (Li et al., 2023b), and MMLU for multitask knowledge and reasoning (Hendrycks et al., 2020). The benchmark examples define the evaluation tasks, while the observed scores 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. AlpacaEval uses an AlpacaEval 2.0 pairwise score matrix over 153 leaderboard models and 805 instructions; entries are continuous scores in $[0, 1]$ rather than binary correctness values. 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

Algorithm 1 Gittins index evaluation for a Bayesian bandit

Require: Prior mean and variance (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.

- 1: Set $c_k = \lambda \tilde{c}_k$ for all arms k .
 - 2: Precompute $\{r_n(c, v_0, \tau^2, H)\}_{n=0}^{H-1}$ for each needed cost c .
 - 3: Initialize posterior mean $\mu_{k,0} = \mu_0$, posterior variance $v_{k,0} = v_0$, and count $n_k(0) = 0$ for all k ; set $t = 0$.
 - 4: Set $\hat{k}_0 \in \arg \max_k \mu_{k,0}$ as the recommendation by posterior mean.
 - 5: **while** budget remains **do**
 - 6: **if** $n_k(t) = H$ for all arms k **then**
 - 7: **return** $\hat{k}_t$.
 - 8: **end if**
 - 9: For completed arms ($n_k(t) = H$), set terminal index $G_{k,t} = \mu_{k,H}$.
 - 10: For unfinished arms ($n_k(t) < H$), set continuation index $G_{k,t} = G_{n_k(t)}^{c_k}(\mu_{k,n_k(t)})$.
 - 11: **if** early stopping is enabled **then**
 - 12: **if** an arm in $\arg \max_k G_{k,t}$ is completed **then**
 - 13: **return** $\arg \max_{k:n_k(t)=H} \mu_{k,H}$.
 - 14: **end if**
 - 15: **end if**
 - 16: Select arm $A_t \in \arg \max_{k:n_k(t) < H} G_{k,t}$.
 - 17: Observe batch mean $\bar{Y}_{A_t, n_{A_t}(t)+1}$.
 - 18: Update $(\mu_{A_t, \cdot}, v_{A_t, \cdot})$ using Eq. (4.8).
 - 19: 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 $t \leftarrow t + 1$.
 - 20: Set $\hat{k}_t \in \arg \max_k \mu_{k,n_k(t)}$ as the recommendation by posterior mean.
 - 21: **end while**
 - 22: **return** $\hat{k}_t$.
-

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 C.2.

Baselines. We compare two versions of the proposed Gittins index policy with UCB-E, a classical frequentist bandit method for best-arm identification (Audibert and Bubeck, 2010), UCB-E-LRF, the low-rank-factorization variant used for faster language model evaluation by Zhou et al. (2025), SySRs, a bandit method based on successive rejection that exploits similarities among model response vectors (Lyu et al., 2026), and configuration-level Bayesian optimization baselines using PBGI, LogEI, and its cost-aware variant, LogEIPC. PBGI is inherently cost-aware. We abbreviate UCB-E-LRF as LRF in figures and captions. Gittins-G denotes Gittins with the general prior, while Gittins-S selects a shared data-specific prior for each dataset or MMLU subject from four difficulty buckets: easy $\mathcal{N}(0.75, 0.01)$, medium $\mathcal{N}(0.6, 0.02)$, hard $\mathcal{N}(0.4, 0.02)$, and very hard $\mathcal{N}(0.2, 0.01)$. The exact assignments are listed in Table C.5. 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 evaluating each sampled arm 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 $\text{SR}_t = \max_k \theta_k^{\text{mat}} - \theta_{\hat{k}_t}^{\text{mat}}$ using the finite-matrix target in Eq. (4.3). 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 . In the cost-aware setting, recalling that c_k denotes the per-example evaluation cost of arm k , the allocation at step s costs $b_s c_{A_s}$; we therefore plot simple regret against cumulative evaluation cost $C_t = \sum_{s=1}^t b_s c_{A_s}$. For configuration-level Bayesian optimization baselines, one query evaluates a configuration on all benchmark examples and consumes its corresponding all-example 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.3.

For the bandit methods, we set the fixed evaluation budget to 10% of the resources required for exhaustive evaluation: 10% of the response-matrix entries in the unit-cost setting and 10% of the total exhaustive 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 4.3.4 would terminate. The Bayesian optimization baselines follow a configuration-level protocol, where evaluating one candidate on all benchmark examples reveals its aggregate score and consumes that configuration’s all-example evaluation cost. 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 Bayesian optimization runs stop before exceeding the 10% exhaustive-evaluation-cost budget. For the main

GSM8K, PIQA, and AlpacaEval plots, UCB-E and Gittins use batch size $B = 8$, while UCB-E-LRF uses $B = 32$. For the main MMLU aggregate plots, the small size bucket (at most 150 examples) uses $B = 2$ and the large size bucket (more than 400 examples) uses $B = 8$ 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, PIQA, and AlpacaEval under unit-cost and cost-aware evaluation. Each curve compares the Gittins variants, UCB-E, UCB-E-LRF, SySRs, and Bayesian optimization baselines on the same response matrices. Unless otherwise stated, shaded bands denote one standard error of the mean across repeated runs. Figure 4.2 presents these main curves. Figure 4.3 summarizes the corresponding MMLU trends. Dashed vertical markers indicate average stopping locations when available: for Gittins, these correspond to the Gittins stopping rule; for Bayesian optimization, they correspond to the Bayesian optimization budget or acquisition stopping point. These markers are distinct from the fixed 10% budget used for UCB-E and UCB-E-LRF. In the main figures, the x -axis reports the percentage of the exhaustive evaluation budget consumed in the unit-cost or cost-aware setting, while the y -axis shows raw simple regret. Appendix C.4 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 pre-computed for the common prior variance, noise, batch size, horizon, and cost scale used in each condition. During evaluation, the policy updates posterior means

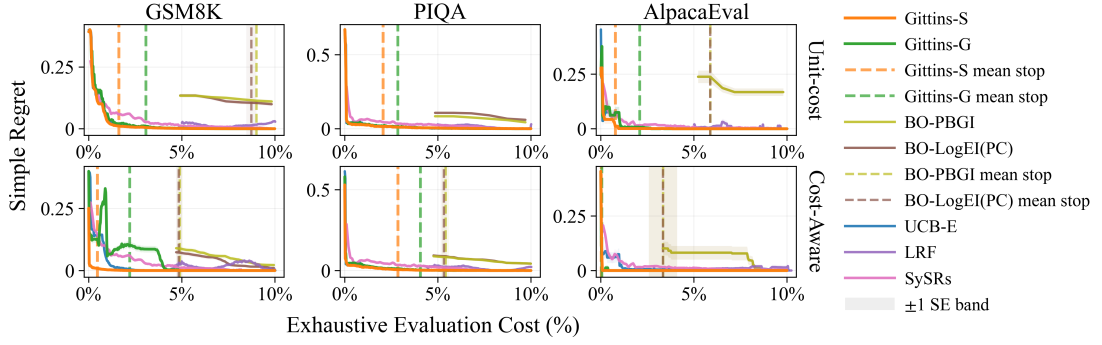


Figure 4.2: Main GSM8K, PIQA, and AlpacaEval results. The top row shows unit-cost evaluation and the bottom row shows cost-aware evaluation; simple regret is plotted against the percentage of the corresponding exhaustive evaluation budget consumed. UCB-E and Gittins use batch size $B = 8$, while LRF uses $B = 32$. Each GSM8K and PIQA panel averages over 100 independent runs (5 ground-truth matrices $\times$ 20 randomized trials); AlpacaEval panels average over the available response matrix and randomized trials. Shaded bands denote ± 1 standard error. Bayesian optimization curves are shown after the 5% configuration-level random-initialization phase. Dashed vertical lines show mean stopping locations when available: for Gittins, when the index-based stopping rule first triggers; for Bayesian optimization, the recorded Bayesian optimization stopping point. These large-example benchmarks show the main advantage over configuration-level Bayesian optimization: Gittins-S enters the low-regret regime early under both cost settings, while Bayesian optimization lags because all-example configuration evaluations leave little room for acquisition-driven improvement under the same budget fraction. Gittins-S also reduces simple regret faster than SySRs and enters the low-regret regime earlier across these panels. Gittins-G is also competitive, though its cost-aware GSM8K curve shows larger early variation before converging.

and variances using the Gaussian approximation in Section 4.3.1. Because online allocation only requires posterior updates and root-table lookup after this offline pre-computation step, the total runtime of the Gittins variants remains close to UCB-E and much smaller than UCB-E-LRF; Figures C.11 and C.12 in Appendix C.4 report runtime comparisons. The Bayesian optimization 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. We implement these baselines in BoTorch using Gaussian processes with categorical kernels, encoding BanditEval

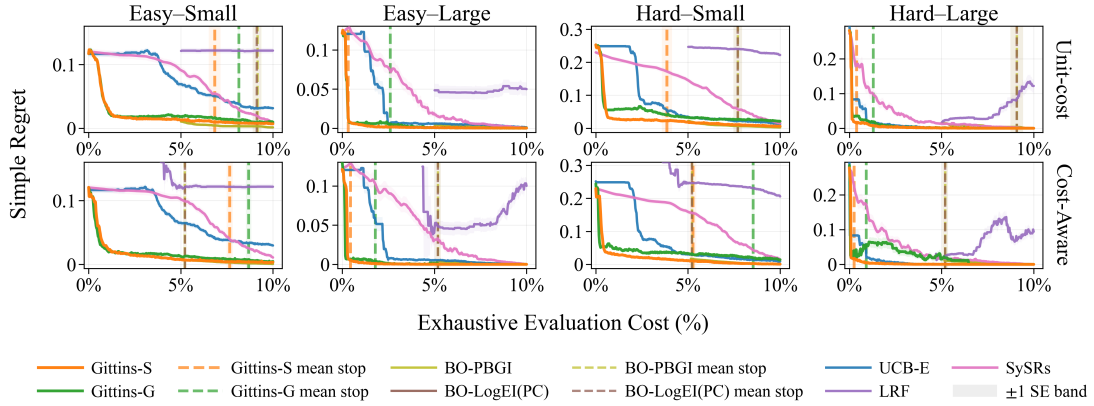


Figure 4.3: Aggregate MMLU results by subject size and difficulty. Columns show the easy-small, easy-large, hard-small, and hard-large buckets; the top row shows unit-cost evaluation and the bottom row shows cost-aware evaluation. Simple regret is plotted against the percentage of the corresponding exhaustive evaluation budget consumed. For UCB-E and Gittins, the small size bucket uses batch size $B = 2$ and the large size bucket uses $B = 8$; LRF uses batch size $B = 32$. Each panel averages over all subjects in the size-difficulty bucket and 20 independent runs per subject. Shaded bands denote ± 1 standard error. Bayesian optimization curves are shown after the 5% configuration-level random-initialization phase. Dashed vertical lines show mean stopping locations when available: for Gittins, when the index-based stopping rule first triggers; for Bayesian optimization, the recorded Bayesian optimization stopping point. These 1500-arm tasks show the main advantage over cost-unaware frequentist bandits: Gittins-G and Gittins-S usually enter the low-regret regime earlier than UCB-E and LRF, especially in the cost-aware and harder buckets. Across both cost settings and all four size-difficulty buckets, Gittins-S also reduces simple regret faster than SySRs and enters the low-regret regime earlier.

configurations for GSM8K and PIQA by four categorical coordinates and DOVE configurations for MMLU by two categorical coordinates. For GSM8K and PIQA, we average over five response matrices, reflecting LLM output randomness, and run 20 randomized algorithm trials on each matrix; for AlpacaEval, we run the same randomized trials on the available response 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; in many settings, the arm recommended at the Gittins stopping point attains near-zero simple regret after only about 1–10% of the exhaustive evaluation cost. On GSM8K, PIQA, and AlpacaEval, each candidate has hundreds or 1000 benchmark examples, so the main separation is between Gittins and configuration-level Bayesian optimization. Bayesian optimization is less competitive because each configuration-level 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 Bayesian optimization budget, leaving relatively few acquisition-driven updates. This effect is especially visible on AlpacaEval, where the Bayesian optimization curves remain at noticeably higher regret than Gittins over much of the budget. Gittins can instead allocate partial batches across configurations and reach low regret early under both unit-cost and cost-aware evaluation.

The MMLU aggregate results show the large-candidate regime. Each MMLU subject has 1500 model–template arms, so the policy must search over a much larger candidate set than in GSM8K or PIQA. Here the main comparison changes: Gittins, especially Gittins-S, is markedly stronger than the cost-unaware frequentist bandit baselines UCB-E and UCB-E-LRF, which spread a fixed budget across many arms without using priors or cost-normalized index values. Notably, UCB-E-LRF has substantially higher simple regret than standard UCB-E across the MMLU aggregates. This agrees with the broader empirical finding of Lyu et al. (2026) that UCB-E-LRF usually underperforms UCB-E. Bayesian optimization remains competitive on small and easy tasks, where all-example configuration evaluations are relatively cheap. These results suggest that when the number of available examples per configuration is small, configuration-level Bayesian optimization may

be preferable because the low cost of a full configuration evaluation reduces the benefit of fine-grained bandit allocation. Across the reported size–difficulty buckets, however, the Gittins variants typically reach low regret earlier.

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 Bayesian optimization 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 after using around 1% of the resources required for full response-matrix evaluation, 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 task difficulties.

4.5 Conclusion

We formulated LLM evaluation as cost-aware Bayesian bandit evaluation and implemented it with a practical Gittins index policy. The empirical takeaway is simple: Gittins outperforms configuration-level Bayesian optimization on large-example benchmarks such as GSM8K, PIQA, and AlpacaEval, and markedly outperforms cost-unaware frequentist bandit methods when the candidate set is large, as in MMLU. When the number of available examples per configuration is small, however, configuration-level Bayesian optimization may be preferable because a full configuration evaluation is relatively inexpensive, reducing the benefit of fine-grained bandit allocation. After index precomputation, the policy is lightweight, handles heterogeneous evaluation costs, and provides a stopping signal once regret has flattened.

CHAPTER 5

BAYESIAN-OPTIMAL DECISION-MAKING UNDER COST-AWARE MULTI-STAGE FEEDBACK VIA THE GITTINS INDEX

This chapter is based on ongoing joint work with Theodore Brown, Ziv Scully, and Alexander Terenin.

Abstract. We study a class of bandit-like decision problems, under a Bayesian formulation, where information about the optimal action is revealed gradually over multiple stages, with each stage incurring a cost. In our setting, one can characterize the Bayesian-optimal algorithm, and show that it makes decisions by optimizing the Gittins index associated with each action. However, the Gittins index is only known to be computable for a very limited class of models, such as finite-state Markov chains, which has limited its practical use. In this work, we develop practical numerical algorithms for Gittins-index-based decision-making, by giving an efficient algorithm for computing Gittins indices of Gaussian state-space models—a broad class of multi-stage continuous-state models suitable for a large set of applications. This is achieved by using Gaussianity to reduce the n -dimensional case to the one-dimensional case, which we then solve efficiently using a dynamic program for the associated optimal stopping problem, where each step involves a fast Fourier transform. We evaluate the resulting Gittins index algorithm through synthetic Bayesian-regret experiments in the model-matched setting, isolating the regime where the modeling assumptions and optimality guarantee apply.

5.1 Introduction

Many real-world problems of decision-making under uncertainty involve situations where information about an action’s value is revealed gradually over a series of stages, each of which involves paying costs. In such problems, the key explore-exploit tradeoff is how to balancing the value of knowing how well an action will ultimately perform against the costs of obtaining this information.

These decision-making problems involving multi-stage feedback appear frequently in scientific research, where evaluating candidates can involve running expensive computational models at scale, long-running clinical trials, or costly laboratory experiments. A particularly challenging example is in nuclear fusion, where simulating a plasma requires solving coupled non-linear partial differential equations (PDEs) such as the 5D gyrokinetics equations. As plasma turbulence is high dimensional, multi-scale, and extremely sensitive to gradients, these simulations take tens or hundreds of thousands of core-hours on exascale compute. Given the extreme cost, it is important to efficiently identify the most promising candidates for fusion scenario optimization.

Similar multi-stage decision-making problems also arise in automated machine learning, especially hyperparameter tuning, where model performance is revealed progressively as training proceeds and evaluations can be paused or resumed based on partially-observed training curves. This setting has been studied in freeze-thaw Bayesian optimization (Swersky et al., 2014) and bandit-like budgeted resource allocation formulations such as Hyperband (Li et al., 2018). Notably, the cost of obtaining additional observations of model performance, measured in training time or computational resources, can be varying across sets of hyperparameters, but such cost heterogeneity is not explicitly incorporated in these frameworks.

In this work, we focus on the setting where uncertainty about each action’s value is quantified via a probabilistic model, and different actions’ values are independent from one another. Given such a model, the decision on which action to evaluate next is captured by a Markov decision process (MDP) with multiple Markov chains to proceed with, whose optimal algorithm is called a *Bayesian-optimal* algorithm. Following the seminal work of Gittins (1979b), for certain classes of such MDPs, the Bayesian-optimal algorithm takes the form of ranking each action by a number, called its *Gittins index*, which is defined as the solution of a certain optimization problem.

Despite the natural appeal of making Bayesian-optimal decisions, the vast majority of research on Gittins indices has focused on settings which are simple-enough that their respective optimization problems can be solved analytically—such as Pandora’s Box problem with Gaussian reward distributions (Xie et al., 2024), or M/G/1 queue problems (Scully et al., 2020). In general, complex Markov decision processes can yield Gittins indices whose computation is intractable. For example, recent work on chain-of-thought selection for large language models formulates the problem as a Markov chain selection problem, but notes that implementing the Gittins index algorithm for this formulation is challenging, leading them to consider Gittins-inspired policies based on simplified formulations Zabounidis et al. (2025).

At present, settings where Gittins indices are non-analytic—but can be reliably computed numerically—have largely yet to be identified. Our aim is to identify and develop one such class—which is rich enough to capture the structure of our motivating problems—and evaluate the resulting policies in model-matched synthetic settings where the assumptions can be controlled. Following the work of Xie et al. (2024) connecting Gittins index theory with Bayesian optimization, we

view our work as the next significant step towards integrating ideas from Gittins index theory into machine learning.

Contributions. In this work, we (i) define a suitable class of Markov decision processes—based on selecting actions whose values evolve according to Gaussian state-space models—which simultaneously capture a broad class of multi-stage decision-making problems of practical interest, yet admit an explicit optimal algorithm via a Gittins Index Theorem. We (ii) show the resulting Gittins index algorithm can be expressed as a dynamic program, whose steps consist of one-dimensional root-finding problems with certain additional structure. We (iii) develop an efficient fast Fourier transform based method for solving these root-finding problems, thereby providing the first efficient numerical method for computing the Gittins index algorithm of this class. We (iv) use the resulting theory to design and implement a policy class for cost-aware multi-stage decision-making, and evaluate it on synthetic model-matched Bayesian-regret tasks.

5.2 Bayesian-optimal Decision-making via Gittins Indices

We are interested in Bayesian-optimal decision-making for the *Markov chain selection* problem—a general Markov decision problem class. Before introducing this in full generality, we will begin by first describing the simplest concrete example, namely a sequential decision-making problem called *Pandora’s Box*, which we now proceed to.

5.2.1 A Review of Pandora's Box

In Pandora's Box, a decision-making agent is presented with a set of N closed boxes, indexed by $n = 1, \dots, N$. Each box has an associated random reward $r_n \in \mathbb{R}$ with distribution $p_n \in \mathcal{M}_1(\mathbb{R})$, and deterministic cost $c_n > 0$. At each iteration $t = 1, \dots, T$ (T can be a flexible stopping time), the agent is allowed to either:

1. *Open a closed box*: letting n_t be the chosen box, the agent incurs a cost of c_{n_t} , but learns the exact value $r_{n_t} \sim p_{n_t}$. After this, box n_t is considered *open*.
2. *Select an open box*: in this case, the decision problem ends, and the agent receives the reward r_{n_T} inside the box.

Obviously, if the agent selects an open box, they should always select the best open box. Note that the agent pays the cost to open *each* box they choose, but only receives a reward from *one* of the boxes they have opened. Finally, we assume that the agent is not allowed to select boxes that have not yet been opened, and that all boxes' reward distributions are independent. The agent's total reward is given by

$$\mathbb{E} \max_{t=1, \dots, T} r_{n_t} - \mathbb{E} \sum_{t=1}^T c_{n_t}. \quad (5.1)$$

Remarkably, it is possible to analytically characterize the Bayesian-optimal algorithm for Pandora's Box, due to a result of Weitzman (1978). It says that selecting n_t according to

$$n_t = \arg \max_{n=1, \dots, N} \alpha^*(n) \quad (5.2)$$

where $\alpha^*(n) = r_n$ for an open box, and

$$\alpha^*(n) = \inf\{g \in \mathbb{R} : \mathbb{E} \max(0, r_n - g) = c_n\} \quad (5.3)$$

for a closed box, is Bayesian-optimal. Note that this infimum yields the solution of a *root-finding problem* with respect to the variable g , with the convention that if there are multiple roots we select the smallest one.

The interpretation is as follows: for each closed box n , $\alpha^*(n)$ computes the value of an *equivalent open box*. The optimal algorithm, then, is to rank all boxes according to the *fair value* given by their respective equivalent open boxes, open them in that order, and select a box once its revealed value is higher than that of any remaining closed boxes.

5.2.2 From Pandora’s Box to Markov Chain Selection

Markov Chain Selection can be viewed as a far-reaching generalization of Pandora’s Box, which unifies it with Bayesian bandits and other decision problems. Conceptually, we begin by viewing each individual Pandora’s Box as a transient *Markov chain* defined over the state space $\{\boxtimes\} \cup \mathbb{R} \cup \{\checkmark\}$. Among these states, the symbol $\boxtimes$ represents a closed box, $r \in \mathbb{R}$ represents an open box with reward r , and $\checkmark$ represents the box selected at the end of the decision problem. The transitions of this Markov chain describe what happens to a given box each time it is chosen by the agent.

The key idea behind the general Markov chain selection problem is to replace the specific kind of Markov chain described above with a collection of independent Markov chains satisfying the standard admissibility conditions for Markov chain selection. In the undiscounted formulation used here, each chain has designated terminal states, reaches a terminal state in finite local time almost surely from every admissible initial state, and has finite expected cumulative absolute reward before termination (Scully and Terenin, 2025, Assumption 3.3). Thus, “general”

refers to the dynamics and state spaces allowed within this class, rather than to literally unrestricted Markov chains. For the finite-stage models studied below, the states at local stage T are terminal, and the required integrability follows from the assumed finite reward moments. We allow each chain to be time-heterogeneous, where time means the chain’s own local stage and advances only when that chain is selected. This is compatible with the time-homogeneous formulation of the Gittins index theorem: a chain with stage-dependent state spaces X_t and transition kernels p_t can be represented by a time-homogeneous chain whose states are pairs (t, x) , where $t \in \{0, \dots, T\}$ and $x \in X_t$, with transition $(t, x) \mapsto (t + 1, x')$ for $x' \sim p_t(\cdot \mid x)$. Thus, stage is part of the Markov state whenever we invoke a time-homogeneous result. At local stage τ , chain n is equipped with a *reward function* $r_{n,\tau} : X_{n,\tau} \rightarrow \mathbb{R}_{\geq 0}$ and a *cost function* $c_{n,\tau} : X_{n,\tau} \rightarrow \mathbb{R}_{\geq 0}$, both of which are non-negative.¹ At each global iteration $k = 1, 2, \dots$, the agent chooses one of the N chains $n_k \in \{1, \dots, N\}$. The chosen chain transitions from its current stage-state (τ_n, x_n) to $(\tau_n + 1, x'_n)$, while every other chain remains in its previous stage-state. The goal, now, is to maximize

$$\mathbb{E} \sum_{k=1}^T (r_{n_k, \tau_{n_k}}(x_{n_k}) - c_{n_k, \tau_{n_k}}(x_{n_k})). \quad (5.4)$$

The remarkable fact about Markov chain selection is that—just as in Pandora’s Box—the Bayesian-optimal algorithm retains a surprisingly simple form.

Result 11. *There exist local value functions $V_{n,\tau,\text{loc}}^*$ such that the algorithm selecting the chain with the largest current Gittins index, $n_k \in \arg \max_n \alpha^*(n; \tau_n, x_n)$, where*

$$\alpha^*(n; \tau_n, x_n) = \inf\{g \in \mathbb{R} : V_{n,\tau_n,\text{loc}}^*(x_n; g) = g\}, \quad (5.5)$$

is Bayesian-optimal.

¹This is equivalent to the formulation of Scully and Terenin (2025), by splitting their signed reward function into its positive and negative parts.

A proof is given in Scully and Terenin (2025, Theorem 3.7), and additional details on notation are given in Appendix D.1. This result is central to our work: our primary goal is to understand how to compute α^* , and therefore the Bayesian-optimal action, numerically. The key challenge will be to understand how to compute $V_{n,\tau,\text{loc}}^*$ —which, beyond Pandora’s Box, almost never has a simple analytic form—so that we can solve the defining root-finding problem. Since our development will focus on detailed properties of $V_{n,\tau,\text{loc}}^*$, we will return to it and describe its precise definition in the sequel.

For finite-state Markov chains, general methods for computing α^* have been developed by Katta and Sethuraman (2004). At present, settings which involve infinite-state Markov chains—which are already present in Pandora’s Box, and various Markovian queues—involve case-by-case techniques which rely on problem-specific structure; for example, Xie et al. (2024) considers Pandora’s Box with Gaussian distributions.

5.2.3 Linear Gaussian State-space Models

In this work, we consider Markov chain selection between a set of chains given by linear Gaussian state-space models (LGSSMs). LGSSM is a flexible model class that is used for time series analysis in a wide variety of domains including finance (Harvey, 1990), biology (Linderman et al., 2016) and robotics (Thrun, 2002). For simplicity, we state our results initially for fully-observed LGSSMs, and extend to the general case with hidden states later.

Definition 12 (Time-heterogeneous linear Gaussian state-space model). *A linear Gaussian state-space model $LGSSM(\{\mathbf{F}_t, \mathbf{Q}_t\}_{t=0}^{T-1})$ generates a finite sequence*

$(\mathbf{x}_t)_{t=0}^T$ according to

$$\mathbf{x}_{t+1} = \mathbf{F}_t \mathbf{x}_t + \boldsymbol{\varepsilon}_{t+1}, \quad \boldsymbol{\varepsilon}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t), \quad (5.6)$$

where the random vectors $\boldsymbol{\varepsilon}_{t+1}$ are mutually independent. They are called the *INNOVATIONS*, or *process noises*, because $\boldsymbol{\varepsilon}_{t+1}$ is the new random information added to the state when the model advances from stage t to stage $t + 1$. Here, $\mathbf{x}_t \in \mathbb{R}^{d_t}$, $\mathbf{F}_t \in \mathbb{R}^{d_{t+1} \times d_t}$, and $\mathbf{Q}_t \in \mathbb{R}^{d_{t+1} \times d_{t+1}}$. The model is *TIME-HOMOGENEOUS* when $d_t = d$, $\mathbf{F}_t = \mathbf{F}$, and $\mathbf{Q}_t = \mathbf{Q}$ for all t .

The time-heterogeneous notation is needed even when the latent dynamics are time-homogeneous: the Kalman-filter belief process in Section 5.3.4 generally has a stage-dependent innovation covariance. As above, augmenting the state with its local stage converts this model into a time-homogeneous Markov chain without changing its value functions or Gittins indices.

5.3 Gittins Indices for Gaussian State Space Models

The Markov chain selection problem, introduced in Section 5.2, provides a general framework for performing Bayesian-optimal decision-making. At optimality, the expected reward from selecting chain n_t is exactly balanced against its cost c_{n_t} , where the expectation is taken over future state transitions induced by that selection. By choosing appropriate reward functions, cost functions, and transition distributions, one can place a wide range of problems into this framework—including advanced variants of Pandora’s box which incorporate nested boxes and other complications, various optimal scheduling and queueing problems, and other bandit-like decision problems.

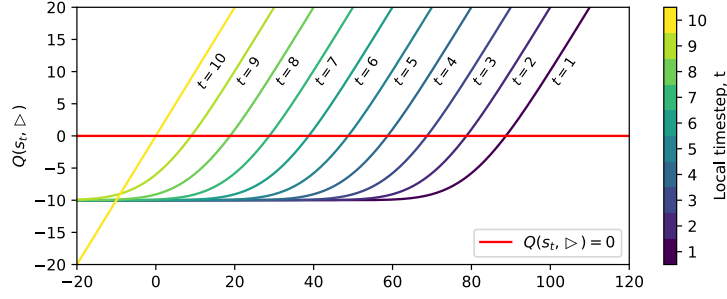


Figure 5.1: Backward solution of the dynamic program. At the terminal stage, $Q_{T,\text{loc}}^*(x) = x$; at each earlier stage, the next-stage Q -function is thresholded at zero, convolved with the stage-specific Gaussian increment density, and shifted by the evaluation cost.

Following Result 11, optimal actions for Markov chain selection can be obtained by computing each chain’s Gittins index at its current local stage-state. The fundamental question therefore becomes: *how can we reliably compute this quantity using numerical methods?* We proceed to study this question, focusing on Gaussian state-space models as an initial non-trivial setting to start with.

5.3.1 Reduction to One-Dimensional Gaussian Random Walks

We now begin our technical development, and assume that we are given a Markov chain selection problem where each Markov chain is given by a Gaussian state-space model. To proceed, following Scully and Terenin (2025), we first identify the function $V_{n,\text{loc}}^*$ introduced in Section 5.2.

Definition 13. *Fix a time-heterogeneous Markov chain with state spaces X_t , transition kernels p_t , reward functions r_t , and cost functions c_t , and let $\alpha \in \mathbb{R}$. Define the LOCAL MARKOV DECISION PROCESS as follows:*

1. *States:* $X_{\text{loc}} = \{(t, x) : t \in \{0, \dots, T\}, x \in X_t\} \cup \{\checkmark\}$, where $\checkmark$ is the unique

terminal state.

2. *Actions:* $A_{\text{loc}} = \{\triangleright, \square\}$, called *GO* and *STOP*.
3. *Rewards:* $r_{\text{loc}}((t, x), \square) = \alpha$ and $r_{\text{loc}}((t, x), \triangleright) = r_t(x) - c_t(x)$.
4. *Transition kernel:* from (t, x) with $t < T$ under $\triangleright$, the next state is $(t + 1, x')$ with $x' \sim p_t(\cdot \mid x)$; from (T, x) under $\triangleright$, from $\checkmark$, or under $\square$, the next state is $\checkmark$.

With this notation, $V_{n,t,\text{loc}}^*(x_n; \alpha)$ is the optimal value function of the local MDP for the n th chain at local stage t and outside option α . To solve the root-finding problem from Result 11 for the optimal algorithm, we need to solve the Bellman equation of the local MDP for $V_{n,t,\text{loc}}^*(x_n; \cdot)$. At first glance, this appears to be difficult even for Gaussian state-space models, as the MDP is defined over a d -dimensional space.

We will show that the above problem can be reduced analytically to an equivalent problem in *one* dimension. To do so, we need a definition from Markov chain theory. Let (X_t, p_t, r_t, c_t) and $(\bar{X}_t, \bar{p}_t, \bar{r}_t, \bar{c}_t)$ be two time-heterogeneous Markov chains with the same horizon. We say that a family of measurable functions $f_t : X_t \rightarrow \bar{X}_t$ is a *lumping map* (Buchholz, 1994) if

$$p_t(f_{t+1}^{-1}(A) \mid x) = \bar{p}_t(A \mid f_t(x)) \quad (5.7)$$

for every $t < T$, $x \in X_t$, and measurable $A \subseteq \bar{X}_{t+1}$, together with $r_t(x) = \bar{r}_t(f_t(x))$ and $c_t(x) = \bar{c}_t(f_t(x))$.

While appearing technical, this definition encodes a simple idea: one should think of (X_t, p_t, r_t, c_t) as the *larger chain*, and $(\bar{X}_t, \bar{p}_t, \bar{r}_t, \bar{c}_t)$ as the *smaller chain*. With this intuition, f_t is akin to a *sufficient statistic* which lumps together states

that yield the same costs, rewards, and reduced-state transition law. We now state the key result.

Theorem 14 (Value preservation under lumping). *Let (X_t, p_t, r_t, c_t) and $(\bar{X}_t, \bar{p}_t, \bar{r}_t, \bar{c}_t)$ be two time-heterogeneous Markov chains related by a lumping map $(f_t)_{t=0}^T$, and let $V_{t,\text{loc}}^*$ and $\bar{V}_{t,\text{loc}}^*$ be their respective local value functions. Then*

$$V_{t,\text{loc}}^*(x; \alpha) = \bar{V}_{t,\text{loc}}^*(f_t(x); \alpha) \quad (5.8)$$

for every t , $x \in X_t$, and $\alpha \in \mathbb{R}$.

Theorem 14 is proved in Appendix D.1. Because the Gittins index is defined from the local value function, lumping maps preserve Gittins indices as well. For the LGSSMs of interest, the appropriate scalar state is determined by the terminal quantity being optimized.

Definition 15 (Terminal-value signal map). *Fix a terminal stage T and a terminal linear quantity $Z = \ell^\top \mathbf{x}_T$. For $0 \leq t \leq T$, define*

$$\Phi_{t,T} := \begin{cases} \mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_t, & t < T, \\ \mathbf{I}, & t = T, \end{cases} \quad f_t(\mathbf{x}) := \ell^\top \Phi_{t,T} \mathbf{x}, \quad (5.9)$$

and call $s_t := f_t(\mathbf{x}_t)$ the *TERMINAL-VALUE SIGNAL*. Thus $s_t = \mathbb{E}[Z \mid \mathbf{x}_t]$ in the fully observed model. For a time-homogeneous LGSSM, this reduces to $s_t = \ell^\top \mathbf{F}^{T-t} \mathbf{x}_t$.

The terminal reward is therefore a function of $s_T = Z = \ell^\top \mathbf{x}_T$, not of $\ell^\top \mathbf{x}_t$ at an intermediate stage. More generally, to apply lumping, we assume that any stage- t reward and cost factor through the terminal-value signal: for some scalar functions $\bar{r}_t, \bar{c}_t$,

$$r_t(\mathbf{x}) = \bar{r}_t(f_t(\mathbf{x})), \quad c_t(\mathbf{x}) = \bar{c}_t(f_t(\mathbf{x})). \quad (5.10)$$

This includes the principal case in which intermediate stages incur stage-dependent evaluation costs and the only positive reward is the terminal value Z .

Theorem 16 (Terminal-value signal reduction). *Consider the fully observed LGSSM in Definition 12, and suppose its rewards and costs factor through the terminal-value signal maps in Definition 15. Then $(f_t)_{t=0}^T$ is a lumping map, and the reduced process is the time-heterogeneous Gaussian random walk*

$$s_{t+1} = s_t + \zeta_{t+1}, \quad \zeta_{t+1} \sim \mathcal{N}(0, \sigma_{t+1}^2), \quad (5.11)$$

where the increments are independent and

$$\sigma_{t+1}^2 = \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t+1,T} \mathbf{Q}_t \boldsymbol{\Phi}_{t+1,T}^\top \boldsymbol{\ell}. \quad (5.12)$$

Consequently, the original LGSSM and the scalar random walk have identical local value functions and Gittins indices at corresponding states.

Theorem 16 is proved in Appendix D.1. The partially observed case is handled separately in Section 5.3.4: first the complete observation history is lumped to the current Kalman belief, and then the terminal-value signal map is applied to that belief.

As a consequence, to compute the Gittins index of a general Gaussian state-space model, it suffices to understand the one-dimensional case, thereby avoiding the usual curse of dimensionality associated with dynamic programming. We therefore proceed to study Gaussian random walks.

5.3.2 Local Value Functions of Gaussian Random Walks

We now work with the scalar, time-heterogeneous Gaussian random walk obtained above. We specialize to the multi-stage evaluation setting used in the sequel:

continuing at a nonterminal stage $t < T$ incurs a signal-independent cost $c_t \geq 0$, and continuing at stage T receives terminal reward x . Let $Q_{t,\text{loc}}^*(x, a; \alpha)$ and $V_{t,\text{loc}}^*(x; \alpha)$ denote the stage- t action-value and value functions with outside option α . They satisfy

$$Q_{t,\text{loc}}^*(x, \triangleright; \alpha) = \begin{cases} -c_t + \mathbb{E}[V_{t+1,\text{loc}}^*(x + \zeta_{t+1}; \alpha)], & t < T, \\ x, & t = T, \end{cases} \quad (5.13)$$

$$Q_{t,\text{loc}}^*(x, \sqcap; \alpha) = \alpha, \quad (5.14)$$

where $\zeta_{t+1} \sim \mathcal{N}(0, \sigma_{t+1}^2)$. To ease notation, write $Q_{t,\text{loc}}^*(x; \alpha) := Q_{t,\text{loc}}^*(x, \triangleright; \alpha)$. Note that the optimal value function satisfies

$$V_{t,\text{loc}}^*(x; \alpha) = \max\{Q_{t,\text{loc}}^*(x, \triangleright; \alpha), Q_{t,\text{loc}}^*(x, \sqcap; \alpha)\} \quad (5.15)$$

$$= \max\{Q_{t,\text{loc}}^*(x; \alpha), \alpha\}. \quad (5.16)$$

At first glance, computing $Q_{t,\text{loc}}^*(x; \alpha)$ appears to require solving a two-parameter dynamic program over (x, α) at every stage. The random-walk identity $x_{t+1} = x_t + \zeta_{t+1}$ is exactly what makes the following reduction possible.

Lemma 17 (Translational equivariance of the Q-function). *For every stage t and $\beta \in \mathbb{R}$,*

$$Q_{t,\text{loc}}^*(x; \alpha) + \beta = Q_{t,\text{loc}}^*(x + \beta; \alpha + \beta). \quad (5.17)$$

Corollary 18.

$$Q_{t,\text{loc}}^*(x; \alpha) = Q_{t,\text{loc}}^*(x - \alpha; 0) + \alpha. \quad (5.18)$$

Figure 5.2 illustrates this translational equivariance. By the corollary, it suffices at each stage to compute the single-argument function $Q_{t,\text{loc}}^*(x) := Q_{t,\text{loc}}^*(x; 0)$. We have therefore eliminated the outside-option dimension from the dynamic program.

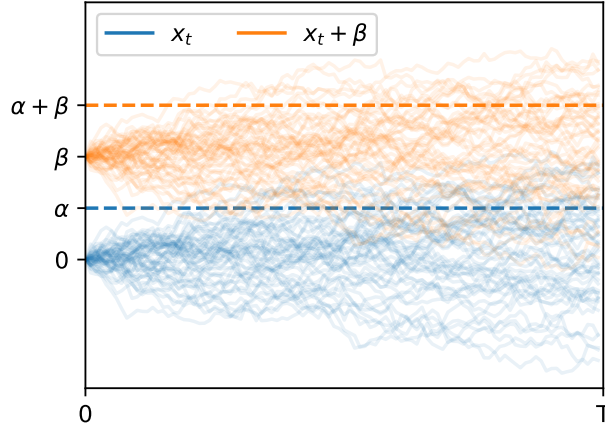


Figure 5.2: Translational equivariance of $Q_{t,\text{loc}}^*$: shifting the state and threshold by β shifts the Q -function by β .

At this stage, the main remaining challenge is that $Q_{t,\text{loc}}^*$ is defined over an unbounded domain: to handle this, we prove the following structural properties.

Proposition 19. *For each $t < T$, the function $Q_{t,\text{loc}}^* : \mathbb{R} \rightarrow \mathbb{R}$ is monotone and convex. Moreover,*

$$\lim_{x \rightarrow -\infty} Q_{t,\text{loc}}^*(x) = -c_t, \quad \lim_{x \rightarrow \infty} Q_{t,\text{loc}}^*(x) = \infty, \quad (5.19)$$

with $\lim_{x \rightarrow \infty} \frac{d}{dx} Q_{t,\text{loc}}^*(x) = 1$. At the terminal stage, $Q_{T,\text{loc}}^*(x) = x$.

The key idea here is that each limit corresponds to an algorithm. If x is very small, then the optimal action at the next time step is overwhelmingly likely to be $\square$, so continuing now has value approaching the immediate cost $-c_t$. On the other hand, if x is very large, then it is likely one should play $\triangleright$ indefinitely, resulting in a value equal to approximately x in expectation. Other values of x lie in-between these extreme cases.

5.3.3 Efficient Approximate Dynamic Programming via Fast Fourier Transforms

Given the structural properties from the previous section, we can now develop a numerical method for computing the functions $Q_{t,\text{loc}}^*(\cdot)$ backward in t , from which we recover the full value functions and compute Gittins indices by root finding.

With outside option zero, the recursion from Equations (5.14) and (5.16) is

$$Q_{T,\text{loc}}^*(x) = x, \quad (5.20)$$

$$Q_{t,\text{loc}}^*(x) = -c_t + \mathbb{E}[\max\{Q_{t+1,\text{loc}}^*(x + \zeta_{t+1}), 0\}], \quad t < T. \quad (5.21)$$

Writing ϕ_{t+1} for the density of $\mathcal{N}(0, \sigma_{t+1}^2)$ and $Q_{t+1,\text{loc}}^+ := \max\{Q_{t+1,\text{loc}}^*, 0\}$, the nonterminal update is

$$Q_{t,\text{loc}}^*(x) = -c_t + \int_{-\infty}^{\infty} Q_{t+1,\text{loc}}^+(s) \phi_{t+1}(x - s) ds \quad (5.22)$$

$$= -c_t + (Q_{t+1,\text{loc}}^+ * \phi_{t+1})(x). \quad (5.23)$$

The behavior of this recursion is shown in Fig. 5.1.

To solve this backward recursion approximately, we replace $Q_{t+1,\text{loc}}^+$ with a finite-basis approximation, perform the convolution, and approximate the result within the same basis. Using Lemma 17 and Proposition 19, we note that $Q_{t+1,\text{loc}}^+ : \mathbb{R} \rightarrow \mathbb{R}$ can be well-approximated by a finite-dimensional function class of convex monotone functions with the correct limiting behavior. The simplest choice satisfying these requirements is a piecewise linear function class. We denote by $\widehat{Q}_{t+1,\text{loc}}^+$ the approximation of $Q_{t+1,\text{loc}}^+$ within this class, which can be defined by a grid of evenly-spaced points ξ_i ($i = 0, \dots, P - 1$) with space δ and a set of slopes $m_1, \dots, m_P$:

$$\widehat{Q}_{t+1,\text{loc}}^+(x) = \widehat{Q}_{t+1,\text{loc}}^+(\xi_0) + m_1(x - \xi_0) + \sum_{i=1}^{P-1} (m_{i+1} - m_i) \max(x - \xi_i, 0). \quad (5.24)$$

With this parameterization, the convolution in the recursion Eq. (5.23) admits a simple form.

Proposition 20. *Define the EI kernel $\text{EI}_{t+1}(z) := \mathbb{E}_{\xi \sim \mathcal{N}(0, \sigma_{t+1}^2)} \max(z - \xi, 0)$. Then for each grid point $x = \xi_j$ ($j = 0, \dots, P - 1$), we have*

$$\hat{Q}_{t+1, \text{loc}}^+ * \phi_{t+1}(x) = \hat{Q}_{t+1, \text{loc}}^+(\xi_0) + m_1(x - \xi_0) + \sum_{i=1}^{P-1} (m_{i+1} - m_i) \text{EI}_{t+1}(x - \xi_i). \quad (5.25)$$

We call this kernel EI kernel based on its relationship with the expected improvement acquisition function in Bayesian optimization (Jones et al., 1998; Moćkus, 1974).

Note the indices of the summation: it is a discrete convolution, which can be computed efficiently using a fast Fourier transform (FFT), resulting in an overall computational cost of $\mathcal{O}(PT \log P)$, where T is the time horizon. See Appendix D.2 for details of the FFT formulation.

5.3.4 Extension to LGSSMs with hidden states

A more flexible variant of an LGSSM is constructed by only making a linear combination of the underlying states available to an observer. We call this a *partially observed* LGSSM.

Definition 21 (Partially observed linear Gaussian state-space model). *A partially observed LGSSM augments the latent dynamics in Definition 12 with*

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{x}_t + \boldsymbol{\eta}_t, \quad \boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t), \quad (5.26)$$

where the observation noises are mutually independent and are also independent of the state innovations $\boldsymbol{\varepsilon}_t$. Here, $\mathbf{y}_t \in \mathbb{R}^{m_t}$, $\mathbf{H}_t \in \mathbb{R}^{m_t \times d_t}$, and $\mathbf{R}_t \in \mathbb{R}^{m_t \times m_t}$.

This additional structure greatly increases the expressivity of the model, enabling the representation of any Gaussian process within the LGSSM framework (Hartikainen and Särkkä, 2010; Sarkka et al., 2013). Unfortunately $(\mathbf{y}_t)$ is not a Markov chain, which means we cannot directly apply the Gittins algorithm for MCS to this model class. However, the Kalman filter can be used to construct a fully observed, time-heterogeneous LGSSM from a partially observed one.

Result 22 (Kalman filtering of LGSSMs). *Let $\boldsymbol{\mu}_{t|t} := \mathbb{E}[\mathbf{x}_t \mid \mathbf{y}_{1:t}]$ and $\boldsymbol{\Sigma}_{t|t} := \text{Cov}(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, starting from a Gaussian prior $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_{0|0}, \boldsymbol{\Sigma}_{0|0})$. The deterministic covariance recursion is*

$$\boldsymbol{\Sigma}_{t|t-1} = \mathbf{F}_{t-1} \boldsymbol{\Sigma}_{t-1|t-1} \mathbf{F}_{t-1}^\top + \mathbf{Q}_{t-1}, \quad (5.27)$$

$$\mathbf{S}_t = \mathbf{H}_t \boldsymbol{\Sigma}_{t|t-1} \mathbf{H}_t^\top + \mathbf{R}_t, \quad (5.28)$$

$$\mathbf{K}_t = \boldsymbol{\Sigma}_{t|t-1} \mathbf{H}_t^\top \mathbf{S}_t^{-1}, \quad (5.29)$$

$$\boldsymbol{\Sigma}_{t|t} = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \boldsymbol{\Sigma}_{t|t-1}. \quad (5.30)$$

Moreover,

$$\boldsymbol{\mu}_{t|t} \mid \boldsymbol{\mu}_{t-1|t-1} \sim \mathcal{N}(\mathbf{F}_{t-1} \boldsymbol{\mu}_{t-1|t-1}, \mathbf{K}_t \mathbf{S}_t \mathbf{K}_t^\top). \quad (5.31)$$

Thus $(\boldsymbol{\mu}_{t|t})$ is a fully observed, time-heterogeneous LGSSM with transition matrices $\mathbf{F}_{t-1}$ and innovation covariances $\mathbf{K}_t \mathbf{S}_t \mathbf{K}_t^\top$.

These standard Kalman filtering identities follow, for example, from West and Harrison (1997, Chapter 4); see also Welch, Bishop, et al. (1995). No new proof of the filtering recursion is required here.

To connect this result precisely to lumping, let $\mathcal{I}_t = (\boldsymbol{\mu}_{0|0}, \dots, \boldsymbol{\mu}_{t|t}, \mathbf{y}_1, \dots, \mathbf{y}_t)$ denote the complete information history. The projection

$$\mathcal{I}_t \longmapsto \boldsymbol{\mu}_{t|t} \quad (5.32)$$

is a lumping map whenever rewards and costs depend on the history only through the current belief. Indeed, the displayed transition law shows that the conditional distribution of the next belief depends on $\mathcal{I}_t$ only through $\boldsymbol{\mu}_{t|t}$. Hence Theorem 14 implies that retaining the full observation history and retaining only the current Kalman belief give the same local value function and Gittins index.

For the terminal quantity $Z = \boldsymbol{\ell}^\top \mathbf{x}_T$, applying the terminal-value signal map f_t from Definition 15 to the current belief gives the second mapping

$$\boldsymbol{\mu}_{t|t} \xrightarrow{f_t} \hat{s}_t := f_t(\boldsymbol{\mu}_{t|t}) = \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t,T} \boldsymbol{\mu}_{t|t} = \mathbb{E}[Z \mid \mathbf{y}_{1:t}]. \quad (5.33)$$

Applying Theorem 16 to the belief LGSSM shows that $(\hat{s}_t)$ is specifically the time-heterogeneous Gaussian random walk

$$\hat{s}_{t+1} = \hat{s}_t + \hat{\zeta}_{t+1}, \quad \text{Var}(\hat{\zeta}_{t+1}) = \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t+1,T} \mathbf{K}_{t+1} \mathbf{S}_{t+1} \mathbf{K}_{t+1}^\top \boldsymbol{\Phi}_{t+1,T}^\top \boldsymbol{\ell}. \quad (5.34)$$

In summary, the two-step reduction is

$$\mathcal{I}_t \longmapsto \boldsymbol{\mu}_{t|t} \xrightarrow{f_t} \hat{s}_t. \quad (5.35)$$

It extends the one-dimensional Gittins computation to partially observed LGSSMs while keeping the observation history separate from the terminal-value signal map.

5.4 Synthetic Bayesian Regret Experiments

We evaluate the Gittins index policy in a model-matched synthetic setting, where the data-generating process is the same Gaussian state-space model family used by the policy. This isolates the regime in which the Bayesian-optimality guarantee is intended to apply, without adding complications from empirical data. We compare against random search and successive halving baselines, and sweep the Gittins

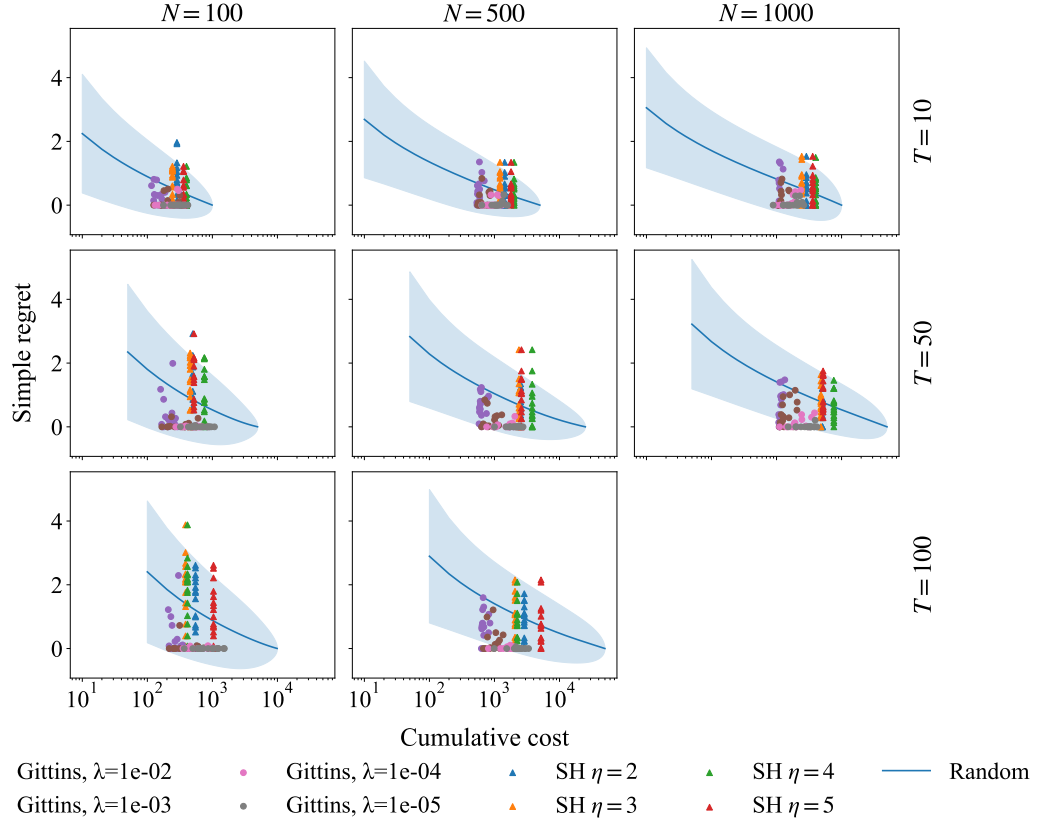


Figure 5.3: Synthetic model-matched Bayesian-regret experiments. Each panel shows the tradeoff between evaluation cost and Bayesian regret for a fixed number of arms and horizon. Columns correspond to horizons $T \in \{10, 50, 100\}$; rows correspond to arm counts $N \in \{100, 500, 1000\}$. The Gittins policy is evaluated across cost-scaling parameters λ and compared with random search and successive halving baselines. Relative to random search, Gittins reaches the Pareto frontier; the successive-halving results are dominated and therefore do not reach this frontier.

cost-scaling parameter λ to trace the cost-regret tradeoff. Successive halving is the resource-allocation subroutine used by Hyperband: it allocates a small initial budget to many arms and repeatedly eliminates a fraction of the least promising arms (Jamieson and Talwalkar, 2016; Li et al., 2018).

Figure 5.3 reports the resulting Pareto frontiers. Across the synthetic grid, changing λ moves the Gittins policy along the expected exploration–cost tradeoff: larger values spend less budget, while smaller values continue more arms and reduce

regret. These experiments are intentionally limited to the model-matched setting; empirical robustness evaluation is outside the scope of this thesis version. Further experimental details are provided in Appendix D.3.

5.5 Conclusion

We developed a numerical route for applying Gittins-index methods to Gaussian state-space models by reducing the relevant computation to a one-dimensional dynamic program and solving the resulting Bellman updates efficiently with FFT-based convolutions. In model-matched synthetic experiments, varying the cost-scaling parameter allows the Gittins policy to trace a favorable cost–regret frontier: it Pareto-dominates random search across the tested settings, whereas successive halving does not consistently improve upon the random-search frontier. These results demonstrate that the proposed policy can translate its Bayesian optimality structure into an effective practical tradeoff between evaluation cost and decision quality when the modeling assumptions hold. Empirical robustness to model misspecification and real-world data remains to be evaluated in future work.

APPENDIX A
SUPPLEMENT TO CHAPTER 2

A.1 Illustrations

Here we provide a set of additional illustrations to aid understanding of our results.

A.1.1 Visualization of synthetic benchmark functions

To better understand how the behavior of Bayesian optimization algorithms on the three different synthetic benchmark functions might be affected by their geometric shape, Figure A.1 provides a visual illustration of their two-dimensional variants.¹ This allows us to visually see the multimodality of the Ackley function, multimodality and ridge-like regions in the Levy function, and banana shape of the Rosenbrock function. While we use higher-dimensional versions of these in our experiments, this illustration provides some intuition for what the resulting the optimization landscape might look like, helping contextualize results.

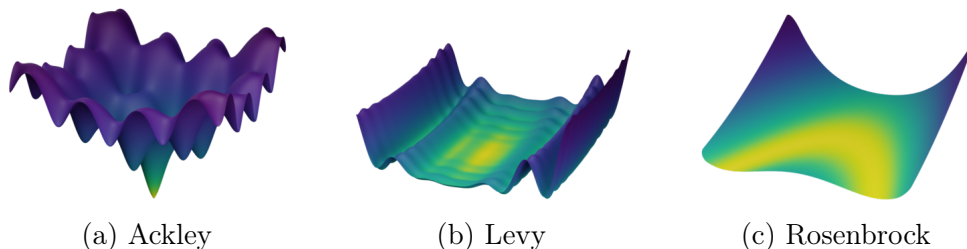


Figure A.1: An illustration of the two-dimensional Ackley, Levy, and Rosenbrock functions.¹ From the visual, one can see that these functions differ in terms of multimodality and ridge-regions within the optimization landscape.

A.1.2 Comparison of behaviors between EI and PBGI

In Figure A.2, we see that EI’s maximum occurs at a point near $x \approx 0.365$, which is located closer to the observed data and is fairly close to the maximum of the posterior mean. Similarly, large- λ PBGI’s maximum occurs at a similar point near $x \approx 0.358$, indicating that the algorithm makes similar decisions in spite of the fact that the acquisition function’s shape is different. In contrast, small- λ PBGI’s maximum occurs at a point near $x \approx 0.86$: it favors a riskier approach, preferring a location where the mean is smaller, but the variance is larger.

This example shows potential differences in behavior between EI and PBGI, and illustrates how these differences are mediated by the parameter λ .

A.1.3 An example where expected improvement underperforms

In Section 2.3.3, we showed a cost-aware Bayesian optimization problem on which the logarithmic form of expected improvement per unit cost (LogEIPC) has poor performance. Here, we show that this problem can be modified so that the logarithmic form of ordinary expected improvement (LogEI), which ignores the cost function, also has poor performance—a somewhat obvious, but nonetheless important sanity check that we make to ensure that costs play a sufficiently-important role in problems of this class to merit their consideration. This is shown in Figure A.3. It is not hard to construct a less-visualization-friendly variant of these problems on which both expected improvement per unit cost and ordinary expected improvement perform poorly, by considering cost functions which are appropriately-weighted sums of bump functions.

¹This visual originally appeared in Terenin (2022), and is reproduced here with permission.

A.2 Theory and calculations

Below, we provide additional ideas to help understand the Pandora's Box problem and acquisition function that results from its considerations.

A.2.1 Additional intuition on Pandora's Box

In what follows, we sketch a viewpoint from which one can see the key idea behind why Theorem 1 holds. Rather than considering the full Pandora's Box problem with a general set of open and closed boxes, consider first the case where there is exactly one closed and one open box. To slightly simplify notation, let f denote the random reward inside the closed box, let c denote the cost of opening the closed box, assumed deterministic, and let g denote the visible reward of the open box. Our possible actions are as follows:

1. *Open the closed box.* In this case, we pay a cost of c , but subsequently get to choose between taking the realized value f , or instead taking g from the box that was originally open. In expectation, the total value obtained by taking this action is $\mathbb{E}(\max(f, g)) - c$.
2. Take the reward from the open box. The total value obtained is g .

We can therefore analytically solve for the optimal policy of this respective Markov decision process: we open the closed box if $\mathbb{E}(\max(f, g)) - c \geq g$, and take the reward from the open box if $\mathbb{E}(\max(f, g)) - c \leq g$, with both actions optimal in the case of equality. As consequence, if g is such that both actions are optimal, the same value is obtained no matter whether one chooses to open the box or not.

Rewriting the preceding expressions slightly, this occurs when

$$\mathbb{E} \max(f - g, 0) = c \tag{A.1}$$

where in the case of multiple boxes the left-hand-side becomes the expected improvement function. The insight of Weitzman (1979)—and indeed of Gittins (1979a) in a much more general setting—is that one can modify the Pandora’s Box Markov decision process by replacing closed boxes with open boxes whose value g satisfies (A.1) without changing the optimal policy. As a consequence, the optimal policy is precisely the Gittins index policy of Theorem 1.

As a final point, note that the assumption that c is deterministic is made without loss of generality: if c is instead stochastic but has finite expectation, the same reasoning applies, but with the value c in (A.1) replaced with its expected value. We will make use of this in Appendix A.2.4.

A.2.2 Gradient of the PBGI acquisition function

In most Bayesian optimization setups, gradient-based methods including multi-start stochastic gradient descent, BFGS, and L-BFGS-B, are used to effectively optimize analytical acquisition functions, such as EI and UCB. These methods can also be used to optimize the PBGI acquisition function. To facilitate this, we provide the gradient formula for the PBGI acquisition function. In what follows, mirroring the preceding and following sections, if costs are stochastic then $c(x)$ should be replaced with its respective mean. We also omit time subscripts to ease notation.

Proposition 23 (Gradient of PBGI). *Let $\mu(x)$ and $\sigma(x)$ be the mean and standard deviation of the posterior Gaussian process $(f \mid x_{1:t}, y_{1:t})(x)$. With this notation,*

the gradient of the acquisition function $\alpha^{\text{PBGI}}(x)$ is given by

$$\nabla \alpha^{\text{PBGI}}(x) = \nabla \mu(x) + \frac{\phi\left(\frac{\mu(x) - \alpha^{\text{PBGI}}(x)}{\sigma(x)}\right) \nabla \sigma(x) - \lambda \nabla c(x)}{\Phi\left(\frac{\mu(x) - \alpha^{\text{PBGI}}(x)}{\sigma(x)}\right)}, \quad (\text{A.2})$$

where ϕ and Φ denote the density and cumulative distribution function of a standard normal distribution, respectively.

Proof. Recall that when $\psi(x) \sim \text{N}(\mu(x), \sigma(x))$ is Gaussian, the expected improvement with respect to the comparator y is given as

$$\text{EI}_{\psi}(x; y) = (\mu(x) - y) \Phi\left(\frac{\mu(x) - y}{\sigma(x)}\right) + \sigma(x) \phi\left(\frac{\mu(x) - y}{\sigma(x)}\right). \quad (\text{A.3})$$

Next, note by definition of α^{PBGI} , we have

$$\text{EI}_{f|x_{1:t}, y_{1:t}}(x; \alpha^{\text{PBGI}}(x)) = \lambda c(x). \quad (\text{A.4})$$

Differentiating this with respect to x on both sides gives

$$\nabla \text{EI}_{f|x_{1:t}, y_{1:t}}(x; \alpha^{\text{PBGI}}(x)) = \lambda \nabla c(x). \quad (\text{A.5})$$

Applying the product and chain rule to the left-hand-side gives

$$\nabla \text{EI}_{f|x_{1:t}, y_{1:t}}(x; \alpha^{\text{PBGI}}(x)) = (\nabla \mu(x) - \nabla \alpha^{\text{PBGI}}(x)) \Phi\left(\frac{\mu(x) - \alpha^{\text{PBGI}}(x)}{\sigma(x)}\right) \quad (\text{A.6})$$

$$+ (\mu(x) - \alpha^{\text{PBGI}}(x)) \phi\left(\frac{\mu(x) - \alpha^{\text{PBGI}}(x)}{\sigma(x)}\right) \nabla \left(\frac{\mu(x) - \alpha^{\text{PBGI}}(x)}{\sigma(x)}\right) \quad (\text{A.7})$$

$$+ \nabla \sigma(x) \phi\left(\frac{\mu(x) - \alpha^{\text{PBGI}}(x)}{\sigma(x)}\right) \quad (\text{A.8})$$

$$+ \sigma(x) \phi'\left(\frac{\mu(x) - \alpha^{\text{PBGI}}(x)}{\sigma(x)}\right) \nabla \left(\frac{\mu(x) - \alpha^{\text{PBGI}}(x)}{\sigma(x)}\right). \quad (\text{A.9})$$

Recall the identity for the derivative of the Gaussian density, namely

$$\phi'(x) = -x\phi(x). \quad (\text{A.10})$$

Applying this identity to (A.9) gives

$$\sigma(x)\phi'\left(\frac{\mu(x) - \alpha^{\text{PBG I}}(x)}{\sigma(x)}\right)\nabla\left(\frac{\mu(x) - \alpha^{\text{PBG I}}(x)}{\sigma(x)}\right) \quad (\text{A.11})$$

$$= -(\mu(x) - \alpha^{\text{PBG I}}(x))\phi\left(\frac{\mu(x) - \alpha^{\text{PBG I}}(x)}{\sigma(x)}\right)\nabla\left(\frac{\mu(x) - \alpha^{\text{PBG I}}(x)}{\sigma(x)}\right) \quad (\text{A.12})$$

which is equal to the negation of (A.7), hence (A.7) and (A.9) cancel: we get

$$(\nabla\mu(x) - \nabla\alpha^{\text{PBG I}}(x))\Phi\left(\frac{\mu(x) - \alpha^{\text{PBG I}}(x)}{\sigma(x)}\right) + \nabla\sigma(x)\phi\left(\frac{\mu(x) - \alpha^{\text{PBG I}}(x)}{\sigma(x)}\right) = \lambda\nabla c(x). \quad (\text{A.13})$$

Rearranging this gives the expression in the claim. $\square$

A.2.3 Small-cost limit of PBGI acquisition function

We now derive the small- λ limiting expression for PBGI which was presented in Section 2.3.3. According to (A.3) and (A.4), we have

$$\alpha_t^{\text{PBG I}}(\cdot) = \mu_t(\cdot) - \sigma_t(\cdot)u(\cdot) \quad (\text{A.14})$$

where μ_t and σ_t are the mean and the standard deviation of $f \mid x_{1:t}, y_{1:t}$, and $u(\cdot)$ is the solution of

$$u(\cdot)\Phi(u(\cdot)) + \phi(u(\cdot)) = \frac{\lambda c(\cdot)}{\sigma_t(\cdot)}. \quad (\text{A.15})$$

When $\lambda \rightarrow 0$, the solution satisfies $u(\cdot) \rightarrow -\infty$. This implies that both $\Phi(u(\cdot)) \rightarrow 0$ and $\phi(u(\cdot)) \rightarrow 0$. Since $\phi(u(\cdot))$ dominates $\Phi(u(\cdot))$, we can approximate the solution $u(\cdot)$ by $\phi(u(\cdot)) \approx \frac{\lambda c(\cdot)}{\sigma_t(\cdot)}$. By substituting the probability density function of a standard normal distribution for ϕ , we obtain

$$u(\cdot) \approx -\sqrt{2\log\left(\frac{\sigma_t(\cdot)}{\lambda c(\cdot)}\right)}. \quad (\text{A.16})$$

Thus, we can conclude that in the limit $\lambda \rightarrow 0$, the Gittins index becomes

$$\alpha_t^{\text{PBG I}}(x; \lambda) \approx \mu_t(x) + \sigma_t(x)\sqrt{2\log\left(\frac{\sigma_t(x)}{\lambda c(x)}\right)}. \quad (\text{A.17})$$

As consequence, we expect PBGI-D, whose λ -parameter eventually decays to zero, to potentially inherit consistency and other properties known for UCB. Specifically, since PBGI-D decreases the value of λ to zero over time by dividing it with a constant every time the Gittins stopping rule triggers, its behavior should eventually be well-described by the above limit. As a result of this limit, unexplored regions where $\sigma(x) \gg 0$ should eventually have acquisition values that are larger than regions where $\sigma(x) \approx 0$. We believe this should hold regardless of whether the costs are known—our main focus—or unknown, especially if the cost function $c(x)$ is uniformly bounded above and below. This because (A.17)’s dominant term in the $\lambda \rightarrow 0$ limit is $\sigma(x)\sqrt{2\log\frac{1}{\lambda}}$, which does not depend on the cost $c(x)$.

A.2.4 Closed-form expression for the PBGI unknown-cost variant

In the Lunar Lander and Robot Pushing empirical examples of Section 2.4, the cost function does not admit an analytic, automatically-differentiable form, and can only be evaluated in a black-box manner. To handle this, we model the logarithm of the costs as a Gaussian process, and condition this process on the costs observed at locations evaluated so far. This mirrors how unknown costs are handled in Astudillo et al. (2021) for other acquisition functions, such as EIPC and BMSEI.

From the viewpoint of the Pandora’s Box problem, stochastic costs make little difference: following the discussion in Appendix A.2.1, the optimality results of Weitzman (1979) continue to hold even if costs are stochastic, so long as the costs in the formula for α^* are replaced with expected costs. Mirroring this, if we plug in the mean of a log-normal random variable into the definition of α^{PBGI} , we obtain the following acquisition function.

Definition 24. Let $c(x)$ be log-normal for all x . For a dataset $(x_1, y_1), \dots, (x_t, y_t)$, let $\mu_{\ln c}$ and $\sigma_{\ln c}$ be the posterior mean and posterior standard deviation of the log-costs. Define the UNKNOWN-COST PANDORA’S BOX GITTINS INDEX ACQUISITION FUNCTION by

$$\alpha_t^{\text{PBGI-U}}(x) = g \quad \text{where } g \text{ solves } \mathbb{E}I_{f|x_{1:t}, y_{1:t}}(x; g) = \lambda \exp\left(\mu_{\ln c}(x) + \frac{\sigma_{\ln c}(x)^2}{2}\right). \quad (\text{A.18})$$

The interpretation of λ , namely as a hyperparameter that determines the expected budget the algorithm will use before reaching its respective stopping time, is the same as in the known-cost setting. One can define cost-per-sample and decay variants in the same manner as well.

A.2.5 Relationship between expected budget-constrained and cost-per-sample problems

We now prove Theorem 2. In what follows, recall that the assumptions of Theorem 1 are (i) X is discrete, (ii) $\mathbb{E}|f(x)| < \infty$ for all x , (iii) $f(x)$ and $f(x')$ are independent for $x \neq x'$, and (iv) the budget satisfies $B > \min_{x \in X} c(x)$. Further, Theorem 1 was stated for deterministic costs: more generally, we allow for stochastic costs satisfying $0 < \mathbb{E}c(x) < \infty$, and in such cases instead define α^* using the expected costs.

We begin by defining the Markov decision process (MDP) under study and stating the properties of it that we will use. Define:

1. States: let $\mathcal{S} = \mathcal{S}^\circ \cup \partial\mathcal{S}$ be the union of two disjoint sets, namely the set $\mathcal{S}^\circ$ of non-terminal states and set $\partial\mathcal{S}$ of terminal states. These are described below:

- (a) Non-terminal states: let $\mathcal{S}^\circ$ consist of all finite sequences of length $|X|$ taking values in $\mathbb{R} \cup \{\boxtimes\}$, where numbers represent the reward inside an open box, and $\boxtimes$ represents a closed box, along with terminal states described below.
 - (b) Terminal states: let $\partial\mathcal{S} = \mathbb{R}$ represent the reward of the box chosen by the learner.
2. Actions: let $\mathcal{A} = X$, where actions represent either opening a closed box, or taking a reward from an open box.
 3. Costs: define the non-terminal cost function $c : \mathcal{S}^\circ \times \mathcal{A} \rightarrow \mathbb{R}^*$ by $c(s, a) = c(a)\mathbb{1}\{s_a = \boxtimes\}$.
 4. Rewards: define the terminal reward function $\partial r : \partial\mathcal{S} \rightarrow \mathbb{R}$ by $\partial r(s) = s$.
 5. Transition kernel: define a Markov transition kernel such that for a non-terminal state s and an action a :
 - (a) If box i is closed, that is, $s_i = \boxtimes$, and a corresponds to opening this closed box, then the MDP transitions to a new state s' , where s'_i represents a random draw of the value in box i and all other components s'_j for $j \neq i$ remains the same as s_j ;
 - (b) If box i is open, that is, $s_i \in \mathbb{R}$, and a corresponds to taking the reward s_i from this open box, then the MDP deterministically transitions to a terminal state $s' = s_i$.

This defines a class of time-homogeneous undiscounted Markov decision processes, parameterized by the reward distribution f of bounded expected value and cost function c . We consider Markov policies, which are probability kernels mapping states to probability measures over actions. For this class of MDPs, all such policies

are guaranteed to terminate in finite time, because at most one can open all the boxes before being forced to select one and thereby enter a terminal state. As consequence, by standard MDP theory:

1. The value function $V^{(\pi,c)} : \mathcal{S} \rightarrow \mathbb{R}$ is well-defined, where the superscripts denote the policy and the cost with respect to which the MDP is defined.
2. The optimal value function $V^{(*,c)} : \mathcal{S} \rightarrow \mathbb{R}$ is also well-defined.
3. There exists an optimal policy $\pi^{(*,c)}$ which achieves the optimal value $V^{(*,c)}$.
4. The map $\mathbb{R}_+^{|X|} \rightarrow \mathbb{R}$ defined by $c \mapsto V^{(\pi,c)}$ is affine for all π .
5. The map $\mathbb{R}_+^{|X|} \rightarrow \mathbb{R}$ defined by $c \mapsto V^{(*,c)}$ is convex, since it is a supremum of affine functions.

We immediately note that this formulation extends to cover two variations of interest:

1. *Stochastic costs.* One can handle this by replacing c with a probability kernel. In this case, letting m be the mean costs, we have $V^{(\pi,c)} = V^{(\pi,m)}$. Using this, we henceforth work with deterministic costs without loss of generality.
2. *Simple regret as a terminal reward.* One can also consider an MDP with a stochastic terminal reward which subtracts the best-in-hindsight term $\sup_{x \in X} f(x)$ from the terminal rewards ∂r given above—this gives an objective equal to simple regret up to a minus sign. Since this only changes the rewards up to a random constant which is independent of the chosen actions, by linearity of expectation, the value function of this modified MDP is equal to those of our MDP up to a constant. We therefore omit this term without loss of generality.

The expected budget-constrained setting does not directly incorporate costs into the MDP itself: more precisely, it takes $c(s, a) = 0$ within the MDP formulation, which means the value function of this modified MDP is $V^{(\pi, 0)}$. Instead, costs are incorporated as a constraint set on the policy class one considers. With this notation, the values of the optimization problems for Bayesian-optimal policy in the expected budget-constrained and cost-per-sample problems are

$$V_{\text{ebc}}^{(*,c)} = \sup_{\pi \in \Pi_B^{(c)}} V^{(\pi, 0)} \quad \quad \quad V^{(*,c)} = \sup_{\pi \in \Pi} V^{(\pi, c)} \quad (\text{A.19})$$

where Π is the set of all Markov policies, and the feasible set for the expected budget-constrained problem is

$$\Pi_B^{(c)} = \left\{ \pi \in \Pi : \mathbb{E} \sum_{t=1}^{T^{(\pi)}} c(x_t) \leq B \right\} \quad (\text{A.20})$$

and $T^{(\pi)}$ is policy π 's stopping time, that is

$$T^{(\pi)} = \inf\{t \geq 1 : s_t \in \partial S\} \quad (\text{A.21})$$

which is bounded above by $|X| < \infty$. This defines the optimization problems under study. Define the set of maximizers

$$\Pi^{(*,c)} = \{\pi \in \Pi : V^{(\pi, c)} = V^{(*,c)}\} \quad (\text{A.22})$$

which is non-empty, since, as said above, by MDP theory the supremum defining $V^{(*,c)}$ is achieved. Define also the set of feasible policies which satisfy the constraints in a tight manner, namely

$$\Pi_{B, \text{eq}}^{(c)} = \left\{ \pi \in \Pi_B^{(c)} : \mathbb{E} \sum_{t=1}^{T^{(\pi)}} c(x_t) = B \right\}. \quad (\text{A.23})$$

We are now ready to prove the claim in question. For this, we employ a Lagrangian duality argument: loosely speaking, this will reveal the cost-per-sample

λ to be the Lagrange multipliers associated with the expected budget constraint. We prove the main claim via a series of lemmas, starting from handling the Lagrange-multiplier-part of the argument.

Lemma 25. *Define the function*

$$\mathcal{A} : [0, \infty) \rightarrow \mathbb{R} \quad \mathcal{A}(\lambda) = V^{(*, \lambda c)} + \lambda B. \quad (\text{A.24})$$

Suppose that the infimum of $\mathcal{A}$ is achieved, and denote it by $\lambda_B^ \in [0, \infty)$. Suppose further that there exists an optimal policy for the cost-per-sample MDP for which the expected budget constraint is tight, namely $\hat{\pi} \in \Pi^{(*, \lambda_B^* c)} \cap \Pi_{B, \text{eq}}^{(c)}$. Then we have*

$$V_{\text{ebc}}^{(\hat{\pi}, c)} = V_{\text{ebc}}^{(*, c)}. \quad (\text{A.25})$$

Proof. We start by expressing the expected budget-constrained optimization problem in an unconstrained form via Lagrange multipliers, obtaining

$$V_{\text{ebc}}^{(*, c)} = \sup_{\pi \in \Pi_B^{(c)}} V^{(\pi, 0)} = \sup_{\pi \in \Pi} \inf_{\lambda \geq 0} \left(V^{(\pi, 0)} - \lambda \left(\mathbb{E} \sum_{t=1}^{T(\pi)} c(x_t) - B \right) \right) \quad (\text{A.26})$$

$$= \inf_{\lambda \geq 0} \sup_{\pi \in \Pi} \left(V^{(\pi, 0)} - \lambda \left(\mathbb{E} \sum_{t=1}^{T(\pi)} c(x_t) - B \right) \right) \quad (\text{A.27})$$

$$= \inf_{\lambda \geq 0} (V^{(*, \lambda c)} + \lambda B) = \inf_{\lambda \geq 0} \mathcal{A}(\lambda) \quad (\text{A.28})$$

where the second line follows from the Lemma 30 since the respective Lagrangian equals $V^{(\pi, \lambda c)}$ up to a constant, and the third line follows by definition of $V^{(*, \lambda c)}$ is by definition the terminal reward minus the cumulative costs.

Now, suppose the infimum of $\mathcal{A}$ is achieved, and let $\lambda_B^* \in [0, \infty)$ be any minimizer. Then $V_{\text{ebc}}^{(*, \lambda_B^* c)} = V^{(*, \lambda_B^* c)} + \lambda_B^* B$. Using this, and the fact that the policy $\hat{\pi}$ by assumption achieves the supremum over $\sup_{\pi \in \Pi} V^{(\pi, \lambda_B^* c)}$ and satisfies $\hat{\pi} \in \Pi_B^{(c)}$, we get

$$\sup_{\pi \in \Pi_B^{(c)}} V^{(\pi, 0)} = \sup_{\pi \in \Pi} V^{(\pi, \lambda_B^* c)} + \lambda_B^* B = \sup_{\pi \in \Pi_B^{(c)}} V^{(\pi, \lambda_B^* c)} + \lambda_B^* B. \quad (\text{A.29})$$

Thus, the optimization objectives, defining the expected budget-constrained problem and the cost-per-sample problem with costs $\lambda_B^* c$, are equal up to a constant. Therefore, their minimizer sets coincide, and the claim follows. $\square$

Lemma 25 reveals that as long as the optimization problems arising from Lagrange multipliers are achieved, and the resulting policy $\hat{\pi}$ is feasible, the expected budget-constrained problem will admit the same optimum as its associated cost-per-sample problem. By MDP theory, we know the supremum over Π is achieved: we now show the infimum involving λ is achieved as well, which essentially amounts to ruling out arbitrarily large λ values.

Lemma 26. *The map $\mathcal{A}$ is convex. Moreover, the infimum $\inf_{\lambda \geq 0} \mathcal{A}(\lambda)$ is achieved.*

Proof. First, note that convexity follows straightforwardly from convexity of $c \mapsto V^{(*,c)}$ and the fact that the sum of convex functions is convex. Next, since the feasible set is $\lambda \in [0, \infty)$ and the objective is convex, either the infimum is achieved, or the objective is non-increasing. We prove the latter property cannot hold. For this, first consider the policy π' that deterministically opens some box $x' \in X$ whose cost is $c(x') < B$ —note that our assumptions guarantee the existence of at least one such box—and selects the value in it. Therefore if we define $\mathcal{A}'(\lambda) = V^{(\pi', \lambda c)} + \lambda B$, we have

$$\mathcal{A}'(\lambda) = V^{(\pi', \lambda c)} + \lambda B \leq V^{(*, \lambda c)} + \lambda B = \mathcal{A}(\lambda). \quad (\text{A.30})$$

At the same time, we have $V^{(\pi', \lambda c)} = \mathbb{E} f(x') - \lambda c(x')$ therefore

$$\mathcal{A}'(\lambda) = \mathbb{E} f(x') - \underbrace{\lambda (c(x') - B)}_{\text{negative}} \quad (\text{A.31})$$

which means the map $\mathcal{A}'$ is affine and strictly increasing with respect to λ . Thus, the map $\mathcal{A}$ is lower-bounded by a strictly increasing affine function, and therefore cannot be non-increasing. We conclude that the infimum is achieved. $\square$

The next part is to show that the optimal cost-per-sample policy is feasible for the expected budget-constrained problem. Then to do so, we need to verify a certain convergence criterion, given below. In the following, note that λ' —which can be negative—is never used in the definition of any optimal policy, only to compute the value of a given policy, which still makes sense even with negative costs.

Lemma 27. *For any monotone sequence $\lambda_n \rightarrow \lambda$ where $\lambda_n > 0$ and $\lambda > 0$, and any $\lambda' \in \mathbb{R}$, there exist policies $\pi_n^* \in \Pi^{(*, \lambda_n c)}$ and $\pi^* \in \Pi^{(*, \lambda c)}$ such that*

$$V(\pi_n^*, \lambda' c) \rightarrow V(\pi^*, \lambda' c). \quad (\text{A.32})$$

Proof. First, note that

$$V(\pi_n^*, \lambda' c) = \underbrace{V(\pi_n^*, \lambda c)}_{(a)} + (\lambda' - \lambda_n) \underbrace{\mathbb{E} \sum_{t=1}^{T(\pi_n^*)} c(x_t)}_{(b)}. \quad (\text{A.33})$$

Using this, by passing limits through the respective sums and products, it suffices to prove convergence of (a) and (b) separately, with the same sequence choice π_n^* in both cases. By Weitzman (1979)—see Kleinberg et al. (2016, Theorem 1), for an alternative proof—any policies π_n^*, π^* which maximize the respective Gittins indices α_n^*, α^* are optimal: we will therefore choose π_n^*, π^* from this set of policies. This choice is not unique due to the possibility of ties, both in terms of which boxes to open, and when to stop: we will show that tie-breaking choices do not affect convergence of (a), and will make a suitable choice of tie-breaking rules, depending on the sequence λ_n in the claim’s assumptions, to prove convergence of (b).

Part I: convergence of (a). Define $\kappa_n(x) = \min(f(x), \alpha_n^*(x))$, and define κ analogously. By Kleinberg et al. (2016, Theorem 1), we have

$$V(\pi^*, \lambda_n c) = \mathbb{E} \max_{x \in X} \kappa_n(x) \quad (\text{A.34})$$

and analogously for λ , α^* , and κ . We now argue that $\alpha_n^* \rightarrow \alpha^*$ monotonically pointwise. Since $\lambda_n \rightarrow \lambda$ converges monotonically, consider

$$\text{EI}_f(x, g) = \lambda_n c(x). \quad (\text{A.35})$$

Recall that $\text{EI}_f(x, g)$ is continuous and strictly decreasing in g : hence, its inverse in g exists and is also strictly decreasing and continuous. We conclude that $\alpha_n^* \rightarrow \alpha^*$ monotonically pointwise. From this, convergence of the respective expectations $\mathbb{E} \max_{x \in X} \kappa_n(x)$, and therefore convergence of (a), follows by the Monotone Convergence Theorem.

Part II: convergence of (b). We will need an identity involving the order in which boxes are opened: for this, we first prove that once a tie-breaking rule is chosen, for large enough n , π_n^* and π^* open boxes in the same order. Since the number of boxes is finite, and $\alpha_n^* \rightarrow \alpha^*$ monotonically pointwise from (a): it follows for n large enough that, if there are no ties in $\alpha^*(x)$, then π_n^* opens boxes in the same order as π^* . If there are ties in α^* , we choose a tie-breaking rule so that π_n^* and π^* open boxes in the same order. It follows that, once this choice is made, for n large enough, all policies open boxes in the same order $x_1, \dots, x_{|X|}$, with different π_n and π^* possibly stopping at different times.

The identity we seek will differ depending on how tie-breaking rules regarding when to stop are handled: recall that ties can occur when opening the box $x_t \in X$ at time t , we have $\alpha^*(x_t) = f_t^*$, where f_t^* is the best observed value up to time t . We adopt the following tie-breaking rule for stopping, depending on whether or not λ_n is an increasing or decreasing sequence:

- If λ_n is increasing: let $\pi^* = \pi_+^*$ be the policy that opens the best closed box in the event of a tie.

- If λ_n is decreasing: let $\pi^* = \pi_-^*$ be the policy that stops in the event of a tie.

Note also that if λ_n is both increasing and decreasing, it is constant, and the claim we want to prove holds: therefore, we suppose it is not, which makes the choice between π_+^* and π_-^* uniquely determined. We make the same tie-breaking choices for π_n^* , letting $\pi_{n,+}^*$ and $\pi_{n,-}^*$ be the respective policies. Let $x_1, \dots, x_{|X|}$ denote boxes in the order they are opened. Then the expected total costs are

$$\mathbb{E} \sum_{t=1}^{T(\pi_+^*)} c(x_t) \stackrel{(i)}{=} \sum_{i=1}^{|X|} \mathbb{P}(T(\pi_+^*) \geq i) c(x_i) \stackrel{(ii)}{=} \sum_{i=1}^{|X|} \mathbb{P}(f_j^* \leq \alpha^*(x_j), 1 \leq j \leq i) c(x_i) \quad (\text{A.36})$$

$$\stackrel{(iii)}{=} \sum_{i=1}^{|X|} \mathbb{P}(f(x_j) \leq \alpha^*(x_i), 1 \leq j \leq i) c(x_i) \stackrel{(iv)}{=} \sum_{i=1}^{|X|} \prod_{j=1}^{i-1} \mathbb{P}(f(x_j) \leq \alpha^*(x_i)) c(x_i) \quad (\text{A.37})$$

where (i) follows by writing the probability as an expectation of an appropriate indicator, (ii) follows by noting that under the policy π_+^* , the event $T(\pi_+^*) \geq i$ occurs if and only if at each time $j = 1, \dots, i$, the best observed value f_j^* is not higher than the Gittins index of x_j , (iii) follows by expanding the maximum which defines f_j^* and using monotonicity of $\alpha^*(x_j)$ in j to simplify the resulting events, and (iv) follows by independence of $f(x_j)$ and $f(x_{j'})$ for $j \neq j'$. Using this, it suffices to show

$$\mathbb{P}(f(x_j) \leq \alpha_n^*(x_i)) \rightarrow \mathbb{P}(f(x_j) \leq \alpha^*(x_i)) \quad \text{if } \pi^* = \pi_+^*, \text{ and} \quad (\text{A.38})$$

$$\mathbb{P}(f(x_j) < \alpha_n^*(x_i)) \rightarrow \mathbb{P}(f(x_j) < \alpha^*(x_i)) \quad \text{if } \pi^* = \pi_-^*. \quad (\text{A.39})$$

The former equals the cumulative distribution function of $f(x_j)$, which is right-continuous, evaluated at $\alpha_n^*(x_i)$. The latter is similar but is instead left-continuous. For $\pi_{n,+}^*$, since λ_n is increasing, we have that $\alpha_n(x)$ is decreasing, and the claim follows by right-continuity. For $\pi_{n,-}^*$ since λ_n is decreasing, we have that $\alpha_n(x)$ is increasing, and the claim follows by left-continuity. $\square$

We are now ready to prove the key property needed to apply Lemma 25.

Lemma 28. *If $\lambda_B^* > 0$, then there exists a $\hat{\pi} \in \Pi^{(*, \lambda_B^* c)} \cap \Pi_{B, \text{eq}}^{(c)}$.*

Proof. By the Envelope Theorem—specifically, Lemma 33—we have for $\lambda \in [0, \infty)$ that

$$\partial_v \mathcal{A}(\lambda) = \partial_v (V^{(*, \lambda c)} + \lambda B) \quad (\text{A.40})$$

$$= \partial_v \left(\sup_{\pi \in \Pi} \left(V^{(\pi, 0)} - \lambda \left(\mathbb{E} \sum_{t=1}^{T(\pi)} c(x_t) - B \right) \right) \right) \quad (\text{A.41})$$

$$= \max_{\pi' \in \Pi^{(*, \lambda c)}} \partial_v \left(V^{(\pi', 0)} - \lambda \left(\mathbb{E} \sum_{t=1}^{T(\pi')} c(x_t) - B \right) \right) \Big|_{\pi=\pi'} \quad (\text{A.42})$$

$$= vB - \min_{\pi \in \Pi^{(*, \lambda c)}} v \mathbb{E} \sum_{t=1}^{T(\pi)} c(x_t) \quad (\text{A.43})$$

where all Gâteaux derivatives—see Definition 31—are taken with respect to λ and exist by convexity, and the pointwise convergence condition of Lemma 33 follows by first noting that

$$V^{(\pi, 0)} - \lambda \left(\mathbb{E} \sum_{t=1}^{T(\pi)} c(x_t) - B \right) = V^{(\pi, \lambda c)} + \lambda B \quad (\text{A.44})$$

and applying Lemma 27. By the first-order optimality conditions, we have

$$\partial_v \mathcal{A}(\lambda_B^*) \geq 0. \quad (\text{A.45})$$

Combining this with the above, we conclude

$$\min_{\pi \in \Pi^{(*, \lambda c)}} v \mathbb{E} \sum_{t=1}^{T(\pi)} c(x_t) \leq vB. \quad (\text{A.46})$$

Since $\lambda_B^* > 0$, this expression holds for v equal to ± 1 : plugging this in, we get

$$\min_{\pi \in \Pi^{(*, \lambda_B^* c)}} \mathbb{E} \sum_{t=1}^{T(\pi)} c(x_t) \leq B \quad B \leq \max_{\pi \in \Pi^{(*, \lambda_B^* c)}} \mathbb{E} \sum_{t=1}^{T(\pi)} c(x_t). \quad (\text{A.47})$$

Now, let $\hat{\pi}_-$ be a policy from the minimizer set, and let $\hat{\pi}_+$ be a policy from the maximizer set. Define a third policy $\hat{\pi}_\alpha$ which randomizes between the two, choosing $\hat{\pi}_-$ with probability α and $\hat{\pi}_+$ with probability $1 - \alpha$. Since $\hat{\pi}_+$ and $\hat{\pi}_-$ are optimal, they achieve the same value, so by convexity $\hat{\pi}_\alpha$ also achieves the same value: therefore, $\hat{\pi}_\alpha \in \Pi^{(*, \lambda_B c)}$. On the other hand, the expected total costs of $\hat{\pi}_\alpha$ are a convex combination of the expected costs of $\hat{\pi}_-$ and $\hat{\pi}_+$: since the former lower-bounds B and the latter upper-bounds B , there exists an $\alpha \in [0, 1]$ for which the costs of π_α are exactly B . Taking $\hat{\pi} = \hat{\pi}_\alpha$ for this value of α , we obtain $\hat{\pi} \in \Pi_{B, \text{eq}}^{(c)}$ and the claim follows. $\square$

To complete the proof, we show the minimizer set of $\mathcal{A}$ does not contain zero.

Lemma 29. *Let $\lambda_B^* \in [0, \infty)$ be any minimizer of $\mathcal{A}$, then $\lambda_B^* > 0$.*

Proof. From the derivative calculation used in proof of Lemma 28, plugging in $v = 1$ into the respective Gâteaux derivative, we obtain

$$\frac{d}{d\lambda} \mathcal{A}(\lambda) = B - \min_{\pi \in \Pi^{(*, \lambda c)}} \mathbb{E} \sum_{t=1}^{T(\pi)} c(x_t). \quad (\text{A.48})$$

If $\lambda = 0$, then the costs are zero: thus, any policy that opens every box, then selects the highest reward among opened boxes, is optimal. Therefore, we have $T^{(\pi)} = |X|$, and every optimal policy's cost is equal to the total cost of all the boxes. This gives

$$\frac{d}{d\lambda} \mathcal{A}(0) = B - \sum_{x \in X} c(x) < 0 \quad (\text{A.49})$$

where the inequality follows by the assumption that the expected budget constraint is active. The claim follows. $\square$

With these results at hand, we are now ready to prove the main claim.

Theorem 2. *Consider the expected budget-constrained problem, with the assumptions of Theorem 1. Assume the problem is feasible and the constraint is active, namely $\min_{x \in X} c(x) < B < \sum_{x \in X} c(x)$. Then there exists a $\lambda > 0$ and a tie-breaking rule such that the policy defined by maximizing the Gittins index acquisition function $\alpha^*(\cdot)$, defined using costs $\lambda c(x)$, is Bayesian-optimal.*

Proof. Combine Lemma 25 with Lemmas 26, 28 and 29. □

We conclude by comparing this claim and proof with the results of Aminian et al. (2024):

1. The objective of their expected budget-constrained problem is more general: while we focus on minimizing simple regret, Aminian et al. (2024) study how to maximize utility, which is defined in a broader context—one example being the difference between the best observed value and cumulative costs, which we study here. Our expected budget constraint is similarly a special case, which, in their terminology, takes all weights on indicators denoting inspection to be identical, and takes all selection weights to be zero.
2. We do not assume that $f(x)$ has finite support for all x . This results in a significant technical difference: we must apply a sharp envelope theorem with an explicit supremum to conclude that the expected budget constraint is satisfied. More-straightforward results such as those presented by Milgrom and Segal (2002), or their subdifferential-formulated analogues as found in the proof of Aminian et al. (2024, Proposition 1 (iii)), would not suffice: due to an explicit counterexample involving upside-down-absolute-value functions, without suitable structure, one cannot conclude that the supremum is achieved and the resulting inequality is tight. As an alternative to our approach, one

could instead appeal to finiteness of the class of deterministic policies—which Aminian et al. (2024) assume, leading to piecewise-linear value functions. We avoid these assumptions, leading to a more technical but more general argument.

We conclude by noting that one can also consider variants of the problems we have studied under an almost-sure budget constraint, for instance

$$\Pi_{B,\text{as}}^{(c)} = \left\{ \pi \in \Pi : \mathbb{P} \left(\sum_{t=1}^{T(\pi)} c(x_t) \leq B \right) = 1 \right\} \quad V_{\text{asbc}}^{(*,c)} = \sup_{\pi \in \Pi_{B,\text{as}}^{(c)}} V^{(\pi,0)}. \quad (\text{A.50})$$

For this problem, this optimal value is upper-bounded by the optimal value of the expected budget-constrained optimization—specifically, $V_{\text{asbc}}^{(*,c)} \leq V_{\text{ebc}}^{(*,c)}$, since $\Pi_{B,\text{as}}^{(c)} \subseteq \Pi_B^{(c)}$.

A.2.6 Auxillary results from optimization theory

Below we state and prove three results from optimization theory: a Lagrange Multiplier Theorem for optimization problems with inequality constraints, and two Envelope Theorems. These results are not new, but are difficult to find at the level of generality we need them at: most references begin by assuming the domain of optimization is $\mathbb{R}^d$, or, if not that, that it is a Banach space, whereas we need results that hold when the domain of optimization is an arbitrary set, without a linear structure or a topology. In light of this, we have found it easier to simply prove the claims we need.

Lemma 30. *Let X be an arbitrary set, let $f : X \rightarrow \mathbb{R}$ and let $g : X \rightarrow \mathbb{R}$. Then defining $\mathcal{L}(x, \lambda) = f(x) - \lambda(g(x) - y)$ for $\lambda \geq 0$, we have*

$$\sup_{\substack{x \in X \\ g(x) \leq y}} f(x) = \sup_{x \in X} \inf_{\lambda \geq 0} \mathcal{L}(x, \lambda). \quad (\text{A.51})$$

Moreover, suppose there exist x^*, λ^* which satisfy $\mathcal{L}(x^*, \lambda^*) = \inf_{\lambda \geq 0} \sup_{x \in X} \mathcal{L}(x, \lambda)$ and $g(x^*) = y$. Then we have

$$\sup_{x \in X} \inf_{\lambda \geq 0} \mathcal{L}(x, \lambda) = \inf_{\lambda \geq 0} \sup_{x \in X} \mathcal{L}(x, \lambda). \quad (\text{A.52})$$

Proof. First, we show that for all x , we have

$$\inf_{\lambda \geq 0} \mathcal{L}(x, \lambda) = \begin{cases} f(x) & g(x) \leq y \\ -\infty & g(x) > y. \end{cases} \quad (\text{A.53})$$

To show this, suppose first that $g(x) \leq y$. Then $\lambda(g(x) - y) \leq 0$, hence its negation is non-negative, and minimized at $\lambda = 0$. Now, suppose the converse. Then $\lambda(g(x) - y) > 0$, so its negation is negative, and the objective can be made arbitrarily close to $-\infty$ by scaling λ . Using this, write

$$\sup_{x \in X} \inf_{\lambda \geq 0} \mathcal{L}(x, \lambda) = \max \left(\sup_{\substack{x \in X \\ g(x) \leq y}} \inf_{\lambda \geq 0} \mathcal{L}(x, \lambda), \sup_{\substack{x \in X \\ g(x) > y}} \inf_{\lambda \geq 0} \mathcal{L}(x, \lambda) \right) \quad (\text{A.54})$$

$$= \max \left(\sup_{\substack{x \in X \\ g(x) \leq y}} f(x), -\infty \right) = \sup_{\substack{x \in X \\ g(x) \leq y}} f(x). \quad (\text{A.55})$$

We now argue that, under the claim's additional assumption, one can swap the order of the supremum and infimum. Let x^*, λ^* be a pair for which the constraints are tight. Then

$$\inf_{\lambda \geq 0} \sup_{\substack{x \in X \\ g(x) \leq y}} \mathcal{L}(x, \lambda) \stackrel{(i)}{=} \sup_{\substack{x \in X \\ g(x) \leq y}} \mathcal{L}(x, \lambda^*) \stackrel{(ii)}{=} \sup_{\substack{x \in X \\ g(x) = y}} \mathcal{L}(x, \lambda^*) \quad (\text{A.56})$$

$$\stackrel{(iii)}{=} \sup_{\substack{x \in X \\ g(x) = y}} \inf_{\lambda \geq 0} \mathcal{L}(x, \lambda) \stackrel{(iv)}{\leq} \sup_{\substack{x \in X \\ g(x) \leq y}} \inf_{\lambda \geq 0} \mathcal{L}(x, \lambda) \quad (\text{A.57})$$

where (i) follows by definition of λ^* , (ii) follows by the fact that x^* achieves the supremum and satisfies $g(x^*) = y$, (iii) follows by the fact that, when restricted to the set $\{x \in X : g(x) = y\}$, $\mathcal{L}(x, \lambda)$ is constant in λ for all x -values, hence the

infimum is taken over constant functions, (iv) follows by making the feasible set larger. Combining this with the sup-inf inequality (Boyd and Vandenberghe, 2004)

$$\sup_{\substack{x \in X \\ g(x) \leq y}} \inf_{\lambda \geq 0} \mathcal{L}(x, \lambda) \leq \inf_{\lambda \geq 0} \sup_{\substack{x \in X \\ g(x) \leq y}} \mathcal{L}(x, \lambda) \quad (\text{A.58})$$

gives the claim. $\square$

It is easy to see that this argument holds even if one lets y take values in an infinite-dimensional vector space, as long as λ takes values within a convex cone which is suitably paired with the aforementioned vector space, but we will not need this level of generality. The arguments we employ will be cleanest if we work with directional derivatives in the sense of Gâteaux, as opposed to left-derivatives and right-derivatives of real-valued functions, which in our setting are equivalent but require more management of minus signs. For this, we adopt the following notation.

Definition 31. *Let $\mathcal{X}$ be a topological vector space, let $X \subseteq \mathcal{X}$, and let $f : X \rightarrow \mathbb{R}$ be a function. The GÂTEAUX DERIVATIVE $\partial_v f(x) \in \mathbb{R}$ of a function f at a point $x \in X$ in the direction $v \in \mathcal{X}$, if it exists, is defined as*

$$\partial_v f(x) = \lim_{\varepsilon \rightarrow 0^+} \frac{f(x + \varepsilon v) - f(x)}{\varepsilon}. \quad (\text{A.59})$$

Note that we require no properties of our Gâteaux derivatives: in particular, $\partial_v f(x)$ can be non-linear in v , though one can easily see that it will always satisfy $\partial_{\alpha v} f(x) = \alpha \partial_v f(x)$ for $\alpha \geq 0$. If f is a convex function, one can show by monotonicity of finite differences that $\partial_v f(x)$ always exists—in the non-extended-valued sense defined above—on the relative interior of the effective domain of f .

Our arguments need an appropriate Envelope Theorem. The first statement we need is essentially equivalent to Milgrom and Segal (2002, Theorem 1), which is

formulated in the language of left-derivatives and right-derivatives: Milgrom and Segal (2002) state in a footnote that their claim also holds in a general normed vector space, provided one works with directional differentiation. In fact, the claim is even more general than that, and does not require a norm. Since we find this variant to be particularly clean, elegant, and instructive, we prove the claim in full generality below.

Lemma 32. *Let Θ be a subset of a topological vector space, and let X be an arbitrary set. Let $f : X \times \Theta \rightarrow \mathbb{R}$ be bounded above in its first argument, and define $\mathcal{V}$ to be*

$$\mathcal{V}(\theta) = \sup_{x \in X} f(x, \theta) \quad (\text{A.60})$$

Suppose that, for every $\theta \in \Theta$, the supremum is achieved, and let $X^(\theta)$ be the maximizer set. For any v , suppose that the Gâteaux derivatives $\partial_v \mathcal{V}(\theta)$ and $\partial_v f(x, \theta)$ exist and are finite-valued, with the convention that the Gâteaux derivative of f is taken in its second argument. Then*

$$\sup_{x^* \in X^*(\theta)} \partial_v f(x^*, \theta) \leq \partial_v \mathcal{V}(\theta). \quad (\text{A.61})$$

Proof. Fix $\theta \in \Theta$, and let $x^*(\theta) \in X^*(\theta)$ be an arbitrary maximizer. Note that, for all $\theta' \in \Theta$ and all $x^*(\theta') \in X^*(\theta')$, by optimality we have

$$f(x^*(\theta), \theta') \leq f(x^*(\theta'), \theta') = \mathcal{V}(\theta') \quad (\text{A.62})$$

with equality for $\theta' = \theta$. Letting $\varepsilon > 0$ be sufficiently small, taking $\theta' = \theta + \varepsilon v$, subtracting $f(x^*(\theta), \theta) = \mathcal{V}(\theta)$ from both sides, and dividing by ε , we get

$$\frac{f(x^*(\theta), \theta + \varepsilon v) - f(x^*(\theta), \theta)}{\varepsilon} \leq \frac{\mathcal{V}(\theta + \varepsilon v) - \mathcal{V}(\theta)}{\varepsilon} \quad (\text{A.63})$$

thus taking limits as $\varepsilon \rightarrow 0$ gives

$$\partial_v f(x^*(\theta), \theta) \leq \partial_v \mathcal{V}(\theta). \quad (\text{A.64})$$

This holds for all choices $x^*(\theta) \in X^*(\theta)$, and the claim follows. $\square$

For our arguments to go through, we need a sharper version of this result, with the inequality replaced with an equality. Results like this go back at least to Danskin (1967), and are proven by Bonnans and Shapiro (2013, Proposition 4.12), Bařar and Bernhard (2008, Theorem 10.1), and Bertsekas (1999): however, all of their claims require topological assumptions on the domain of optimization. Moreover, it is easy to see—for instance by considering an envelope made up of upside-down absolute value functions, where $\mathcal{V}(\theta)$ is constant but $f(x, \theta)$ is not—that the inequality cannot be tight without similar assumptions. Below, we show that being affine in θ and a certain convergence criterion suffice, even if X is an arbitrary set. The argument is similar to that of the aforementioned references.

Lemma 33. *With the assumptions and notations in Lemma 32, suppose further that $f(x, \theta)$ is affine in θ for all x , and that for any θ and any v , there exists $x_v^*(\theta) \in X^*(\theta)$ such that, for all sufficiently-small $\varepsilon > 0$, there exist $x_v^*(\theta + \varepsilon v) \in X^*(\theta + \varepsilon v)$, such that we have $f(x_v^*(\theta + \varepsilon v), v) \rightarrow f(x^*(\theta), v)$. Then we have*

$$\sup_{x^* \in X^*(\theta)} \partial_v f(x^*, \theta) = \max_{x^* \in X^*(\theta)} \partial_v f(x^*, \theta) = \partial_v f(x^*(\theta), \theta) = \partial_v \mathcal{V}(\theta). \quad (\text{A.65})$$

Proof. For any ε , note that

$$\frac{\mathcal{V}(\theta + \varepsilon v) - \mathcal{V}(\theta)}{\varepsilon} = \frac{f(x^*(\theta + \varepsilon v), \theta + \varepsilon v) - f(x^*(\theta), \theta)}{\varepsilon} \quad (\text{A.66})$$

$$= \frac{f(x^*(\theta + \varepsilon v), \theta + \varepsilon v) - f(x^*(\theta + \varepsilon v), \theta)}{\varepsilon} \quad (\text{A.67})$$

$$+ \frac{\overbrace{f(x^*(\theta + \varepsilon v), \theta) - f(x^*(\theta), \theta)}^{\text{negative}}}{\varepsilon} \quad (\text{A.68})$$

$$\leq \frac{f(x^*(\theta + \varepsilon v), \theta + \varepsilon v) - f(x^*(\theta + \varepsilon v), \theta)}{\varepsilon} \quad (\text{A.69})$$

$$= f(x^*(\theta + \varepsilon v), v) \quad (\text{A.70})$$

where the respective term is negative because $f(x^*(\theta + \varepsilon v), \theta) \leq f(x^*(\theta), \theta)$, which holds by optimality of $x^*(\theta)$. Taking limits gives

$$\partial_v \mathcal{V}(\theta) \leq f(x^*(\theta), v) = \partial_v f(x^*(\theta), \theta) \quad (\text{A.71})$$

where the final equality follows from the expression for the Gâteaux derivative of a linear function. Combining this expression with the Lemma 32 shows that the respective supremum is achieved, and gives the claim. $\square$

A.3 Experimental setup

We implement all experiments in BOTORCH (Balandat et al., 2020). Following standard practice, we initialize each optimization algorithm with $2(d + 1)$ values drawn using a quasirandom Sobol sequence, where d is the dimension of the domain. All computations were run on CPU, with individual experiments ran in parallel on various nodes of the Cornell G2 cluster, each allocated up to 4GB of memory, except for certain exceptions. These exceptions include KG, MSEI, and BMSEI for higher dimensions, which required substantially more memory, up to 32GB. Most individual runs took several minutes at most, with exception of the more-expensive KG, MSEI and BMSEI baselines: more information with a direct runtime comparison is given in Appendix A.4.2.

Gaussian process models. In the Bayesian regret experiments shown in Figure 2.4, we use Matérn kernels with identical fixed hyperparameters—namely smoothness $5/2$ and length scale 10^{-1} —for both the Gaussian process prior used to sample the objective function, and the Gaussian process used for Bayesian optimization. To maintain consistency, we do not standardize data in this variant. For the synthetic and empirical experiments shown in Figure 2.5 and Figure 2.6,

we use Matérn kernels with smoothness $5/2$ and length scales learned from data via maximum marginal likelihood optimization, and standardize input variables to be in $[0, 1]^d$ along with output variables to be zero-mean and unit-variance, following BoTorch defaults. Categorical variables in the Pest Control problems are treated as ordinal variables following (Li et al., 2024), although categorical variables typically require special treatment (e.g., one-hot encoding or embedding) in Gaussian process models. In the unknown-cost experiments, we model the objective and the logarithm of the cost function using independent Gaussian processes. Additional Bayesian regret experiment results, showing the effect of varying the kernel smoothness and length scale hyperparameters, are provided in Appendix A.4.6.

Acquisition function optimization. This is done as follows. We begin by computing acquisition values at $200d$ points spread across the domain X , where d is the dimension of X . For all acquisition functions except MSEI and BMSEI, which use a modification described below, the initial $200d$ points are generated using a Sobol sequence design. From these, $10d$ points are selected according to the initialization heuristic used by BoTorch, detailed in Balandat et al. (2020, Appendix F.1). We then use multi-start L-BFGS-B to optimize the acquisition function from each selected point. The point with highest acquisition value among the $10d$ optimized points is chosen as the next evaluation point.

We now detail the modified strategy used for MSEI and BMSEI: here, the initial $200d$ points are selected using the warm-start initialization strategy described in Jiang et al. (2020, Appendix D) and Astudillo et al. (2021, Appendix F). This strategy uses the optimal solution from the previous iteration, targeting the branch that originates from the tree’s root and whose fantasy sample most closely matches the actual observed value of the previously suggested candidate on the true function.

This modification favors MSEI and BMSEI, slightly disadvantaging PBGI and other baselines.

Acquisition function hyperparameters. For PBGI, we choose the constant hyperparameter to be $\lambda = 10^{-4}$. For PBGI-D, we choose the initial value to be $\lambda_0 = 10^{-1}$ and the constant decay factor $\beta = 2$. For both variants, we compute the Gittins indices using 100 iterations of bisection search without any early stopping or other performance and reliability optimizations.

For LogEICC, we follow Lee et al. (2020) to use $\alpha_t = \frac{B-t}{B}$ as the cost exponent where B is set to be the cumulative cost cap shown on our plots. For UCB, we follow the schedule of Srinivas et al. (2009) given by $\eta_t = 2 \log(dt^2\pi^2/6\delta)$, where d is the dimension. We also adopt the choice of $\delta = 10^{-1}$ and a scale-down factor of 5, as used in that work’s experiments. For MSEI and BMSEI, we use 4 lookahead steps, each with a batch size of 1 and a single fantasy point.

Omitted baselines. We omit MSEI from the Bayesian regret plots for $d = 16$ with $\kappa = 10^{-1}$ and for $d = 32$ with all length scale choices because we were unable to get it to work reliably in these settings: the implementation of Jiang et al. (2020) results in frequent crashes due to running out of memory and related issues when used on higher-difficulty problems. We also omit KG from $d = 32$, BMSEI from the cost-aware Pest Control and Robot Pushing experiments, and MES from the Bayesian regret experiment with $d = 32$, for the same reasons.

In addition to the baselines mentioned in Section 2.4, we also implemented the *predictive entropy search (PES)* acquisition function of Hernández-Lobato et al. (2014), but could not get its computations to run reliably in an automatic-

differentiation-based environment without resulting in NaN gradients. Hernández-Lobato et al. (2014) document this behavior, and suggest using finite-differencing in situations where it occurs: however, from initial examination, we found this to decrease performance on higher-dimensional problems. We therefore opted to restrict ourselves to automatically-differentiable baselines and omit PES, to ensure that performance differences seen can reliably be attributed to the acquisition functions used, and not to gradient computation.

Objective functions: Bayesian regret. For Bayesian regret, this is straightforward: the objective is simply a draw from a Fourier feature approximation of the respective Gaussian process prior, drawn in such a way that different baselines with the same random number seed share the same objective, but objectives for different seeds are different draws from the same prior. We use a total of 1024 Fourier features.

Objective functions: synthetic benchmarks. The synthetic benchmark functions we use are as follows. We use variants with dimension $d = 4, 8, 16$ in our experiments.

Ackley: this is

$$f_A(x_1, \dots, x_d) = 20 - 20 \exp \left(-0.2 \sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2} \right) - \exp \left(\frac{1}{d} \sum_{i=1}^d \cos(2\pi x_i) \right) + e \quad (\text{A.72})$$

with search domain $X = [-1, 1]^d$.

Levy: this is $f_L = 100 \hat{f}_L$ where

$$\hat{f}_L(x_1, \dots, x_d) = s_1 + \sum_{i=1}^{d-1} (w_i - 1)^2 (1 + 10 \sin(\pi w_i + 1)^2) + (w_d - 1)^2 (1 + \sin(2\pi w_d)^2) \quad (\text{A.73})$$

with $w_i = 1 + \frac{x_i - 1}{4}$ and $s_1 = \sin(\pi w_1)^2$, and search domain $X = [-10, 10]^d$.

Rosenbrock: this is $f_R = 10^5 \hat{f}_R$ where

$$\hat{f}_R(x_1, \dots, x_d) = \sum_{i=1}^{d-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \quad (\text{A.74})$$

with search domain $X = [-5, 10]^d$.

Cost function. In the cost-aware Bayesian regret and synthetic benchmark experiments, we use the cost function

$$c(x) = 20\|S(x)\|_1 + 1 \quad (\text{A.75})$$

where S is an affine map used to standardize the input domain: specifically, $S(x) = Ax + b$ where A is a diagonal matrix and b is a vector, both chosen so that the image of X under S is $[0, 1]^d$.

Objective functions: empirical. We now detail the empirical objective functions.

Pest Control ($d = 25$). The pest control problem, as described in Oh et al. (2019) aims to minimize the spread of pests as well as the costs of prevention treatment. We adopt the experiment setup from Li et al. (2024), framing this as a categorical optimization problem with 25 variables, each representing a stage of intervention with 5 values reflecting different treatments. The objective function combines the spread of pests and the costs of prevention.

In our cost-aware experiment, we use the cost of prevention as the cost function. This can be computed in an automatically-differentiable manner, thus this problem is a known-cost problem.

Lunar Lander ($d = 12$). Following the setup in Eriksson et al. (2019), we consider a reinforcement learning problem optimizing a controller for the lunar lander as implemented in OpenAI Gym, which includes 12 continuous input dimensions for engine throttle adjustments. The state space captures the lander’s position, angle, time derivatives, and leg contact status. The controller’s actions allow for directional booster firings or inaction. The objective is to maximize the average final reward over 50 randomly generated environments.

For cost-aware experimentation, we choose the cost to be the average number of simulation time steps, assuming batch processing in groups of 16. This assumption is based on the implementation found in the code associated with Li et al. (2024). Note that this objective involves the number of actual simulation steps used, and is therefore not automatically-differentiable: we thus consider this problem to be an unknown-cost problem.

Robot Pushing ($d = 14$). In this work, we adapt one of the three versions of the robot pushing problem designed by Wang and Jegelka (2017), where two robots work to push two objects to their specified targets. The problem’s complexity is captured through 14 control parameters, including each robot’s initial placement and motion settings. The objective minimizes the sum of the distances from the final position of each object to its respective target.

In our cost-aware experiments, we test both known-cost and unknown-cost variants. The known-cost variant is the maximum of the two robots’ operational duration representing the evaluation time, and the unknown-cost variant is the sum of their traversal distances representing the total energy use, similar to the cost function used for energy-aware robot pushing benchmark of Astudillo et al. (2021), but with one modification: we use the distance traversed by the robot

arms instead of the distance the objects are moved. To understand the effect of this difference, we include the results for both the unknown-cost and known-cost versions in Appendix A.4.4.

A.4 Additional experimental results

Here, we provide additional experimental results to better understand performance differences and other aspects of policy behavior, including the effect of various problem hyperparameters.

A.4.1 Impact of log-scaling and numerical stability on EI and PBGI

Ament et al. (2023) introduce the LogEI variant of Expected Improvement (EI), which is equal to the logarithm of EI but incorporates improved numerical properties. Their work demonstrates that LogEI significantly outperforms EI in practice. In this appendix, we aim to investigate two key questions:

1. What specific properties of LogEI contribute to its improved performance over ordinary EI?
2. Can the stable numerical primitives proposed by Ament et al. (2023) also benefit PBGI?

The LogEI acquisition function, denoted as $\alpha^{\text{LogEI}}(x)$, is defined as the logarithm of the EI acquisition function, $\alpha^{\text{EI}}(x)$, but is computed using a numerical helper function `log_h` designed to provide better numerical stability. This introduces two potential sources of improvement for LogEI over EI: (a) the enhanced numerical

stability due to the use of $\log_h$, and (b) the effect of logarithmic scaling, which alters the gradients of the acquisition function and may lead to more effective optimization of $\alpha^{\text{LogEI}}(x)$. To disentangle these two factors, we compare not only LogEI and ordinary EI—the latter of which we refer to as *unstable EI (Unst. EI)* in this section for disambiguation—but also two additional variants: $\text{Log}(\text{Unst. EI})$, which directly applies a logarithmic transformation to unstable EI ($\log(\alpha^{\text{EI}}(x))$) without using $\log_h$ or other numerical stability tricks, and $\text{Exp}(\text{LogEI})$ ($\exp(\alpha^{\text{LogEI}}(x))$), which applies the numerical helper function from LogEI, but then undoes the logarithmic transformation. For $\text{Log}(\text{Unst. EI})$, we compute $\log(\alpha^{\text{EI}}(x) + 10^{-6})$ to avoid infinite floating-point evaluations.

To test whether PBGI can benefit from the improved numerical stability, we also test PBGI and the following variant, which we call *stable PBGI (S-PBGI)*, which uses the following gradient formula:

$$\nabla_x \alpha^{\text{S-PBGI}}(x) = \nabla_x \mu(x) + \frac{\nabla_x \sigma(x) - \sigma(x) \nabla_x \log c(x) - z \log_h'(z) \nabla_x \sigma(x)}{\log_h(z)} \quad (\text{A.76})$$

where $\log_h(z)$ is given by (9) of Ament et al. (2023). We do not consider logarithmic scaling of the PBGI acquisition function value because it can be negative.

From Figure A.4 we see that the improved performance of LogEI over EI appears to be mostly attributable to the log-scaling, though the numerically stable computation provides significant benefit, too. In contrast, there appears to be little-to-no difference between PBGI and S-PBGI.

A.4.2 Runtime comparison

Here, we provide a runtime comparison between PBGI with 100 bisection iterations and various baselines, including inexpensive baselines such as LogEI and TS, and expensive ones such as MSEI. We do so in the Ackley function setting, using the same hyperparameter settings as the main experiments. We do not perform any explicit caching in computing μ or σ or their gradients, and simply rely on standard BoTorch routines to obtain these at each iteration. We measure the CPU time (in seconds) to compute and optimize the acquisition function.

Results are in Figure A.5. We see that PBGI is slightly slower than LogEI and TS, but significantly faster than either KG or MSEI, though the runtime of the latter decreases substantially as it accumulates more data. Overall, we conclude that PBGI’s runtime is closer to that of classical acquisition functions than sophisticated lookahead-based variants. If necessary, PBGI’s runtime can potentially be reduced further by optimizing the computation of $\nabla\mu$ and $\nabla\sigma$, or using fewer bisection iterations.

A.4.3 Hyperparameter choice for PBGI-D

Next, we examine the behavior of different hyperparameter choices for λ_0 and β in PBGI-D. Recall that this variant sets $\lambda_t = \frac{\lambda_{t-1}}{\beta}$ at times when the Gittins stopping rule triggers, and $\lambda_t = \lambda_{t-1}$ at all other times. To understand this, we examine regret curves in the Bayesian regret setting of Section 2.4 with $d = 8$ under different choices. Results are in Figure A.6. We see that behavior of different λ_0 -values is qualitatively similar to that of different λ -values in PBGI, but with substantially smaller differences between variants, which are detectable primarily

due to the relatively-large number of seeds in each experiment. Smaller values of λ_0 tend to lead to very slightly higher regret on small time scales, but very slightly lower regret on larger time scales. Behavior for β is similar: larger values lead to a more gradual decay of λ , but with minimal overall impact on performance, especially in comparison to between-seed variability. We conclude that, in terms of regret, PBGI-D is less sensitive to hyperparameter choice compared to PBGI, whose behavior was previously shown in Figure 2.3.

A.4.4 Effect of unknown costs

The Robot Pushing empirical benchmark involves two cost functions: a known-cost variant based on the maximum operational duration, reflecting the evaluation time, and an unknown-cost variant based on the total distance traversed, a proxy for energy use similar to the variant considered by Astudillo et al. (2021). One can therefore ask: is the resulting algorithm behavior similar in these two settings? Figure A.7 shows that the answer is *yes*: for both the known-cost and the unknown-cost settings, LogEIPC, LogEICC, and PBGI-D outperform FMES and the non-myopic BMSEI baseline.

A.4.5 Alternative visualization of experimental results: means and standard errors

Figure A.8 gives an alternative version of Figures 2.4 to 2.6: instead of showing the median and quartiles, it shows the mean and standard error under 95% confidence, without Bonferroni correction. This can be used, for instance, to assess the influence of the number of random seeds used on our results.

A.4.6 Kernel and problem hyperparameters

Choice of kernel. To check whether our results are sensitive to the kernel used for the Gaussian process model, we replicated the Bayesian regret experiments with Matérn kernels with smoothness parameters $\nu = 3/2, 5/2$, as well as the squared exponential kernel, which is the limit of Matérn kernels as $\nu \rightarrow \infty$ (Rasmussen and Williams, 2006a).

Similar to the original results of Figure 2.4, we can clearly see from Figure A.9 and Figure A.10 that behavior splits into three regimes:

1. *Easy*: d sufficiently-small, most policies achieve similar performance.
2. *Medium-hard*: d moderate-to-large, both PBGI variants match or outperform baselines.
3. *Very hard*: d sufficiently large, no policy outperforms random search.

We also see that $d = 32$ lands in the very-hard regime for the uniform-cost case but not for the cost-aware case: intuitively, this occurs because costs can reduce the effective volume of the search space, since high-cost regions without promising points can be excluded from search.

This behavior is consistent among different kernels, but where the exact threshold at which regimes switch differs. In particular, for the squared exponential kernel, the separation between the medium and the hard regime appears earlier than for the other variants: all policies there have similar performance to random search when $d = 16$.

Choice of length scale. To check whether our results are sensitive to the Gaussian process model’s length scale, we compare $\kappa = 10^{-1}$ with $\kappa = 5 \cdot 10^{-1}$ and $\kappa = 10^0$. From Figure A.11 and Figure A.12, which show uniform-cost and cost-aware results, respectively, we can see that PBGI variants’ performance tends to improve relative to most baselines as dimension increases in both scenarios. Since $\kappa = 10^0$ and $\kappa = 5 \times 10^{-1}$ result in easier problems than $\kappa = 10^{-1}$, in the uniform-cost case under $d = 32$, these two length scales land into the medium-hard regime rather than the very-hard regime.

Synthetic benchmark dimension. To better understand the effect of problem dimension in settings outside of Bayesian regret, we repeat the synthetic benchmark experiments with $d = 4$, $d = 8$ and $d = 16$. Results in Figure A.14. Since $d = 4$ and $d = 8$ are less-difficult, PBGI-D usually performs comparably to baselines, while PBGI either under-performs or over-performs, depending on the objective. This behavior might be explainable by PBGI’s sensitivity to hyperparameters in comparison with PBGI-D, whose performance is more-uniform.

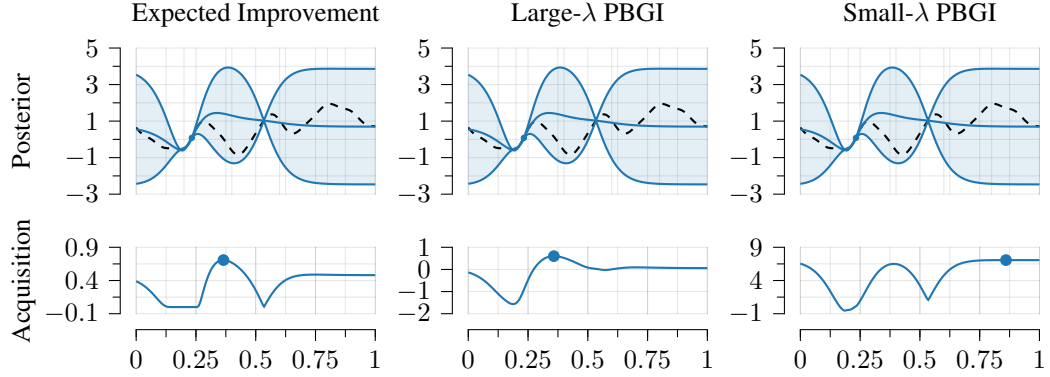


Figure A.2: Comparison between three acquisition functions, namely Expected Improvement (EI), Pandora’s Box Gittins Index (PBGI) with a large λ -value of $\lambda = 10^0$, and Pandora’s Box Gittins Index with a small λ -value of $\lambda = 10^{-5}$. All three are computed using the same posterior distribution with four data points. The maximum is shown as a large dot.

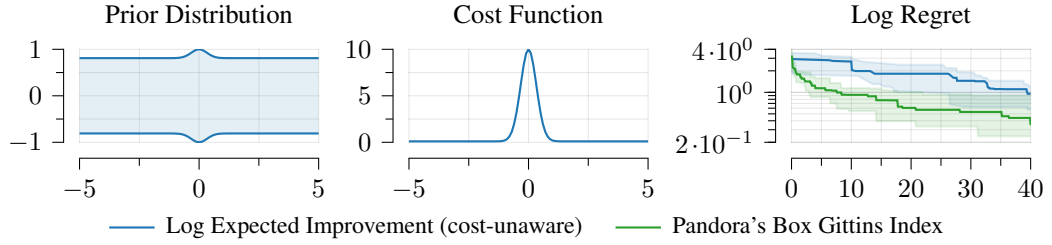


Figure A.3: A Bayesian optimization problem with varying costs on which LogEI, a numerically stable implementation of EI—which by construction ignores costs—has poor performance. See Section 2.4 and Appendix A.4.1 for additional details. Like Figure 2.2, this construction also mirrors Astudillo et al. (2021, Section A). The domain is $X = [-500, 500]$, which we visualize on the subinterval $[-5, 5]$. Left: illustration of the non-uniform prior variance, which is given by a Matérn-5/2 kernel scaled by a narrow bump function, plus a constant. Center: the cost function, which is a narrow bump-shaped function. Right: regret curves, shown as medians and quartiles.

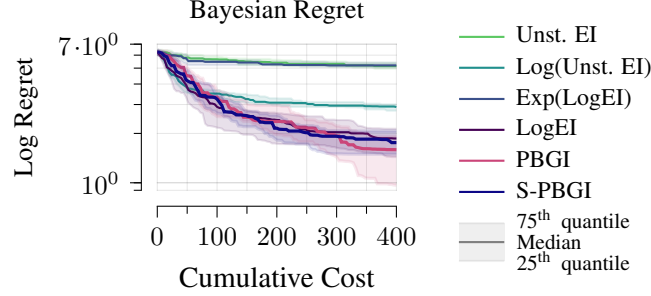


Figure A.4: Experimental results comparing the impact of log-scaling and numerical stability on the performance of EI and PBGI, tested on Bayesian regret with $d = 16$ and shown using quantiles. We see that $\text{Exp}(\text{LogEI})$ is close to Unst. EI, and similarly for S-PBGI vs PBGI. However, $\text{Log}(\text{Unst. EI})$ improves significantly over EI, and LogEI improves even more.

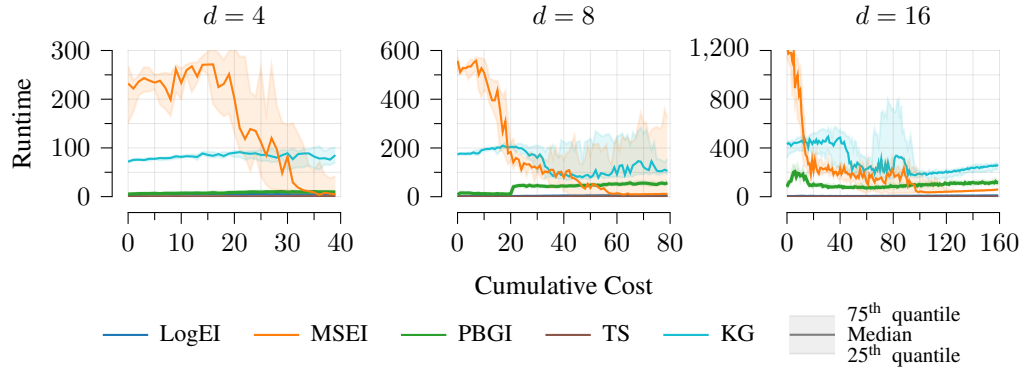


Figure A.5: Runtime comparison of PBGI against baselines for computing the acquisition function on the Ackley benchmark across different dimensions ($d = 4, 8, 16$). We see that runtime of PBGI is slower than LogEI and TS, but faster than KG and MSEI.

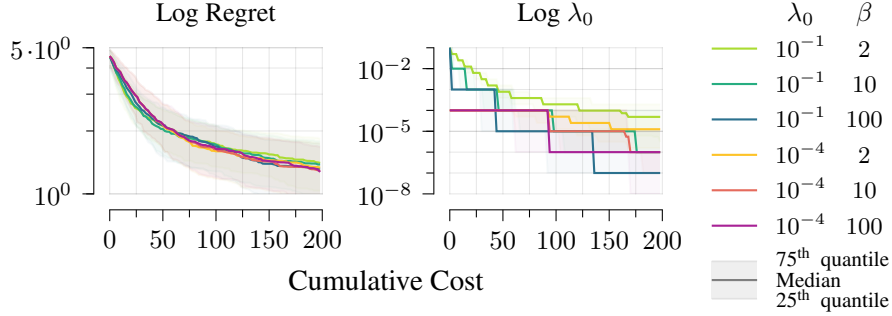


Figure A.6: Behavior of PBGI-D under different choices of the initial value λ_0 , and the coefficient of decay β , on the Bayesian regret experiment with $d = 8$ and other parameters set the same way as in Section 2.4. We show medians under $n = 256$ samples, along with quartiles to assess variability. We see that smaller λ_0 -values act similarly to the behavior seen in Figure 2.3 for PBGI, trading off risk-seeking vs. risk-averse behavior, but with a substantially smaller gap between the two variants. Larger β -values lead to more-abrupt decay curves for λ , with an effect qualitatively similar to λ_0 . In both cases, though our relatively-large sample size allows us to see small differences between methods, their overall impact on performance is close-to-nonexistent on the scale of variability.

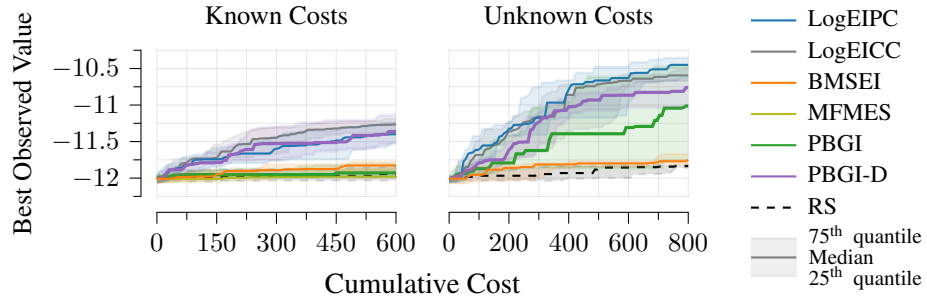


Figure A.7: Experimental results for the Robot Pushing empirical benchmark, with the known-cost variant (left) and unknown-cost variant (right). We see that LogEIPC, LogEICC, and PBGI-D performing best in both cases.

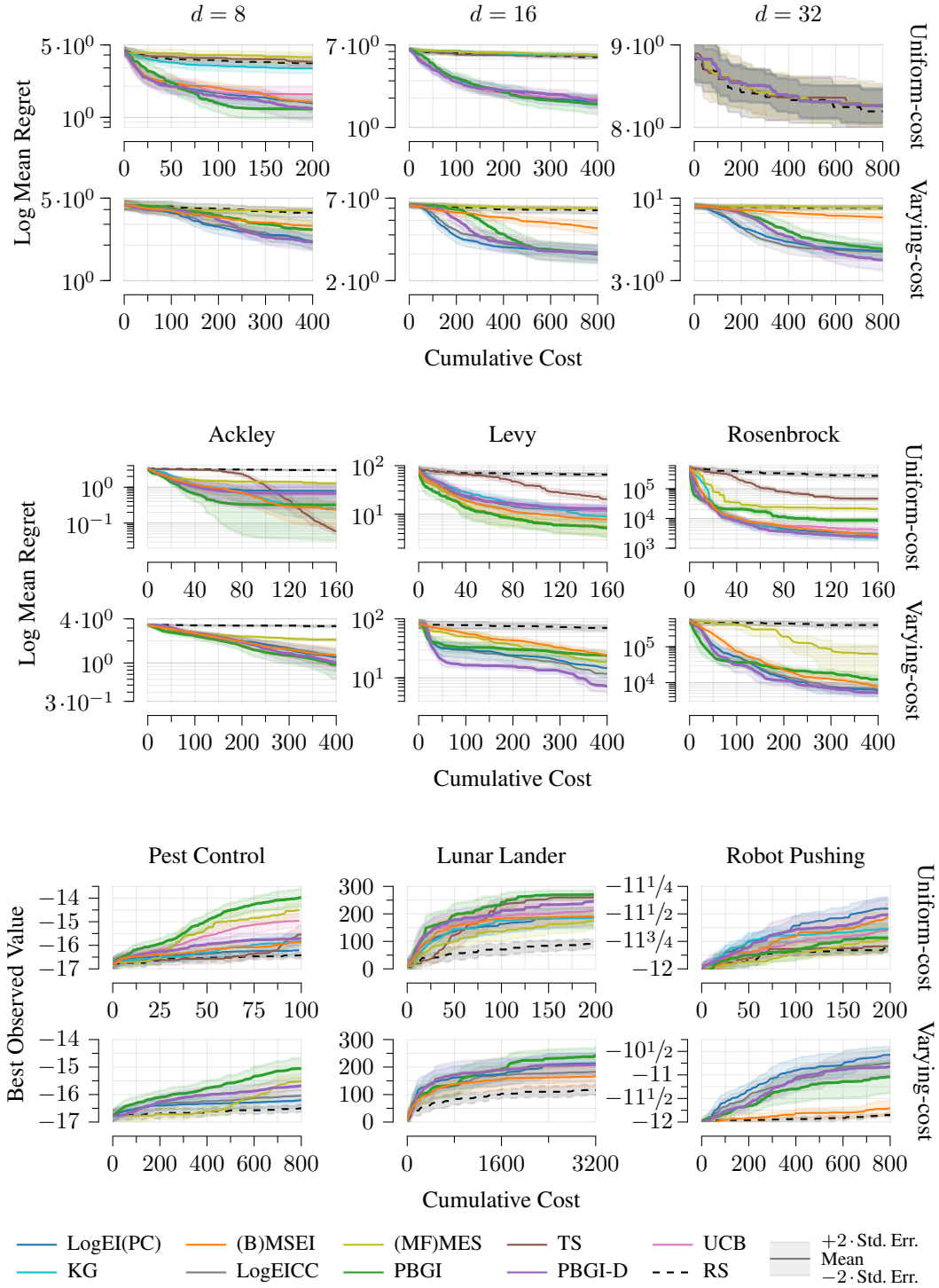


Figure A.8: Regret curves for the Bayesian regret and synthetic benchmark experiments, as well as best observed values for the empirical experiments, showing using means (solid lines) and two times standard errors (shaded regions). See Figures 2.4 to 2.6 for details.

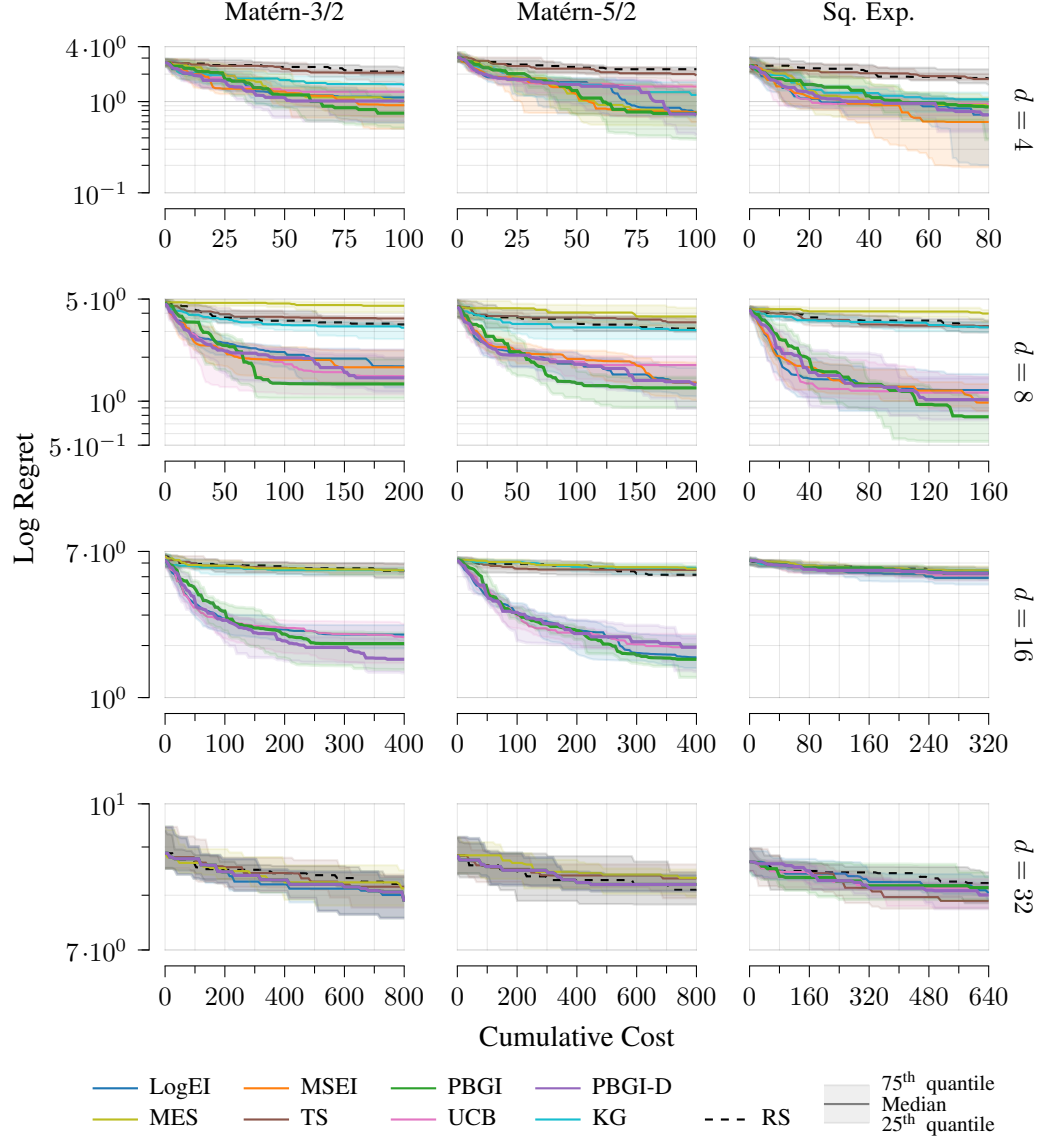


Figure A.9: Comparison of Bayesian regret across Gaussian process priors with different kernels over different dimensions, in the uniform-cost setting. All length scales are $\kappa = 10^{-1}$. We see that overall behavior is similar, but the precise thresholds at which each example switches between the easy, medium-hard, and very hard regimes differ.

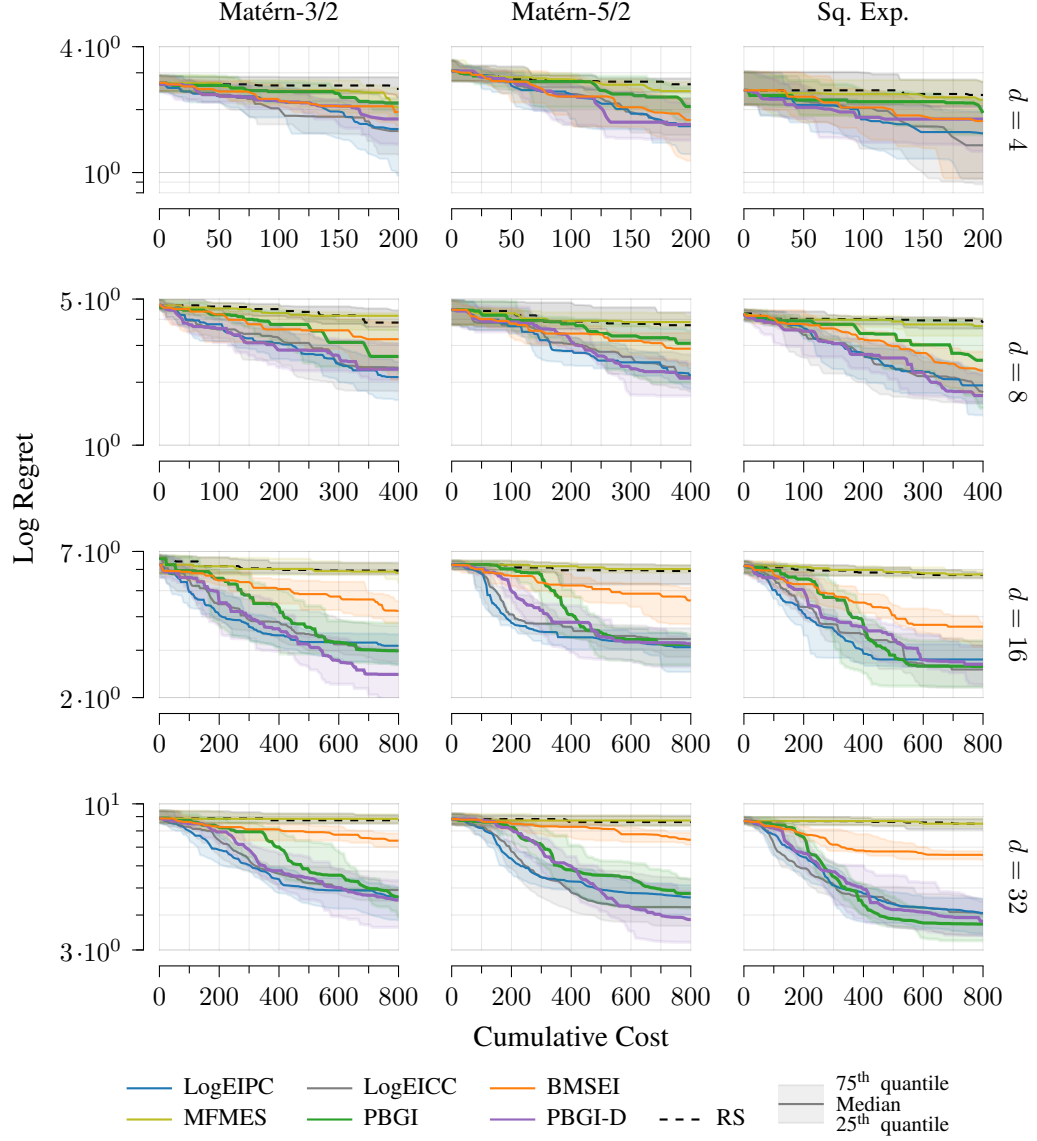


Figure A.10: Comparison of Bayesian regret across Gaussian process priors with different kernels over different dimensions, in the cost-aware setting. All length scales are $\kappa = 10^{-1}$. We see that overall behavior is similar, but the precise thresholds at which each example switches between the easy, medium-hard, and very hard regimes differ.

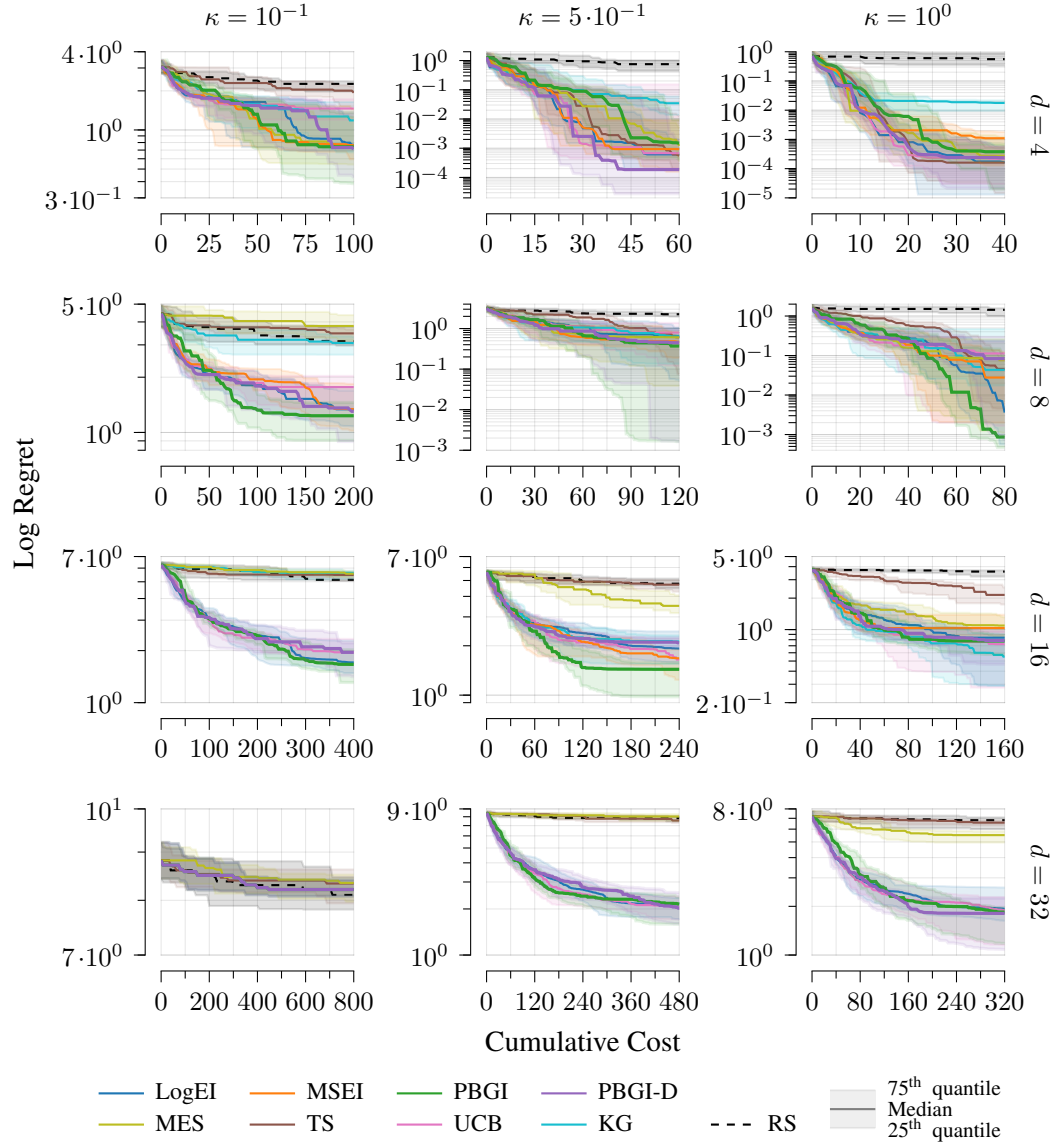


Figure A.11: Comparison of Bayesian regret across different length scales and dimensions, with a Matérn-5/2 kernel, in the uniform-cost setting. We see similar overall behavior, but each example switches between the easy, medium-hard, and very hard regimes at different precise thresholds.

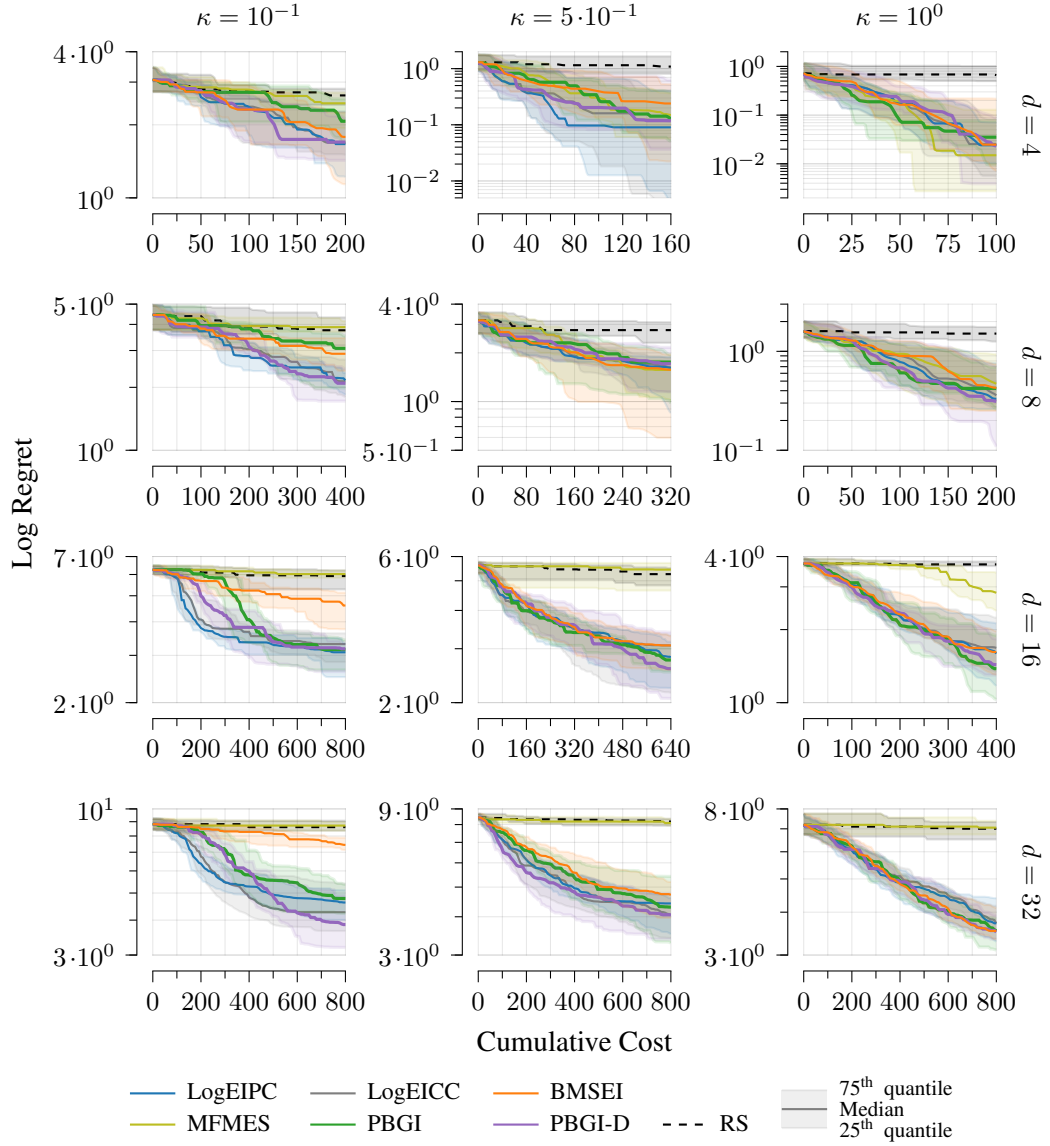


Figure A.12: Comparison of Bayesian regret across different length scales and dimensions, with a Matérn-5/2 kernel, in the cost-aware setting. We see similar overall behavior, but each example switches between the easy, medium-hard, and very hard regimes at different precise thresholds.

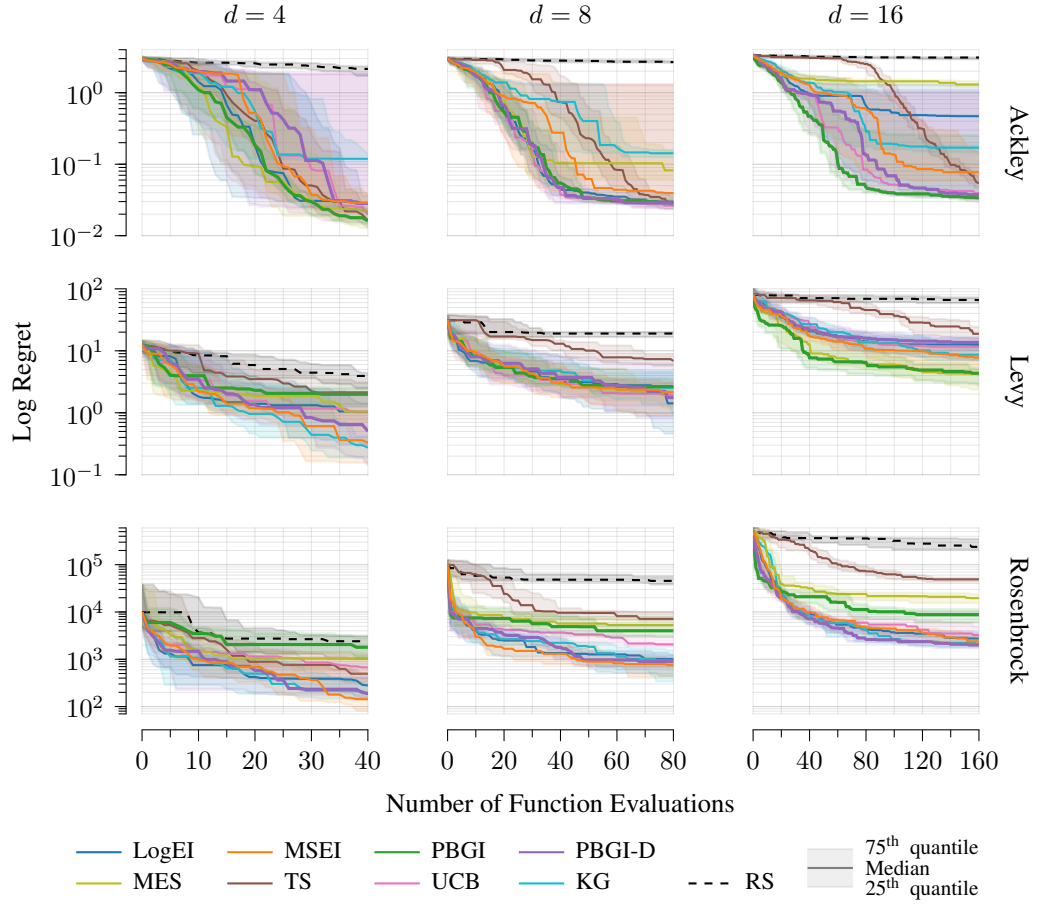


Figure A.13: Comparison of regret for synthetic benchmark functions under different dimensions, in the uniform-cost setting. We see that most methods perform similarly for $d = 4$, with differences between the most-competitive methods emerging as dimension increases to $d = 8$ and $d = 16$.

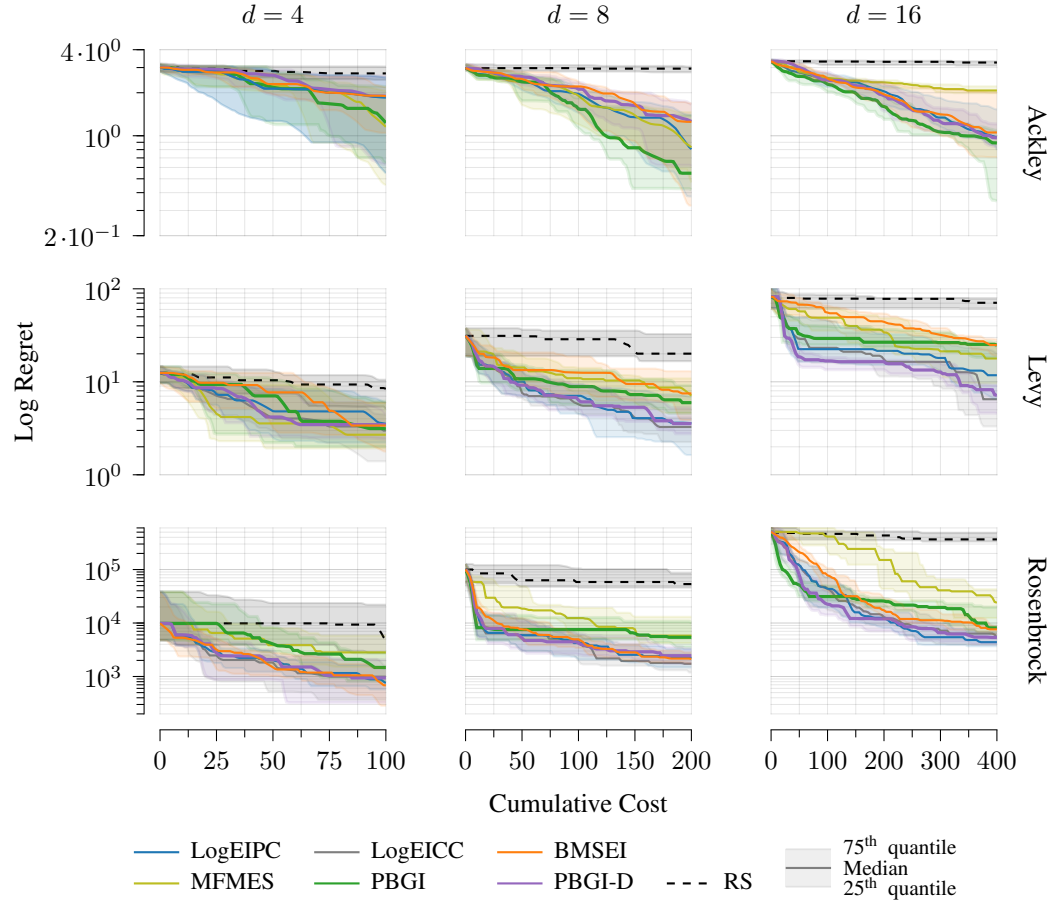


Figure A.14: Comparison of regret for synthetic benchmark functions under different dimensions, in the cost-aware setting. We see that most methods perform similarly for $d = 4$, with differences between the most-competitive methods emerging as dimension increases to $d = 8$ and $d = 16$.

APPENDIX B

SUPPLEMENT TO CHAPTER 3

B.1 Theoretical Analysis and Calculations

In the following lemma, we prove a point-wise lower bound on the expected improvement before stopping for our recommended pairing of PBGI/LogEIPC stopping rule paired with either PBGI or LogEIPC acquisition function.¹

Lemma 3. *Let X be compact, and let $f : X \rightarrow \mathbb{R}$ be a random function with prior mean $\mu(\cdot)$. Consider a Bayesian optimization algorithm that begins at some initial point $x_1 \in X$ with cost $C = c(x_1)$, acquires subsequent points using either the PBGI or LogEIPC acquisition function, and terminates according to the PBGI/LogEIPC stopping rule. Let $\tau = \min_{t \geq 1} \{\sup_{x \in X} \alpha_t^{\text{LogEIPC}}(x) \leq 0\}$ be the algorithm's stopping time, and denote the posterior expected improvement function by $\alpha_t^{\text{EI}}(x) = \text{EI}_{f|x_{1:t}, y_{1:t}}(x; y_{1:t}^*)$. Then, for all $t < \tau$, $\alpha_t^{\text{EI}}(x_{t+1}) \geq c(x_{t+1})$.*

Proof. While stopping has not occurred, meaning $t < \tau$, by the stopping criteria definition we have $\max_{x \in X} \alpha_t^{\text{EI}}(x)/c(x) \geq 1$. Hence, there exists at least one point with $\alpha_t^{\text{EI}}(x)/c(x) \geq 1$. We now argue for each algorithm.

PBGI. For each x we defined a threshold $\alpha_t^{\text{PBGI}}(x)$ by $\text{EI}_{f|x_{1:t}, y_{1:t}}(x; \alpha_t^{\text{PBGI}}(x)) = c(x)$. Since $\text{EI}_{f|x_{1:t}, y_{1:t}}$ is increasing in its second argument,

$$y_{1:t}^* \geq \alpha_t^{\text{PBGI}}(x) \iff \alpha_t^{\text{EI}}(x)/c(x) \geq 1. \quad (\text{B.1})$$

The existence of a point with ratio at least 1 therefore implies that the set $\mathcal{S}_t = \{x : y_{1:t}^* \geq \alpha_t^{\text{PBGI}}(x)\}$ is non-empty. PBGI chooses x_{t+1} with the *smallest* threshold,

¹In fact, any acquisition function (e.g., expected improvement-cost (EIC) (Hu et al., 2025)) that ensures the one-step expected improvement is worth the evaluation cost before the stopping time τ can apply here.

and thus

$$\alpha_t^{\text{EI}}(x_{t+1}) = \text{EI}_{f|x_{1:t}, y_{1:t}}(x_{t+1}; y_{1:t}^*) \geq \text{EI}_{f|x_{1:t}, y_{1:t}}(x_{t+1}; \alpha_t^{\text{PBG}}(x_{t+1})) = c(x_{t+1}). \quad (\text{B.2})$$

(Log)EIPC. By definition x_{t+1} maximizes $\log(\alpha_t^{\text{EI}}(x)/c(x))$ and $\alpha_t^{\text{EI}}(x)/c(x)$, hence $\alpha_t^{\text{EI}}(x_{t+1}) \geq c(x_{t+1})$.

Thus for *both* algorithms, we have

$$\alpha_t^{\text{EI}}(x_{t+1}) \geq c(x_{t+1}), \quad \text{for all } t < \tau. \quad (\text{B.3})$$

□

We can now use Lemma 3 to prove the following theorem, where we show that our PBGI/LogEIPC stopping rule (paired with the PBGI or LogEIPC acquisition function) also achieves cost-adjusted simple regret no worse than a naive baseline—stopping-immediately (*Immediate*).

Theorem 4. *Consider the setting and algorithm specified in Lemma 3. Let $U := \mu(x_1) - \mathbb{E}[\min_{x \in X} f(x)] < \infty$, then the algorithm's expected cost-adjusted simple regret is bounded by*

$$\mathbb{E} \left[y_{1:\tau}^* - \min_{x \in X} f(x) + \sum_{t=1}^{\tau} c(x_t) \right] \leq \mathbb{E} \left[\underbrace{y_1 - \min_{x \in X} f(x) + c(x_1)}_{\text{cost-adjusted regret of immediate stopping}} \right] = U + C. \quad (3.12)$$

Proof. We treat the two algorithms—meaning, the two acquisition function and stopping rule pairs—together and write $\mathcal{F}_t = \sigma(\{x_i, y_i\}_{i=1}^t)$ for the filtration generated by the observations. Since we are minimizing, the one-step improvement after iteration t is $y_{1:t-1}^* - y_{1:t}^*$, where recall that $y_{1:t}^* = \min_{1 \leq i \leq t} y_i$. By our assumption

about the random function f , $\mathbb{E}[y_{1:1}^*] = \mu(x_1)$ and the quantity $y_{1:1}^* - \min_{x \in X} f(x)$ has finite expectation $U < \infty$. Denote the posterior expected improvement function as $\alpha_t^{\text{EI}}(x) = \mathbb{E}_{f|x_{1:t}, y_{1:t}}(x; y_{1:t}^*)$.

By Lemma 3, $c(x_t) \leq \alpha_{t-1}^{\text{EI}}(x_t)$ for all $t \leq \tau$. Set $\Delta_t = y_{1:t-1}^* - y_{1:t}^* \geq 0$ for $2 \leq t \leq \tau$. Conditional on $\mathcal{F}_{t-1}$ and the choice of x_t , we have

$$\mathbb{E}[\Delta_t \mid \mathcal{F}_{t-1}, x_t] = \alpha_{t-1}^{\text{EI}}(x_t). \quad (\text{B.4})$$

Taking expectations and summing up from 2 to τ ,

$$\begin{aligned} \mathbb{E} \left[\sum_{t=2}^{\tau} \alpha_{t-1}^{\text{EI}}(x_t) \right] &= \mathbb{E} \left[\sum_{t=2}^{\infty} \mathbf{1}_{\{t \leq \tau\}} \alpha_{t-1}^{\text{EI}}(x_t) \right] \\ &= \mathbb{E} \left[\sum_{t=2}^{\infty} \mathbf{1}_{\{t \leq \tau\}} \mathbb{E}[\Delta_t \mid \mathcal{F}_{t-1}, x_t] \right] \\ &= \mathbb{E} \left[\sum_{t=2}^{\infty} \mathbf{1}_{\{t \leq \tau\}} \Delta_t \right] \quad (\text{tower property}) \\ &= \mathbb{E} \left[\sum_{t=2}^{\tau} \Delta_t \right] = \mathbb{E}[y_{1:1}^* - y_{1:\tau}^*]. \end{aligned} \quad (\text{B.5})$$

Summing (B.3) over $t \leq \tau$, taking expectations, and applying (B.5), we have

$$\mathbb{E} \left[\sum_{t=2}^{\tau} c(x_t) \right] \leq \mathbb{E} \left[\sum_{t=2}^{\tau} \alpha_{t-1}^{\text{EI}}(x_t) \right] \leq \mathbb{E}[y_{1:1}^* - y_{1:\tau}^*]. \quad (\text{B.6})$$

Adding the term $y_{1:\tau}^*$ on both sides of (B.6) gives

$$\mathbb{E} \left[y_{1:\tau}^* + \sum_{t=2}^{\tau} c(x_t) \right] \leq \mathbb{E}[y_{1:1}^*]. \quad (\text{B.7})$$

Finally, adding $c(x_1)$ to both sides and subtracting $\mathbb{E}[\min_{x \in X} f(x)]$ yields

$$\begin{aligned} \mathbb{E} \left[y_{1:\tau}^* - \min_{x \in X} f(x) + \sum_{t=1}^{\tau} c(x_t) \right] &\leq \mathbb{E} \left[y_{1:1}^* - \min_{x \in X} f(x) + c(x_1) \right] \\ &= \mathbb{E} \left[y_1 - \min_{x \in X} f(x) + c(x_1) \right] \\ &= \mu(x_1) - \mathbb{E} \left[\min_{x \in X} f(x) \right] + \mathbb{E}[c(x_1)] \\ &= U + C, \end{aligned}$$

since $\mathbb{E}[y_1] = \mu(x_1)$, $C = c(x_1)$, and $U = \mu(x_1) - \mathbb{E}[\min_{x \in X} f(x)]$. This is exactly (3.12). $\square$

Remark 34. *The quantities U and C depend on the choice of the initial point x_1 . Choosing x_1 to minimize $\mu(x_1) + c(x_1)$ yields a tighter bound.*

Remark 35. *In practice, one may center the objective by replacing $f(x)$ with $f(x) - \mu(x)$ and then using a zero-mean GP prior. This changes only the parameterization of the posterior mean (which can be shifted back by $\mu(x)$) and does not affect the implementation of the acquisition functions or stopping rules.*

Notably, this guarantee may not hold for other acquisition–stopping rule pairings. Moreover, in the worst case, this is the best guarantee we can hope for—for instance, the evaluation costs can be uniformly high and no point is worth evaluating. We now formalize this worst-case regime.

Proposition 36 (Immediate stopping is optimal under large costs). *Suppose the objective function is bounded as $f(x) \in [a, b]$ for all $x \in \mathcal{X}$, and define $B := b - a$. Consider any Bayesian optimization policy that starts from an initial evaluation x_1 and may take additional evaluations. If the evaluation cost satisfies*

$$c(x) \geq B \quad \text{for all } x \in \mathcal{X}, \tag{B.8}$$

then the policy that stops immediately after the initial evaluation is optimal with respect to expected cost-adjusted regret.

Proof. Let π_{imm} denote the policy that stops immediately after the initial evaluation x_1 , and let π be any other policy with stopping time $\tau \geq 1$. Recall that the cost-adjusted simple regret is

$$R_c(\pi) := \mathbb{E} \left[\min_{1 \leq t \leq \tau} f(x_t) - \min_{x \in \mathcal{X}} f(x) + \sum_{t=1}^{\tau} c(x_t) \right]. \tag{B.9}$$

Since $y_1 = f(x_1)$ and $f(x) \in [a, b]$ for all $x \in \mathcal{X}$, we have

$$0 \leq y_1 - \min_{x \in \mathcal{X}} f(x) \leq b - a = B. \quad (\text{B.10})$$

The immediate-stopping policy has $\tau = 1$, so

$$R_c(\pi_{\text{imm}}) = \mathbb{E} \left[y_1 - \min_{x \in \mathcal{X}} f(x) + c(x_1) \right]. \quad (\text{B.11})$$

Now consider an arbitrary policy π .

If $\tau = 1$, then π coincides with immediate stopping and incurs the same cost-adjusted simple regret.

If $\tau \geq 2$, then the simple regret term is nonnegative, and the policy incurs at least one additional evaluation beyond x_1 . Hence,

$$\min_{1 \leq t \leq \tau} f(x_t) - \min_{x \in \mathcal{X}} f(x) + \sum_{t=1}^{\tau} c(x_t) \geq c(x_1) + c(x_2). \quad (\text{B.12})$$

Using $c(x) \geq B$ for all x ,

$$c(x_1) + c(x_2) \geq c(x_1) + B \geq c(x_1) + \left(y_1 - \min_{x \in \mathcal{X}} f(x) \right). \quad (\text{B.13})$$

Therefore, in all cases,

$$\min_{1 \leq t \leq \tau} f(x_t) - \min_{x \in \mathcal{X}} f(x) + \sum_{t=1}^{\tau} c(x_t) \geq y_1 - \min_{x \in \mathcal{X}} f(x) + c(x_1). \quad (\text{B.14})$$

Taking expectations yields

$$R_c(\pi) \geq R_c(\pi_{\text{imm}}), \quad (\text{B.15})$$

which proves optimality of immediate stopping. $\square$

Corollary 5. *Consider the setting and algorithm specified in Lemma 3. Then the expected cumulative cost of the algorithm is bounded by $\mathbb{E} [\sum_{t=1}^{\tau} c(x_t)] \leq U + C$.*

Further, if evaluation costs are uniformly bounded below by a constant $c_0 > 0$, i.e., $c(x) \geq c_0, \forall x \in X$, then for any $\delta \in (0, 1)$, the algorithm terminates in at most $\frac{U+C}{\delta \cdot c_0}$ iterations with probability $1 - \delta$.

Proof. Immediately by Theorem 4 and the fact that $y_{1:\tau}^* - \min_{x \in X} f(x) \geq 0$ pointwise, we have that the expected cumulative cost of the algorithm

$$\mathbb{E} \left[\sum_{t=1}^{\tau} c(x_t) \right] \leq U + C. \quad (\text{B.16})$$

By Markov's inequality, for any $\delta \in (0, 1)$,

$$\Pr \left[\sum_{t=1}^{\tau} c(x_t) \leq \frac{U + C}{\delta} \right] \geq 1 - \delta.$$

Since $c(x_t) \geq c_0$ for all $t < \tau$, it follows that

$$\tau = \sum_{t=1}^{\tau} 1 \leq \sum_{t=1}^{\tau} \frac{c(x_t)}{c_0} = \frac{1}{c_0} \sum_{t=1}^{\tau} c(x_t).$$

Therefore,

$$\Pr \left[\tau \leq \frac{U + C}{\delta \cdot c_0} \right] \geq \Pr \left[\frac{1}{c_0} \sum_{t=1}^{\tau} c(x_t) \leq \frac{U + C}{\delta \cdot c_0} \right] = \Pr \left[\sum_{t=1}^{\tau} c(x_t) \leq \frac{U + C}{\delta} \right] \geq 1 - \delta.$$

□

Corollary 7. Consider the setting, algorithm, and notation specified in Lemma 3 and Theorem 4, but with costs rescaled by a factor $\lambda > 0$: both the acquisition values and the stopping conditions are computed using $\lambda c(\cdot)$. For some budget $B > C$, if the cost-scaling factor is set to $\lambda = \frac{U}{B-C}$, then the algorithm's expected cumulative unscaled cost satisfies $\mathbb{E} [\sum_{t=1}^{\tau} c(x_t)] \leq B$.

Proof. Since the Bayesian optimization algorithm considers the post-scaling cost, by Theorem 4 and the fact that $y_{1:\tau}^* - \min_{x \in X} f(x) > 0$ pointwise, we have

$$\mathbb{E} \left[\sum_{t=1}^{\tau} \lambda \cdot c(x_t) \right] \leq \mathbb{E} \left[y_{1:\tau}^* - \min_{x \in X} f(x) + \sum_{t=1}^{\tau} \lambda \cdot c(x_t) \right] \leq \mu(x_1) - \mathbb{E} \left[\min_{x \in X} f(x) \right] + \lambda C = U + \lambda C. \quad (\text{B.17})$$

Since $\lambda = U/(B - C)$, we have

$$\mathbb{E} \left[\sum_{t=1}^{\tau} c(x_t) \right] \leq \frac{U + \lambda C}{\lambda} = \frac{U}{\lambda} + C = B. \quad (\text{B.18})$$

□

Remark 37. *In the Bayesian regret setting where f is drawn from a Gaussian process, i.e., $f \sim \mathcal{GP}(\mu(\cdot), K)$, the term $U = \mu(x_1) - \mathbb{E}[\min_{x \in X} f(x)]$ can be further bounded above using classical results on the expected supremum/infimum of Gaussian processes; see, for example, Lifshits (2012, Theorem 10.1). In many practical settings, the objective is naturally bounded. For example, in our AutoML experiments the objective is accuracy, which lies in $[0, 100]$. This implies*

$$U = \mu(x_1) - \mathbb{E} \left[\min_{x \in X} f(x) \right] \leq 100, \quad (\text{B.19})$$

providing a simple and interpretable upper bound on U .

B.2 Experimental Setup

All experiments are implemented based on BoTorch (Balandat et al., 2020). Each Bayesian optimization procedure is initialized with $2(d + 1)$ random samples, where d is the dimension of the search domain. For Bayesian regret experiments, we follow the standard practice to generate the initial random samples using a quasirandom Sobol sequence. For empirical experiments, we randomly sample configuration IDs from a fixed pool of candidates—2,000 for LCBench and 32,768 for NATS-Bench. All computations are performed on CPU.

Each experiment is repeated with 50 random seeds, and we report the mean with error bars, given by two times the standard error, for each stopping rule. We also impose a cap on the number of iterations: 100 for 1D Bayesian regret, 500 for

8D Bayesian regret, and 200 for empirical experiments. If a stopping rule is not triggered before reaching this cap, we treat the stopping time as equal to the cap.

Gaussian process models. For all experiments, we follow the standard practice to apply Matérn kernels with smoothness $5/2$ and length scales learned from data via maximum marginal likelihood optimization, and standardize input variables to be in $[0, 1]^d$. For empirical experiments, we standardize output variables to be zero-mean and unit-variance, but not for Bayesian regret experiments. In this work, we consider the noiseless setting and set the fixed noise level to be 10^{-6} . In the unknown-cost experiments, we follow Astudillo et al. (2021) to model the objective and the logarithm of the cost function using independent Gaussian processes.

Acquisition function optimization. For the 1D Bayesian regret experiments, we optimize over 10,001 grid points. For the 8D Bayesian regret experiment, we use BoTorch’s ‘gen_candidates_torch’, a gradient-based optimizer for continuous acquisition function maximization, as it avoids reproducibility issues caused by internal randomness in the default scipy optimizer. For the empirical experiments, since LCBench and NATS-Bench provide only 2,000 and 32,768 configurations respectively, we optimize the acquisition function by simply applying an argmin/argmax over the acquisition values of the unevaluated configurations, without using any gradient-based methods.

Acquisition function and stopping rule parameters. For PBGI, we follow Xie et al. (2024) to compute the Gittins indices using 100 iterations of bisection search without any early stopping or other performance and reliability optimizations.

For UCB/LCB-based acquisition functions and stopping rules, we follow the

original GP-UCB paper Srinivas et al. (2009) and the choice in UCB/LCB based stopping rules (Makarova et al., 2022; Ishibashi et al., 2023) to use the schedule $\beta_t = 2\log(dt^2\pi^2/6\delta)$, where d is the dimension. We also adopt their choice of $\delta = 10^{-1}$ and a scale-down factor of 5.

For PRB, we follow Wilson (2024) to use the schedule $N_t = \max(\lceil 64 * 1.5^{t-1} \rceil, 1000)$ for number of posterior samples, risk tolerance $\delta = 0.05$. The error bound ϵ is set to be 0.1 for Bayesian regret experiments and 0.5% of the best test error (here, the misclassification rate) among all configurations for the empirical experiments.

For the 8D Bayesian regret experiments, for all acquisition-value-based stopping rules, we apply moving average over 20 iterations to mitigate the fluctuations due to the imperfect acquisition function optimization. Figure B.1 illustrates the challenges these oscillations pose when computing stopping rule statistics and shows the improvement with moving average. For consistency, we also apply the 20-iteration averaging to the non-acquisition-value-based GSS and Convergence baselines.

Omitted baselines. We omit the KG acquisition function and stopping rule due to its high computational cost, as they are shown to be computationally intensive in the runtime experiments of Xie et al. (2024).

Objective functions: Bayesian regret. In all Bayesian regret experiments, each objective function f is sampled from a Gaussian process prior with a Matérn 5/2 kernel and a length scale of 0.1, using a different seed in each of the 50 trials.

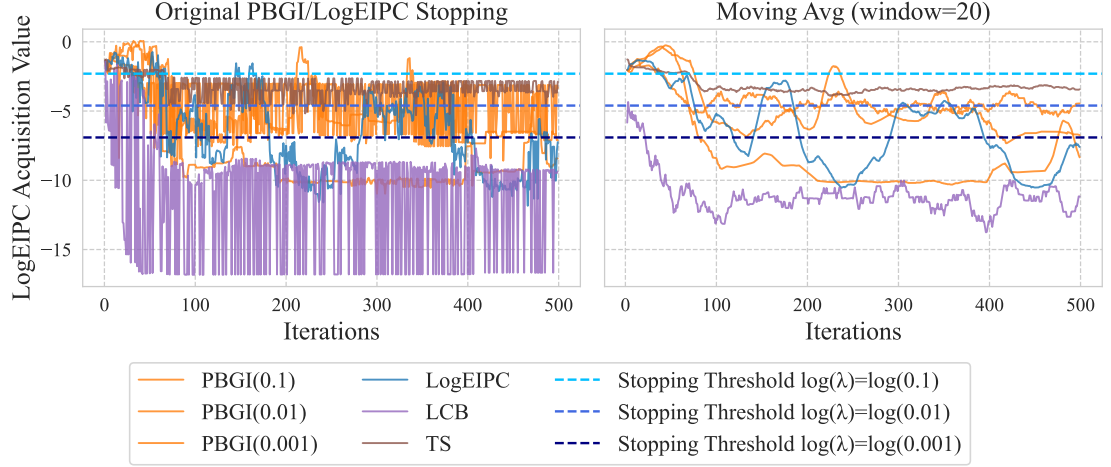


Figure B.1: Comparison of the raw and moving-averaged PBGI/LogEIPC stopping rule signals (i.e., the LogEIPC acquisition values) in Bayesian regret 8D experiments for multiple acquisition functions, with linear cost and stopping thresholds at $\log(0.1)$, $\log(0.01)$ and $\log(0.001)$. *Left*: The unaveraged signals exhibit large, high-frequency fluctuations due to the difficulty of acquisition optimization in high dimensions. *Right*: Applying moving average (window=20) smooths these wiggles, yielding more stable stopping signals.

Objective functions: empirical. In the empirical experiments, the validation error (scaled out of 100) is used as the objective function during the Bayesian optimization procedure, while the cost-adjusted simple regret is reported based on the corresponding test error.

Cost functions: Bayesian regret. In Bayesian regret experiments, we consider three types of evaluation costs: uniform, linear, and periodic. These costs are normalized such that $\mathbb{E}_{x \in [0,1]^d} [c(x)]$ is approximately 1 and their expressions (prior to cost scaling) are given below.

In the *uniform* cost setting, each evaluation incurs a constant cost of 1.

In the *linear* cost setting, the cost increases proportionally with the average

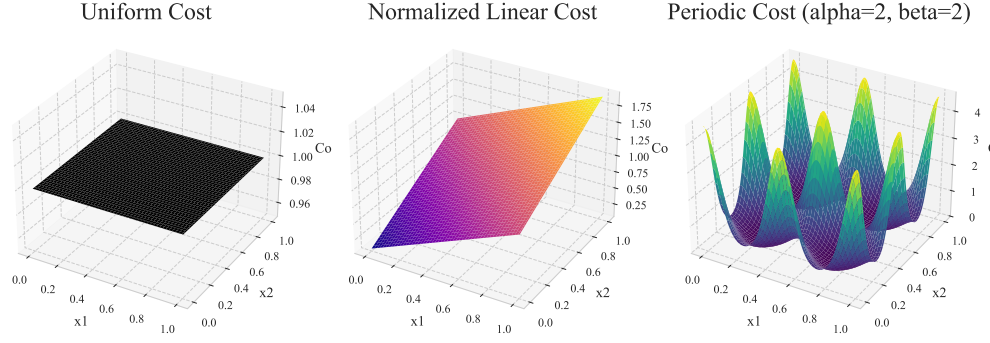


Figure B.2: Surface plots of cost functions over $[0, 1]^2$: (Left) Uniform cost of 1 across domain. (Middle) Normalized linear cost increasing with the mean of x_1 and x_2 . (Right) Periodic cost with $\alpha = 2$, $\beta = 2$, normalized by Bessel-based factor.

coordinate value of the input:

$$\text{linear_cost}(x) = \frac{1 + 20 \cdot \left(\frac{1}{d} \sum_{i=1}^d x_i \right)}{11}. \quad (\text{B.20})$$

In the *periodic* cost setting, the evaluation cost fluctuates across the domain.

Following (Astudillo et al., 2021), we define the periodic cost as

$$\text{periodic_cost}(x) = \frac{\exp \left(\frac{\alpha}{d} \sum_{i=1}^d \cos(2\pi\beta(x_i - x_i^*)) \right)}{[I_0(\frac{\alpha}{d})]^d}, \quad (\text{B.21})$$

where x_i^* denotes the coordinate of the global optimum of f , and I_0 is the modified Bessel function of the first kind, which acts as a normalization constant. We set $\alpha = 2$ and $\beta = 2$ to induce noticeable variation in cost across the domain, while ensuring that costly evaluations can still be worthwhile.

A visualization of the three cost functions is provided in Figure B.2.

Cost functions: empirical. In the *unknown-cost* experiments, we treat runtime—meaning, the provided full model training time (200 epochs for LCBench and 90 epochs for NATS)—as evaluation costs (prior to cost scaling).

In the *known-cost* experiments, for LCBench, we estimate the runtime cost from

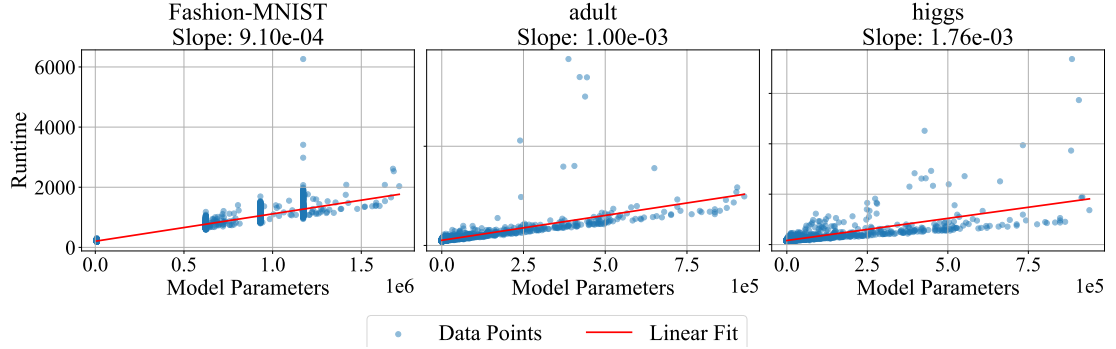


Figure B.3: Empirical relationship between the number of model parameters and runtime for three LCBench datasets. Each subplot shows a scatter plot of actual runtime (y -axis) against number of model parameters (x -axis), along with a fitted linear regression line. The observed linear trend supports using 0.001 times the number of model parameters as a proxy for runtime. For *Fashion-MNIST* and *adult*, the fitted slopes are close to 0.001. The slope for *higgs* is slightly higher, possibly due to a few outliers.

the number of model parameters p . Specifically, for the first three datasets in the six-dataset lite version, we use 0.001 times the number of model parameters as a proxy for runtime. This proxy is motivated by our observation of an approximately linear relationship between the number of model parameters and the actual runtime, with slope close to 0.001 (see Figure B.3). For the full 35-dataset version, where the slope varies across datasets, we instead use a regression-derived coefficient α and the proxy cost is αp . Importantly, the number of model parameters can be computed in advance, before the Bayesian optimization procedure, based on the network structure and classification task, as we explain in detail below.

In a feedforward neural network like `shapedmlpnet` with shape ‘funnel’, the model parameters are determined by input size (number of features), output size (e.g., number of classes), number of layers, size of each layer. The input size and output size are given by:

- Fashion-MNIST:

- Input dimension: 784 (each image has 28×28 pixels, flattened into a vector of length 784)
- Number of Output Features (output_feat): 10 (corresponding to 10 clothing categories)
- Adult:
 - Input Dimension: 14 (the dataset comprises 14 features, including both numerical and categorical attributes)
 - Number of Output Features: 1 (binary classification: income $> 50K$ or $\leq 50K$)
- Higgs:
 - Input Dimension: 28 (each instance has 28 numerical features)
 - Number of Output Features: 1 (binary classification: signal or background process)

The number of layers (num_layers) and size of each layer (max_unit) can be obtained from the configuration. With these information, we can compute the total number of model parameters (weights and biases) based on the layer-wise structure as follows:

$$\text{layer_params}_{i \rightarrow i+1} = \text{layer}_i \cdot \text{layer}_{i+1} + \text{layer}_{i+1} \quad (\text{weights} + \text{biases}) \quad (\text{B.22})$$

$$\text{model_params} = \sum_{i=0}^{L-1} \text{layer_params}_{i \rightarrow i+1} \quad (\text{B.23})$$

$$\text{where } \text{layer}_0 = \text{input_dim}, \quad \text{layer}_L = \text{output_feat} \quad (\text{B.24})$$

Similarly for NATS-Bench, we use $\alpha \times F + \beta$ as a proxy for runtime (see Figure B.4 for a visualization of the linear relationship), where F is the number

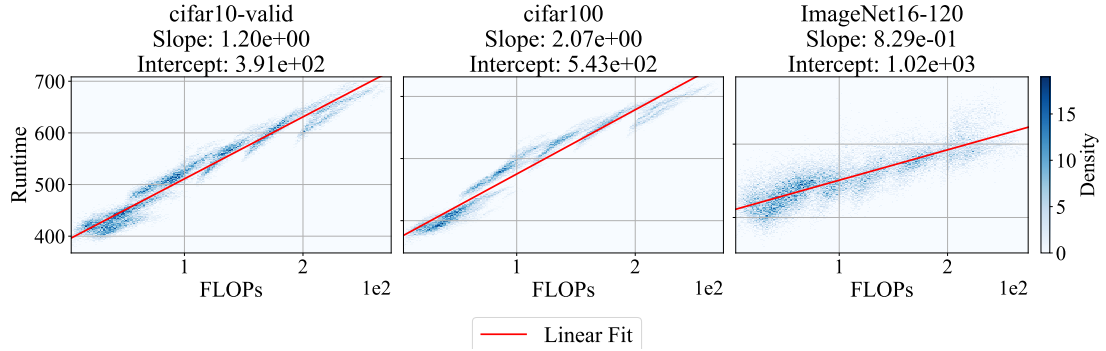


Figure B.4: Empirical relationship between the number of FLOPs and runtime for the three NATS-Bench datasets. Each subplot shows a heatmap of actual runtime (y -axis) against number of FLOPs (x -axis), along with a fitted linear regression line. The observed linear trend supports using $\alpha \times \text{\#FLOPs} + \beta$ as a proxy for runtime.

of floating point operations (FLOPs). Specifically, for *cifar10-valid*, we set $\alpha = 1$, $\beta = 400$; for *cifar100*, we set $\alpha = 2$, $\beta = 550$; and for *ImageNet16-120*, we set $\alpha = 1$, $\beta = 1000$.

FLOPs can also be computed in advance, as it is determined solely by the architecture’s structure and the fixed input shape. Specifically, they are precomputed and stored for each architecture. Since each architecture corresponds to a deterministic computational graph and all inputs (e.g., CIFAR-10 images) have a fixed shape, the FLOPs required for a forward pass can be calculated analytically—without executing the model on data.

B.3 Additional Experiment Results

In this section, we present additional experimental results to further evaluate the performance of different acquisition-function–stopping-rule pairs across various settings. We also include alternative visualizations to aid interpretation of the results.

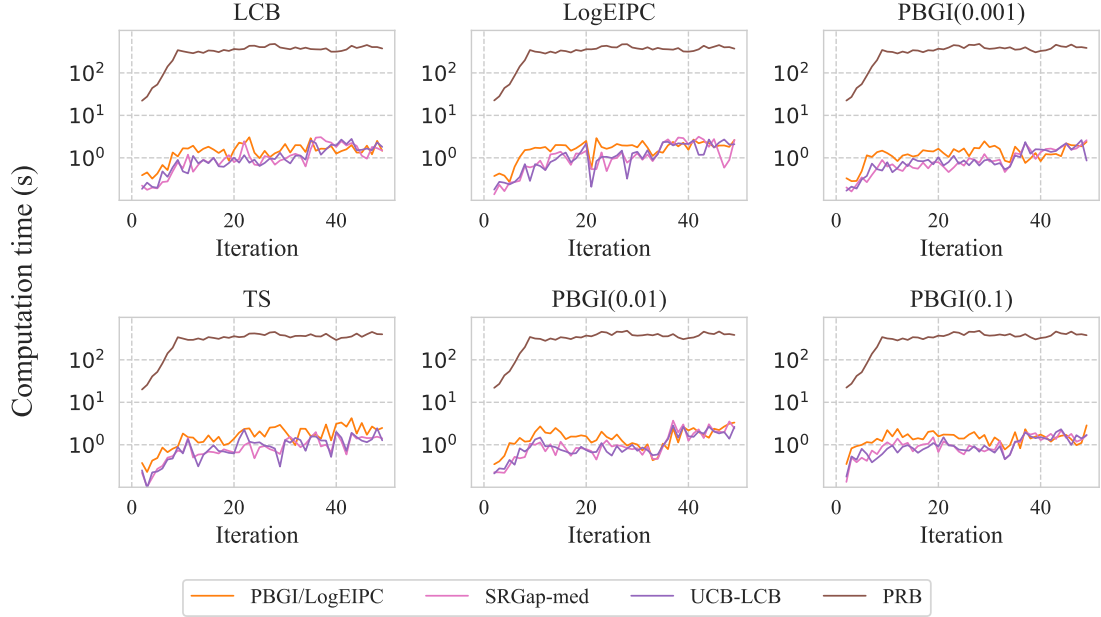


Figure B.5: Evolution of per-iteration computation time (in log scale) for different stopping rules when paired with six acquisition policies on the 8-dimensional Bayesian regret benchmark. Each subplot shows the average runtime (in seconds) over 50 iterations under one acquisition function—LCB, Thompson Sampling, LogEIPC, PBGI($\lambda = 10^{-1}$), PBGI($\lambda = 10^{-2}$), and PBGI($\lambda = 10^{-3}$). Curves correspond to four stopping criteria: our PBGI/LogEIPC stopping rule, SRGap-med, UCB-LCB, and the probabilistic regret bound (PRB). Convergence and GSS can be applied using only the best observed value and thus require no additional computation time, thus they are omitted here. LogEIPC-med relies on the same underlying statistical computations as the LogEIPC rule, and therefore its runtime is not measured separately. The results should that PRB incurs significant computational overhead compared to other stopping rules.

B.3.1 Runtime Comparison

First, we compare the runtime between our PBGI/LogEIPC stopping rule with several baselines. We measure the CPU time (in seconds) of the computation of the stopping rule, excluding the acquisition function computation and optimization.

From results in Figure B.5 we can see that our PBGI/LogEIPC stopping rule is roughly as efficient as SRGap-med and UCB-LCB. In contrast, PRB is significantly more time-consuming, as it involves optimizing over up to 1000 samples.

B.3.2 Order of Stopping and Posterior Updates

Following the discussions in Section 3.3, we always compute our proposed stopping rule with respect to the optimal acquisition function value of the *next round*—namely, the one which is obtained after posterior updates have been performed. One could alternatively consider checking the stopping criteria *before* posterior updates, which is backward-looking rather than forward-looking. Figure B.6 provides an empirical comparison between the two choices, showing that stopping *after* the posterior update leads to stronger empirical performance.

This suggests that the theoretical guarantee for the Gittins index policy in the correlated Pandora’s Box setting by Gergatsouli and Tzamos (2023), which is based on the *before-posterior-update* stopping, could potentially be improved by adopting the *after-posterior-update* stopping.

B.3.3 Additional Experiment Results: Empirical

This subsection presents additional results for hyperparameter optimization on the LCBench datasets and neural architecture–size search on the three NATS-Bench datasets. For LCBench, we provide results for the first three datasets from the six-dataset lite version of the benchmark: Fashion-MNIST, adult, and higgs under alternative settings (fixed budget, actual runtime, and unknown cost), along with an alternative visualization. We also include per-dataset results under the proxy-runtime setting for the full set of 35 datasets.

Simple regret under the fixed-budget setting. To isolate the effect of the acquisition function on cost-adjusted regret, we report the simple regret of several

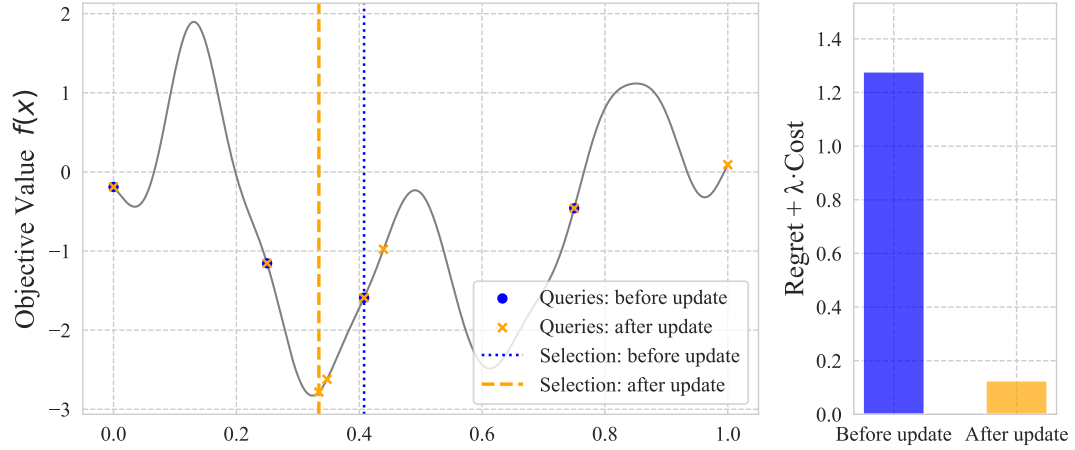


Figure B.6: Illustration of a single draw from a Matérn-5/2 Gaussian process on $[0, 1]$ with lengthscale 0.1, optimized using PBGI acquisition function under uniform cost and cost-scaling factor $\lambda = 0.01$. We compare two variants of PBGI stopping rules: the *before-posterior-update* (this-round) stopping rule and the *after-posterior-update* (next-round) stopping rule. **Left:** The latent objective function (solid gray) and evaluation sequences for *before-posterior-update* stopping (blue circles) and *after-posterior-update* stopping (orange crosses). The dotted blue line and the dashed orange line mark the best observed value under each respective rule. **Right:** Cost-adjusted regret for each stopping rule. In this example, *after-posterior-update* stopping achieves strictly lower cost-adjusted regret despite performing more evaluations.

acquisition functions in the fixed-budget setting. We compare LogEIPC, PBGI, LCB, and TS, and additionally include PBGI-D with our recommended choice $\lambda_0 = U/(B - C)$ from Section 3.3.1, where $U = 50$ (reflecting the $[0, 100]$ range of classification accuracy) and $B - C = 10,000$ (corresponding to a budget after initial evaluation of 10,000 seconds, or roughly 3 hours). Cost-aware methods (LogEIPC and the PBGI variants) outperform cost-unaware ones (UCB and TS), with PBGI at smaller λ consistently better than LogEIPC. This mirrors the findings of Xie et al. (2024) and helps explain the strong performance of the matched PBGI combination in our main experiments.

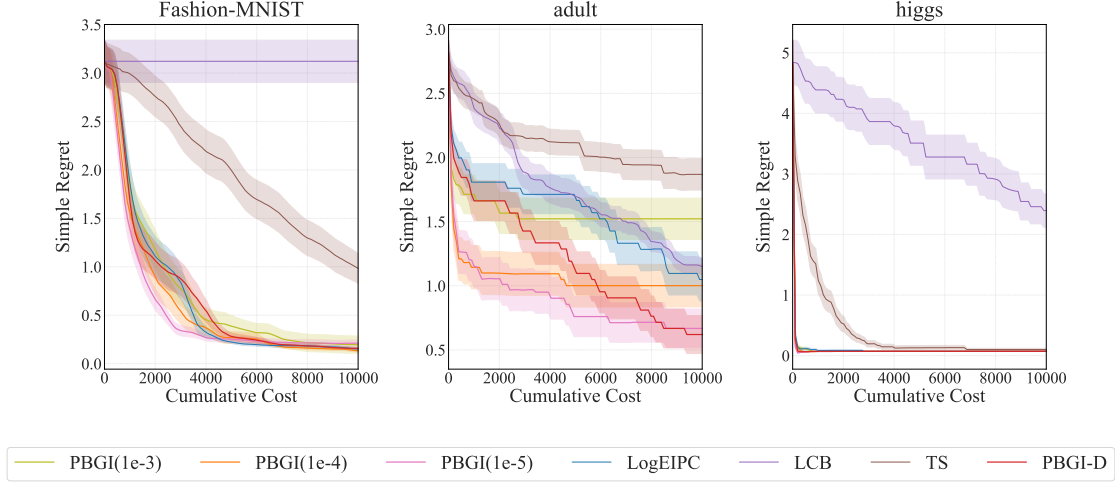


Figure B.7: Comparison of simple regret of five acquisition functions: LogEIPC, PBGI($\lambda = 10^{-4}$), LCB, TS, and PBGI-D on LCBench datasets, using scaled proxy runtime as evaluation cost. We can see that indeed the cost-aware LogEIPC and PBGI outperforms the cost-unaware UCB and TS. PBGI-D with out recommended λ_0 is also competitive.

Number of trials where stopping fails. We count the number of trials in which a stopping rule fails to trigger within our iteration cap of 200 and present the results in Table B.1. From the table, we observe that on datasets from the NATS benchmark, regret-based and acquisition-based stopping rules—except for ours—often fail to stop early. On LCBench datasets, some regret-based stopping rules such as SRGap-med and UCB-LCB also frequently exceed the cap. In contrast, our stopping rule consistently stops early, which aligns with our theoretical result in Corollary 7.

Alternative visualization: cost-adjusted simple regret vs iteration. We provide an alternative visualization of cost-adjusted simple regret by plotting its mean and error bars at fixed iterations, along with the mean and error bars of the stopping iterations for each rule. This allows us to compare the performance of adaptive stopping rules not only against the hindsight-optimal adaptive stopping

Table B.1: Number of trials (out of 50) where each stopping rule failed to trigger within 200 iterations, for each dataset in the LCBench (first three) and NATS (last three) benchmarks and each acquisition function. Results are identical across acquisition functions.

Dataset	PBGI	LogEIPC-med	SRGap-med	UCB-LCB	GSS	Convergence	PRB
Fashion-MNIST	0	0	0	50	0	0	0
adult	5	0	7	38	0	0	6
higgs	0	0	31	50	0	0	0
Cifar10	0	32	50	50	0	0	26
Cifar100	3	17	50	50	0	0	2
ImageNet	0	23	50	50	0	0	0

but also against the hindsight-optimal fixed-iteration stopping.

As shown in the empirical setting in Figures B.8 to B.13, cost-adjusted regret generally decreases in the early iterations and then increases. The turning point is exactly the hindsight-optimal fixed-iteration stopping point, and our PBGI/LogEIPC stopping rule consistently performs close to this optimum, particularly when paired with the PBGI acquisition function.

Cost model misspecification: proxy runtime vs actual runtime. In the known-cost setting of hyperparameter tuning, a practical approach is to use a proxy for runtime as the evaluation cost during the Bayesian optimization procedure. In our case, we use the number of model parameters scaled by a constant factor, which can be known in advance and has been shown to correlate well with the actual runtime. However, for reporting performance, one may prefer to use the actual runtime to better reflect real-world cost. To assess the impact of this cost model misspecification, we compare the cost-adjusted simple regret obtained when evaluation costs are computed using either the proxy runtime or the actual runtime. As shown in Figure B.14, our PBGI/LogEIPC stopping rule remains close to the

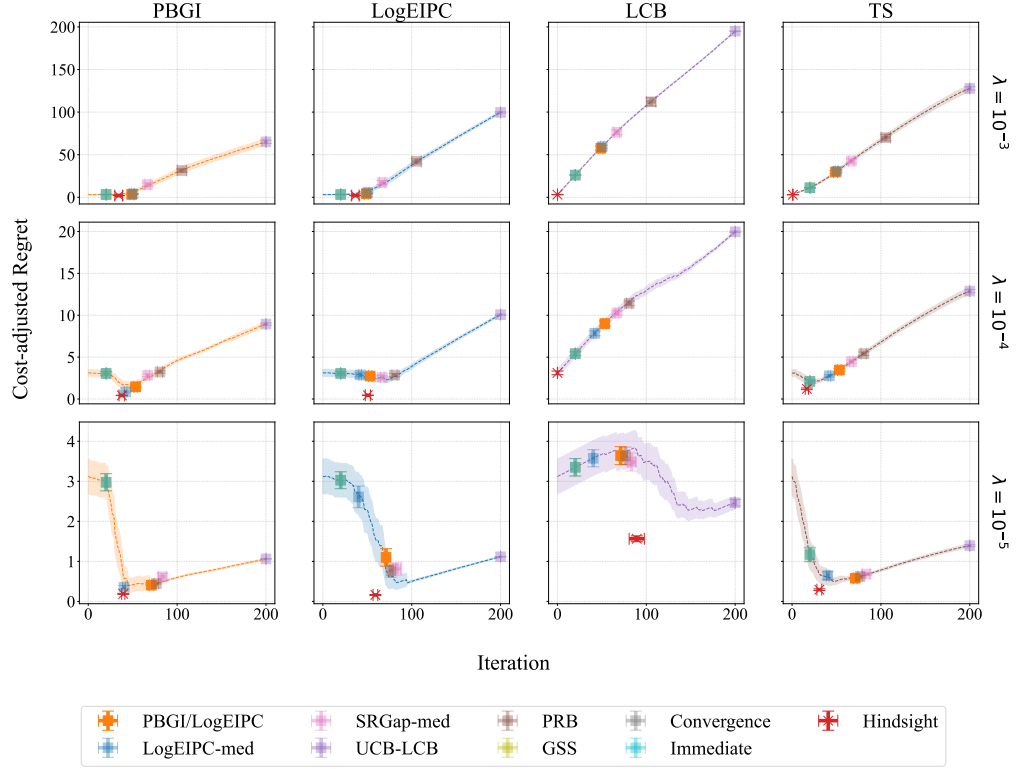


Figure B.8: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the *Fashion-MNIST* dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-3}, 10^{-4}, 10^{-5}$. We can see that the PBGI/LogEIPC stopping rule consistently achieves cost-adjusted regret close to the hindsight optimal adaptive stopping (shown by the red marker) as well as the hindsight optimal fixed-iteration stopping (given by the minimum point along the fixed-budget curve) when paired with the PBGI acquisition function, though not always the best.

hindsight optimal even when there is a misspecification, although its ranking may shift slightly (e.g., from best to second-best on the *higgs* dataset).

Unknown-cost. Astudillo et al. (2021, Proposition 2) proposed modeling unknown cost $c(x)$ via

$$\mathbb{E}[1/c(x)]^{-1} = \exp(\mu_{\ln c}(x) - (\sigma_{\ln c}(x))^2/2). \quad (\text{B.25})$$

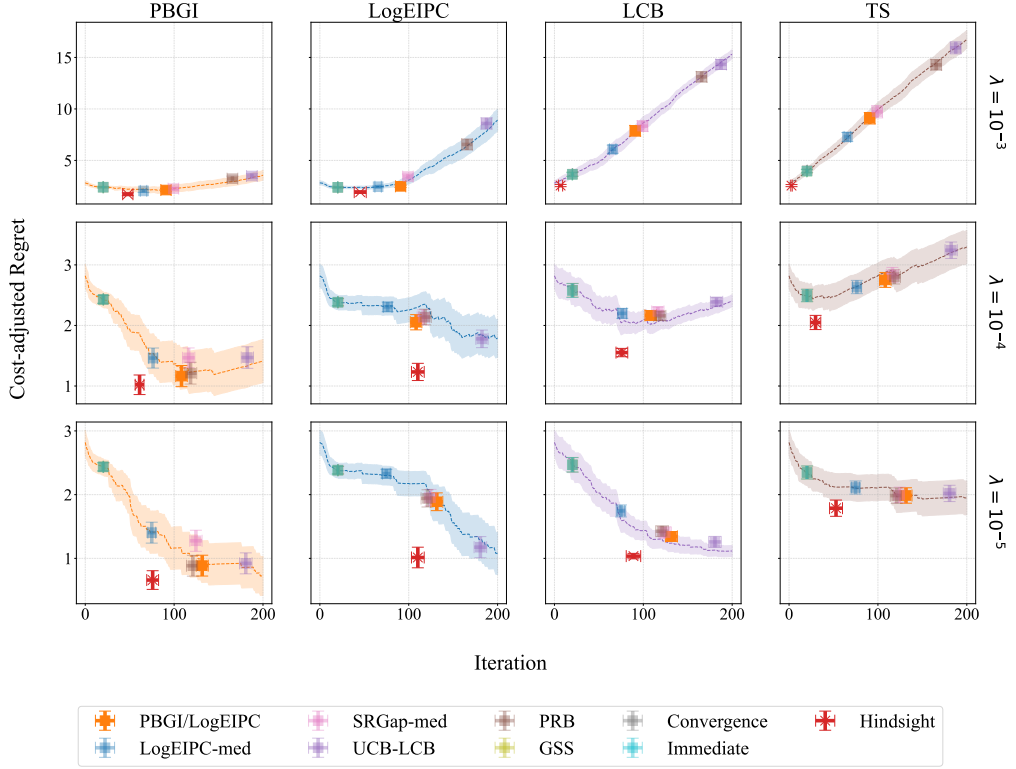


Figure B.9: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the *adult* dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-3}, 10^{-4}, 10^{-5}$. We can see that the PBGI/LogEIPC stopping rule consistently achieves the cost-adjusted regret close to hindsight optimal adaptive stopping (shown by the red marker) as well as the hindsight optimal fixed-iteration stopping (given by the minimum point along the fixed-budget curve) when paired with the PBGI or TS acquisition function, particularly with PBGI.

An alternative is

$$\mathbb{E}[c(x)] = \exp(\mu_{\ln c}(x) + (\sigma_{\ln c}(x))^2/2). \quad (\text{B.26})$$

The difference in sign before the variance term reflects how each formulation handles predictive uncertainty: (B.25) encourages more exploration than (B.26). For PBGI under the unknown-cost setting, it is more natural to replace $c(x)$ in (3.5) with $\mathbb{E}[c(x)]$ using (B.26), as this aligns with how costs enter the root-finding problem. For LogEIPC, both variants are possible—we refer to the (B.25) version as *LogEIPC-inv* and the (B.26) version as *LogEIPC-exp*. However, equivalence

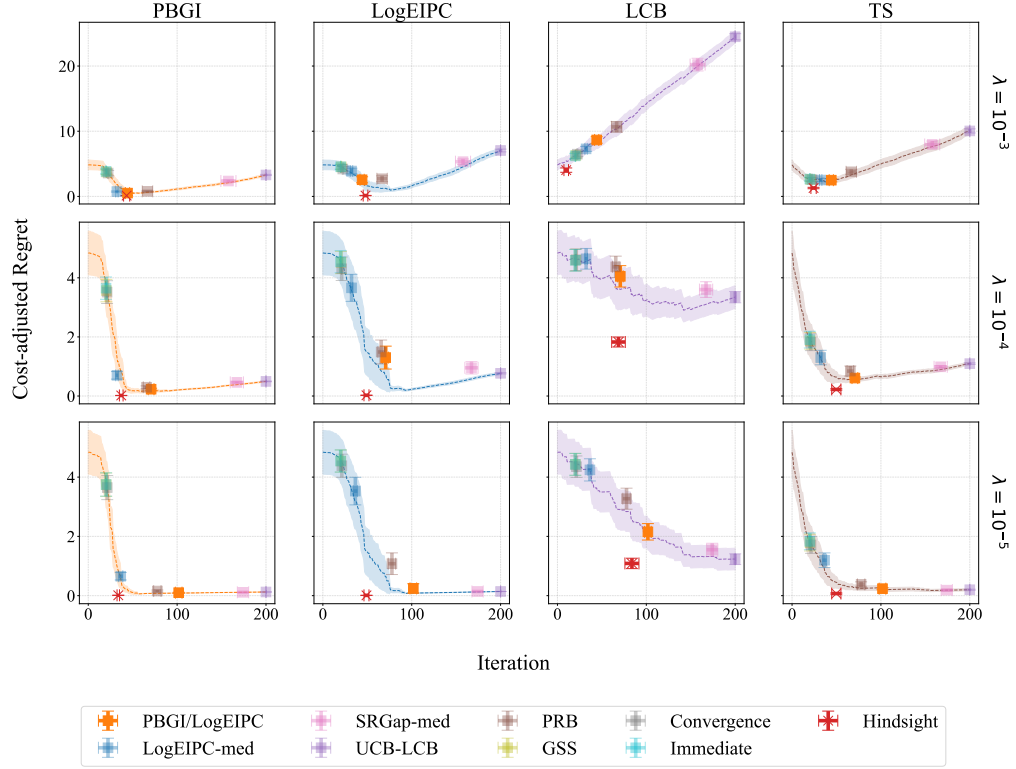


Figure B.10: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the *higgs* dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-3}, 10^{-4}, 10^{-5}$. We can see that the PBGI/LogEIPC stopping rule consistently achieves the best cost-adjusted regret when paired with the LogEIPC, PBGI, or TS acquisition function, particularly with PBGI. These pairs not only approach the hindsight optimal adaptive stopping (shown by the red marker) but also perform comparably to the hindsight optimal fixed-iteration stopping (the minimum point along the fixed-budget curve).

between PBGI and LogEIPC stopping rules and our theoretical guarantees hold only with (B.26) but not with (B.25). Accordingly, we use (B.26) for both methods in our experiments to maintain consistency and preserve this equivalence. Figure B.15 shows performance of acquisition function and stopping rule pairs under the unknown-cost setting, which are qualitatively similar to the known-cost setting.

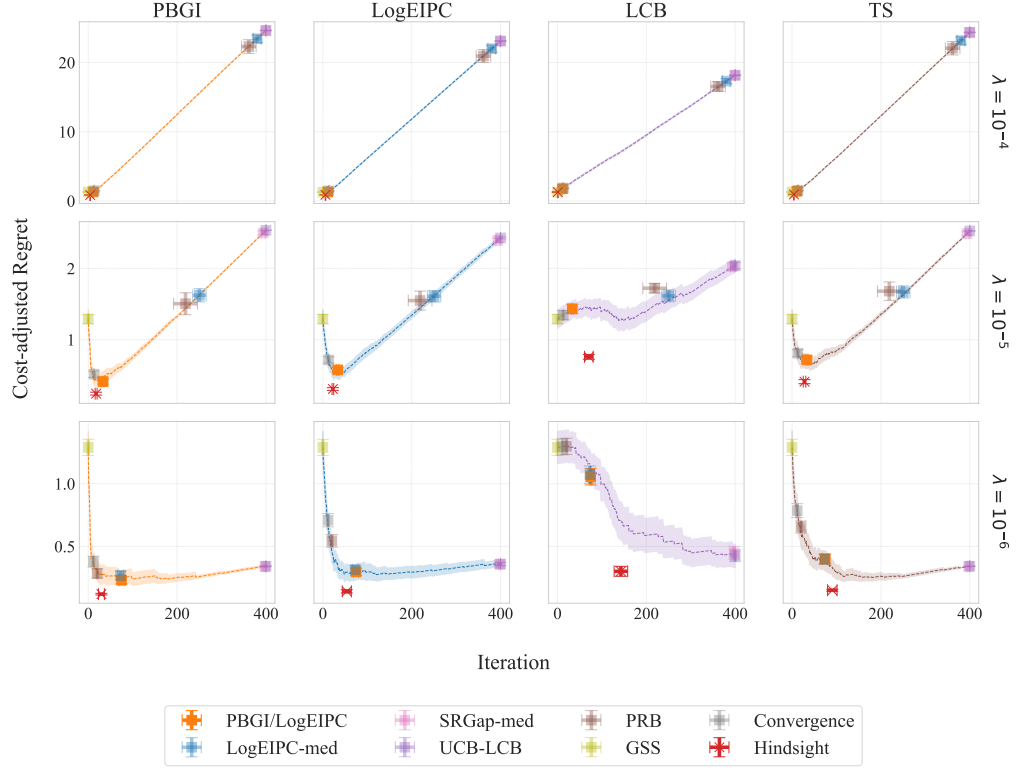


Figure B.11: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the *cifar10-valid* dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-4}, 10^{-5}, 10^{-6}$. We can see that the PBGI/LogEIPC stopping rule consistently achieves the best cost-adjusted regret when paired with the LogEIPC, PBGI, or TS acquisition function, particularly with PBGI. These pairs not only approach the hindsight optimal adaptive stopping but also perform comparably to the hindsight optimal fixed-iteration stopping.

Cost-adjusted simple regret of all LCBench datasets. Due to space constraints, for LCBench, Figure 3.3 in the main text reports min-max normalized cost-adjusted simple regret, defined as

$$\frac{r - r_{\min}}{r_{\max} - r_{\min}}, \quad (\text{B.27})$$

where r denotes cost-adjusted regret of a given acquisition function-stopping rule pair, and $r_{\min}$ and $r_{\max}$ are the minimum (including the hindsight optimal) and maximum cost-adjusted regret across all pairs. This normalization enables

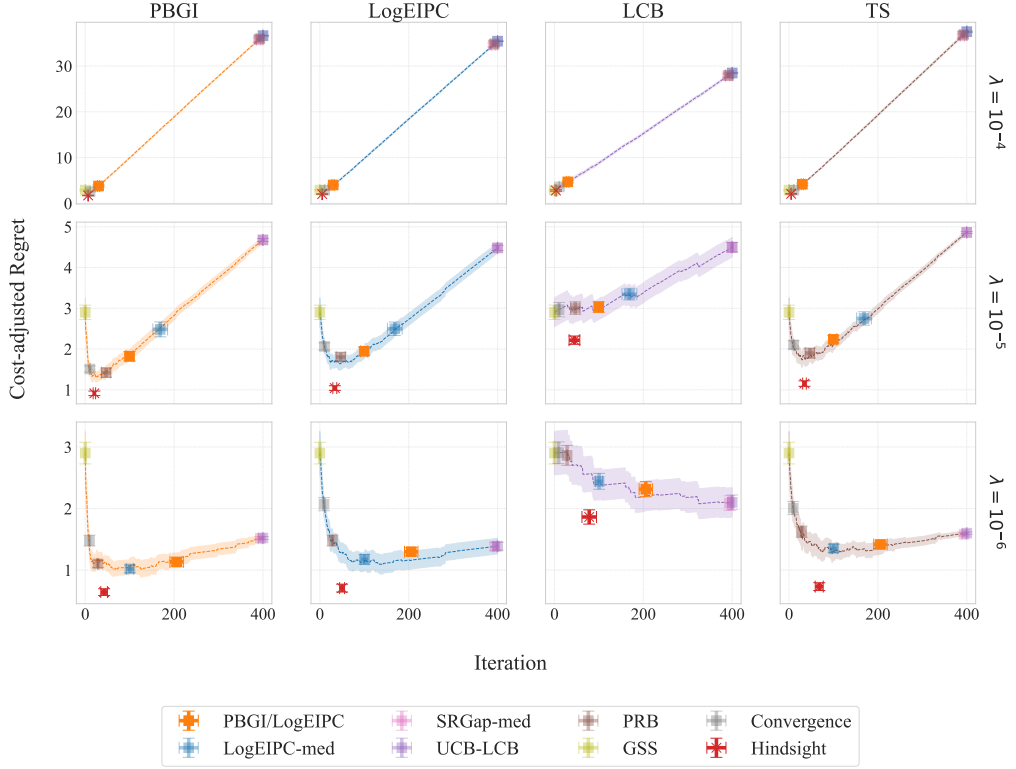


Figure B.12: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the *cifar100* dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-4}, 10^{-5}, 10^{-6}$. The PBGI/LogEIPC stopping rule remains competitive at $\lambda = 10^{-4}$ and 10^{-6} , though not always the best. At $\lambda = 10^{-5}$, unlike in most experiments, it stops noticeably late (though still outperforming several other rules), even when paired with its matching acquisition function. By Lemma 3, under model match, the PBGI/LogEIPC rule with the corresponding acquisition function should never incur negative expected cost-adjusted regret before stopping. The suboptimal behavior observed here points to significant model misspecification on the *cifar100* dataset.

aggregation across datasets with different regret scales.

Here we present the *unnormalized* bar-plot results for each LCBench dataset (OpenML dataset size in parentheses) under all three cost-scaling parameters in Figures B.16 to B.18. As discussed in the main text, our PBGI/LogEIPC stopping rule—when paired with either PBGI or LogEIPC—achieves competitive performance on roughly 75% of the 35 datasets. However, the matched PBGI pair consis-

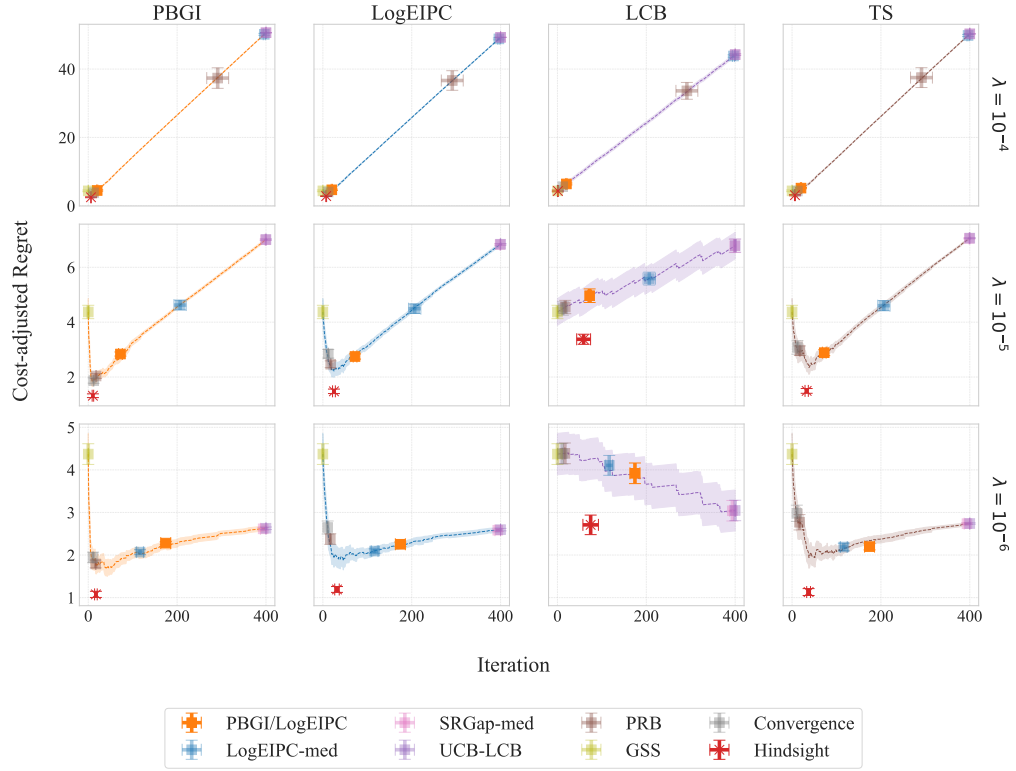


Figure B.13: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs on the *ImageNet16-120* dataset. The objective function is the validation error, and the evaluation cost is the proxy runtime, scaled by three different cost-scaling factors $\lambda = 10^{-4}, 10^{-5}, 10^{-6}$. The PBGI/LogEIPC stopping rule remains competitive at $\lambda = 10^{-4}$ and 10^{-6} , though not always the best. At $\lambda = 10^{-5}$, unlike in most experiments, it stops noticeably late (though still outperforming several other rules), even when paired with its matching acquisition function. By Lemma 3, under model match, the PBGI/LogEIPC rule with the corresponding acquisition function should never incur negative expected cost-adjusted regret before stopping. The suboptimal behavior observed here points to significant model misspecification on the *ImageNet16-120* dataset.

tently underperforms across all three λ values on the following datasets (see subplots with light grey background in Figures B.16 to B.18): Amazon_employee_access (32769), Australian (690), KDDCup09_appetency (50000), cnae-9 (1080), credit-g (1000), fabert (8237), and vehicle (846). Three additional datasets, jasmine (2984), mfeat-factors (2000) and sylvine (5124), show acceptable performance only when $\lambda = 10^{-5}$.

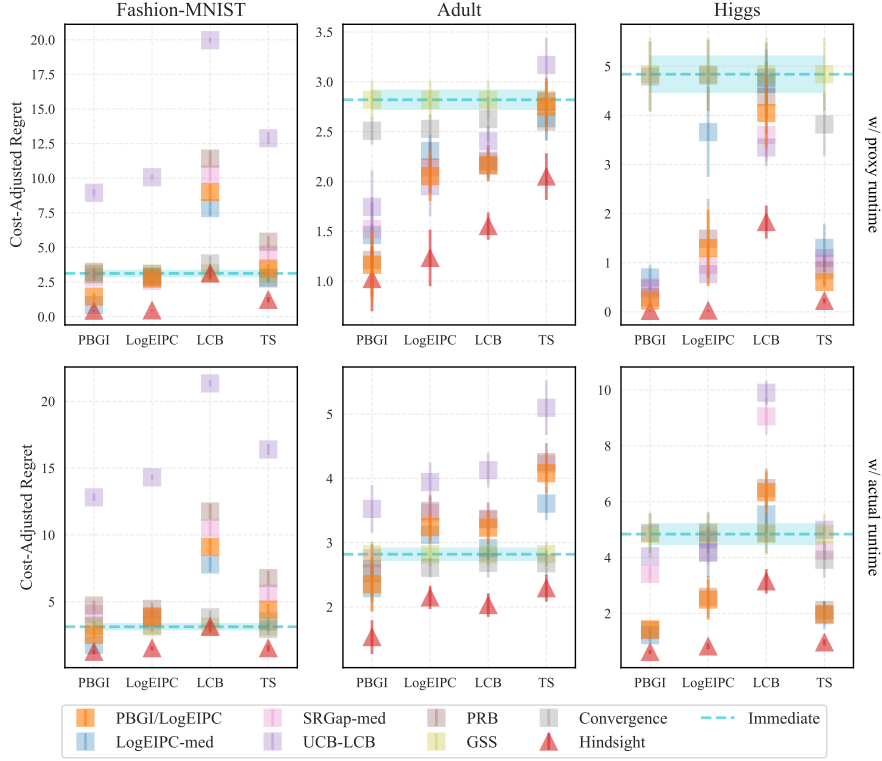


Figure B.14: Comparison of cost-adjusted simple regret on three LCBench datasets with $\lambda = 10^{-4}$, using scaled proxy runtime vs. scaled actual runtime as evaluation cost. While our PBGI/LogEIPC stopping rule performs slightly worse under actual runtime (e.g., dropping from best to second-best on the *higgs* dataset), likely due to cost model misspecification, it remains close to the hindsight optimal in all cases.

With the exception of `Amazon_employee_access` (32769) and `KDD-Cup09_appetency` (50000), all datasets on which the matched PBGI pair underperforms have fewer than 10,000 instances. This suggests that relatively small dataset size may contribute to model misspecification and degraded performance.

B.3.3.1 Diagnosing Underperforming LCBench Datasets via Validation–Test Rank Preservation

We show that the degraded performance of our stopping rule on smaller LCBench datasets is likely caused by a benchmark-intrinsic val/test mismatch rather than a deficiency of the rule itself. We classify a dataset as “underperforming” if the

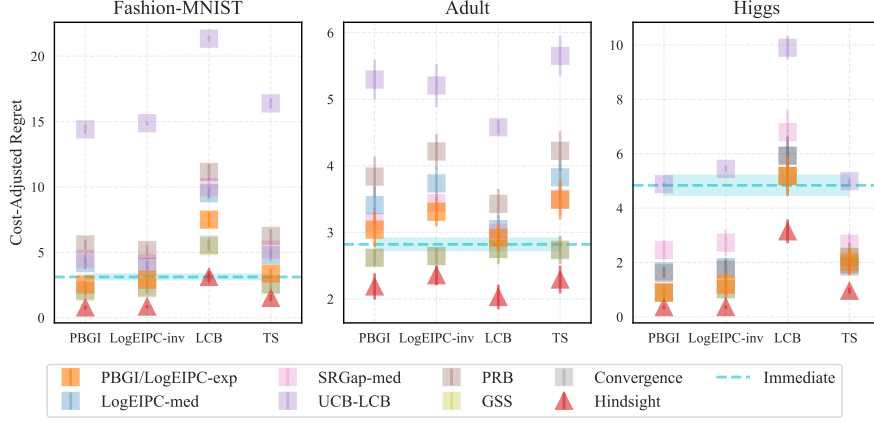


Figure B.15: Cost-adjusted simple regret across acquisition function and stopping rule pairs under the unknown-cost setting on LCBench with $\lambda = 10^{-4}$. Our PBGI/LogEIPC stopping rule remains close to the hindsight optimal when paired with the LogEIPC-exp or PBGI acquisition function, sometimes slightly worse than heuristics such as GSS and Convergence. It is also slightly worse than Immediate on *Adult*, likely due to a cost-model misspecification in the unknown-cost setting.

cost-adjusted simple regret (at $\lambda = 10^{-4}$) of the matched PBGI pair does not rank among the top three PBGI-based acquisition function–stopping rule pairs formed by the seven non-trivial stopping rules we study. Formally, let $\mathcal{S} = \{\text{PBGI/LogEIPC}, \text{LogEIPC-med}, \text{SRGap-med}, \text{UCB-LCB}, \text{PRB}, \text{GSS}, \text{Convergence}\}$ (excluding the Immediate and Hindsight baselines), and let $r_d(s) := \mathcal{R}_c^{(\lambda=10^{-4})}(\text{PBGI}, s; d)$ denote the mean cost-adjusted regret of PBGI paired with rule s on dataset d . With $\rho_d := |\{s \in \mathcal{S} : r_d(s) \leq r_d(\text{PBGI/LogEIPC})\}|$ the within-dataset rank of the matched rule (rank 1 best), we set

$$d \in \begin{cases} \mathcal{D}_{\text{top}} & \text{if } \rho_d \leq 3, \\ \mathcal{D}_{\text{under}} & \text{if } \rho_d \geq 4. \end{cases} \quad (\text{B.28})$$

The LCBench experiments optimize validation error but report test error. We quantify how faithfully validation ranks transfer to test ranks via the top- k overlap fraction. For dataset d with n_d configurations, let $\pi_d^{\text{val}}, \pi_d^{\text{test}} \in S_{n_d}$ be the rankings

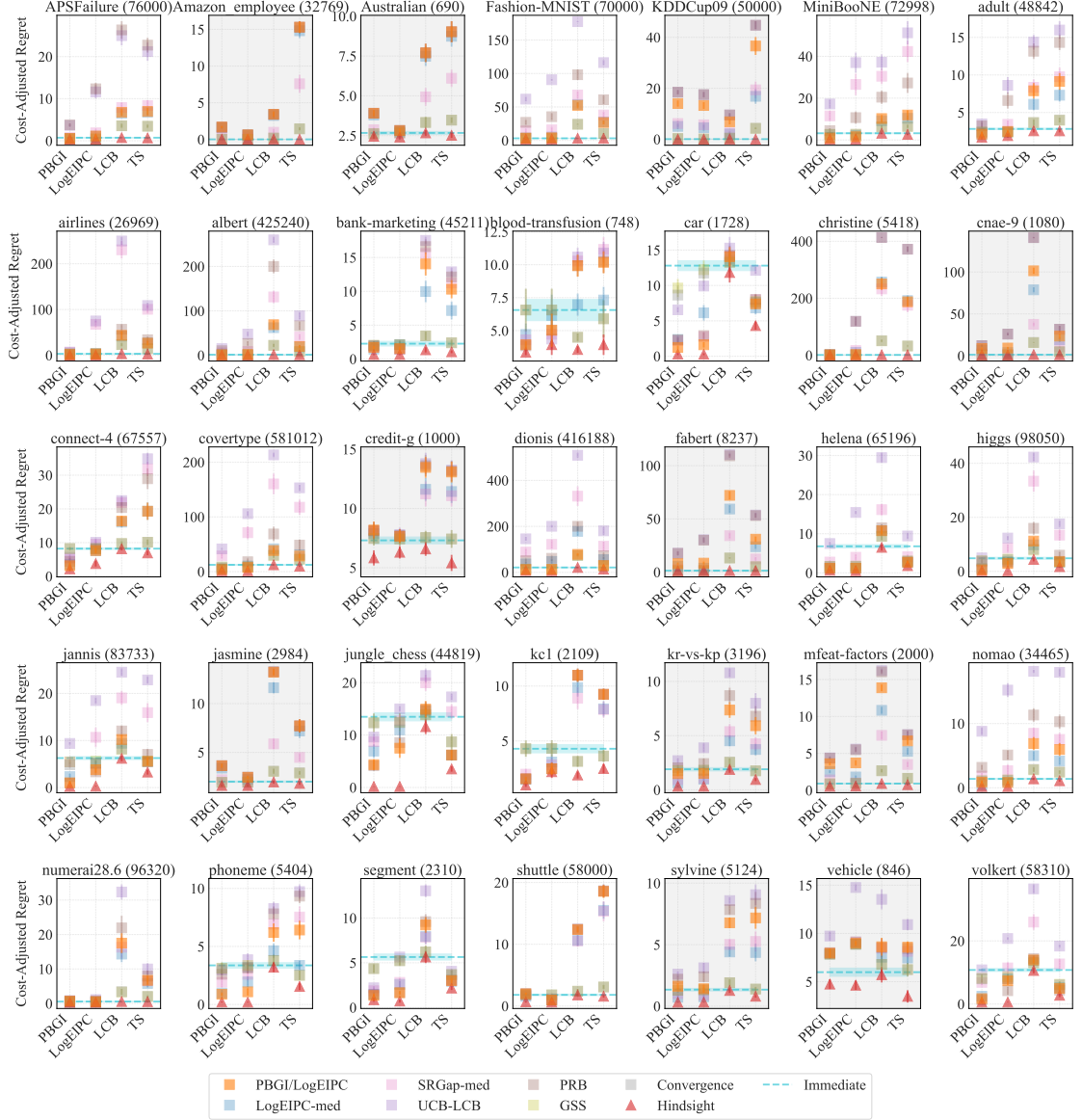


Figure B.16: Cost-adjusted simple regret of all 35 LCBench datasets when $\lambda = 10^{-3}$. Datasets where the matched PBGI pair underperforms, meaning that it does not rank among the top three acquisition function-stopping rule pairs by cost-adjusted simple regret, are highlighted in light grey.

of configurations by validation and test error (ascending, best first); then

$$\text{Top-}k(d) := \frac{1}{k} |\{\pi_d^{\text{val}}(1), \dots, \pi_d^{\text{val}}(k)\} \cap \{\pi_d^{\text{test}}(1), \dots, \pi_d^{\text{test}}(k)\}|. \quad (\text{B.29})$$

We take $k = 20$, comparable to the 200-iteration cap in our experiments and to the top tail of configurations any adaptive method realistically considers as stopping

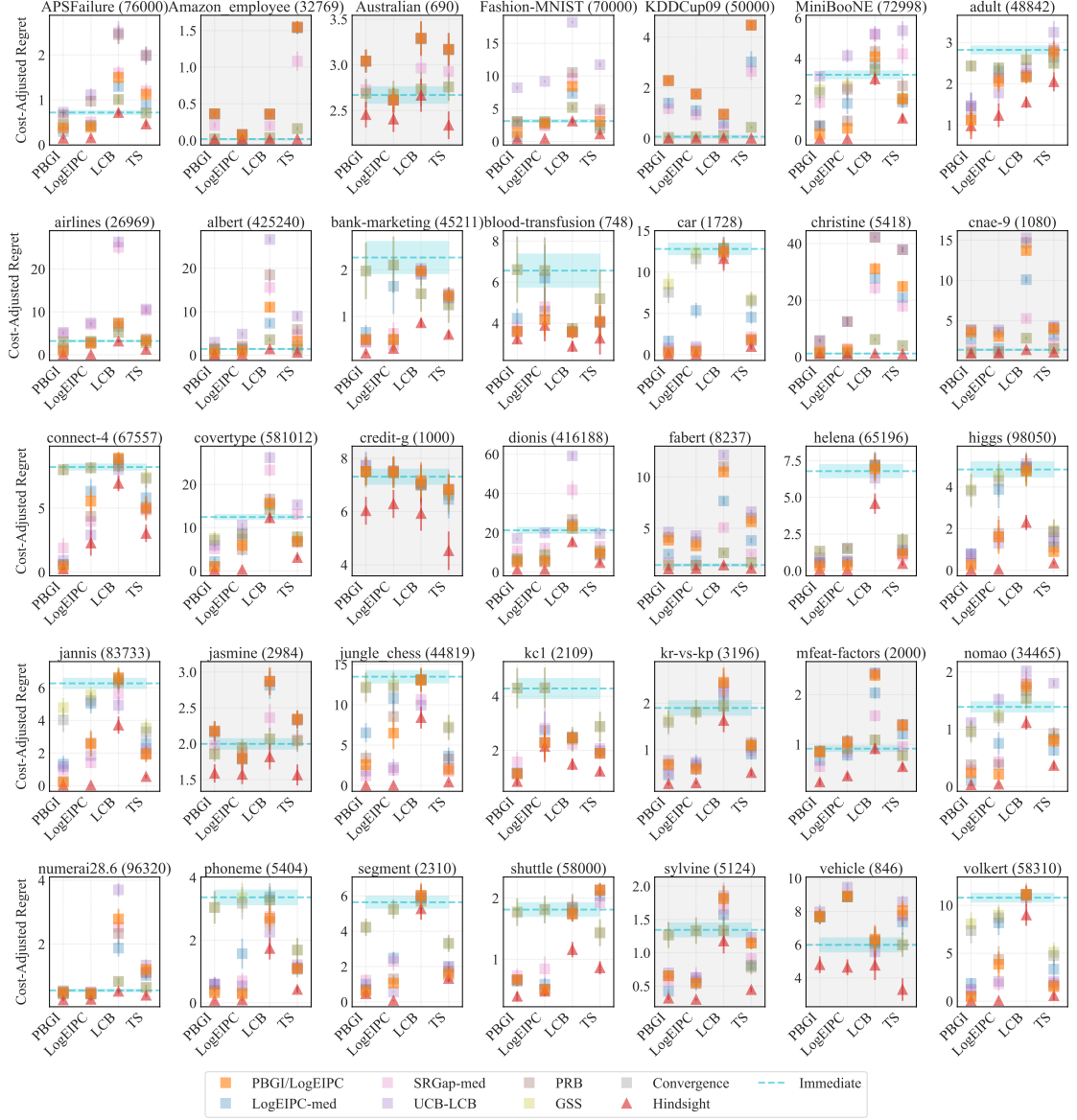


Figure B.17: Cost-adjusted simple regret of all 35 LCBench datasets when $\lambda = 10^{-4}$. Datasets where the matched PBGI pair underperforms, meaning that it does not rank among the top three acquisition function–stopping rule pairs by cost-adjusted simple regret, are highlighted in light grey.

candidates.

Figure B.19 reports $\text{Top-20}(d)$ for every LCBench dataset, color-coded by the above classification. The $\mathcal{D}_{\text{under}}$ datasets have *substantially* lower val–test rank

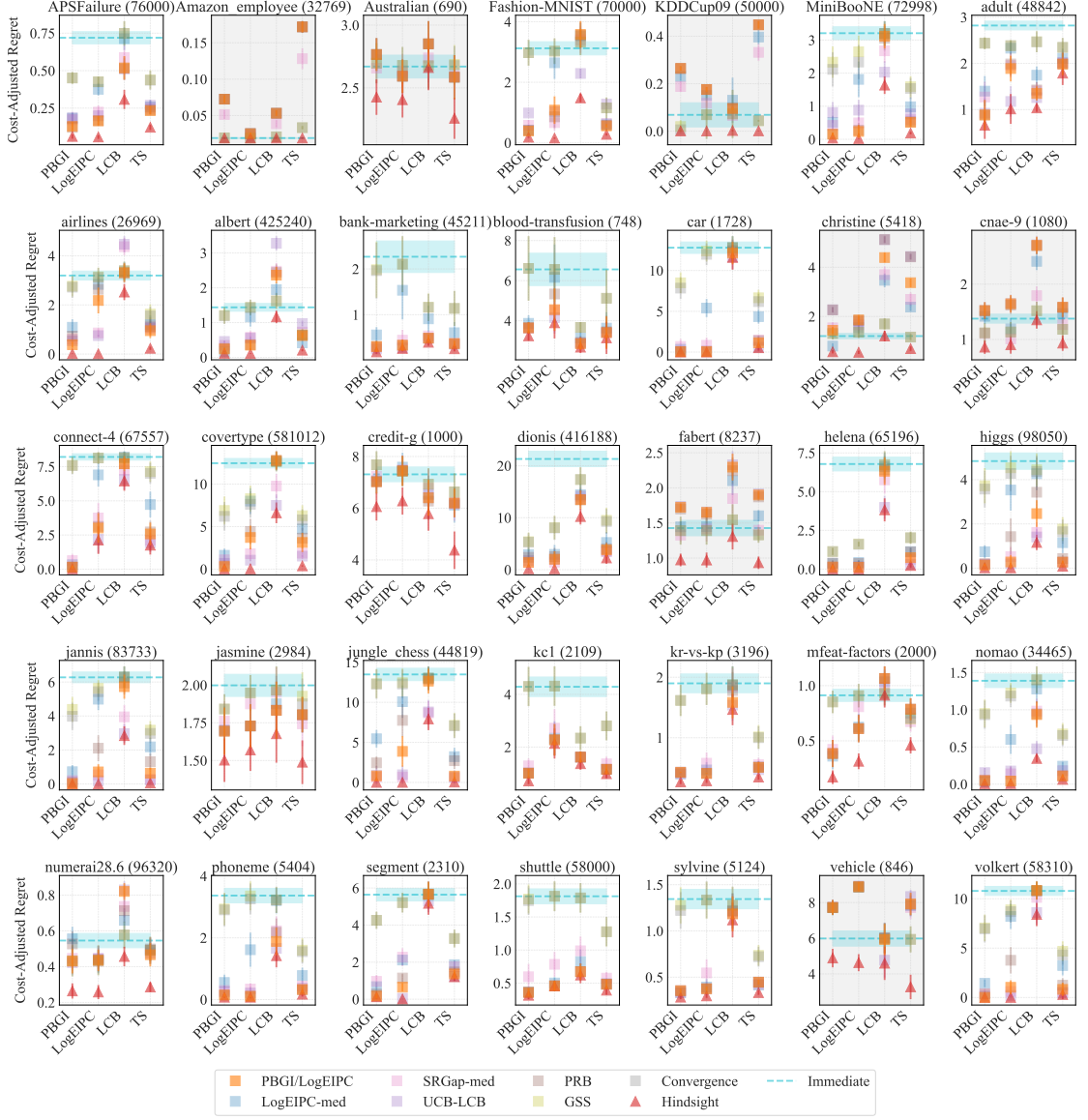


Figure B.18: Cost-adjusted simple regret of all 35 LCBench datasets when $\lambda = 10^{-5}$. Datasets where the matched PBGI pair underperforms, meaning that it does not rank among the top three acquisition function–stopping rule pairs by cost-adjusted simple regret, are highlighted in light grey.

preservation than those in $\mathcal{D}_{\text{top}}$:

$$\overline{\text{Top-20}} \Big|_{\mathcal{D}_{\text{under}}} = \mathbf{36.1\%} \ (n = 14, \text{std} = 35.7\%) \ll \overline{\text{Top-20}} \Big|_{\mathcal{D}_{\text{top}}} = \mathbf{83.6\%} \ (n = 21, \text{std} = 23.8\%)$$

When the validation-selected top- k shares little overlap with the test top- k , no validation-driven stopping rule can recover test-optimal configurations, so the

matched PBGI pairing’s apparent underperformance is best attributed to the benchmark’s intrinsic val/test mismatch rather than to the stopping rule.

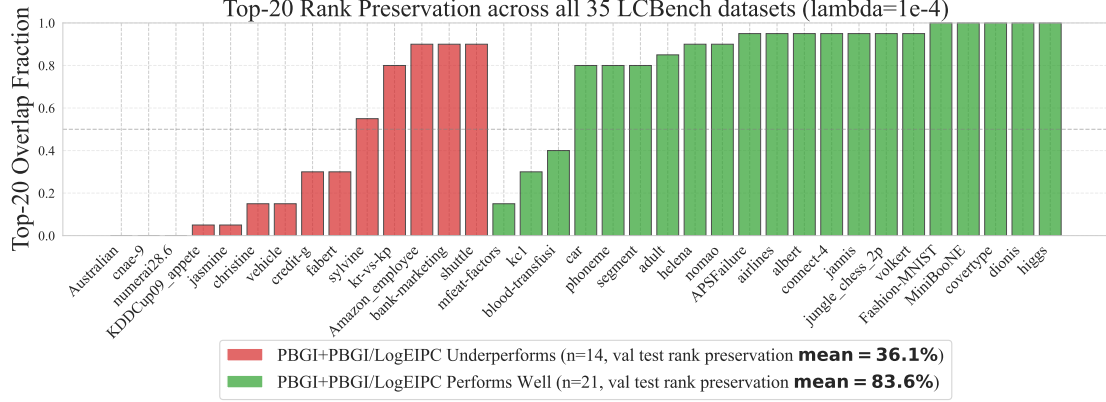


Figure B.19: Top-20 validation–test rank preservation for every LCBench dataset. Datasets are split into $\mathcal{D}_{\text{under}}$ (red) and $\mathcal{D}_{\text{top}}$ (green) based on whether the matched PBGI pair ranks among the top three PBGI-based acquisition function–stopping rule pairs formed by the seven stopping rules in $\mathcal{S}$ at $\lambda = 10^{-4}$. The dashed line marks Top-20 = 0.5. Datasets where the matched PBGI pair underperforms also have markedly lower validation-to-test rank preservation, indicating that their weaker cost-adjusted regret reflects a val/test mismatch in the benchmark rather than a failure mode of the stopping rule itself.

B.3.4 Additional Experiment Results: Bayesian Regret

In this section, we present the complete Bayesian regret results. Figures B.20 to B.22 show the 1D experiments, and Figures B.23 to B.25 show the 8D experiments. Each figure corresponds to one cost setting (uniform, linear or periodic) and three values of the cost-scaling factor, $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$. In all of the experimental results, we observe that PBGI/LogEIPC acquisition function + PBGI/LogEIPC stopping achieves cost-adjusted regret that is not only competitive with the baselines, but is also competitive regarding the *best in hindsight* fixed iteration stopping and often competitive even comparing to *hindsight optimal* stopping. These results indicate that our automatic stopping rule can replace manual selection of stopping

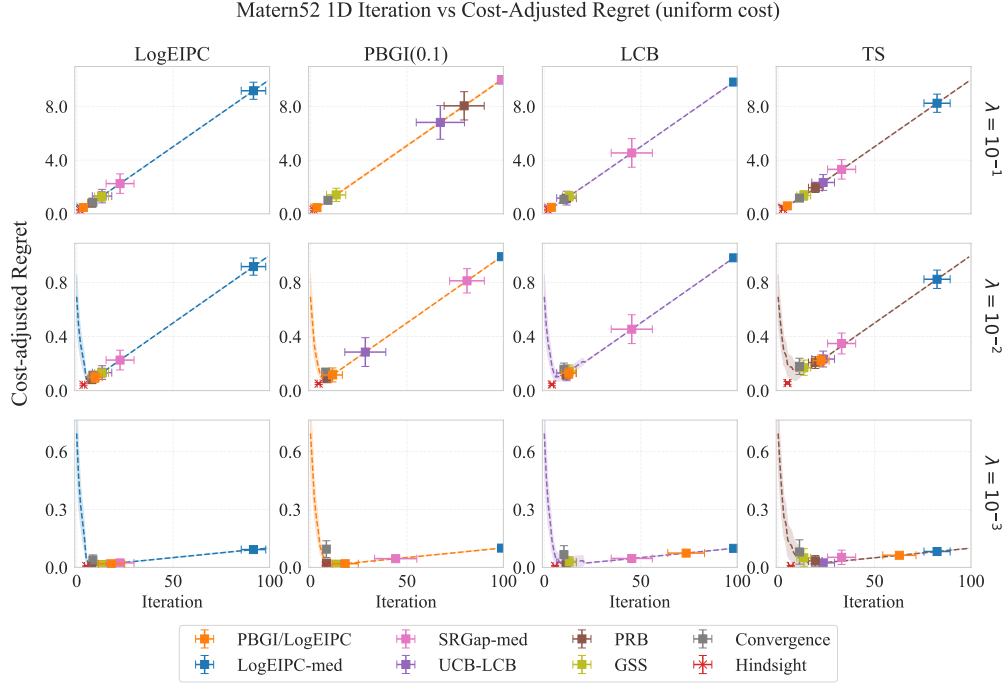


Figure B.20: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 1D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under uniform cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).

times without loss in performance. Figures B.20 to B.25 also show that our PBGI/LogEIPC stopping rule outperforms other baselines when the cost-scaling factor λ is large, indicating that it's an especially suitable stopping criteria when evaluation is expensive.

B.3.4.1 Ablation: Moving-Average Window Size

We ablate the window size $W \in \{1, 5, 10, 20\}$ for the PBGI/LogEIPC stopping rule on the 8D Bayesian regret benchmark, where $W = 1$ corresponds to no smoothing. Figures B.26 to B.29 show how the window size affects the stopping iteration and the resulting cost-adjusted regret along a single acquisition-function path under each cost regime. Aggregate results across 50 seeds for the PBGI/LogEIPC stopping

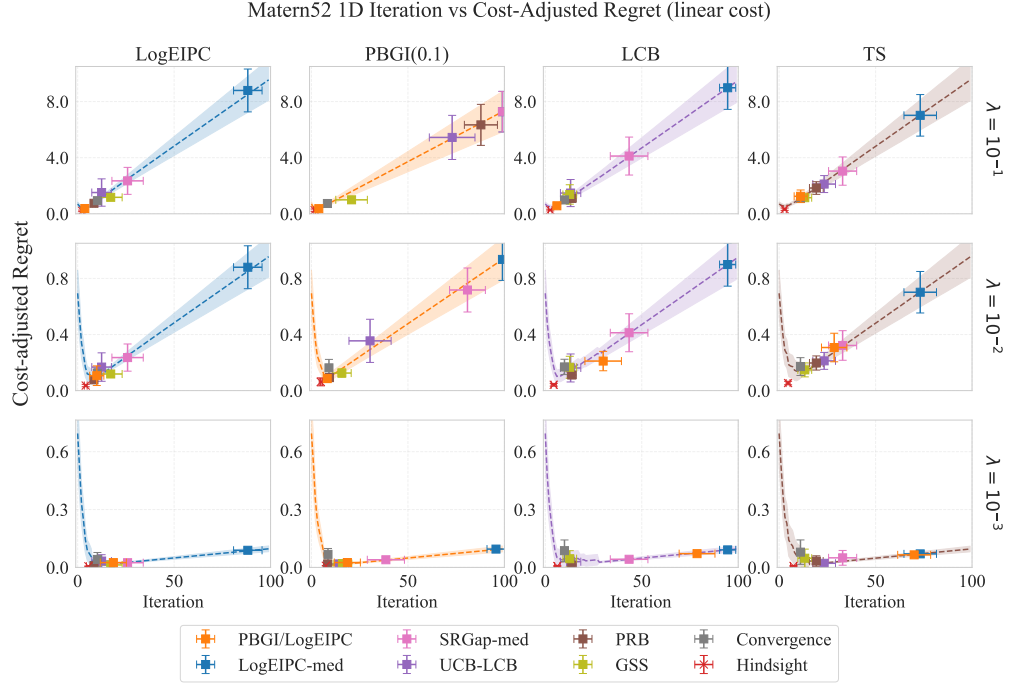


Figure B.21: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 1D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under linear cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).

rule paired with its matched acquisition function under the three cost regimes (uniform, linear, periodic) are summarized in Figures B.30 to B.32.

Across all three cost regimes, applying a moving average consistently improves cost-adjusted regret when the PBGI/LogEIPC stopping rule is paired with its matched acquisition function, with the largest gains observed under periodic costs.

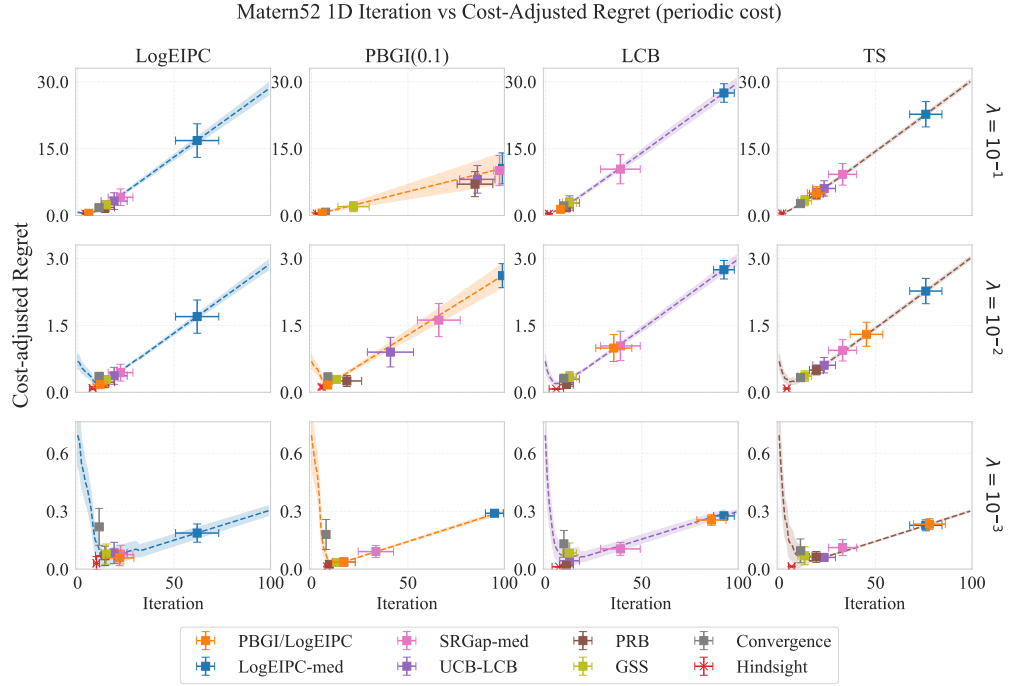


Figure B.22: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 1D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under periodic cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).

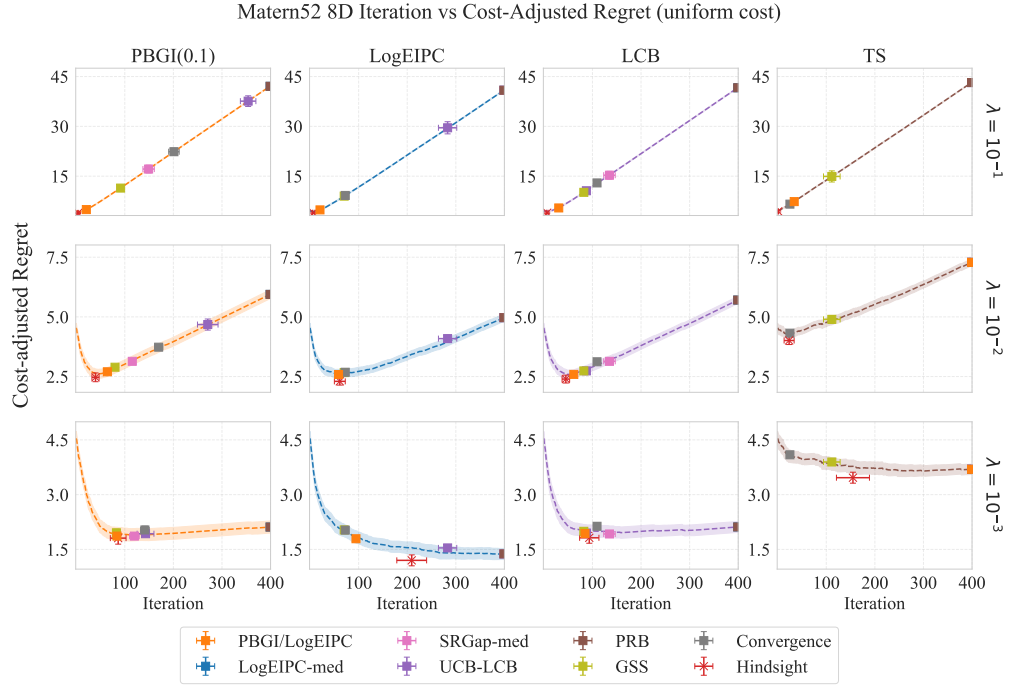


Figure B.23: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 8D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under uniform cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).

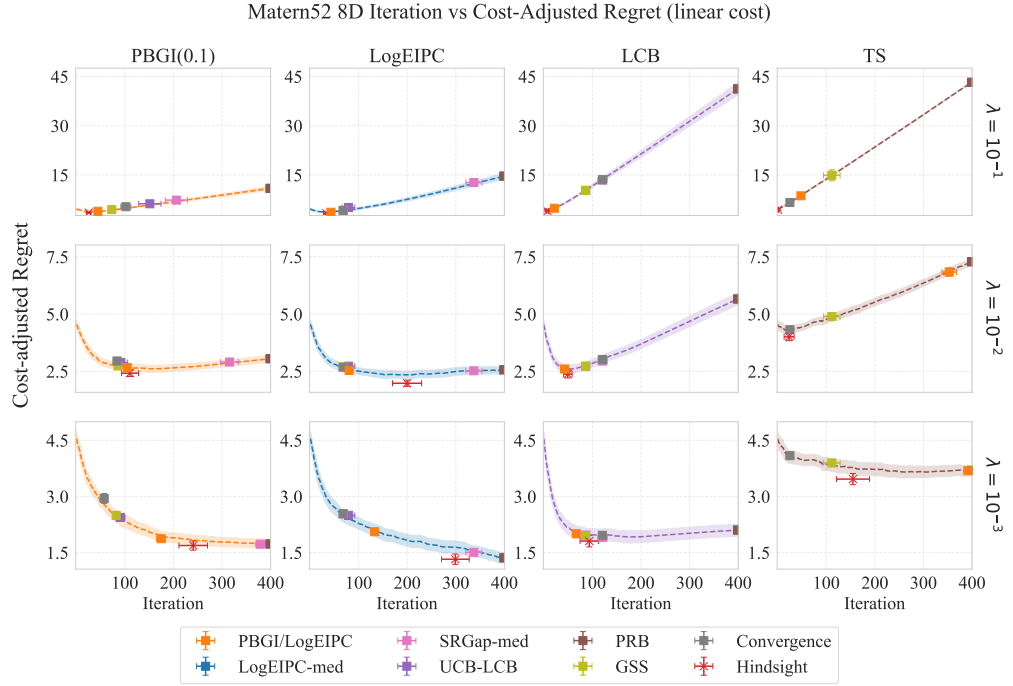


Figure B.24: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 8D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under linear cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).

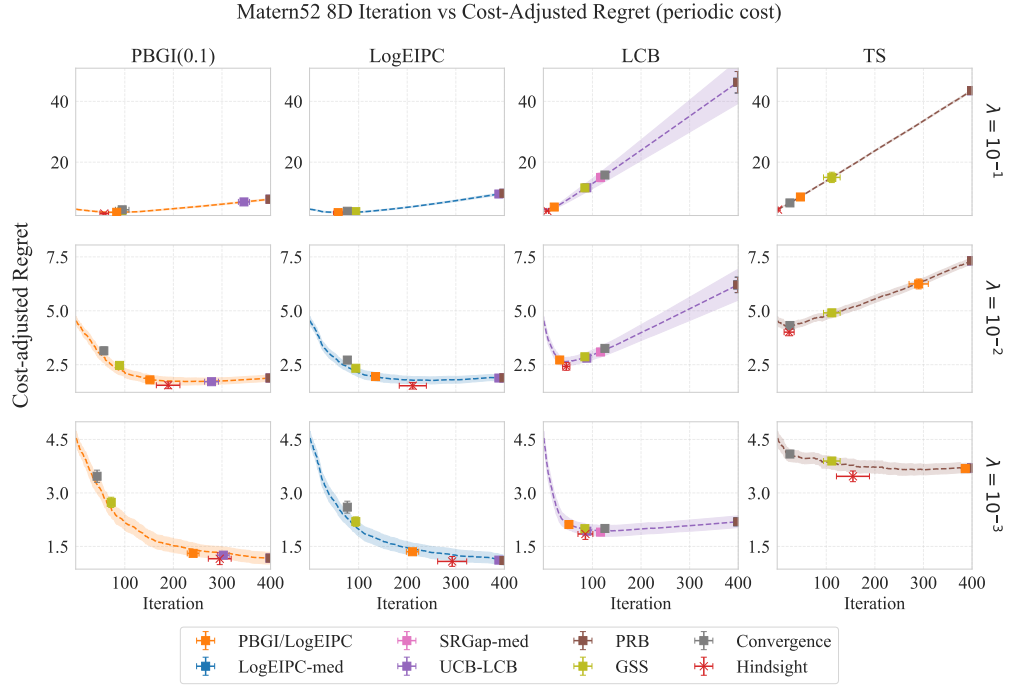


Figure B.25: Comparison of cost-adjusted simple regret across acquisition function and stopping rule pairs in the 8D Bayesian regret experiments, with cost-scaling factor $\lambda = 10^{-1}, 10^{-2}, 10^{-3}$ and under periodic cost. The dashed line in each subplot represent fixed iteration stopping (e.g., the y-axis value of the line at iteration 50 represent the cost-adjusted regret when always stop at iteration 50).

**Sampling: PBGI(0.01) Acquisition | Stopping: PBGI/LogEIPC Rule
(8D Periodic Cost, Seed=0, $\lambda=0.01$)**

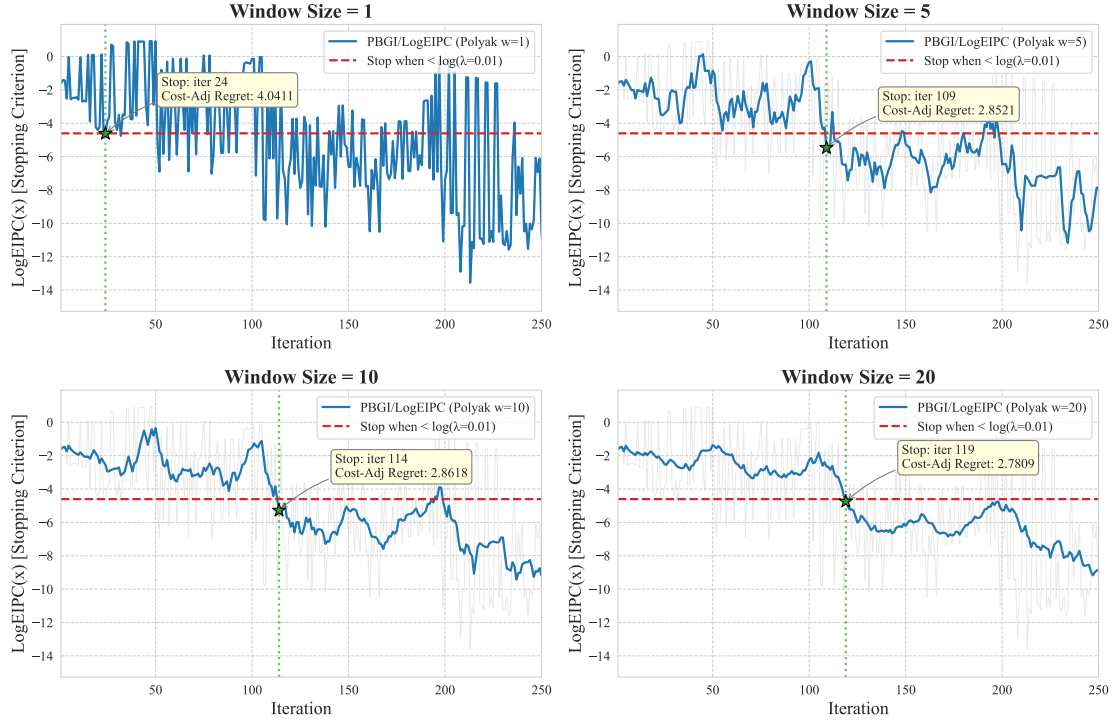


Figure B.26: Single-trajectory illustration of how the moving-average window size W affects the PBGI/LogEIPC stopping signal on the 8D Bayesian regret benchmark under periodic costs (seed 0, $\lambda = 10^{-2}$), with PBGI ($\lambda = 10^{-2}$) as the acquisition function. The light grey curve is the raw $\alpha_t^{\text{LogEIPC}}$, the blue curve its W -iteration trailing average, the red dashed line the stopping threshold $\log \lambda$, and the green star the resulting stopping iteration. Without smoothing ($W = 1$) the rule fires early (iter. 24, regret 4.04) on a noisy dip; smoothing defers the trigger and recovers low cost-adjusted regret (iter. 119, regret 2.78 at $W = 20$).

Sampling: LogEIPC Acquisition | Stopping: PBGI/LogEIPC Rule
(8D Periodic Cost, Seed=0, $\lambda=0.01$)

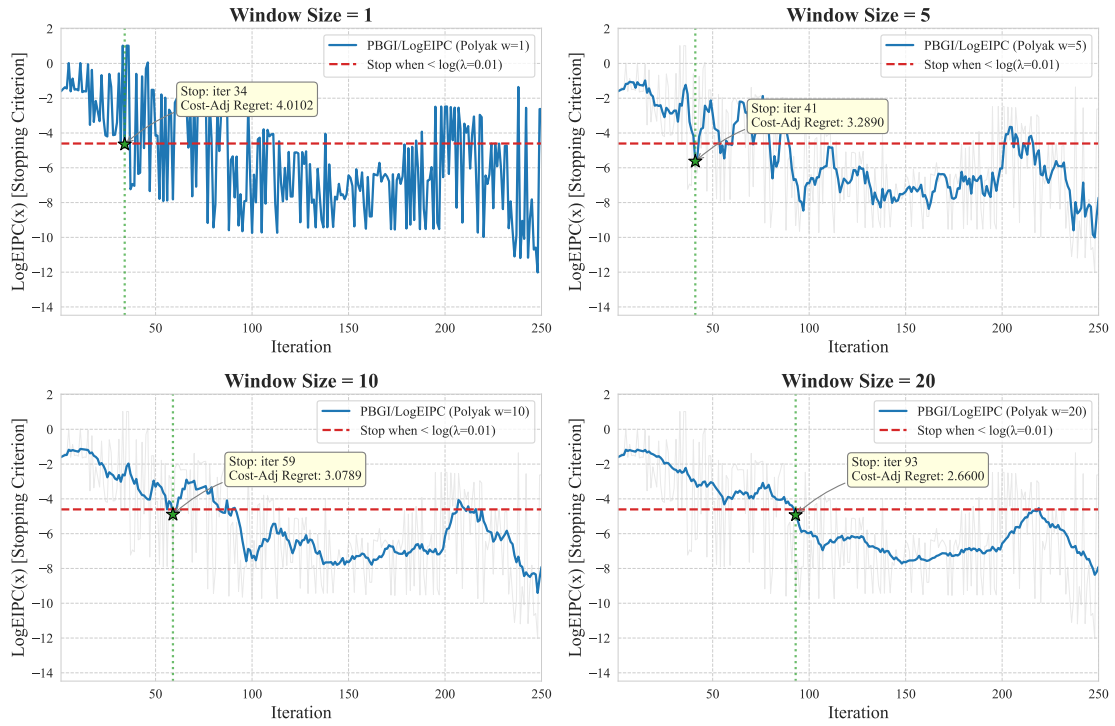


Figure B.27: Single-trajectory illustration analogous to Figure B.26, with the LogEIPC acquisition function (same seed, same cost regime, $\lambda = 10^{-2}$). Without smoothing the rule fires at iter. 34 with regret 4.01; with $W = 20$ it fires at iter. 93 with regret 2.66.

**Sampling: Thompson Sampling (TS) Acquisition | Stopping: PBGI/LogEIPC Rule
(8D Periodic Cost, Seed=0, $\lambda=0.01$)**

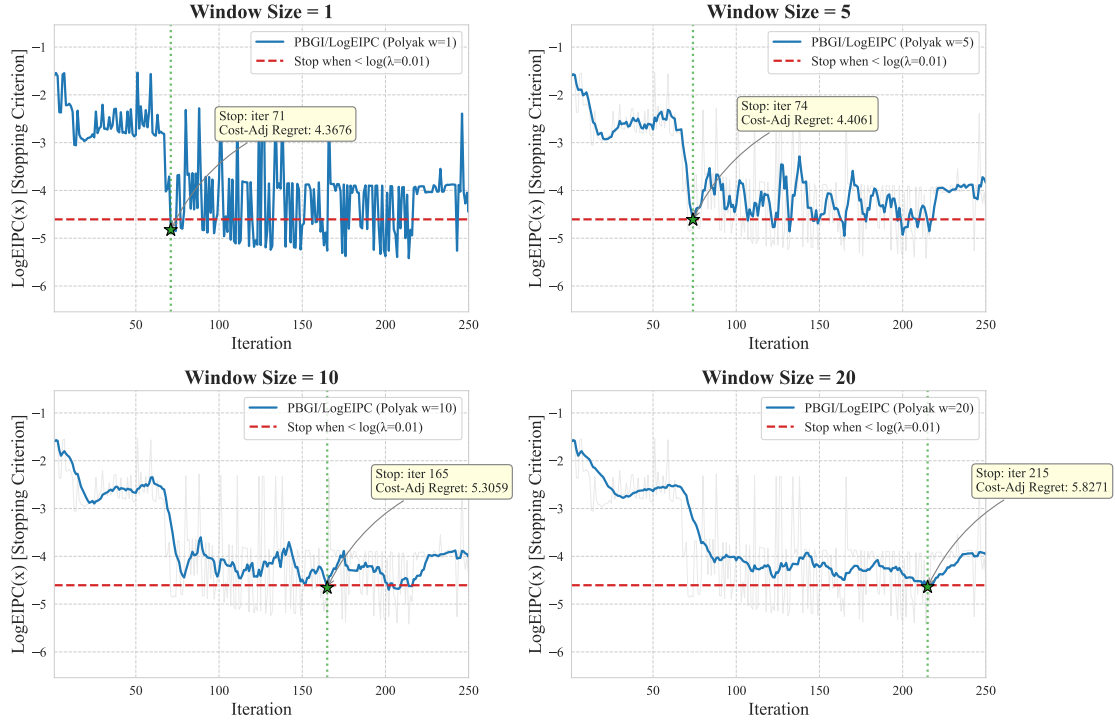


Figure B.28: Single-trajectory illustration analogous to Figure B.26, with Thompson sampling (TS) as the acquisition function. Unlike the matched-acquisition cases, smoothing here pushes the stopping iteration later (iter. 71 at $W = 1$ vs. iter. 215 at $W = 20$) without a corresponding drop in cost-adjusted regret. TS does not satisfy the expected-improvement-exceeds-cost property of Lemma 3, so the PBGI/LogEIPC stopping rule has no matched guarantee for it.

**Sampling: LCB Acquisition | Stopping: PBGI/LogEIPC Rule
(8D Periodic Cost, Seed=0, $\lambda=0.01$)**

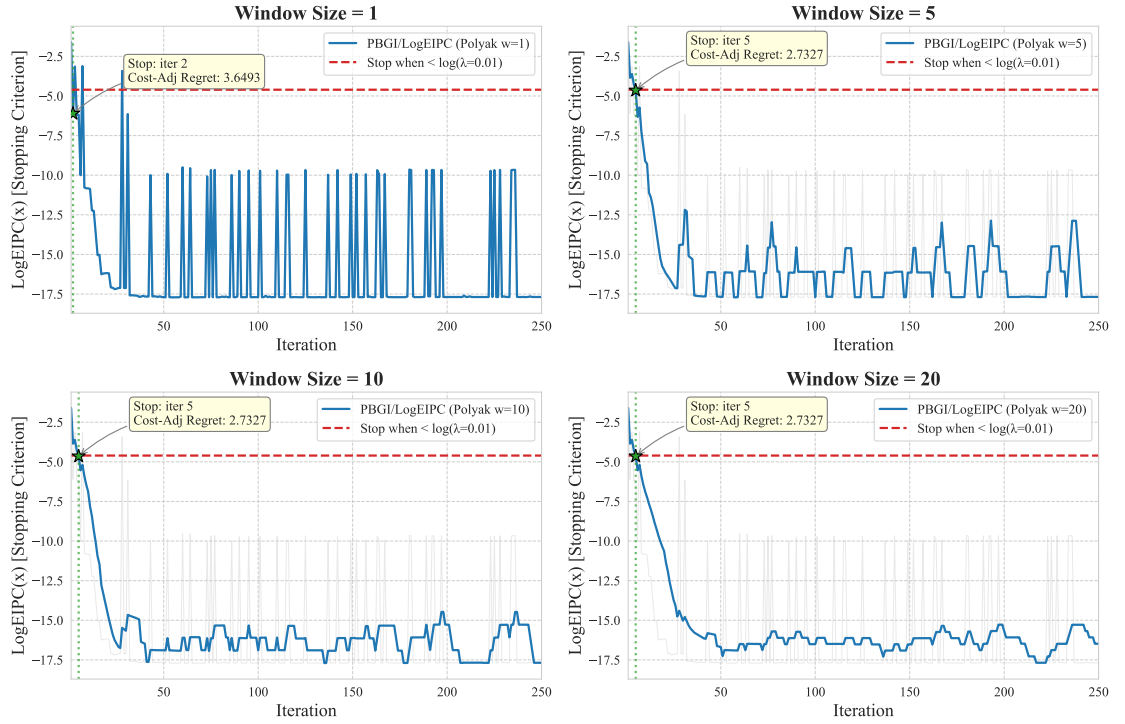


Figure B.29: Single-trajectory illustration analogous to Figure B.26, with LCB as the acquisition function. The raw LogEIPC signal has a clean transient: smoothing changes the stopping iteration only slightly (iter. 2 at $W = 1$ vs. iter. 5 at $W \geq 5$) and the cost-adjusted regret stabilizes at 2.73 for all $W \geq 5$.

Average Stopping Time & Cost-Adjusted Regret vs Window Size
(8D Periodic Cost, $\lambda=0.01$, 50 seeds)

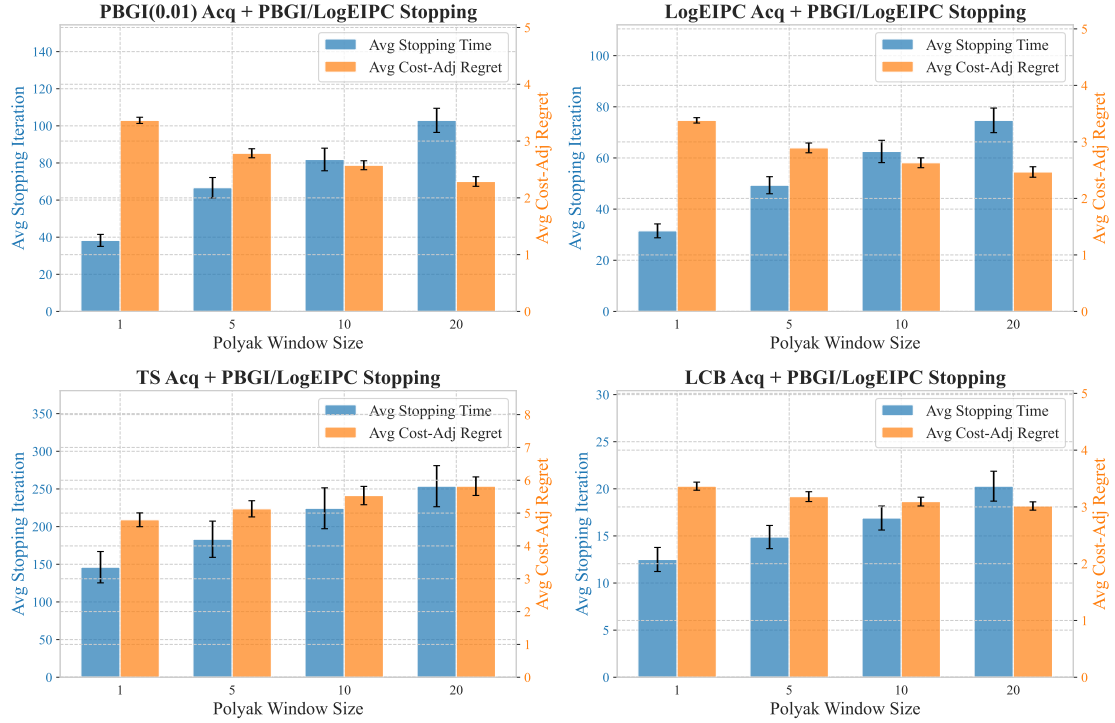


Figure B.30: Effect of the moving-average window size $W \in \{1, 5, 10, 20\}$ on the average stopping iteration (blue, left axis) and the average cost-adjusted simple regret (orange, right axis) for the PBGI/LogEIPC stopping rule on the 8D Bayesian regret benchmark under *periodic* costs ($\lambda = 10^{-2}$, 50 seeds, error bars are ± 2 standard errors). Each subplot fixes one acquisition function. Smoothing yields the largest cost-adjusted regret reductions for the matched PBGI and LogEIPC pairs, a smaller reduction for LCB, and no benefit for TS.

Average Stopping Time & Cost-Adjusted Regret vs Window Size
(8D Uniform Cost, $\lambda=0.01$, 50 seeds)

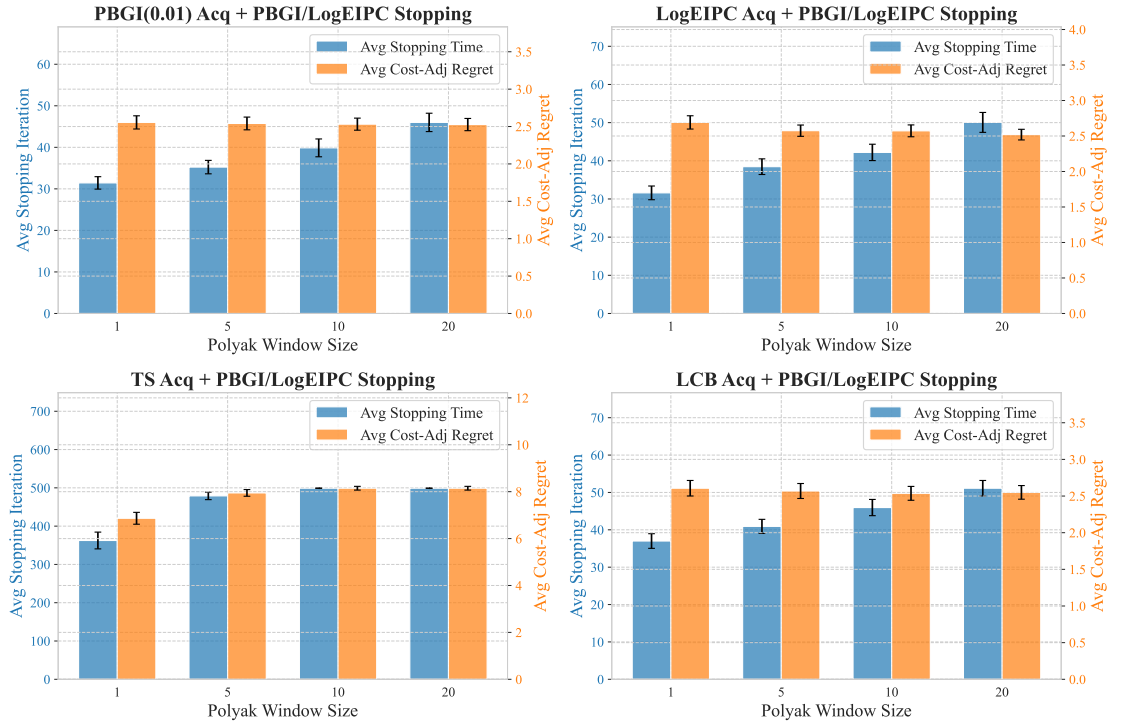


Figure B.31: Aggregate ablation analogous to Figure B.30 under *uniform* costs. Smoothing slightly reduces cost-adjusted regret for the matched PBGI and LogEIPC pairs and is roughly neutral for LCB and TS.

Average Stopping Time & Cost-Adjusted Regret vs Window Size
(8D Linear Cost, $\lambda=0.01$, 50 seeds)

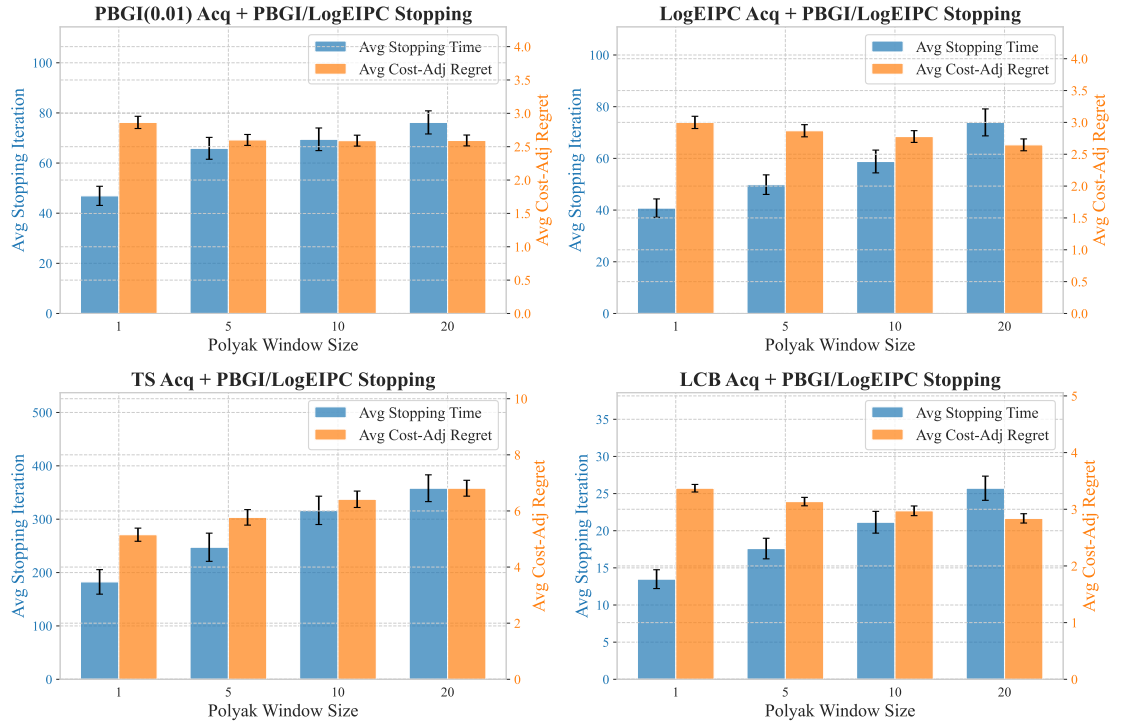


Figure B.32: Aggregate ablation analogous to Figure B.30 under *linear* costs. Smoothing reduces cost-adjusted regret for the matched PBGI and LogEIPC pairs and for LCB, with no benefit for TS.

APPENDIX C

SUPPLEMENT TO CHAPTER 4

C.1 Theoretical Details

C.1.1 Gaussian Random-Walk Details

We derive the posterior update and the induced random-walk transition used in Section 4.3. Fix an arm and suppress the arm index. Let $\bar{Y}_{n+1}$ 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

$$\theta \mid \mathcal{D}_n \sim \mathcal{N}(\mu_n, v_n), \quad (\text{C.1})$$

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

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

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

$$p(\theta \mid \mathcal{D}_{n+1}) \propto p(\bar{Y}_{n+1} \mid \theta) p(\theta \mid \mathcal{D}_n). \quad (\text{C.3})$$

Substituting the Gaussian likelihood and the current Gaussian posterior gives

$$p(\theta \mid \mathcal{D}_{n+1}) \propto \exp \left(-\frac{1}{2} \left[\frac{(\theta - \mu_n)^2}{v_n} + \frac{(\bar{Y}_{n+1} - \theta)^2}{\tau^2} \right] \right). \quad (\text{C.4})$$

Expanding the terms that depend on θ ,

$$\frac{(\theta - \mu_n)^2}{v_n} + \frac{(\bar{Y}_{n+1} - \theta)^2}{\tau^2} = (v_n^{-1} + \tau^{-2}) \theta^2 - 2 \left(\frac{\mu_n}{v_n} + \frac{\bar{Y}_{n+1}}{\tau^2} \right) \theta + \text{const.} \quad (\text{C.5})$$

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

$$\exp\left(-\frac{(\theta - \mu_{n+1})^2}{2v_{n+1}}\right) \quad (\text{C.6})$$

gives

$$v_{n+1}^{-1} = v_n^{-1} + \tau^{-2}, \quad \frac{\mu_{n+1}}{v_{n+1}} = \frac{\mu_n}{v_n} + \frac{\bar{Y}_{n+1}}{\tau^2}. \quad (\text{C.7})$$

Therefore,

$$v_{n+1} = (v_n^{-1} + \tau^{-2})^{-1}, \quad \mu_{n+1} = v_{n+1} \left(\frac{\mu_n}{v_n} + \frac{\bar{Y}_{n+1}}{\tau^2} \right). \quad (\text{C.8})$$

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

$$\mu_{n+1} = \mu_n + \frac{v_n}{v_n + \tau^2} (\bar{Y}_{n+1} - \mu_n). \quad (\text{C.9})$$

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

$$\frac{v_n}{v_n + \tau^2}. \quad (\text{C.10})$$

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

$$\bar{Y}_{n+1} = \theta + \varepsilon_{n+1}, \quad \theta \mid \mathcal{D}_n \sim \mathcal{N}(\mu_n, v_n), \quad \varepsilon_{n+1} \sim \mathcal{N}(0, \tau^2), \quad (\text{C.11})$$

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

$$\bar{Y}_{n+1} \mid \mathcal{D}_n \sim \mathcal{N}(\mu_n, v_n + \tau^2). \quad (\text{C.12})$$

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

$$S_{n+1} = s + \frac{v_n}{v_n + \tau^2} (\bar{Y}_{n+1} - s). \quad (\text{C.13})$$

Since

$$\bar{Y}_{n+1} - s \mid S_n = s \sim \mathcal{N}(0, v_n + \tau^2), \quad (\text{C.14})$$

we have

$$S_{n+1} \mid S_n = s \sim \mathcal{N}\left(s, \left(\frac{v_n}{v_n + \tau^2}\right)^2 (v_n + \tau^2)\right). \quad (\text{C.15})$$

Therefore,

$$S_{n+1} \mid S_n = s \sim \mathcal{N}\left(s, \frac{v_n^2}{v_n + \tau^2}\right). \quad (\text{C.16})$$

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

$$v_{n+1} = (v_n^{-1} + \tau^{-2})^{-1} \quad (\text{C.17})$$

does not depend on the realized observation $\bar{Y}_{n+1}$. 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.

C.1.2 Proof of Bayesian Optimality

Proposition 38 (Restatement of Proposition 8). *Consider the Gaussian Bayesian bandit model in Section 4.3.1 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_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*

$$\sup_{\pi \in \Pi_H^{\text{req}}} \mathbb{E}_\pi \left[\theta_{\hat{k}_T} - \sum_{t=1}^T c_{A_t} \right], \quad (\text{C.18})$$

an optimal policy is to assign terminal index $G_{k,t} = \mu_{k,H}$ to completed arms, assign $G_{k,t} = G_{n_k(t)}^{c_k}(\mu_{k,n_k(t)})$ 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 proposition 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 $n_k \in \{0, \dots, H\}$ 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

$$s' \mid s_k, n_k \sim \mathcal{N}\left(s_k, \frac{v_{n_k}^2}{v_{n_k} + \tau^2}\right). \quad (\text{C.19})$$

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

By Eq. (4.8), the state of arm k can be represented by (μ_{k,n_k}, n_k) . Under fixed τ^2 , Appendix C.1.1 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., 2003b; Scully and 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. $\square$

C.1.3 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

$$V_H(s; \alpha) = \max\{s, \alpha\}, \quad (\text{C.20})$$

and, for $n < H$,

$$\begin{aligned} Q_n^{\text{cont}}(s; \alpha) &= -c + \mathbb{E}[V_{n+1}(S_{n+1}; \alpha) \mid S_n = s], \\ Q_n^{\text{stop}}(s; \alpha) &= \alpha, \\ V_n(s; \alpha) &= \max\{Q_n^{\text{cont}}(s; \alpha), Q_n^{\text{stop}}(s; \alpha)\}. \end{aligned} \quad (\text{C.21})$$

The transition kernel depends only on differences in posterior means:

$$S_{n+1} \mid S_n = s \sim \mathcal{N}(s, \sigma_n^2), \quad \sigma_n^2 = \frac{v_n^2}{v_n + \tau^2}. \quad (\text{C.22})$$

Proof of Lemma 9. The terminal value satisfies

$$V_H(s + \beta; \alpha + \beta) = \max\{s + \beta, \alpha + \beta\} = V_H(s; \alpha) + \beta. \quad (\text{C.23})$$

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

$$\begin{aligned} Q_n^{\text{cont}}(s + \beta; \alpha + \beta) &= -c + \mathbb{E}[V_{n+1}(S_{n+1} + \beta; \alpha + \beta) \mid S_n = s] \\ &= -c + \mathbb{E}[V_{n+1}(S_{n+1}; \alpha) + \beta \mid S_n = s] \\ &= Q_n^{\text{cont}}(s; \alpha) + \beta. \end{aligned} \quad (\text{C.24})$$

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

Corollary 10 follows by applying Lemma 9 with shift $\beta = \alpha$ to the centered state $x = s - \alpha$ and outside reward 0. Define the centered value

$$W_n(x) = V_n(\alpha + x; \alpha) - \alpha. \quad (\text{C.25})$$

Then the recursion no longer depends on α :

$$W_H(x) = \max\{x, 0\}, \quad (\text{C.26})$$

and

$$\begin{aligned} W_n(x) &= \max\{0, q_n(x)\}, \\ q_n(x) &= -c + \mathbb{E}[W_{n+1}(X_{n+1}) \mid X_n = x], \end{aligned} \quad (\text{C.27})$$

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

$$q_n(r_n) = 0. \quad (\text{C.28})$$

The Gittins index for cost c is therefore

$$G_n^c(s) = s - r_n. \quad (\text{C.29})$$

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 evaluated efficiently using the FFT. Because the resulting index schedule depends only on the fixed cost and variance schedule, it can be precomputed once offline and reused cheaply during adaptive evaluation.

C.2 Dataset Details

C.2.1 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, Llama-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.

AlpacaEval. We use AlpacaEval (Li et al., 2023b) to evaluate instruction-following quality. Unlike GSM8K and PIQA, AlpacaEval does *not* use the BanditEval model×prompt configuration matrices. Instead, we construct a response matrix from the AlpacaEval 2.0 leaderboard comparisons annotated by `weighted_alpaca_eval_gpt4_turbo`. Each row corresponds to one AlpacaEval leaderboard model, and each column corresponds to one of the 805 fixed AlpacaEval

instructions. Entry (k, n) stores the model’s pairwise score against the GPT-4 Turbo baseline on instruction n . These scores are continuous values in $[0, 1]$ derived from the auto-annotator’s preference probabilities; they are *not* binary correctness labels. Accordingly, the finite-matrix target θ_k^{mat} in Eq. (4.3) is the row mean of these pairwise scores.

The resulting matrix contains 153 leaderboard-model arms and 805 instruction examples. It keeps the AlpacaEval pairwise scores as continuous values rather than converting them to binary outcomes. Each arm is therefore a single leaderboard model rather than a model $\times$ sampling-configuration pair. In contrast to GSM8K and PIQA, we use one fixed AlpacaEval matrix rather than five independently generated response matrices; algorithm randomness enters only through the 20 randomized trial seeds used in each sweep.

For configuration-level Bayesian optimization baselines, we convert this matrix into 153 configuration-level arms. Each Bayesian optimization arm evaluates one model on all 805 instructions, revealing the row-average score and consuming that model’s all-example evaluation cost. Cost-aware experiments use model-specific token prices from our Alpaca pricing table, with an input-to-output token ratio of 1:8 estimated from typical AlpacaEval prompt/response lengths (Appendix C.3). For Gittins-S on AlpacaEval, we use the dataset-specific prior mean/variance (0.2, 0.01); Gittins-G uses the general prior (0.5, 0.04) listed in Table C.5.

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 C.1.

C.2.2 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 score over benchmark examples. For binary correctness matrices, this score is mean accuracy; for AlpacaEval, it is the mean pairwise preference score. 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 C.1 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.

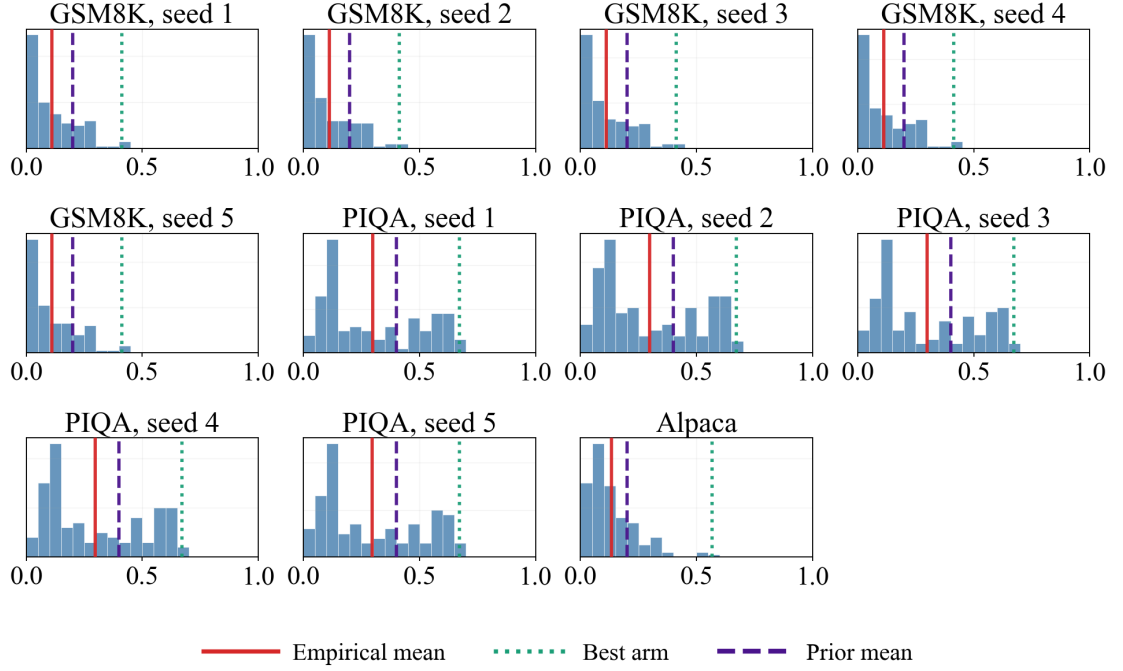


Figure C.1: Distribution of per-arm quality for GSM8K, PIQA, and AlpacaEval. GSM8K has 122 arms and 1,000 examples, PIQA has 103 arms and 1,000 examples, and AlpacaEval has 153 arms and 805 examples. For GSM8K and PIQA, each subplot corresponds to one dataset–seed pair across five random seeds; for AlpacaEval, the subplot uses the no-rounding-debias matrix used in our experiments. Bars show the histogram of arm quality, defined as mean accuracy or win rate over queried examples. Vertical lines mark the empirical mean (red), best arm (green), and prior mean (purple). The GSM8K and PIQA distributions are highly similar across seeds, suggesting that the observed variation is small when the only source of randomness is 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 C.2–C.4 show the per-arm quality distributions for the three buckets.

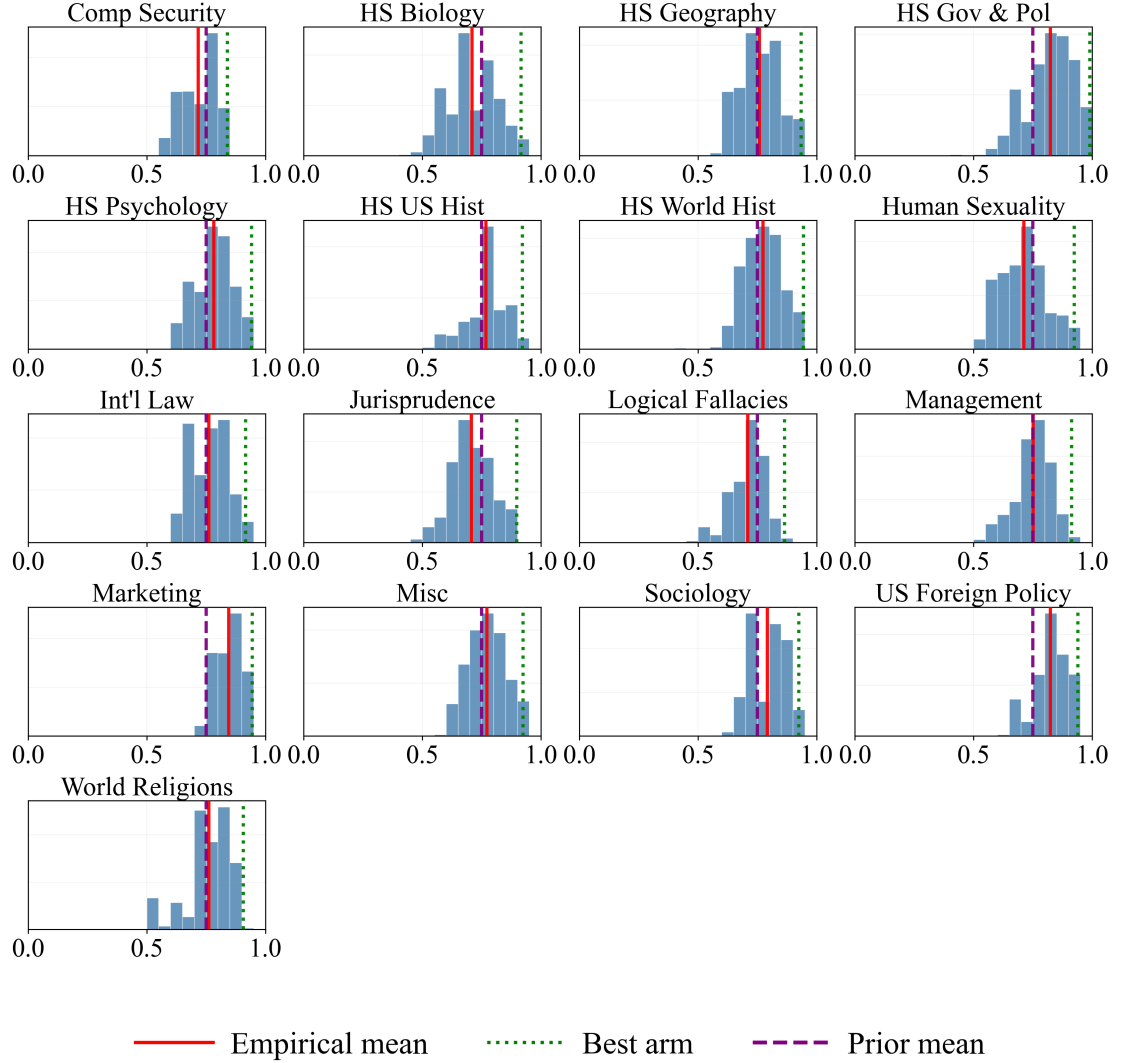


Figure C.2: Easy subjects: Distribution of per-arm quality for MMLU subjects assigned to the high informative-prior bucket ($\mu_0 = 0.75$). Each subplot corresponds to one subject; bars show the histogram of arm quality, defined as mean accuracy over questions. Vertical lines mark the empirical mean (red), best arm (green), and prior mean (purple).

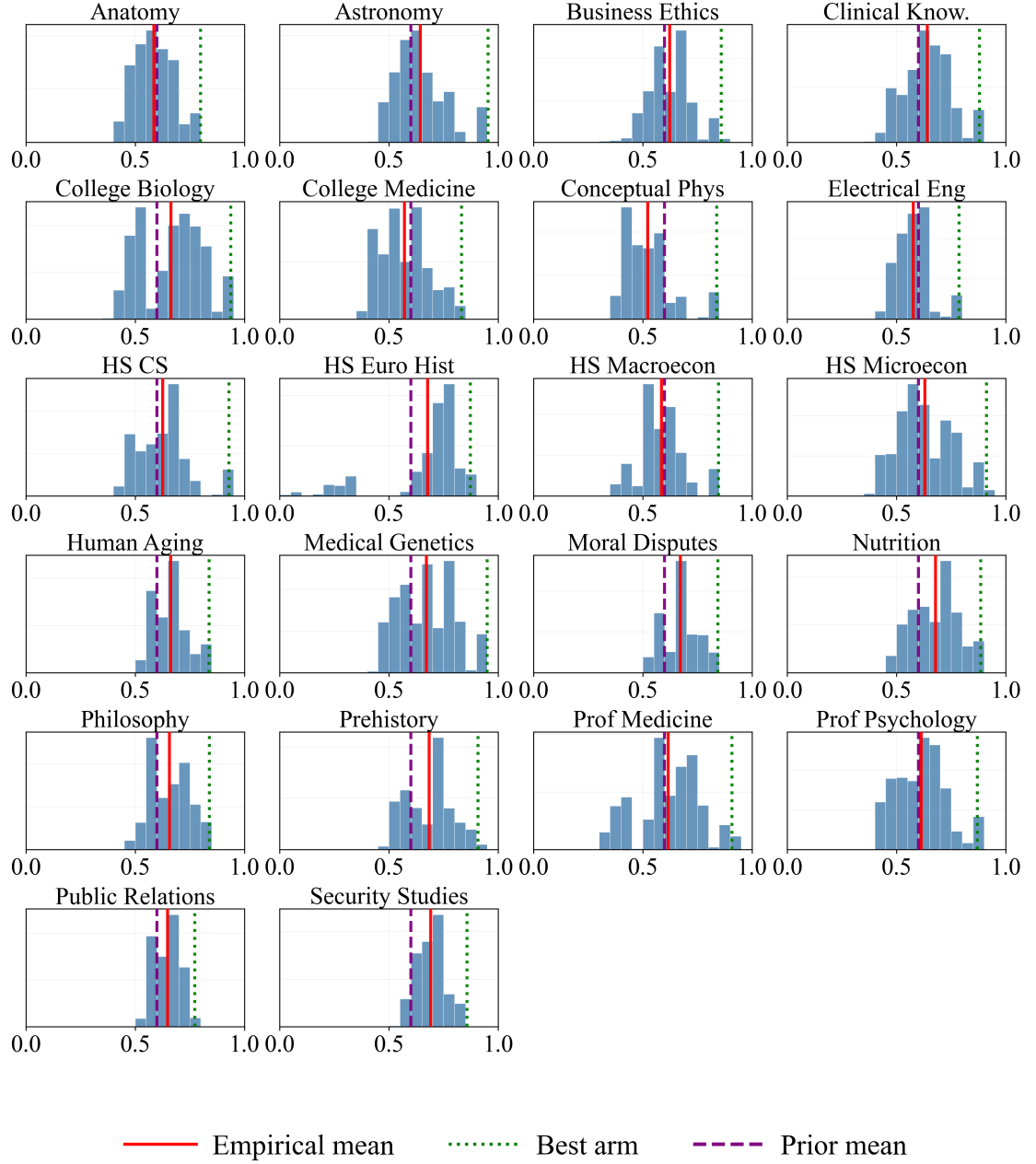


Figure C.3: Medium subjects: Distribution of per-arm quality for MMLU subjects assigned to the medium informative-prior bucket ($\mu_0 = 0.6$). Each subplot corresponds to one subject; bars show the histogram of arm quality, defined as mean accuracy over questions. Vertical lines mark the empirical mean (red), best arm (green), and prior mean (purple).

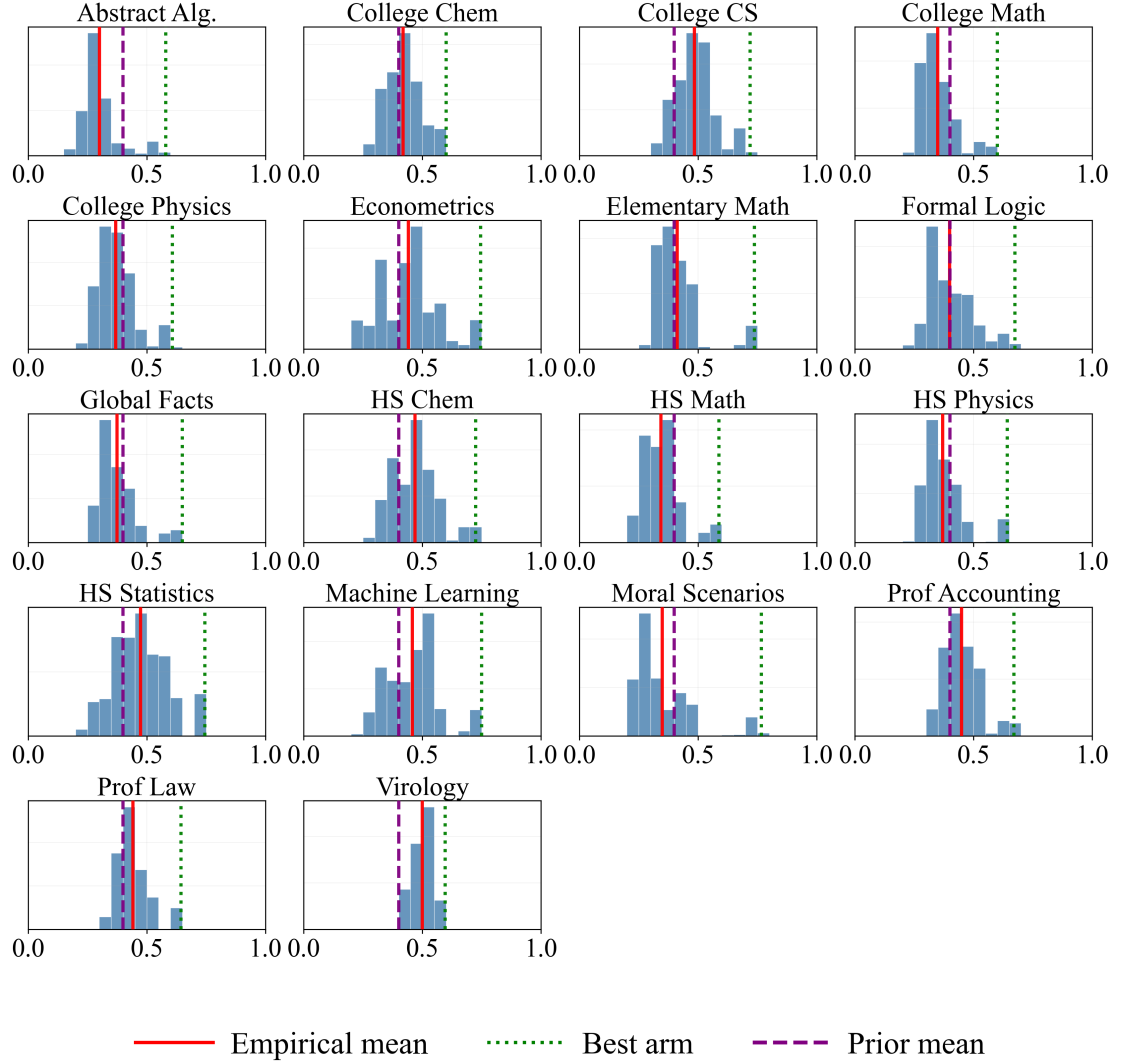


Figure C.4: Hard subjects: Distribution of per-arm quality for MMLU subjects assigned to the low informative-prior bucket ($\mu_0 = 0.4$). Each subplot corresponds to one subject; bars show the histogram of arm quality, defined as mean accuracy over questions. Vertical lines mark the empirical mean (red), best arm (green), and prior mean (purple).

Table C.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

Continued on next page

Table C.1: MMLU datasets used in our evaluation (continued).

Dataset	Rows	Difficulty	Size	Full Name
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

Continued on next page

Table C.1: MMLU datasets used in our evaluation (continued).

Dataset	Rows	Difficulty	Size	Full Name
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

Continued on next page

Table C.1: MMLU datasets used in our evaluation (continued).

Dataset	Rows	Difficulty	Size	Full Name
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.3 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 (arm, question) entries, while per-arm matrix means are used to compute simple regret. For the Bayesian optimization

baselines, the response matrices are converted into configuration-level inputs, where evaluating one candidate reveals its aggregate score. For GSM8K and PIQA, each converted BanditEval configuration corresponds to a model $\times$ prompt arm. For MMLU, each converted DOVE configuration corresponds to a model $\times$ template arm. For AlpacaEval, each converted Bayesian optimization configuration corresponds to one leaderboard model evaluated on all 805 instructions. 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 AlpacaEval, we use a single fixed pairwise score matrix with 153 models and 805 instructions. As in MMLU, we run 20 independent algorithm trials on this matrix and report averages over those trials. Unlike GSM8K and PIQA, we do not average over multiple independently generated response matrices for AlpacaEval. For MMLU, we run 20 randomized algorithm trials for each subject and aggregate results across the corresponding set of subjects. The Bayesian optimization baselines use the same randomized-trial structure on their converted configuration-level inputs.

Stopping-rule variants. For Gittins-based methods, we record two post-pull stopping diagnostics. The primary diagnostic used in the main figures is the *index-induced stopping rule*: after each batch is evaluated and posterior Gittins indices are updated, the policy stops when an arm attaining the largest current Gittins

score is already fully observed. This corresponds to the natural termination point of the required-completion Gittins policy, since the policy’s own index maximizer has become a completed arm.

We also record a *recommendation-aware stopping rule*. Let Γ_i denote the current Gittins score of arm i , and let μ_i denote its posterior mean. This diagnostic stops when

$$\max_{i \in \mathcal{I}_{\text{incomplete}}} \Gamma_i < \max_j \mu_j, \quad (\text{C.30})$$

that is, when the best remaining exploration opportunity has lower index than the current best posterior-mean recommendation. This rule is more directly tied to the anytime recommendation criterion, whereas the index-induced rule follows the exploration policy itself. We use the index-induced rule in the main figures because it is the stopping diagnostic most directly implied by the required-completion Gittins policy: it asks when the policy’s own index maximizer has become a completed arm. The recommendation-aware rule is included only to clarify an alternative diagnostic tied to the anytime recommendation criterion, but it is not used for the stopping markers in the reported main figures.

Baseline initialization and warm-up. For a response matrix with K arms and N examples, the Bayesian optimization baselines use a configuration-level random-initialization phase. In our main Bayesian optimization 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 ($K = 103$), 8 for AlpacaEval ($K = 153$), and 75 for each MMLU subject ($K = 1500$). Under the nominal 10% Bayesian optimization budget, the corresponding total

configuration budgets are 12, 10, 15, 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 arms evaluated on all benchmark examples.

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. For GSM8K and PIQA, 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. For MMLU, each arm is a model-prompt pair, and model-level input prices are repeated across the corresponding prompt arms. For AlpacaEval, each arm is a single leaderboard model rather than a model $\times$ sampling-configuration pair. Depending on the available metadata, costs 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 Bayesian optimization baselines, each converted input row corresponds to a complete configuration, and its cost is the estimated all-example 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 Bayesian optimization baselines use configuration-level costs for cost-aware acquisition and budget accounting.

AlpacaEval evaluation costs. AlpacaEval uses model-specific API or proxy prices for the 153 leaderboard models in the response matrix. For each model, we record separate input and output costs in USD per 1M tokens and form the evaluation cost as

$$\text{cost} = c_{\text{in}} + 8 c_{\text{out}}, \quad (\text{C.31})$$

using the estimated AlpacaEval input-to-output token ratio 1:8. These model-level costs define the per-arm cost vector for cost-aware bandit experiments and the all-example evaluation costs used by the converted Bayesian optimization inputs. Because AlpacaEval arms are individual leaderboard models rather than BanditEval-style model×prompt configurations, we do not repeat a single model price across multiple prompting arms as in MMLU. Tables C.2 and C.3 therefore do not apply to AlpacaEval; the complete Alpaca pricing table is reported in Table C.4.

Table C.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

Table C.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

Table C.4: AlpacaEval model-level pricing table for the 153-model response matrix. Input and output prices are in USD per 1M tokens. The combined cost is input cost + $8\times$ output cost, matching the AlpacaEval input-to-output token ratio used for cost-aware runs.

Model	Input	Output	Combined
Conifer-7B-DPO	0.2	0.2	1.8
Contextual-KTO-Mistral-PairRM	0.2	0.2	1.8
FsfairX-Zephyr-Chat-v0.1	0.2	0.2	1.8
LMCocktail-10.7B-v1	0.2	0.2	1.8
Meta-Llama-3-70B-Instruct	0.9	0.9	8.1
Meta-Llama-3-8B-Instruct	0.2	0.2	1.8
Mistral-7B-Instruct-v0.2	0.2	0.2	1.8
Mistral-7B-ReMax-v0.1	0.2	0.2	1.8
Mixtral-8x22B-Instruct-v0.1	1.2	1.2	10.8
Mixtral-8x7B-Instruct-v0.1	0.5	0.5	4.5
Mixtral-8x7B-Instruct-v0.1_concise	0.5	0.5	4.5
Mixtral-8x7B-Instruct-v0.1_verbose	0.5	0.5	4.5
Nanbeige-Plus-Chat-v0.1	0.2	0.2	1.8
Nanbeige2-8B-Chat	0.2	0.2	1.8
OpenHermes-2.5-Mistral-7B	0.2	0.2	1.8
Qwen-14B-Chat	0.2	0.2	1.8
Qwen1.5-1.8B-Chat	0.1	0.1	0.9
Qwen1.5-110B-Chat	0.9	0.9	8.1
Qwen1.5-14B-Chat	0.2	0.2	1.8
Qwen1.5-72B-Chat	0.1	0.1	0.9
Qwen1.5-7B-Chat	0.2	0.2	1.8

Continued on next page

Table C.4: AlpacaEval model-level pricing table (continued).

Model	Input	Output	Combined
Samba-CoE-v0.1	0.2	0.2	1.8
Samba-CoE-v0.2-best-of-16	3.2	3.2	28.8
Samba-CoE-v0.2	0.2	0.2	1.8
Snorkel-Mistral-PairRM-DPO-best-of-16	3.2	3.2	28.8
Snorkel-Mistral-PairRM-DPO	0.2	0.2	1.8
Starling-LM-7B-alpha	0.2	0.2	1.8
TempNet-LLaMA2-Chat-13B-v0.1	0.2	0.2	1.8
TempNet-LLaMA2-Chat-70B-v0.1	0.9	0.9	8.1
TempNet-LLaMA2-Chat-7B-v0.1	0.2	0.2	1.8
Yi-34B-Chat	0.9	0.9	8.1
airoboros-33b	0.9	0.9	8.1
airoboros-65b	0.9	0.9	8.1
aligner-2b_claude-3-opus-20240229	0.1	0.1	0.9
aligner-2b_qwen1.5-72b-chat	0.1	0.1	0.9
LLaMA-7B	0.2	0.2	1.8
alpaca-7b	0.2	0.2	1.8
alpaca-7b_concise	0.2	0.2	1.8
alpaca-7b_verbose	0.2	0.2	1.8
alpaca-farm-ppo-human	0.2	0.2	1.8
alpaca-farm-ppo-sim-gpt4-20k	0.2	0.2	1.8
baichuan-13b-chat	0.2	0.2	1.8
baize-v2-13b	0.2	0.2	1.8
baize-v2-7b	0.2	0.2	1.8
causallm-14b	0.2	0.2	1.8
chatglm2-6b	0.2	0.2	1.8

Continued on next page

Table C.4: AlpacaEval model-level pricing table (continued).

Model	Input	Output	Combined
claude-2.1	8	24	200
claude-2.1_concise	8	24	200
claude-2.1_verbose	8	24	200
claude-2	8	24	200
claude-3-opus-20240229	15	75	615
claude-3-sonnet-20240229	3	15	123
claude-instant-1.2	0.8	2.4	20
claude	8	24	200
claude2-alpaca-13b	0.2	0.2	1.8
cohere	1	2	17
cut-13b	0.2	0.2	1.8
dbrx-instruct	1.2	1.2	10.8
deepseek-llm-67b-chat	0.9	0.9	8.1
deita-7b-v1.0	0.2	0.2	1.8
dolphin-2.2.1-mistral-7b	0.2	0.2	1.8
evo-7b	0.2	0.2	1.8
evo-v2-7b	0.2	0.2	1.8
falcon-40b-instruct	0.9	0.9	8.1
falcon-7b-instruct	0.2	0.2	1.8
gemini-pro	0.5	1.5	12.5
gemma-2b-it	0.1	0.1	0.9
gemma-7b-it	0.2	0.2	1.8
gpt-3.5-turbo-0301	1.5	2	17.5
gpt-3.5-turbo-0613	1.5	2	17.5
gpt-3.5-turbo-1106	1	2	17

Continued on next page

Table C.4: AlpacaEval model-level pricing table (continued).

Model	Input	Output	Combined
gpt-3.5-turbo-1106_concise	1	2	17
gpt-3.5-turbo-1106_verbose	1	2	17
gpt-4-0125-preview	10	30	250
gpt35_turbo_instruct	1.5	2	17.5
gpt4	30	60	510
gpt4_0314	30	60	510
gpt4_0613	30	60	510
gpt4_0613_concise	30	60	510
gpt4_0613_verbose	30	60	510
gpt4_1106_preview	10	30	250
gpt4_1106_preview_concise	10	30	250
gpt4_gamed	30	60	510
guanaco-13b	0.2	0.2	1.8
guanaco-33b	0.9	0.9	8.1
guanaco-65b	0.9	0.9	8.1
guanaco-7b	0.2	0.2	1.8
humpback-llama-65b	0.9	0.9	8.1
humpback-llama2-70b	0.9	0.9	8.1
internlm2-chat-20b-ppo	0.9	0.9	8.1
jina-chat	0.2	0.2	1.8
llama-2-13b-chat-hf	0.2	0.2	1.8
llama-2-70b-chat-hf	0.9	0.9	8.1
llama-2-7b-chat-hf	0.2	0.2	1.8
llama-2-chat-7b-evol70k-neft	0.2	0.2	1.8
minichat-1.5-3b	0.1	0.1	0.9

Continued on next page

Table C.4: AlpacaEval model-level pricing table (continued).

Model	Input	Output	Combined
minichat-3b	0.1	0.1	0.9
minotaur-13b	0.2	0.2	1.8
mistral-large-2402	8	24	200
mistral-medium	2.7	8.1	67.5
mistral-orpo-beta	0.2	0.2	1.8
nous-hermes-13b	0.2	0.2	1.8
oasst-rlhf-llama-33b	0.9	0.9	8.1
oasst-sft-llama-33b	0.9	0.9	8.1
oasst-sft-pythia-12b	0.1	0.1	0.9
openbuddy-falcon-40b-v9	0.9	0.9	8.1
openbuddy-falcon-7b-v6	0.2	0.2	1.8
openbuddy-llama-30b-v7.1	0.9	0.9	8.1
openbuddy-llama-65b-v8	0.9	0.9	8.1
openbuddy-llama2-13b-v11.1	0.2	0.2	1.8
openbuddy-llama2-70b-v10.1	0.9	0.9	8.1
openchat-13b	0.2	0.2	1.8
openchat-v2-13b	0.2	0.2	1.8
openchat-v2-w-13b	0.2	0.2	1.8
openchat-v3.1-13b	0.2	0.2	1.8
openchat8192-13b	0.2	0.2	1.8
opencoderplus-15b	0.2	0.2	1.8
pairrm-Yi-34B-Chat	0.9	0.9	8.1
pairrm-tulu-2-13b	0.2	0.2	1.8
pairrm-tulu-2-70b	0.9	0.9	8.1
pairrm-zephyr-7b-beta	0.2	0.2	1.8

Continued on next page

Table C.4: AlpacaEval model-level pricing table (continued).

Model	Input	Output	Combined
phi-2-dpo	0.1	0.1	0.9
phi-2-sft	0.1	0.1	0.9
phi-2	0.1	0.1	0.9
platolm-7b	0.2	0.2	1.8
pythia-12b-mix-sft	0.1	0.1	0.9
recycled-wizardlm-7b-v1.0	0.2	0.2	1.8
recycled-wizardlm-7b-v2.0	0.2	0.2	1.8
text_davinci_001	20	20	180
text_davinci_003	20	20	180
tulu-2-dpo-13b	0.2	0.2	1.8
tulu-2-dpo-70b	0.9	0.9	8.1
tulu-2-dpo-7b	0.2	0.2	1.8
ultralm-13b-best-of-16	3.2	3.2	28.8
ultralm-13b-v2.0-best-of-16	3.2	3.2	28.8
ultralm-13b-v2.0	0.2	0.2	1.8
ultralm-13b	0.2	0.2	1.8
vicuna-13b-v1.3	0.2	0.2	1.8
vicuna-13b-v1.5	0.2	0.2	1.8
vicuna-13b	0.2	0.2	1.8
vicuna-33b-v1.3	0.9	0.9	8.1
vicuna-7b-v1.3	0.2	0.2	1.8
vicuna-7b-v1.5	0.2	0.2	1.8
vicuna-7b	0.2	0.2	1.8
wizardlm-13b-v1.1	0.2	0.2	1.8
wizardlm-13b-v1.2	0.2	0.2	1.8

Continued on next page

Table C.4: AlpacaEval model-level pricing table (continued).

Model	Input	Output	Combined
wizardlm-13b	0.2	0.2	1.8
wizardlm-70b	0.9	0.9	8.1
xwinlm-13b-v0.1	0.2	0.2	1.8
xwinlm-70b-v0.1	0.9	0.9	8.1
xwinlm-7b-v0.1	0.2	0.2	1.8
zephyr-7b-alpha	0.2	0.2	1.8
zephyr-7b-beta	0.2	0.2	1.8

Gaussian approximation. For benchmark scores grouped in batches, a batch of B responses produces an empirical mean score. For binary-accuracy benchmarks such as GSM8K, PIQA, and MMLU, the default observation variance is $\tau^2 = 1/(4B)$, the worst-case binary variance divided by the batch size. For AlpacaEval, matrix entries are continuous pairwise preference scores in $[0, 1]$ rather than binary correctness labels. We therefore do not use the binary worst-case variance $1/(4B)$ as the default observation noise for AlpacaEval; instead, we use the same batch-noise scale as in the main reported Gittins runs for that benchmark. 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 C.5 lists the general default prior and the data-specific common priors used by Gittins-G and

Gittins-S. Figures C.2, C.3, and C.4 show the MMLU latent-accuracy distributions used to motivate the informative prior buckets.

Table C.5: 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	$\mathcal{N}(0.2, 0.01)$	Very hard benchmark
PIQA	$\mathcal{N}(0.4, 0.02)$	Hard benchmark
AlpacaEval	$\mathcal{N}(0.2, 0.01)$	Very hard benchmark
MMLU low-accuracy bucket	$\mathcal{N}(0.4, 0.02)$	Hard subjects
MMLU medium-accuracy bucket	$\mathcal{N}(0.6, 0.02)$	Medium subjects
MMLU high-accuracy bucket	$\mathcal{N}(0.75, 0.01)$	Easy subjects

C.4 Additional Experiment Results

C.4.1 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 C.5. 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

Table C.6: Batch-size settings used in the simple-regret experiments. For UCB-E and both Gittins variants, batch size B denotes the number of examples queried for the selected arm at each allocation step. For AlpacaEval, these examples correspond to instruction-level pairwise preference scores rather than binary correctness labels. UCB-E-LRF uses $B = 32$ in all reported settings. Bayesian optimization 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	$B \in \{8, 16, 32\}$
AlpacaEval	1 pairwise score matrix	805 instructions	$B \in \{8, 16, 32\}$
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	$n_{\text{examples}} > 400$	$B \in \{8, 16, 32\}$

approximation error and posterior noise but make each allocation decision coarser. UCB-E-LRF uses $B = 32$ in the reported settings, while Bayesian optimization baselines operate at the configuration level and therefore do not use a bandit batch size. Table C.6 summarizes the batch-size grids used for GSM8K, PIQA, MMLU, and AlpacaEval. On AlpacaEval, each batch query corresponds to B additional pairwise comparisons for the selected model rather than B independent benchmark examples with binary labels.

Cost-scaling factor. The Gittins index policy uses a cost-scaling factor to map evaluation costs into the continuation penalty in the index computation. In the unit-cost setting, this factor is applied to a common unit cost for each evaluation; in the cost-aware setting, it is applied to the original per-evaluation costs. This parameter controls the effective price of continuing to sample: larger values make additional evaluations more costly and therefore favor earlier stopping, whereas smaller values encourage longer exploration. In our experiments, we sweep three

cost-scaling factors, 10^{-3} , 10^{-4} , and 10^{-5} , for both unit-cost and cost-aware Gittins variants. Unless otherwise specified, the main reported figures use 10^{-4} , which we found to provide a representative balance between stopping early and continuing exploration.

C.4.2 Per-Subject MMLU Results

We provide additional MMLU results under informative priors in Figures C.5–C.10. 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. Gittins-S generally reaches lower simple regret earlier than UCB-E and Gittins-G across difficulty buckets. This suggests that the data-specific prior is broadly helpful for MMLU, especially when subjects are more challenging.

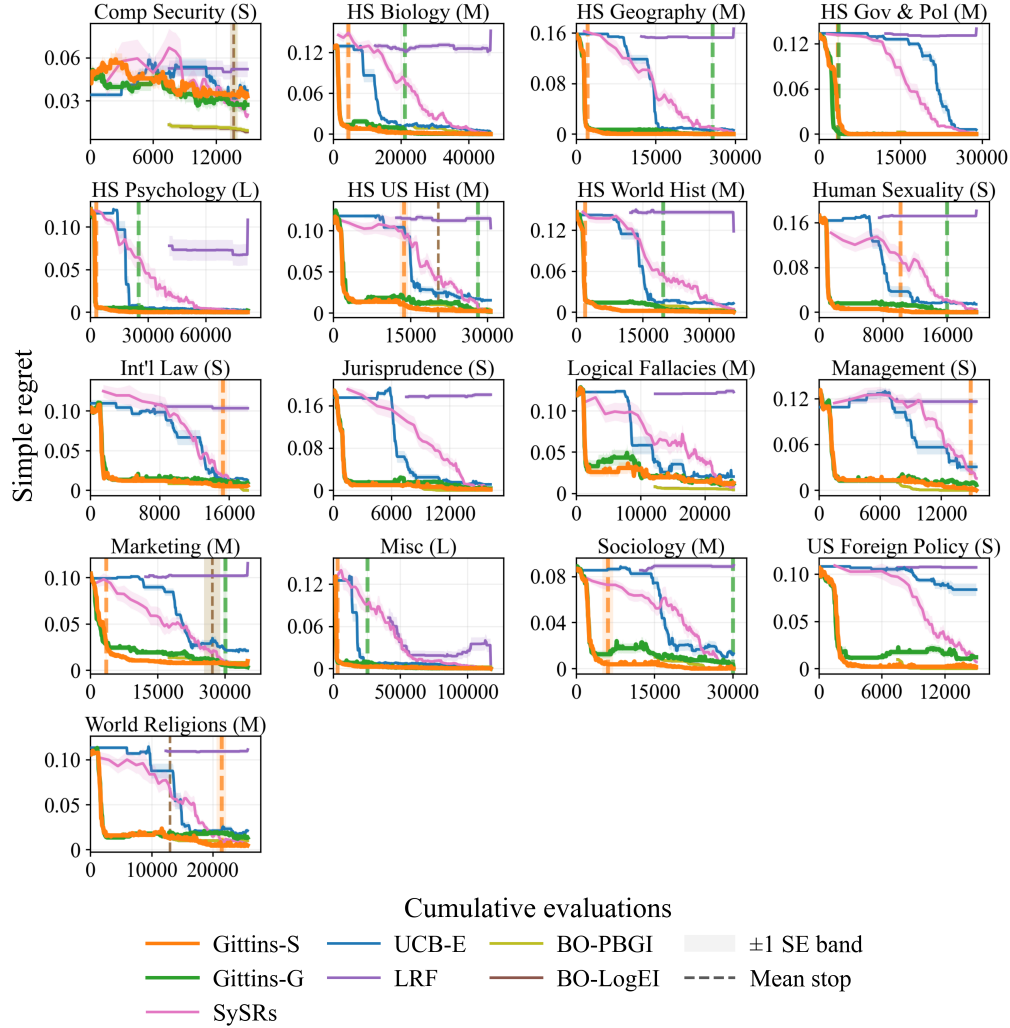


Figure C.5: MMLU unit-cost results for easy subjects (high prior bucket). Simple regret is plotted against cumulative evaluations. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ 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. Bayesian optimization is competitive on several tasks but is less uniformly strong, while LRF is generally less effective and less stable. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.

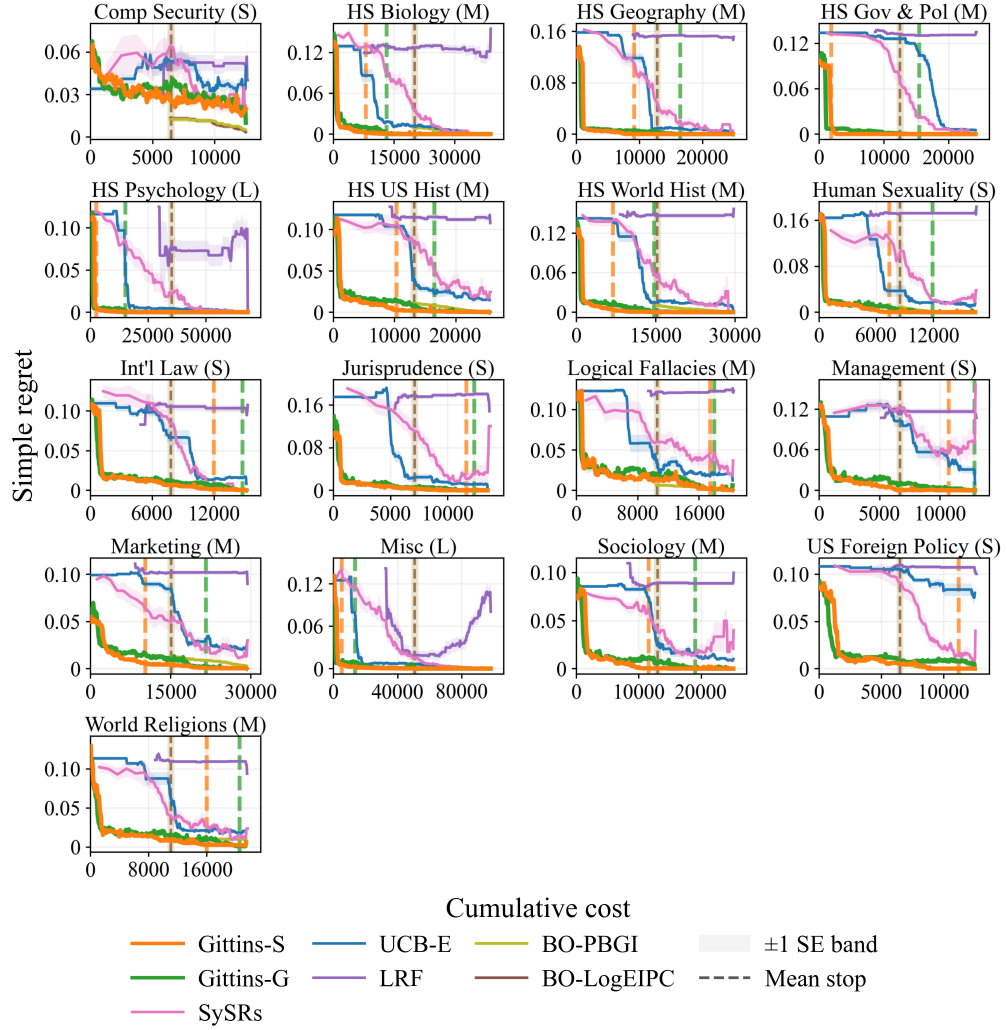


Figure C.6: MMLU cost-aware results for easy subjects (high prior bucket). Simple regret is plotted against cumulative cost. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ throughout. 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. Bayesian optimization is strong on some subjects but is not consistently better than Gittins, and LRF remains comparatively weak and unstable. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.

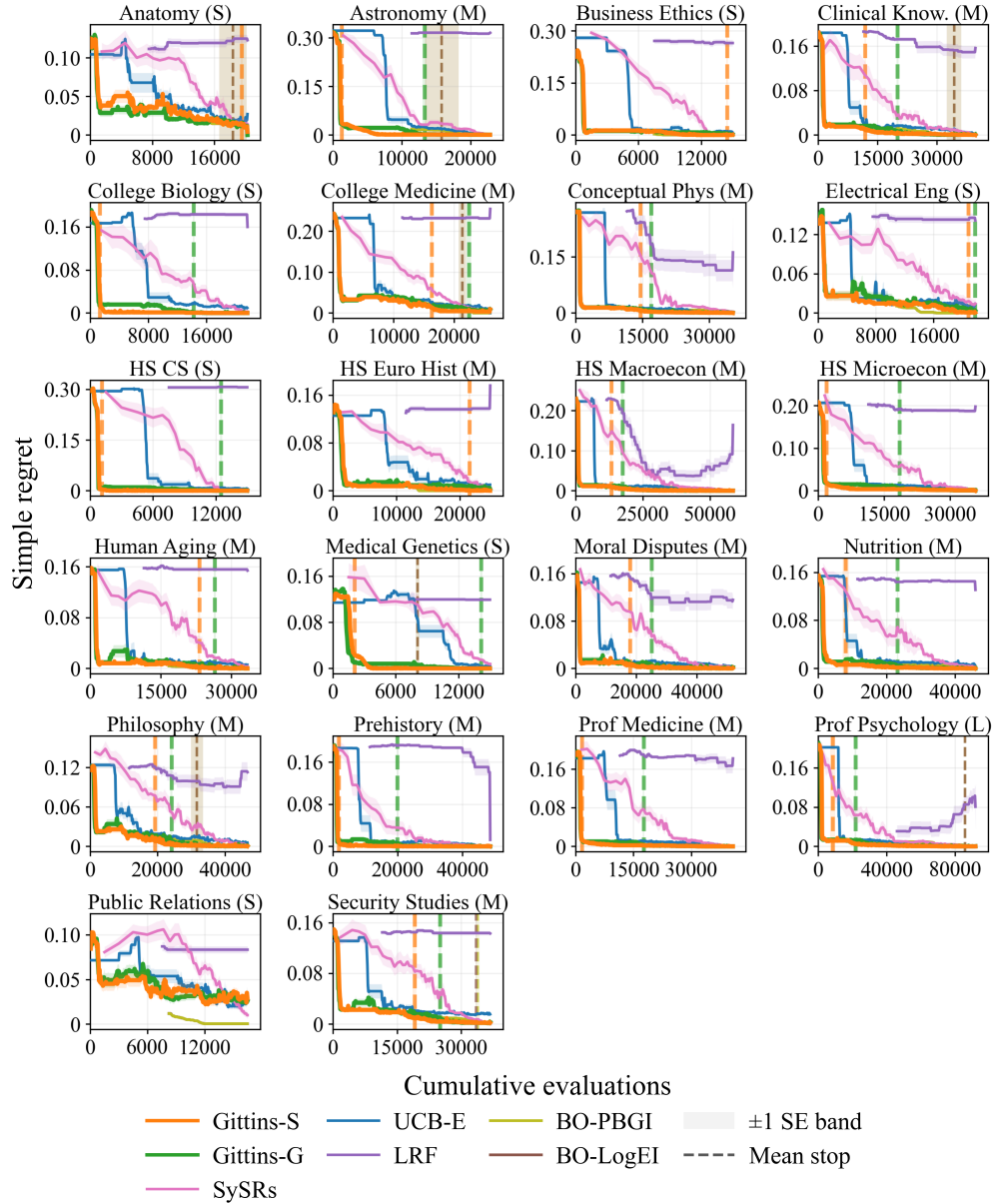


Figure C.7: MMLU unit-cost results for medium-difficulty subjects (medium prior bucket). Simple regret is plotted against cumulative evaluations. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF 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 Bayesian optimization performs well when the arm-level configuration structure is informative. LRF is generally less effective. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.

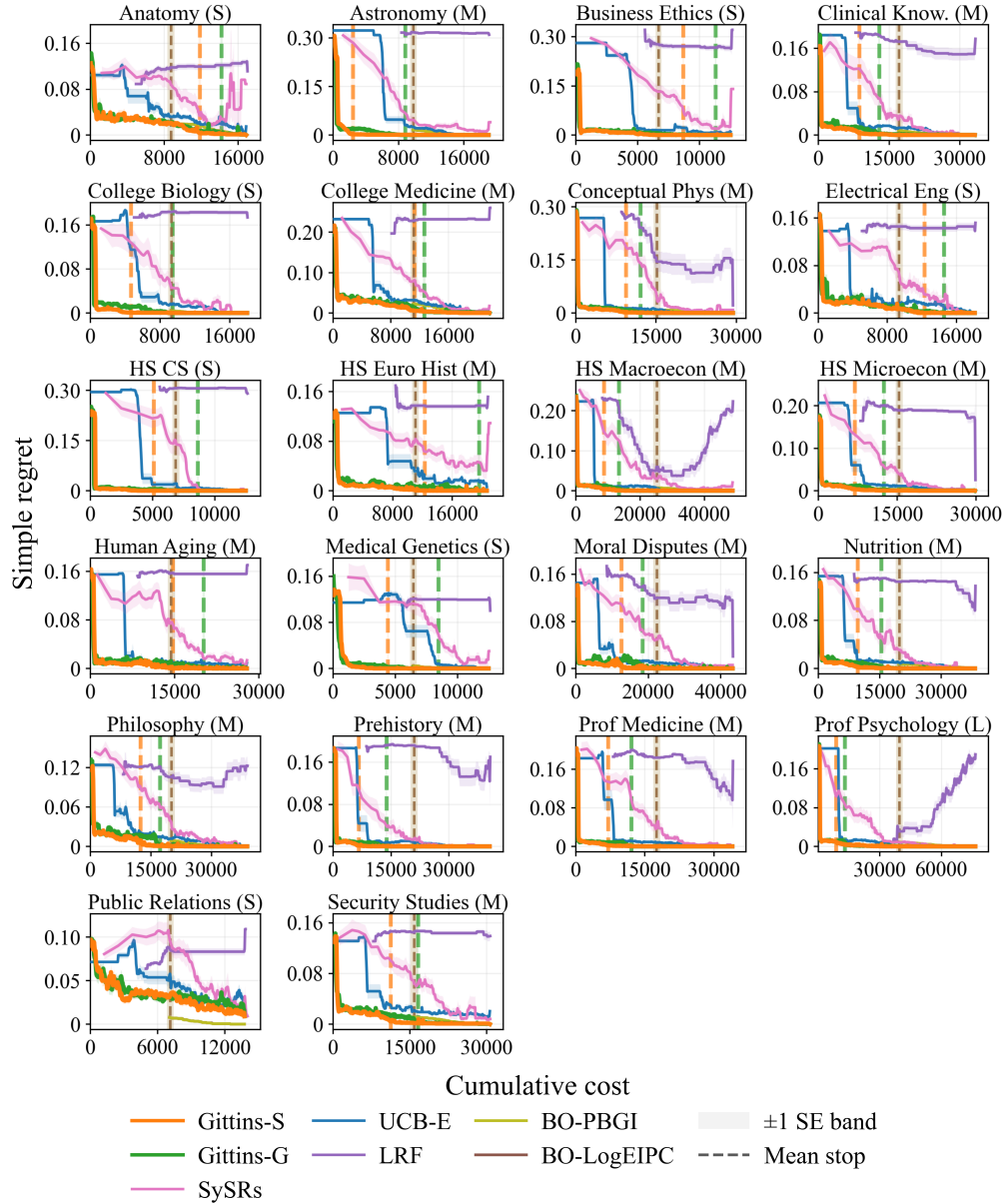


Figure C.8: MMLU cost-aware results for medium-difficulty subjects (medium prior bucket). Simple regret is plotted against cumulative cost. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ 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 Bayesian optimization is competitive on selected subjects. LRF remains less stable and typically underperforms. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.

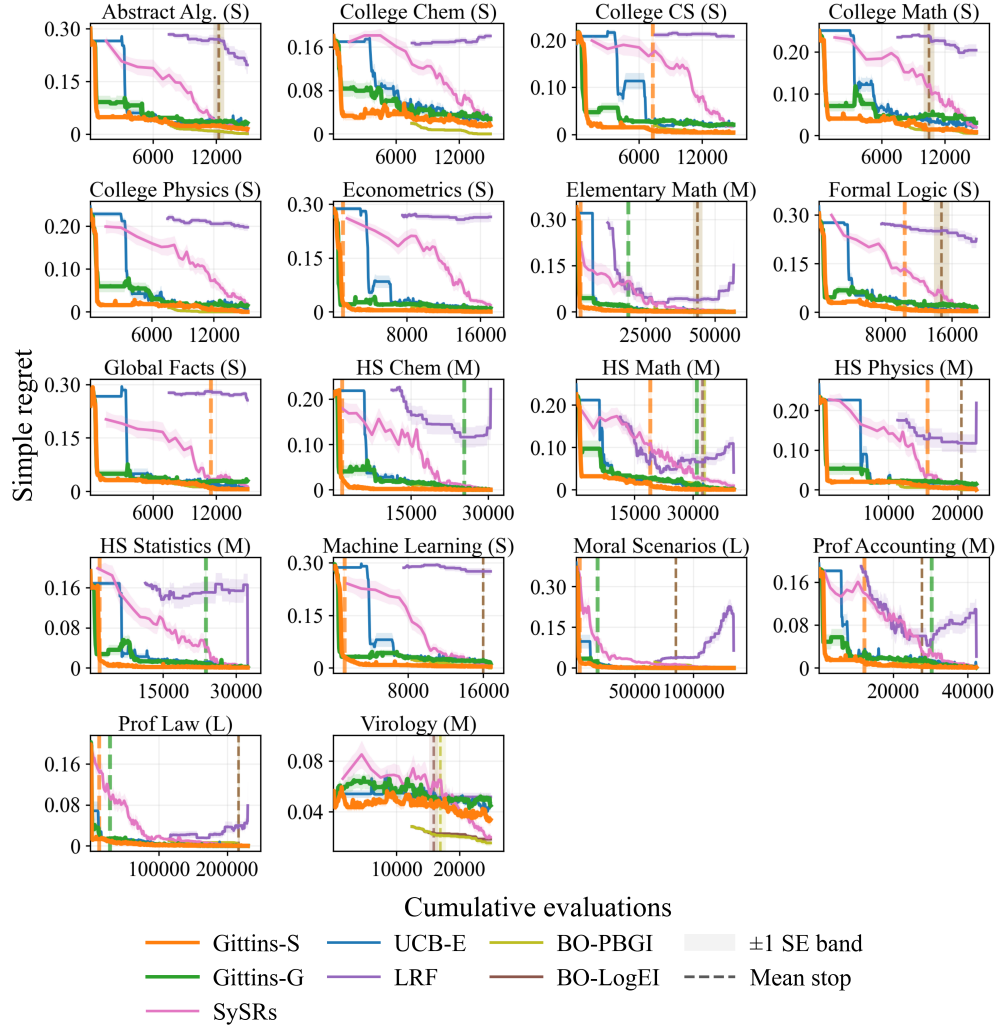


Figure C.9: MMLU unit-cost results for hard subjects (low prior bucket). Simple regret is plotted against cumulative evaluations. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ 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. Bayesian optimization is strong on a few subjects but is not uniformly dominant, while LRF is substantially less reliable. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.

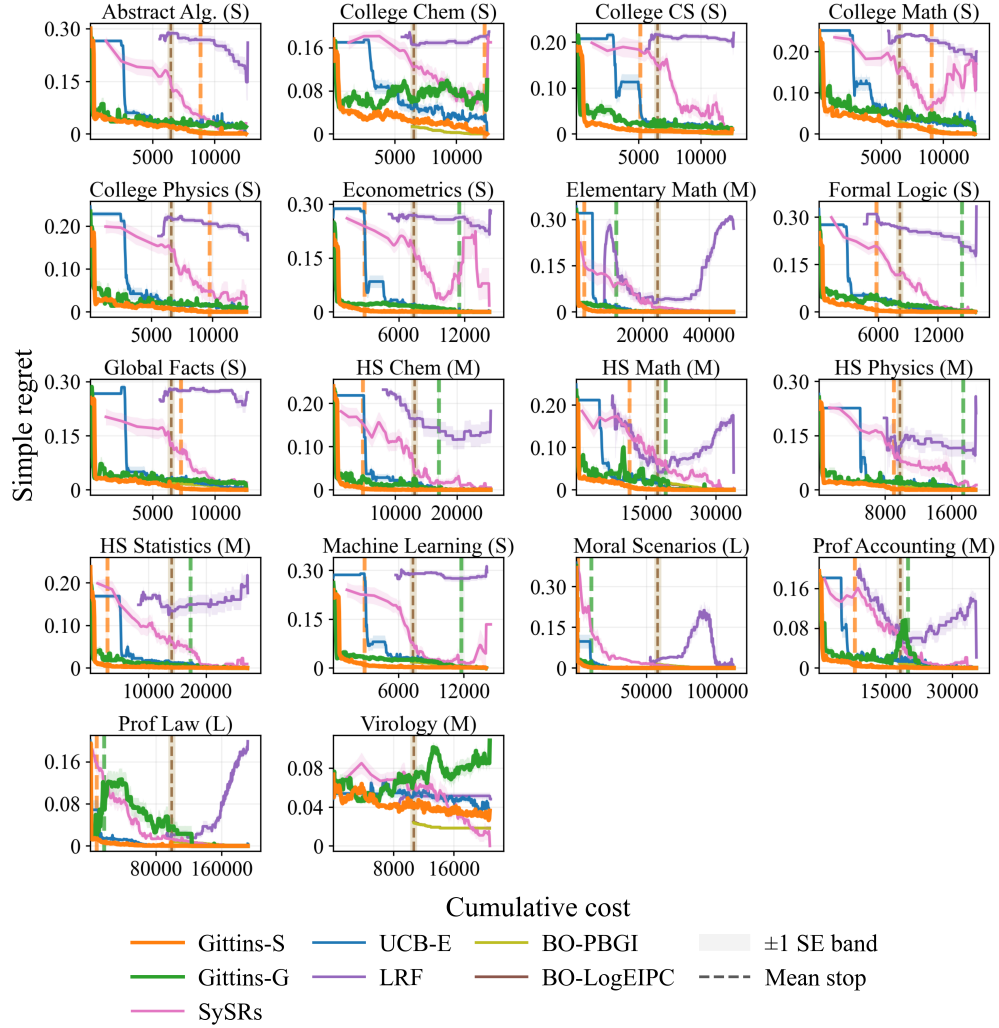


Figure C.10: MMLU cost-aware results for hard subjects (low prior bucket). Simple regret is plotted against cumulative cost. Subplot labels S, M, and L indicate small, medium, and large subject matrices, corresponding to batch sizes $B = 2$, $B = 4$, and $B = 8$, respectively; LRF uses $B = 32$ throughout. 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. Bayesian optimization performs well on selected tasks, while LRF remains the least reliable baseline. Compared with SySRs, Gittins-S generally enters the low-regret regime earlier across these subjects.

C.4.3 Computational Overhead Comparison

Timing protocol. Wall-clock timing measurements report the total runtime of each allocation policy over a completed simulated evaluation run. Since the response matrices are precomputed, each evaluation is implemented as an array lookup rather than an LLM inference call. Therefore, the reported runtime reflects the computational overhead of the allocation procedure itself, including method-specific setup and the repeated online computation required during the run. This includes confidence-bound computation for UCB-E, low-rank-factorization updates for UCB-E-LRF, similarity-based updates for SySRs, Gittins-index computation for the Gittins variants, and Gaussian-process fitting and acquisition computation for the Bayesian optimization baselines. These measurements should be interpreted as simulation-time algorithmic overhead rather than model-serving or benchmark-construction cost.

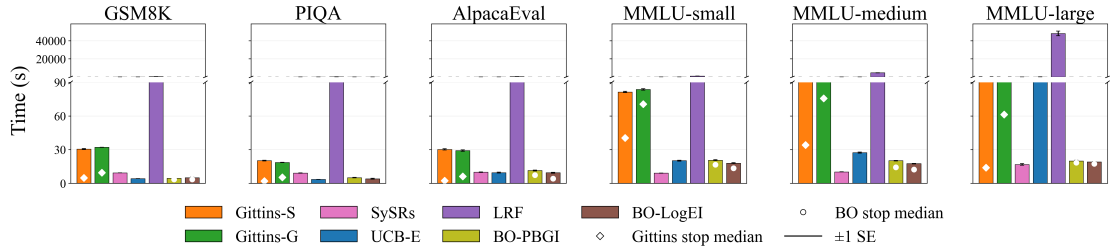


Figure C.11: Total wall-clock runtime under the unit-cost setting. Each panel reports the total runtime for a completed simulated allocation run on GSM8K, PIQA, AlpacaEval, MMLU-small, MMLU-medium, and MMLU-large. Bars show medians across completed runs, with black error bars indicating ± 1 standard error. Hollow diamonds mark median Gittins stopping times, and hollow circles mark median Bayesian optimization stopping times, estimated by scaling each run’s total runtime by the fraction of the evaluation budget used at stopping. UCB-E and Gittins use $B = 8$ on GSM8K/PIQA/AlpacaEval and $B = 2, 4, 8$ on MMLU-small/medium/large, respectively; UCB-E-LRF uses $B = 32$ throughout. Bayesian optimization baselines correspond to PBGI run with unit costs and LogEI. The broken vertical axis is used because UCB-E-LRF is much slower on larger MMLU tasks. Overall, Gittins remains close to UCB-E except on the larger MMLU settings, while UCB-E-LRF incurs substantially larger overhead. SySRs remains computationally light across the benchmark groups. The Bayesian optimization baselines are also computationally light in these total-runtime measurements, partly because their configuration-level protocol requires fewer sequential allocation decisions than the batch-level bandit methods.

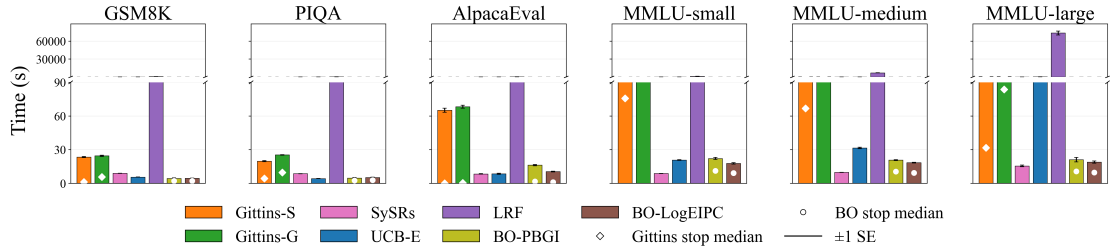


Figure C.12: Total wall-clock runtime under the cost-aware setting. Each panel reports the total runtime for a completed simulated allocation run on GSM8K, PIQA, AlpacaEval, MMLU-small, MMLU-medium, and MMLU-large. Bars show medians across completed runs, with black error bars indicating ± 1 standard error. Hollow diamonds mark median Gittins stopping times, and hollow circles mark median Bayesian optimization stopping times, estimated by scaling each run’s total runtime by the fraction of the cost budget used at stopping. UCB-E and Gittins use $B = 8$ on GSM8K/PIQA/AlpacaEval and $B = 2, 4, 8$ on MMLU-small/medium/large, respectively; UCB-E-LRF uses $B = 32$ throughout. Bayesian optimization baselines correspond to PBGI and LogEIPC. The broken vertical axis is used because UCB-E-LRF is much slower on larger MMLU tasks. Overall, Gittins remains close to UCB-E except on the larger MMLU settings, while UCB-E-LRF incurs substantially larger overhead. SySRs remains computationally light across the benchmark groups. The Bayesian optimization baselines are also computationally light in these total-runtime measurements, partly because their configuration-level protocol requires fewer sequential allocation decisions than the batch-level bandit methods.

APPENDIX D

SUPPLEMENT TO CHAPTER 5

D.1 Theory

This section supplies the proofs of the two theoretical results underlying the one-dimensional reduction in Theorems 14 and 16. We first show, by backward induction, that a lumping map preserves the local value function. We then verify that the terminal-value signal map is such a lumping map and that the resulting scalar process is a time-heterogeneous Gaussian random walk.

Theorem 14 (Value preservation under lumping). *Let (X_t, p_t, r_t, c_t) and $(\bar{X}_t, \bar{p}_t, \bar{r}_t, \bar{c}_t)$ be two time-heterogeneous Markov chains related by a lumping map $(f_t)_{t=0}^T$, and let $V_{t,\text{loc}}^*$ and $\bar{V}_{t,\text{loc}}^*$ be their respective local value functions. Then*

$$V_{t,\text{loc}}^*(x; \alpha) = \bar{V}_{t,\text{loc}}^*(f_t(x); \alpha) \quad (5.8)$$

for every t , $x \in X_t$, and $\alpha \in \mathbb{R}$.

Proof of Theorem 14. Fix $\alpha \in \mathbb{R}$. We prove by backward induction on the local stage t that

$$V_{t,\text{loc}}^*(x; \alpha) = \bar{V}_{t,\text{loc}}^*(f_t(x); \alpha) \quad \text{for all } x \in X_t. \quad (\text{D.1})$$

Base case ($t = T$). At the terminal stage, the two available values are the outside option and the terminal go payoff. Reward and cost preservation give

$$\begin{aligned} V_{T,\text{loc}}^*(x; \alpha) &= \max\{\alpha, r_T(x) - c_T(x)\} \\ &= \max\{\alpha, \bar{r}_T(f_T(x)) - \bar{c}_T(f_T(x))\} \\ &= \bar{V}_{T,\text{loc}}^*(f_T(x); \alpha). \end{aligned} \quad (\text{D.2})$$

Inductive step. Assume the identity holds at stage $t + 1$. For $x \in X_t$, the Bellman recursion for the larger chain is

$$V_{t,\text{loc}}^*(x; \alpha) = \max \left\{ \alpha, r_t(x) - c_t(x) + \mathbb{E}_{X' \sim p_t(\cdot | x)} [V_{t+1,\text{loc}}^*(X'; \alpha)] \right\}. \quad (\text{D.3})$$

By the induction hypothesis and the transition consistency of the lumping map, the pushforward of $p_t(\cdot | x)$ under f_{t+1} is $\bar{p}_t(\cdot | f_t(x))$, so

$$\mathbb{E}_{X' \sim p_t(\cdot | x)} [V_{t+1,\text{loc}}^*(X'; \alpha)] = \mathbb{E}_{\bar{X}' \sim \bar{p}_t(\cdot | f_t(x))} [\bar{V}_{t+1,\text{loc}}^*(\bar{X}'; \alpha)]. \quad (\text{D.4})$$

Reward and cost preservation therefore yield

$$V_{t,\text{loc}}^*(x; \alpha) = \max \left\{ \alpha, \bar{r}_t(f_t(x)) - \bar{c}_t(f_t(x)) + \mathbb{E}_{\bar{X}' \sim \bar{p}_t(\cdot | f_t(x))} [\bar{V}_{t+1,\text{loc}}^*(\bar{X}'; \alpha)] \right\}. \quad (\text{D.5})$$

The right-hand side is $\bar{V}_{t,\text{loc}}^*(f_t(x); \alpha)$, completing the induction. $\square$

Theorem 16 (Terminal-value signal reduction). *Consider the fully observed LGSSM in Definition 12, and suppose its rewards and costs factor through the terminal-value signal maps in Definition 15. Then $(f_t)_{t=0}^T$ is a lumping map, and the reduced process is the time-heterogeneous Gaussian random walk*

$$s_{t+1} = s_t + \zeta_{t+1}, \quad \zeta_{t+1} \sim \mathcal{N}(0, \sigma_{t+1}^2), \quad (\text{5.11})$$

where the increments are independent and

$$\sigma_{t+1}^2 = \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t+1,T} \mathbf{Q}_t \boldsymbol{\Phi}_{t+1,T}^\top \boldsymbol{\ell}. \quad (\text{5.12})$$

Consequently, the original LGSSM and the scalar random walk have identical local value functions and Gittins indices at corresponding states.

Proof of Theorem 16. By definition, $\boldsymbol{\Phi}_{t,T} = \boldsymbol{\Phi}_{t+1,T} \mathbf{F}_t$. Using the LGSSM dynamics,

$$\mathbf{x}_{t+1} = \mathbf{F}_t \mathbf{x}_t + \boldsymbol{\varepsilon}_{t+1}, \quad (\text{D.6})$$

we have

$$\begin{aligned}
s_{t+1} &= \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t+1,T} \mathbf{x}_{t+1} \\
&= \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t+1,T} \mathbf{F}_t \mathbf{x}_t + \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t+1,T} \boldsymbol{\varepsilon}_{t+1} \\
&= \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t,T} \mathbf{x}_t + \zeta_{t+1} \\
&= s_t + \zeta_{t+1},
\end{aligned} \tag{D.7}$$

where $\zeta_{t+1} := \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t+1,T} \boldsymbol{\varepsilon}_{t+1}$. The original innovations are independent and Gaussian, so the transformed increments are independent and

$$\zeta_{t+1} \sim \mathcal{N}(0, \boldsymbol{\ell}^\top \boldsymbol{\Phi}_{t+1,T} \mathbf{Q}_t \boldsymbol{\Phi}_{t+1,T}^\top \boldsymbol{\ell}). \tag{D.8}$$

Thus the conditional distribution of s_{t+1} given $\mathbf{x}_t$ depends on $\mathbf{x}_t$ only through $s_t = f_t(\mathbf{x}_t)$, which verifies transition consistency for $(f_t)_{t=0}^T$. Reward and cost preservation hold by the theorem's factorization assumption. Therefore (f_t) is a lumping map, and equality of the local value functions and Gittins indices follows from Theorem 14. $\square$

D.2 Fast Fourier Transform Details

Derivation of the EI-kernel convolution formula. The reduced random walk satisfies $x_{t+1} \mid x_t = x \sim \mathcal{N}(x, \sigma_{t+1}^2)$. Thus the Bellman continuation expectation at stage t is

$$\mathbb{E} \left[\widehat{Q}_{t+1,\text{loc}}^+(x_{t+1}) \mid x_t = x \right] = \int_{-\infty}^{\infty} \widehat{Q}_{t+1,\text{loc}}^+(s) \phi_{t+1}(x-s) \, ds = (\widehat{Q}_{t+1,\text{loc}}^+ * \phi_{t+1})(x). \tag{D.9}$$

Here we derive the closed-form expression for the Bellman convolution term $(\widehat{Q}_{t+1,\text{loc}}^+ * \phi_{t+1})(x)$ when $\widehat{Q}_{t+1,\text{loc}}^+$ is represented in the piecewise-linear basis

$$\widehat{Q}_{t+1,\text{loc}}^+(x) = \widehat{Q}_{t+1,\text{loc}}^+(\xi_0) + m_1(x - \xi_0) + \sum_{i=1}^{P-1} (m_{i+1} - m_i) \max(x - \xi_i, 0), \tag{D.10}$$

and ϕ_{t+1} is the density of $\mathcal{N}(0, \sigma_{t+1}^2)$:

$$\begin{aligned}
& (\hat{Q}_{t+1, \text{loc}}^+ * \phi_{t+1})(x) \\
&= \int_{-\infty}^{\infty} \hat{Q}_{t+1, \text{loc}}^+(s) \phi_{t+1}(x-s) \, ds \\
&= \hat{Q}_{t+1, \text{loc}}^+(\xi_0) \int_{-\infty}^{\infty} \phi_{t+1}(x-s) \, ds + m_1 \int_{-\infty}^{\infty} (s - \xi_0) \phi_{t+1}(x-s) \, ds \\
&\quad + \sum_{i=1}^{P-1} (m_{i+1} - m_i) \int_{-\infty}^{\infty} \max(s - \xi_i, 0) \phi_{t+1}(x-s) \, ds \\
&= \hat{Q}_{t+1, \text{loc}}^+(\xi_0) + m_1(x - \xi_0) + \sum_{i=1}^{P-1} (m_{i+1} - m_i) \int_{-\infty}^{\infty} \max(x - \xi_i - u, 0) \phi_{t+1}(u) \, du \\
&= \hat{Q}_{t+1, \text{loc}}^+(\xi_0) + m_1(x - \xi_0) + \sum_{i=1}^{P-1} (m_{i+1} - m_i) \mathbb{E}_{\xi \sim \mathcal{N}(0, \sigma_{t+1}^2)} \max(x - \xi_i - \xi, 0) \\
&= \hat{Q}_{t+1, \text{loc}}^+(\xi_0) + m_1(x - \xi_0) + \sum_{i=1}^{P-1} (m_{i+1} - m_i) \text{EI}_{t+1}(x - \xi_i), \tag{D.11}
\end{aligned}$$

where

$$\text{EI}_{t+1}(z) := \mathbb{E}_{\xi \sim \mathcal{N}(0, \sigma_{t+1}^2)} \max(z - \xi, 0) = z \Phi\left(\frac{z}{\sigma_{t+1}}\right) + \sigma_{t+1} \phi\left(\frac{z}{\sigma_{t+1}}\right). \tag{D.12}$$

The resulting expression is used in Equation (5.23) to evaluate Bellman updates on the grid via FFT.

D.3 Experimental Details

All experiments are implemented in JAX (Bradbury et al., 2018).

Synthetic model. The thesis-section experiments use model-matched synthetic linear Gaussian state-space model instances. Each arm is a discretized Matérn-3/2 process with unit time step. We independently draw an arm-specific lengthscale uniformly from $[2, 20]$ and output scale uniformly from $[0.1, 0.5]$; these arm models are held fixed across the sampled worlds. The three-dimensional initial state

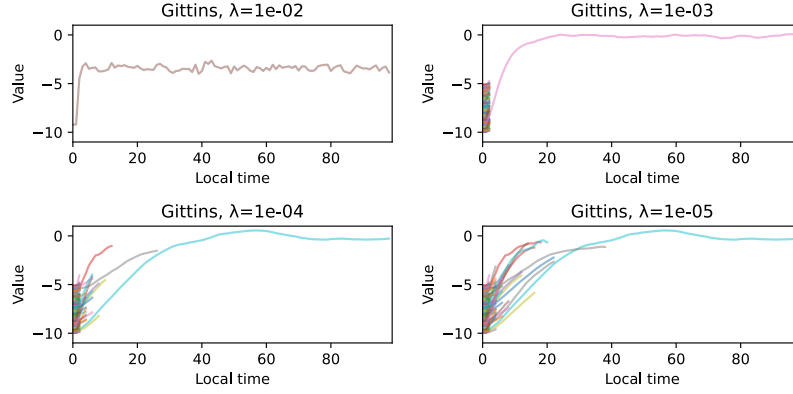


Figure D.1: Policy rollouts for the Gittins policy applied to a synthetic Markov chain selection task. The cost scaling parameter λ reflects the explore/exploit tradeoff.

represents the asymptotic value, initial value, and derivative. For each arm, their prior means are drawn uniformly from $[-5, 0]$, $[-10, -5]$, and the singleton $\{0\}$, respectively, after which the initial state is sampled from a Gaussian distribution with the resulting mean and identity covariance. The policy observes the staged feedback generated by this model and uses the corresponding model parameters. We vary the number of arms and the horizon, using $N \in \{100, 500, 1000\}$ and $T \in \{10, 50, 100\}$. For each (N, T) setting, we sample 16 independent worlds and perform one observation-noise rollout per world. The objective vector $(1, 0, 0)^\top$ makes the arm's asymptotic value the terminal quantity of interest.

Posterior inference. For every arm, the Kalman filter is initialized with the same prior mean used to generate that arm's initial state and covariance $\mathbf{I}_3$. Thus, both the state-space dynamics and the prior used by the policy match the data-generating process. The posterior is updated after each scalar observation, and the resulting sequence of posterior states is supplied to the Gittins-index computation.

Bayesian regret. Within each world, the reference value is the largest posterior mean asymptotic value after conditioning on the complete length- T trajectory of every arm. The selected arm is the first arm that the policy evaluates through the full horizon, and its regret is the difference between this reference value and that arm’s corresponding posterior mean asymptotic value. We report the mean regret across the 16 worlds. Evaluation cost is the total number of arm-level observations made before an arm is selected; together, mean cost and mean regret produce the Pareto frontiers shown in Fig. 5.3. Because the experiments are model-matched, these results are intended to isolate the behavior of the Gittins policy under the assumptions used in the theory.

Fast Fourier Transform. We choose the grid size to be $2^{10} + 1 = 1025$.

Gittins index algorithm. Our Gittins index algorithm uses a cost-scaling factor λ to represent the tradeoff between terminal reward and evaluation cost. In the synthetic experiments, we sweep $\lambda \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. At each decision, the policy evaluates the arm with the largest Gittins index at its current arm-local time and stops when one arm has been evaluated through the horizon. The Gittins indices are calculated from the model-matched state-space posterior used by the synthetic data-generating process, with one observation per stage and a constant per-observation cost λ .

Baselines. We consider two simple baselines.

1. Random search. N random arms are run to completion, and the best arm is selected. We run the policy for 10,000 random seeds and plot the mean and

variance.

2. Successive halving. Given a maximum horizon T and halving rate $\eta \in \{2, 3, 4, 5\}$, successive halving initializes a pool of arms, evaluates each for an initial budget, and repeatedly discards a $1 - 1/\eta$ fraction of arms at each round until one remains (Jamieson and Talwalkar, 2016; Li et al., 2018).

Hardware. Experiments were run with 64 bit precision on NVIDIA GPUs, exploiting vectorised parallelism.

BIBLIOGRAPHY

- S. Aalto, U. Ayesta, and R. Righter. On the Gittins Index in the $M/G/1$ Queue. *Queueing Systems*, 2009. Cited on page 13.
- S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Unexpected improvements to expected improvement for bayesian optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, 2023. Cited on pages 28, 29, 36, 39, 138, 139.
- M. R. Aminian, V. Manshadi, and R. Niazadeh. Markovian Search with Socially Aware Constraints. *Management Science*, 2024. Cited on pages 22, 126, 127.
- R. Astudillo, D. Jiang, M. Balandat, E. Bakshy, and P. I. Frazier. Multi-step Budgeted Bayesian Optimization with Unknown Evaluation Costs. *Advances in Neural Information Processing Systems*, 2021. Cited on pages 12, 17, 23, 26, 28, 38, 49, 114, 133, 137, 141, 144, 161, 164, 173.
- J.-Y. Audibert and S. Bubeck. Best arm identification in multi-armed bandits. *COLT-23th Conference on learning theory-2010*, 13–p, 2010. Cited on pages 66, 79.
- E. Bakshy, L. Dworkin, B. Karrer, K. Kashin, B. Letham, A. Murthy, and S. Singh. AE: A domain-agnostic platform for adaptive experimentation. *Conference on neural information processing systems*, pages 1–8, 2018. Cited on page 40.

- M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020. Cited on pages 17, 27, 132, 133, 160.
- T. Başar and P. Bernhard. *H^∞ -optimal Control and Related Minimax Design Problems: A Dynamic Game Approach*. Springer, 2008. Cited on page 131.
- S. Belakaria, J. R. Doppa, N. Fusi, and R. Sheth. Bayesian Optimization over Iterative Learners with Structured Responses: A Budget-aware Planning Approach. *Artificial Intelligence and Statistics*, 2023. Cited on page 12.
- D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999. Cited on page 131.
- Y. Bisk, R. Zellers, J. Gao, Y. Choi, et al. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI conference on artificial intelligence*, volume 34 of number 05, pages 7432–7439, 2020. Cited on pages 61, 77, 204.
- J. F. Bonnans and A. Shapiro. *Perturbation Analysis of Optimization Problems*. Springer, 2013. Cited on page 131.
- S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. Cited on page 129.
- J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, et al. JAX: composable transformations of Python+ NumPy programs, 2018. Cited on page 243.

- P. Buchholz. Exact and ordinary lumpability in finite Markov chains. *Journal of applied probability*, 31(1):59–75, 1994. Cited on page 97.
- J. M. Cashore, L. Kumarga, and P. I. Frazier. Multi-step Bayesian optimization for one-dimensional feasibility determination. *arXiv preprint arXiv:1607.03195*, 2016. Cited on page 39.
- S. E. Chick and P. Frazier. Sequential sampling with economics of selection procedures. *Management Science*, 58(3):550–569, 2012. Cited on pages 39, 41.
- S. R. Chowdhury and A. Gopalan. On Kernelized Multi-armed Bandits. *International Conference on Machine Learning*, 2017. Cited on page 17.
- K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. Cited on pages 61, 77, 204.
- J. M. Danskin. *The Theory of Max-min and its Application to Weapons Allocation Problems*. Springer, 1967. Cited on page 131.
- X. Dong, L. Liu, K. Musial, and B. Gabrys. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3634–3646, 2021. Cited on pages 53, 54.
- L. Doval. Whether or Not to Open Pandora’s Box. *Journal of Economic Theory*, 2018. Cited on page 13.

- I. Dumitriu, P. Tetali, and P. Winkler. On Playing Golf with Two Balls. *SIAM Journal on Discrete Mathematics*, 2003. Cited on page 13.
- I. Dumitriu, P. Tetali, and P. Winkler. On playing golf with two balls. *SIAM Journal on Discrete Mathematics*, 16(4):604–615, 2003. Cited on pages 72, 202.
- D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable global optimization via local Bayesian optimization. *Advances in Neural Information Processing Systems*, 2019. Cited on pages 31, 137.
- Z. Z. Foumani and R. Bostanabad. Constrained multi-fidelity Bayesian optimization with automatic stop condition. *CoRR*, abs/2503.01126, 2025. DOI: 10.48550/ARXIV.2503.01126. arXiv: 2503.01126. URL: [HTTPS://DOI.ORG/10.48550/ARXIV.2503.01126](https://doi.org/10.48550/ARXIV.2503.01126). Cited on page 41.
- P. Frazier and W. B. Powell. The knowledge-gradient stopping rule for ranking and selection. *2008 Winter Simulation Conference*, pages 305–312. IEEE, 2008. Cited on pages 36, 41.
- P. I. Frazier. Bayesian Optimization, *Recent Advances in Optimization and Modeling of Contemporary Problems*. 2018. Cited on page 15.
- P. I. Frazier, W. Powell, and S. Dayanik. The Knowledge-gradient Policy for Correlated Normal Beliefs. *INFORMS Journal on Computing*, 2009. Cited on page 28.
- R. Garnett. *Bayesian Optimization*. 2023. Cited on pages 14, 15.

- R. Garnett. *Bayesian optimization*. Cambridge University Press, 2023. Cited on pages 38, 40.
- E. Gergatsouli and C. Tzamos. Weitzman’s rule for pandora’s box with correlations. *Advances in Neural Information Processing Systems*, 36:12644–12664, 2023. Cited on pages 43, 169.
- J. Gittins, K. Glazebrook, and R. Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011. Cited on pages 2, 4, 64.
- J. C. Gittins. Bandit Processes and Dynamic Allocation Indices. *Journal of the Royal Statistical Society, Series B: Statistical Methodology*, 1979. Cited on page 111.
- J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 41(2):148–164, 1979. Cited on page 89.
- J. C. Gittins, K. D. Glazebrook, and R. R. Weber. *Multi-Armed Bandit Allocation Indices*. Wiley, 2011. Cited on pages 13, 18, 20.
- K. D. Glazebrook and J. Niño-Mora. Parallel Scheduling of Multiclass $M/M/m$ Queues: Approximate and Heavy-Traffic Optimization of Achievable Performance. *Operations Research*, 2001. Cited on page 13.
- A. Gupta, H. Jiang, Z. Scully, and S. Singla. The Markovian Price of Information. *Integer Programming and Combinatorial Optimization*, 2019. Cited on page 13.

- E. Habba, O. Arviv, I. Itzhak, Y. Perlitz, E. Bandel, L. Choshen, M. Shmueli-Scheuer, and G. Stanovsky. DOVE: A Large-Scale Multi-Dimensional Predictions Dataset Towards Meaningful LLM Evaluation. *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11744–11763. Association for Computational Linguistics, 2025. DOI: 10.18653/v1/2025.FINDINGS-ACL.611. URL: [HTTPS://ACLANTHOLOGY.ORG/2025.FINDINGS-ACL.611/](https://aclanthology.org/2025.FINDINGS-ACL.611/). Cited on pages 77, 206.
- J. Hartikainen and S. Särkkä. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. *2010 IEEE international workshop on machine learning for signal processing*, pages 379–384. IEEE, 2010. Cited on page 104.
- A. C. Harvey. Forecasting, structural time series models and the Kalman filter, 1990. Cited on page 94.
- D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. Cited on pages 61, 77.
- J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. *Advances in Neural Information Processing Systems*, 2014. Cited on pages 134, 135.
- S. Hu, H. Wang, Z. Dai, B. K. H. Low, and S. H. Ng. Adjusted expected improvement for cumulative regret minimization in noisy bayesian optimization. *Journal of Machine Learning Research*, 26(46):1–33, 2025. Cited on page 154.

- H. Ishibashi, M. Karasuyama, I. Takeuchi, and H. Hino. A stopping criterion for Bayesian optimization by the gap of expected minimum simple regrets. *International Conference on Artificial Intelligence and Statistics*, pages 6463–6497. PMLR, 2023. Cited on pages 36, 41, 51, 162.
- K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. *Artificial intelligence and statistics*, pages 240–248. PMLR, 2016. Cited on pages 106, 246.
- S. Jiang, D. Jiang, M. Balandat, B. Karrer, J. R. Gardner, and R. Garnett. Efficient Nonmyopic Bayesian Optimization via One-shot Multi-step Trees. *Advances in Neural Information Processing Systems*, 2020. Cited on pages 12, 28, 133, 134.
- D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998. Cited on page 103.
- A.-K. Katta and J. Sethuraman. A note on bandits with a twist. *SIAM Journal on Discrete Mathematics*, 18(1):110–113, 2004. Cited on page 94.
- R. Kleinberg, B. Waggoner, and E. G. Weyl. Descending price optimally coordinates search. *arXiv preprint arXiv:1603.07682*, 2016. Cited on pages 21, 43, 121.
- E. H. Lee, D. Eriksson, V. Perrone, and M. Seeger. A Nonmyopic Approach to Cost-constrained Bayesian Optimization. *Uncertainty in Artificial Intelligence*, 2021. Cited on page 12.

- E. H. Lee, V. Perrone, C. Archambeau, and M. Seeger. Cost-aware Bayesian optimization. *ICML Workshop on Automated Machine Learning*, 2020. Cited on pages 28, 38, 134.
- L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. Cited on pages 88, 106, 246.
- S. Li, K. Li, and W. Li. “Why Not Looking backward?” A Robust Two-Step Method to Automatically Terminate Bayesian Optimization. *Advances in Neural Information Processing Systems*, 36:43435–43446, 2023. Cited on pages 36, 41.
- X. Li, T. Zhang, Y. Dubois, R. Taori, I. Gulrajani, C. Guestrin, P. Liang, and T. B. Hashimoto. AlpacaEval: An Automatic Evaluator of Instruction-following Models, 2023. URL: [HTTPS://GITHUB.COM/TATSU-LAB/ALPACA_EVAL](https://github.com/Tatsu-Lab/alpaca_eval). Cited on pages 77, 205.
- Y. L. Li, T. G. J. Rudner, and A. G. Wilson. A Study of Bayesian Neural Network Surrogates for Bayesian Optimization. *International Conference on Learning Representations*, 2024. Cited on pages 133, 136, 137.
- M. Lifshits. Lectures on Gaussian processes, *Lectures on Gaussian Processes*, pages 1–117. Springer, 2012. Cited on page 160.
- S. W. Linderman, A. C. Miller, R. P. Adams, D. M. Blei, L. Paninski, and M. J. Johnson. Recurrent switching linear dynamical systems. *arXiv preprint arXiv:1610.08466*, 2016. Cited on page 94.

- M. Locatelli. Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, 10:57–76, 1997. Cited on page 41.
- R. Lorenz, R. P. Monti, I. R. Violante, A. A. Faisal, C. Anagnostopoulos, R. Leech, and G. Montana. Stopping criteria for boosting automatic experimental design using real-time fMRI with Bayesian optimization. *arXiv preprint arXiv:1511.07827*, 2015. Cited on pages 36, 41.
- Z. Lyu, C. Nejma, T. Wegel, F. Yang, and F. E. Dorner. Cutting LLM Evaluation Costs with SySRs: A Bandit Algorithm that Provably Exploits Model Similarity. *arXiv preprint arXiv:2606.07726*, 2026. Cited on pages 63, 79, 84.
- A. Makarova, H. Shen, V. Perrone, A. Klein, J. B. Faddoul, A. Krause, M. Seeger, and C. Archambeau. Automatic termination for hyperparameter optimization. *International Conference on Automated Machine Learning*, pages 7–1. PMLR, 2022. Cited on pages 36, 41, 51, 162.
- R. Martinez-Cantin. Bayesian optimization with adaptive kernels for robot control. *International Conference on Robotics and Automation*, 2017. Cited on pages 12, 35.
- P. Milgrom and I. Segal. Envelope Theorems for Arbitrary Choice Sets. *Econometrica*, 2002. Cited on pages 126, 129, 130.
- J. Moćkus. On Bayesian methods for seeking the extremum. *IFIP Technical Conference on Optimization Techniques*, pages 400–404. Springer, 1974. Cited on page 103.

- V. Nguyen, S. Gupta, S. Rana, C. Li, and S. Venkatesh. Regret for expected improvement over the best-observed value and stopping condition. *Asian conference on machine learning*, pages 279–294. PMLR, 2017. Cited on pages 36, 37, 41, 45.
- C. Oh, J. Tomczak, E. Gavves, and M. Welling. Combinatorial bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems*, 2019. Cited on pages 31, 136.
- E. Persky. *Exploration and Exploitation: From Bandits to Bayesian Optimisation*. MPhil Thesis, University of Cambridge, 2021. Cited on page 13.
- F. M. Polo, R. Xu, L. Weber, M. Silva, O. Bhardwaj, L. Choshen, A. F. de Oliveira, Y. Sun, and M. Yurochkin. Efficient multi-prompt evaluation of llms. *Advances in Neural Information Processing Systems*, 37:22483–22512, 2024. Cited on pages 61, 64, 66.
- C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. Cited on pages 15, 142.
- C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. Cambridge University Press, 2006. Cited on page 38.
- S. Sarkka, A. Solin, and J. Hartikainen. Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013. Cited on page 104.

- Z. Scully, I. Grosz, and M. Harchol-Balter. The Gittins Policy is Nearly Optimal in the $M/G/k$ under Extremely General Conditions. *Measurement and Analysis of Computing Systems*, 2020. Cited on pages 13, 89.
- Z. Scully and A. Terenin. The Gittins Index: A Design Principle for Decision Making under Uncertainty, *Tutorials in Operations Research: Advances in Analytics and Operations Research: Improving Decisions to Secure the Future*, pages 28–70. INFORMS, 2025. Cited on pages 64, 72, 92–94, 96, 202.
- Y.-W. Shang and Y.-H. Qiu. A note on the extended Rosenbrock function. *Evolutionary Computation*, 2006. Cited on page 31.
- C. Shi, K. Yang, Z. Chen, J. Li, J. Yang, and C. Shen. Efficient prompt optimization through the lens of best arm identification. *Advances in Neural Information Processing Systems*, 37:99646–99685, 2024. Cited on pages 63, 66.
- S. Singla. The Price of Information in Combinatorial Optimization. *Symposium on Discrete Algorithms*, 2018. Cited on page 13.
- E. Snelson, Z. Ghahramani, and C. Rasmussen. Warped Gaussian Processes. *Advances in Neural Information Processing Systems*, 2003. Cited on page 28.
- J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 2012. Cited on pages 12, 16, 28, 35, 39, 61.

- N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009. Cited on pages 28, 33, 134, 162.
- K. Swersky, J. Snoek, and R. P. Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014. Cited on page 88.
- S. Takeno, H. Fukuoka, Y. Tsukada, T. Koyama, M. Shiga, I. Takeuchi, and M. Karasuyama. Multi-fidelity Bayesian Optimization with Max-value Entropy Search and its Parallelization. *International Conference on Machine Learning*, 2020. Cited on page 28.
- A. Terenin. *Gaussian Processes and Statistical Decision-making in Non-Euclidean Spaces*. PhD Thesis, Imperial College London, 2022. Cited on page 109.
- S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. Cited on page 94.
- J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014. Cited on page 53.
- Z. Wang and S. Jegelka. Max-value Entropy Search for Efficient Bayesian Optimization. *International Conference on Machine Learning*, 2017. Cited on pages 28, 31, 137.
- R. R. Weber. On the Gittins Index for Multiarmed Bandits. *Annals of Applied Probability*, 1992. Cited on page 13.

- M. Weitzman. *Optimal search for the best alternative*, volume 78 of number 8. Department of Energy, 1978. Cited on page 91.
- M. L. Weitzman. Optimal search for the best alternative. *Econometrica*, 1979. Cited on pages 2, 5, 13, 18, 20, 21, 42, 111, 114, 121.
- G. Welch, G. Bishop, et al. An introduction to the Kalman filter, 1995. Cited on page 104.
- M. West and J. Harrison. *Bayesian forecasting and dynamic models*. Springer, 1997. Cited on page 104.
- J. Wilson. Stopping bayesian optimization with probabilistic regret bounds. *Advances in Neural Information Processing Systems*, 37:98264–98296, 2024. Cited on pages 36, 41, 51, 162.
- J. Wu, S. Toscano-Palmerin, P. I. Frazier, and A. G. Wilson. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. *Uncertainty in Artificial Intelligence*, pages 788–798. PMLR, 2020. Cited on page 17.
- Q. Xie, R. Astudillo, P. Frazier, Z. Scully, and A. Terenin. Cost-aware Bayesian optimization via the Pandora’s Box Gittins index. *Advances in Neural Information Processing Systems*, 37:115523–115562, 2024. Cited on pages 11, 36, 39, 42, 43, 47, 48, 64, 89, 94, 161, 162, 170.
- Q. Xie, L. Cai, A. Terenin, P. I. Frazier, and Z. Scully. Cost-aware stopping for bayesian optimization. *arXiv preprint arXiv:2507.12453*, 2025. Cited on page 34.

- Q. Xie, Y. He, and N. Cao. Efficient Cost-Aware LLM Evaluation via Bayesian Bandit Gittins Indices. *Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning*, 2026. Cited on page 60.
- X. Yue and R. A. Kontar. Why Non-myopic Bayesian Optimization is Promising and How Far Should We Look-ahead? A Study via Rollout. *Artificial Intelligence and Statistics*, 2020. Cited on page 12.
- R. Zabounidis, A. Golatkar, M. Kleinman, A. Achille, W. Xia, and S. Soatto. REFORC: Adaptive Reward Prediction for Efficient Chain-of-Thought Reasoning. *arXiv preprint arXiv:2511.02130*, 2025. Cited on page 89.
- Y. Zhang, D. W. Apley, and W. Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific Reports*, 2020. Cited on pages 12, 35.
- H. Zhou, X. Ma, and M. B. Blaschko. A corrected expected improvement acquisition function under noisy observations. *Asian Conference on Machine Learning*, pages 1747–1762. PMLR, 2024. Cited on pages 36, 37, 41, 45.
- J. P. Zhou, C. K. Belardi, R. Wu, T. Zhang, C. P. Gomes, W. Sun, and K. Q. Weinberger. On speeding up language model evaluation. *The Thirteenth International Conference on Learning Representations*, 2025. URL: [HTTPS://OPENREVIEW.NET/FORUM?ID=3CVWO5DBZN](https://openreview.net/forum?id=3cvwo5dbzn). Cited on pages 61, 63, 66, 75, 77, 79, 80, 85, 204, 218.

L. Zimmer, M. Lindauer, and F. Hutter. Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021. Cited on page 53.