Open-Ended Task Discovery via Bayesian Optimization

Yuki Takezawa [1] **Masaki Adachi** [1]

Abstract

When applying Bayesian optimization (BO) to scientific workflow, a major yet often overlooked source of uncertainty is the *task* itself—namely, *what* to optimize and *how* to evaluate it—which can evolve as evidence accumulates. We introduce *Generate-Select-Refine* (GSR), a open-ended BO framework that alternates between task generation and task optimization. Starting from a user-provided seed task, GSR generates new tasks in a coarse-to-fine manner while a task-acquisition function schedules optimization. Asymptotically, it concentrates evaluations on the best task, incurring only logarithmic regret overhead relative to single-task BO. We apply GSR to new product development, chemical synthesis scaling, algorithm analysis, and patent repurposing, where it outperforms existing LLM-based optimizers.

1. Introduction

Bayesian optimization (BO; (Garnett, 2023)) is a sample-efficient black-box optimizer, widely used in domains such as drug discovery (Gómez-Bombarelli et al., 2018; Pyzer-Knapp, 2018) and materials science (Liang et al., 2021; Adachi et al., 2024a). Its practical impact has grown further with self-driving laboratories (Abolhasani & Kumacheva, 2023), where BO proposes experiments and robotic platforms execute them autonomously. A natural next frontier is planning: deciding *which* optimization problem to pursue and *how* to formulate it. A canonical example is *inverse optimization*: rather than solving a given task, the goal is to identify the task under which a given solution is most valuable. This setting is common in science and engineering: a newly discovered material or algorithm may be promising, yet its best application is often unknown. For instance, around 12,000 new crystals are discovered each year (Hellenbrandt, 2004), but only a small fraction achieve industrial application, partly because discovering the right application is difficult.

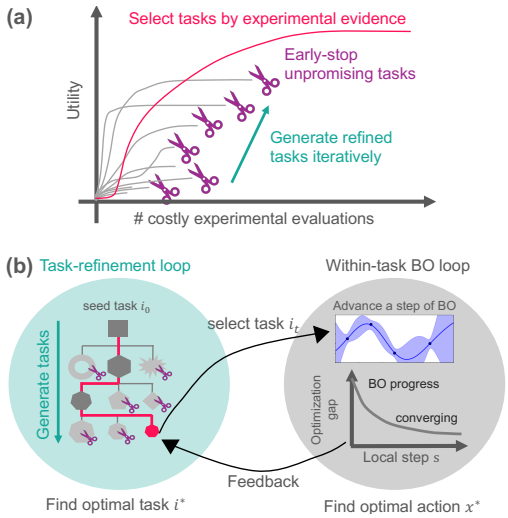

Figure 1. **Conceptual overview**: Starting from a seed task i_0, the new tasks are iteratively generated toward the ϵ-optimal task $i^\star$ and identifies its optimum $x^\star$. (a) Experimental progress: task generation and elimination over time. (b) Two-loop structure: feedback from within-task BO informs task refinement via a coarse-to-fine scheduler.

The main difficulty comes from a finite experimental budget T. While recent advances in large language models (LLMs; (Achiam et al., 2023; Team et al., 2023)) make automatic task generation plausible (Stanley & Lehman, 2015; Du et al., 2025), we must judge *achievability* of the generated tasks within the budget. High-reward tasks may be too ambitious to complete, whereas easy tasks may offer limited value. Moreover, achievability is only revealed through trial and error, evaluating tasks itself consumes experimental resources. Naively exploring many tasks can therefore waste budget on infeasible directions.

We study the problem of generating the most rewarding task that is achievable within budget T. We call such a task ϵ-*optimal* and formalize this setting as *open-ended BO*: tasks are generated online from past experience, budget is allocated to tasks that appear promising and achievable, and standard BO is run within selected tasks to optimize $x^\star$. Unlike standard BO over a predefined task vector, tasks here may be structured, are not enumerated in advance, and reveal their value only indirectly through partial BO trajectories. To solve this problem, we propose *Generate-*

[1]Toyota Motor Corporation. Correspondence to: Yuki Takezawa <yuki_takezawa@mail.toyota.co.jp>.

Proceedings of the 43rd International Conference on Machine Learning Workshop AI4Research, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

Select-Refine (GSR; Fig. 1), which treats task discovery itself as an generative-optimization problem and uses regret bounds as a principled *task manager* for validating generated tasks. We prove that GSR incurs only a logarithmic regret overhead compared with running BO on a single fixed task.

2. Problem Setting

2.1. Tasks, evaluations, and incumbents

Tasks. There is an unbounded collection of tasks indexed by $i \in \mathbb{N}$. Task i is specified by a compact domain $\mathcal{X}^{(i)} \subset \mathbb{R}^{d^{(i)}}$ and an unknown objective $f^{(i)} : \mathcal{X}^{(i)} \to \mathbb{R}$.

Global vs. local time. We index global rounds by $t \in [T]$. At each t, the task-optimizer selects a task i_t and evaluates one design $x_t^{(i_t)} \in \mathcal{X}^{(i_t)}$, observing the noisy oracle $y_t^{(i_t)} = f^{(i_t)}\big(x_t^{(i_t)}\big) + \xi_t^f$, where $\xi_t^f \sim \mathrm{SubGauss}(\sigma_f^2)$. Let $s_t^{(i)}$ be the local counter that task i has been selected until t and $\mathcal{I}_T$ be the set of tasks instantiated by T. For brevity, we write $x_t^{(i)} := x_{s_t^{(i)}}^{(i)}$ (and similarly for other quantities); the notations are summarized in Table 2 in the Appendix.

Incumbent is best-so-far observed value[1] after s local evaluations on task i: $\overline{y}_s^{(i)} := \max_{1 \le r \le s} y_r^{(i)}$.

Optimization gap. Each task runs a standard GP-UCB (Srinivas et al., 2010). For readers unfamiliar with GP-UCB, see Appendix F. We only use an *optimization gap bound*: w.p. $\ge 1 - \delta_f^{(i)}$, after s evaluations,

$$f^{\star(i)} - \overline{y}_s^{(i)} \;\le\; \epsilon_s^f := 2\sqrt{C_\lambda\, \beta_s^{(i)}\, \gamma_s^{(i)}/s}, \qquad (1)$$
$$f^{\star(i)} := \max_{x \in \mathcal{X}^{(i)}} f^{(i)}(x),$$

where $\beta_s^{(i)}$ and $\gamma_s^{(i)}$ are exploration and information-gain terms and $C_\lambda > 0$ is a constant. The optimization gap ϵ_s^f upper-bounds the worst-case error of the global optimum estimate, and satisfies $\epsilon_s^f \to 0$ as $s \to \infty$, i.e., GP-UCB asymptotically identifies the global optimum $f^{\star(i)}$ (no-regret) for commonly used kernels (see (Lee & Javidi, 2025)). ϵ_s^f is computable without access to $f^{\star(i)}$, so it provides a principled cross-task progress metric. For uniformity over instantiated tasks, define $\overline{\epsilon}_s^f := \max_{i \in \mathcal{I}_T} \epsilon_s^f$, so $\epsilon_s^f \le \overline{\epsilon}_s^f$ for all $i \in \mathcal{I}_T$ and $s \in [T]$ (Lemma F.3).

2.2. Latent utility

Since the scale of $f^{(i)}$ can vary across tasks, we introduce a scale-invariant utility function $u^{(i)} : \mathbb{R} \to [0, 1]$ applied to the incumbent $\overline{y}_s^{(i)}$. Any rankable function can be utility; e.g., Bayesian surprise (Agarwal et al., 2026) or human preference (Bradley & Terry, 1952; Adachi et al., 2025). In other words, incomparable tasks are not our scope.

Definition 2.1 (Utility regularity)**.** $u^{(i)}$ is monotone nondecreasing and $L^{(i)}$-Lipschitz: $|u^{(i)}(y) - u^{(i)}(y')| \le L^{(i)}|y - y'|$ for all $y, y' \in \mathbb{R}$. Let $L^\star := \sup_i L^{(i)} < \infty$.

Monotonicity matches the optimization intent (better designs should not be judged worse). Lipschitzness is a standard regularity condition in optimization.

Noisy utility calls. For task i_t selected at t, after observing $y_t^{(i_t)}$, we query the noisy utility oracle $\tilde{u}_t^{(i_t)} = u^{(i_t)}\big(\overline{y}_t^{(i_t)}\big) + \xi_t^u$, where $\xi_t^u \sim \mathrm{SubGauss}\Big(\sigma_u^2\Big)$.

2.3. Long-run value and BO-achievable optimality

Long-run value. A task i has an unknown value $U^{(i)} := u^{(i)}\big(f^{\star(i)}\big) \in [0, 1]$, where $U^\star := \sup_i U^{(i)}$.

Resolution. With a finite T, the optimization gap is nonzero $\epsilon_T^f > 0$. Combining Lipschitzness with (1) gives, $\forall i, s$ $0 \le U^{(i)} - u^{(i)}(\overline{y}_s^{(i)}) \le L^{(i)}\big(f^{\star(i)} - \overline{y}_s^{(i)}\big) \le L^\star\, \epsilon_s^f$, Consequently, even if all budget T is allocated to a single task, $L^\star \epsilon_T^f$ is the finest utility accuracy that can be reliably achieved. We therefore define an *achievable* utility tolerance (the "resolution").

Definition 2.2 (ϵ^U-optimal tasks)**.** For any $\epsilon^U > 0$, define $\mathcal{I}^\star(\epsilon^U) := \big\{i \in \mathbb{N} :\; U^\star - U^{(i)} \le \epsilon^U\big\}$.

Condition 2.3 (BO-comparable target resolution)**.** *Fix* $\epsilon_m^U := \epsilon_0^U 2^{-m}$ *for* $m \in \mathbb{Z}_{\ge 0}$. *Let* $\Lambda_T := \log(eT)$. *Choose a max depth* $\overline{m}_T \in \mathbb{Z}_{\ge 0}$ *and assume there exists an* $m^\star \in \{0, 1, \ldots, \overline{m}_T\}$ *and a constant* $c_\epsilon > 0$ *such that*

$$\epsilon_{m^\star}^U \;\le\; c_\epsilon\, \Lambda_T\, L^\star\, \overline{\epsilon}_T^f.$$

This ensures that the $\epsilon_{m^\star}^U$ is *achievable* within T[2]; otherwise, any finer resolution $\epsilon_{\overline{m}_T}^U < \epsilon_{m^\star}^U$ would exceed what BO can reliably achieve. As T increases, $\overline{\epsilon}_T^f$ shrinks and $\epsilon_{m^\star}^U$ becomes finer.

Goal is to find and optimize any $\epsilon_{m^\star}^U$-optimal task and we employ the task regret as the metric:

$$i_{\epsilon^U}^\star \in \mathcal{I}^\star(\epsilon_{m^\star}^U), \qquad x^{\star(i_{\epsilon^U}^\star)} \in \arg\max_{x \in \mathcal{X}^{(i_{\epsilon^U}^\star)}} f^{(i_{\epsilon^U}^\star)}(x), \qquad (2)$$

$$\mathcal{R}_T := \sum_{t=1}^{T} \Big(U^\star - u^{(i_t)}\big(\overline{y}_t^{(i_t)}\big)\Big). \qquad (3)$$

3. Generate-Select-Refine (GSR)

We propose GSR (Alg. 1), a open-ended-BO framework for *online task discovery* (generate), *budget allocation* (select), and *coarse-to-fine scheduler* (refine). This section is intentionally *algorithm-first*: we define the quantities GSR maintains and explain the control flow. Concrete LLM instantiations are given in §3.2. All formal guarantees are collected in §E.

[1] In analysis, this is noiseless $\overline{y}_s^{(i)} := \max_{1 \le r \le s} f^{(i)}(x_r^{(i)})$.

[2] up to the logarithmic factor needed by the coarse-to-fine reachability schedule in Lemma E.5

Algorithm 1. Generate-Selection-Refine (GSR)

1: **Input:** confidence $(\delta_f, \delta_u, \delta_-)$, Lipschitz bound $\overline{L}$, seed task i_0, batch size J, T, c_g, $\overline{m}_T$.
2: **Init:** $m \leftarrow 0$, $\epsilon_0^U \leftarrow 1$, $\epsilon_m^U \leftarrow \epsilon_0^U$, $\mathcal{I}_1 \leftarrow \{i_0\}$, $\mathcal{B}_0 \leftarrow \mathsf{Gen}(i_0, 0, J)$, $\mathcal{I}_1 \leftarrow \mathcal{I}_1 \cup \mathcal{B}_0$.
3: **for** $t \in [T]$
4: select the next task $i_t \in \arg\max_{j \in \mathcal{I}_t} \overline{U}_t^{(j)}$
5: $x_t^{(i_t)}, y_t^{(i_t)}, \overline{y}_t^{(i_t)} \leftarrow$ RunOneStepGP-UCB(i_t).
6: update envelopes for task i_t by Theorem E.3.
7: select a_t, w_t by Eq. (4), and $\mathcal{I}_{t+1} \leftarrow \mathcal{I}_t$.
8: **if** $m < \overline{m}_T$ **and** $w_t \le c_g \epsilon_m^U$
9: step up $m \leftarrow m+1$; $\epsilon_m^U \leftarrow \epsilon_0^U 2^{-m}$.
10: draw $\mathcal{B}_m \leftarrow \mathsf{Gen}(a_t, m, J)$; $\mathcal{I}_{t+1} \leftarrow \mathcal{I}_{t+1} \cup \mathcal{B}_m$.
11: **end if**
12: **end for**

3.1. Two loops: GP-UCB and task-UCB

Each task i runs a standard GP-based BO procedure (GP-UCB), maintaining: (i) an incumbent $\overline{y}_s^{(i)}$ after s local evaluations, and (ii) a computable optimization gap ϵ_s^f from Eq.(1). The task-optimizer decides (a) which task to evaluate next (select) and (b) when to generate tasks (refine).

Task Selection. Core idea is the confidence intervals (CIs):

$$U^{(i)} \in [\underline{U}_s^{(i)}, \overline{U}_s^{(i)}], \qquad \overline{U}_s^{(i)} = \underbrace{\overline{u}_s^{(i)}}_{\text{Utility UCB}} + \underbrace{\overline{L}\epsilon_s^f}_{\text{optimization gap}},$$

where $\overline{L} \ge L^\star$ bounds the Lipschitz constant. The UCB naturally decomposes into uncertainty from finite utility samples and an optimization gap due to limited evaluation budget. We select the next task i_t via a task-UCB policy (Line 1 in Alg. 1; see Fig.2(a))[3], concentrating budget on the ϵ^U-optimal task while early-stopping suboptimal ones.

Task Refinement. We generate new tasks by mutating and iteratively refining a seed task. A key step is selecting an *anchor* task to mutate. The anchor must be both promising and sufficiently resolved. We therefore choose a_t as the most promising pessimistic task—ranked by $\underline{U}_t^{(i)}$ rather than $\overline{U}_t^{(i)}$—whose uncertainty is below a target threshold:

$$a_t \in \arg\max_{j \in \mathcal{I}_t} \underline{U}_t^{(j)} \text{ s.t. } w_t^{(j)} \le \overline{\epsilon}_t^U, \tag{4}$$

where $w_t^{(i)} := \overline{U}_t^{(i)} - \underline{U}_t^{(i)}$ is the CI width (uncertainty), and $\overline{\epsilon}_t^U := \max\{c_g \epsilon_m^U, \underline{w}_t\}$ is the resolution threshold, with $\underline{w}_t := \min_{j \in \mathcal{I}_t} w_t^{(j)}$. See Fig.2(a) for intuition. This rule selects an anchor that is promising under $\underline{U}$ and sufficiently certain to advance to the next level m (cf. Condition 2.3).

Generation scheduler. We generate new tasks only once the current anchor is sufficiently certain (Line 1; Fig.2(b)).

[3]Ties are broken in favor of the task with fewer evaluations.

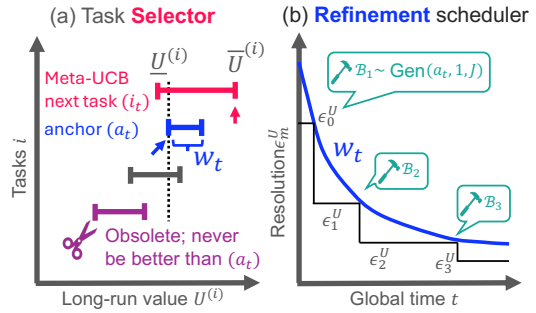

Figure 2. Alg. 1: (a) **Select** chooses the next task i_t and anchor task a_t. (b) **Refine** advances the resolution levels ϵ_m^U and generates J mutations when anchor width $w_t \le c_g \epsilon_m^U$.

We then advance to $m+1$ and generate a new batch of child tasks at a finer mutation scale ϵ_m^U. See Figure 12 for ablation study on generation schedulers.

3.2. LLM-powered openended-BO

We adopt empirically successful paradigms (Fig.3): evolutionary LLMs for task generation (Novikov et al., 2025; Lange et al., 2025). We present only the essential details here and refer readers to Appendix G.1 for background.

3.3. Generate tasks by controlled mutations

We represent each task as a structured specification (e.g., JSON) with editable fields (e.g., objective/domain). Given an anchor task a_t and level m, the generator $\mathsf{Gen}(a_t, m, J)$ prompts an LLM to propose mutated child tasks $\mathcal{B}_m$.

Mutation ratio. To operationalize the resolution levels ϵ_m^U, we control the mutation ratio:

$$\rho(i, a_t) := \frac{\#\text{modified editable fields}}{\#\text{editable fields}} \in [0, 1].$$

We target a decreasing schedule $\rho_m := \rho_0 2^{-m}$, so higher m corresponds to smaller edits.

Scope of the empirical generator. The theory treats Gen as an abstract open-ended generator. In the experiments, however, we use domain-specific JSON schemas and mutate editable fields. Therefore the present empirical instantiation tests *schema-constrained* task discovery: it can discover new task combinations and refinements within the supplied schema, but it is not claimed to invent arbitrary new functional forms unless those structural degrees of freedom are included in the schema.

3.4. Utility elicitation via an LLM committee

When tasks admit an objective comparison, that is ideal. For inverse optimization, for example, we can measure competitiveness by comparing the performance gap between selected method against the best-known baseline. When such comparison is difficult, we can **optionally** use preference-based LLM judges (Rafailov et al., 2023; Fu-

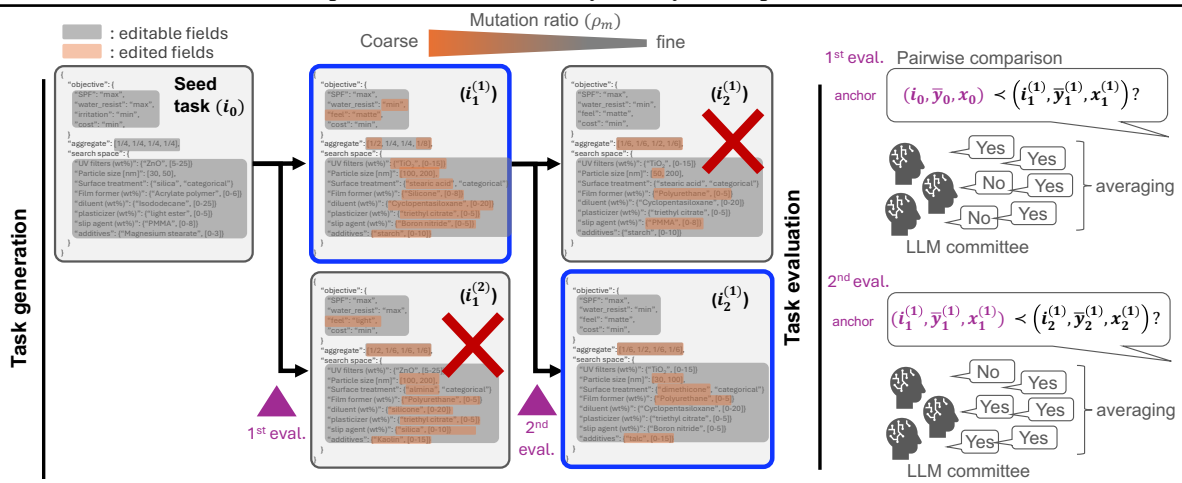

Figure 3. LLM components in GSR: (**Left**) Coarse-to-fine task generation: we mutate a parent task with target mutation ratio ρ_m, producing refined tasks. (**Right**) Committee-based evaluation: committee returns binary votes if the current task–incumbent pair $(i, \overline{y}_s^{(i)})$ is preferred over an anchor $(a_t, \overline{y}_s^{(a_t)})$. By averaging them, we estimate a confidence interval for the utility $u^{(i)}(\overline{y}_s^{(i)})$.

jisawa et al., 2025). Recent LLM-driven discovery work (Agarwal et al., 2026; Bradley et al., 2024; Zhang et al., 2024; Yang et al., 2026) suggests that LLM-based evaluation can support meaningful discovery, and in our experiments LLM judgments correlate strongly with human judgments (Appendix L.2). Following standard LLM-as-a-judge practice (Chen et al., 2024; Lee et al., 2025), an LLM committee compares task pairs and returns binary votes (Fig.3). We adopt the Bradley–Terry model (Bradley & Terry, 1952) to infer latent utilities from these comparisons, defining the utility $u^{(i)}(\cdot)$ as the win rate of task i_t against the seed task i_0, i.e., $\Pr(i_t \succ i_0 \mid \overline{y}_t^{(i_t)}, \overline{y}_0^{(i_0)}, x_t^{(i_t)}, x_0^{(i_0)})$. As both the task i_t and its solution $\overline{y}_t^{(i_t)}$ improve, this win rate provides a consistent cross-task utility. To prevent saturation when i_t is almost always preferred, we instead use an anchor task a_t as an adaptive reference and recover the win rate by chaining pairwise comparisons (e.g., $i_t \succ a_t \succ i_0$). Averaging K_t preference votes yields a single utility estimate $\tilde{u}_t \in [0, 1]$ with $1/(4K_t)$-sub-Gaussian noise, thus it satisfies Definition 2.1 (see Appendix G.1).

4. Experiments

We evaluate GSR across diverse settings. Direct comparison to prior methods is nontrivial due to GSR's open-ended and partial observation nature (see §A). We therefore adapt ShinkaEvolve (Lange et al., 2025) as a baseline, combining four task selectors—random, successive halving (SH; (Jamieson & Talwalkar, 2016)), and Hyperband (Li et al., 2018), and LLM selector (Ferreira et al., 2026) that LLM selects tasks and schedule task generation—with two generation schedulers (fixed and log). We additionally include *seed* (single-task BO on the initial task) and *oracle* (single-task BO on the best task in hindsight). The gap to *seed* quantifies improvement over the initial task, while the gap to *oracle* captures suboptimality in task generation and selection.

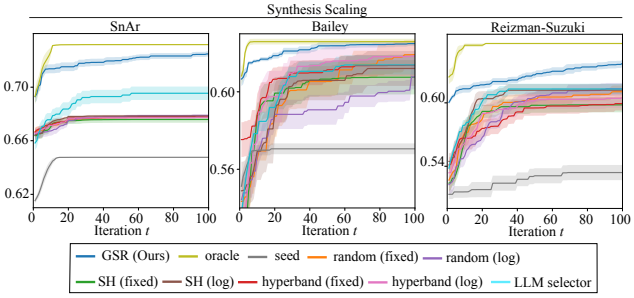

Figure 4. Results on synthesis scaling. The vertical axis represents utility. Solid lines and shaded regions indicate the mean ± one standard error over runs. GSR consistently outperforms the baselines.

All methods are implemented in BoTorch/GPyTorch (Balandat et al., 2020; Gardner et al., 2018). In LLM experiments, GPT-4O-MINI (Achiam et al., 2023) is used only for task generation and committee votes $\tilde{u}_t^{(i)}$; objective evaluations $y_t^{(i)}$ are from the simulator. Full details appear in Appendix K and the anonymous code.[4] Due to space constraints, we present only the results for synthesis scaling here. Other results are provided in B.

Synthesis scaling. Synthesis routes that perform well in the lab often fail to scale to industrial manufacturing; small-scale trials therefore act as a *planning* phase before committing to costly plant construction. We use the SUMMIT chemical process simulator (Felton et al., 2021) with three reactions. Each task is a reaction plan that trades off throughput and sustainability, and cross-task utility is the majority vote of a stakeholder committee (e.g., plant engineers, safety reviewers). We show the results in Fig. 4. The discovered plan is more conservative than the throughput-optimal solution, reflecting committee preferences for sustainability and safety (see also Appendix K).

[4] https://anonymous.4open.science/r/Generate-Select-Refine-2D48

References

Abdolshah, M., Shilton, A., Rana, S., Gupta, S., and Venkatesh, S. Multi-objective Bayesian optimisation with preferences over objectives. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Abolhasani, M. and Kumacheva, E. The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, 2(6):483–492, 2023.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Adachi, M., Hayakawa, S., Jørgensen, M., Wan, X., Nguyen, V., Oberhauser, H., and Osborne, M. A. Adaptive batch sizes for active learning: A probabilistic numerics approach. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 496–504. PMLR, 2024a.

Adachi, M., Planden, B., Howey, D., A. Osborne, M., Orbell, S., Ares, N., Muandet, K., and Lun Chau, S. Looping in the human: Collaborative and explainable Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 238, pp. 505–513, 2024b. URL https://proceedings.mlr.press/v238/adachi24a.html.

Adachi, M., Chau, S. L., Xu, W., Singh, A., Osborne, M. A., and Muandet, K. Bayesian optimization for building social-influence-free consensus. *arXiv preprint arXiv:2502.07166*, 2025.

Adriaensen, S., Rakotoarison, H., Müller, S., and Hutter, F. Efficient bayesian learning curve extrapolation using prior-data fitted networks. *Advances in Neural Information Processing Systems*, 36:19858–19886, 2023.

Agarwal, D., Majumder, B. P., Adamson, R., Chakravorty, M., Gavireddy, S. R., Parashar, A., Surana, H., Mishra, B. D., McCallum, A., Sabharwal, A., and Clark, P. Autodiscovery: Open-ended scientific discovery via bayesian surprise. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=kJqTkj2HhF.

Assumpção, H., Ferreira, D., Campos, L., and Murai, F. CodeEvolve: An open source evolutionary coding agent for algorithm discovery and optimization. *arXiv preprint arXiv:2510.14150*, 2025.

Astudillo, R. and Frazier, P. Bayesian optimization of composite functions. In *International Conference on Machine Learning (ICML)*, pp. 354–363. PMLR, 2019.

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: a framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 21524–21538, 2020.

Berkenkamp, F., Schoellig, A. P., and Krause, A. No-regret Bayesian optimization with unknown hyperparameters. *Journal of Machine Learning Research (JMLR)*, 20(50): 1–24, 2019.

Bradley, H., Dai, A., Teufel, H. B., Zhang, J., Oostermeijer, K., Bellagente, M., Clune, J., Stanley, K., Schott, G., and Lehman, J. Quality-diversity through AI feedback. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=owokKCrGYr.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Chan, T. C., Mahmood, R., and Zhu, I. Y. Inverse optimization: Theory and applications. *Operations Research*, 73 (2):1046–1074, 2025.

Chen, G. H., Chen, S., Liu, Z., Jiang, F., and Wang, B. Humans or LLMs as the judge? a study on judgement bias. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8301–8327, 2024. URL https://aclanthology.org/2024.emnlp-main.474/.

Chew, R. W. T., Nguyen, Q. P., and Low, B. K. H. BILBO: BILevel Bayesian optimization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 10249–10268, 2025. URL https://proceedings.mlr.press/v267/chew25a.html.

Chowdhury, S. R. and Gopalan, A. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pp. 844–853. PMLR, 2017.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4): 547–553, 2009.

Dai, Z., Chen, Y., Yu, H., Low, B. K. H., and Jaillet, P. On provably robust meta-Bayesian optimization. In *Uncertainty in Artificial Intelligence (UAI)*, pp. 475–485. PMLR, 2022.

Dogan, V. and Prestwich, S. Bilevel optimization by conditional bayesian optimization. In *International Conference on Machine Learning, Optimization, and Data Science*, pp. 243–258. Springer, 2023.

Du, Y., Yu, B., Liu, T., Shen, T., Chen, J., Rittig, J. G., Sun, K., Zhang, Y., Song, Z., Zhou, B., et al. Accelerating scientific discovery with autonomous goal-evolving agents. *arXiv preprint arXiv:2512.21782*, 2025.

Felton, K. C., Rittig, J. G., and Lapkin, A. A. SUMMIT: benchmarking machine learning methods for reaction optimisation. *Chemistry-Methods*, 1(2):116–122, 2021.

Ferreira, F., Wobbe, L., Krishnakumar, A., Hutter, F., and Zela, A. Can LLMs beat classical hyperparameter optimization algorithms? a study on autoresearch. *arXiv preprint arXiv:2603.24647*, 2026.

Fu, S., He, F., Tian, X., and Tao, D. Convergence of bayesian bilevel optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fLXpXa7iiz.

Fujisawa, M., Adachi, M., and Osborne, M. A. Scalable valuation of human feedback through provably robust model alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=EaTRrceoU9.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. GPyTorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Garnett, R. *Bayesian optimization*. Cambridge University Press, 2023.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

González, J., Dai, Z., Damianou, A., and Lawrence, N. D. Preferential Bayesian optimization. In *International Conference on Machine Learning (ICML)*, pp. 1282–1291. PMLR, 2017.

Gupta, R., Hartford, J., and Liu, B. LLMs for Bayesian optimization in scientific domains: Are we there yet? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 15482–15510. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-emnlp.838/.

Gupta, S., Rana, S., Ha, H., and Venkatesh, S. Sub-linear regret bounds for Bayesian optimisation in unknown search spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 16271–16281, 2020.

Ha, H., Rana, S., Gupta, S., Nguyen, T., and Venkatesh, S. Bayesian optimization with unknown search space. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Hellenbrandt, M. The inorganic crystal structure database (icsd)—present and future. *Crystallography Reviews*, 10 (1):17–22, 2004.

Hong, K., Li, Y., and Tewari, A. An optimization-based algorithm for non-stationary kernel bandits without prior knowledge. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 3048–3085. PMLR, 2023.

Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1), 2013.

Jamieson, K. and Talwalkar, A. Non-stochastic best arm identification and hyperparameter optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 240–248. PMLR, 2016.

Jerry Lin, Z., Astudillo, R., Frazier, P., and Bakshy, E. Preference exploration for efficient Bayesian optimization with multiple outcomes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151, pp. 4235–4258, 2022. URL https://proceedings.mlr.press/v151/jerry-lin22a.html.

Klein, A., Falkner, S., Bartels, S., Hennig, P., and Hutter, F. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pp. 528–536, 2017. URL https://proceedings.mlr.press/v54/klein17a.html.

Lange, R. T., Imajuku, Y., and Cetin, E. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

Lee, C., Zeng, T., Jeong, J., Sohn, J.-y., and Lee, K. How to correctly report LLM-as-a-judge evaluations. *arXiv preprint arXiv:2511.21140*, 2025.

Lee, M. and Javidi, T. Consequences of kernel regularity for bandit optimization. *arXiv preprint arXiv:2512.05957*, 2025.

Levine, R. D. *Molecular reaction dynamics*. Cambridge university press, 2009.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research (JMLR)*, 18(185):1–52, 2018. URL http://jmlr.org/papers/v18/16-558.html.

Liang, Q., Gongora, A. E., Ren, Z., Tiihonen, A., Liu, Z., Sun, S., Deneault, J. R., Bash, D., Mekki-Berrada, F., Khan, S. A., et al. Benchmarking the performance of Bayesian optimization across multiple experimental materials science domains. *npj Computational Materials*, 7 (1):188, 2021.

Liu, T., Astorga, N., Seedat, N., and van der Schaar, M. Large language models to enhance Bayesian optimization. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=OOxotBmGol.

Maraval, A., Zimmer, M., Grosnit, A., and Bou Ammar, H. End-to-end meta-Bayesian optimisation with transformer neural processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 11246–11260, 2023.

Mockus, J., Tiesis, V., and Zilinskas, A. The application of Bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

Nguyen, V., Schulze, S., and Osborne, M. Bayesian optimization for iterative learning. *Advances in Neural Information Processing Systems*, 33:9361–9371, 2020.

Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J., Mehrabian, A., et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Osselin, P., Adachi, M., Dong, X., and Osborne, M. A. Natural evolutionary search meets probabilistic numerics. *arXiv preprint arXiv:2507.07288*, 2025.

Parkin, S. S., Kaiser, C., Panchula, A., Rice, P. M., Hughes, B., Samant, M., and Yang, S.-H. Giant tunnelling magnetoresistance at room temperature with MgO (100) tunnel barriers. *Nature Materials*, 3(12):862–867, 2004.

Pyzer-Knapp, E. O. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development*, 62(6):2–1, 2018.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 53728–53741, 2023.

Rakotoarison, H., Adriaensen, S., Mallik, N., Garibov, S., Bergman, E., and Hutter, F. In-context freeze-thaw Bayesian optimization for hyperparameter optimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 41982–42008, 2024. URL https://proceedings.mlr.press/v235/rakotoarison24a.html.

Requeima, J., Bronskill, J., Choi, D., Turner, R. E., and Duvenaud, D. LLM processes: Numerical predictive distributions conditioned on natural language. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 109609–109671, 2024.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning (ICML)*, pp. 1015–1022, 2010. URL https://icml.cc/Conferences/2010/papers/422.pdf.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. W. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE transactions on information theory*, 58(5):3250–3265, 2012.

Stanley, K. O. and Lehman, J. *Why greatness cannot be planned: The myth of the objective*. Springer, 2015.

Stein, M. L. *Interpolation of spatial data*. Springer Science & Business Media, 1999.

Suwandi, R. C., Yin, F., Wang, J., Li, R., Chang, T.-H., and Theodoridis, S. Adaptive kernel design for Bayesian optimization is a piece of CAKE with LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=BOEZYnC8nR.

Swersky, K., Snoek, J., and Adams, R. P. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 26, 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/f33ba15effa5c10e873bf3842afb46a6-Paper.pdf.

Swersky, K., Snoek, J., and Adams, R. P. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Volpp, M., Fröhlich, L. P., Fischer, K., Doerr, A., Falkner, S., Hutter, F., and Daniel, C. Meta-learning acquisition functions for transfer learning in Bayesian optimization. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=ryeYpJSKwr.

Wakayama, T. and Suzuki, T. In-context learning is provably bayesian inference: a generalization theory for meta-learning. *arXiv preprint arXiv:2510.10981*, 2025.

Wang, Z. and Jegelka, S. Max-value entropy search for efficient Bayesian optimization. In *International Conference on Machine Learning (ICML)*, pp. 3627–3635. PMLR, 2017.

Wang, Z., Kim, B., and Kaelbling, L. P. Regret bounds for meta Bayesian optimization with an unknown Gaussian process prior. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

Wu, J., Zou, D., Chen, Z., Braverman, V., Gu, Q., and Bartlett, P. How many pretraining tasks are needed for in-context learning of linear regression? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vSh5ePa0ph.

Xie, T. and Grossman, J. C. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120 (14):145301, 2018.

Xu, W., Adachi, M., Jones, C. N., and Osborne, M. A. Principled Bayesian optimization in collaboration with human experts. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 104091–104137, 2024. doi: 10.52202/079017-3308.

Yang, A., Muandet, K., Caprio, M., Chau, S. L., and Adachi, M. Verbalizing llm's higher-order uncertainty via imprecise probabilities. *arXiv preprint arXiv:2603.10396*, 2026.

Zelikman, E., Lorch, E., Mackey, L., and Kalai, A. T. Self-taught optimizer (STOP): Recursively self-improving code generation. In *Conference on Language Modeling (COLM)*, 2024.

Zhang, J., Lehman, J., Stanley, K., and Clune, J. OMNI: Open-endedness via models of human notions of interestingness. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AgM3MzT99c.

Zhang, J., Hu, S., Lu, C., Lange, R., and Clune, J. Darwin Gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.

Ziomek, J., Adachi, M., and Osborne, M. A. Bayesian optimisation with unknown hyperparameters: regret bounds logarithmically closer to optimal. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:86346–86374, 2024.

Ziomek, J., Adachi, M., and Osborne, M. A. Time-varying Gaussian process bandits with unknown prior. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 258, pp. 4294–4302, 2025. URL https://proceedings.mlr.press/v258/ziomek25a.html.

A. Related Work

To our knowledge, no prior work studies our setting. Existing directions such as meta-BO and bi-level BO assume predefined and parameterized task spaces, whereas we consider *open-ended* task optimization: tasks are generated on the fly, may be free-form rather than parameterized, and are evaluated only through partial observations. **Meta BO** either transfers across related tasks (Swersky et al., 2013; Volpp et al., 2020; Dai et al., 2022; Maraval et al., 2023; Osselin et al., 2025) or adapts hyperparameters online (Wang et al., 2018; Berkenkamp et al., 2019; Ziomek et al., 2024; 2025). Instead, our GSR is the *generative optimization* of the task itself. **LLMs + BO** are studied as surrogates (Liu et al., 2024; Requeima et al., 2024; Gupta et al., 2025), collaborative agents (Adachi et al., 2024b; Xu et al., 2024) or kernel generators (Suwandi et al., 2025) but under a fixed task. **Bi-level BO** (Fu et al., 2024; Chew et al., 2025; Dogan & Prestwich, 2023) studies hierarchical optimization where the lower-level problem depends on the upper-level one, such as GP hyperparameter tuning for BO. However, it still assumes a fixed, bounded, continuous search space, unlike our setting with unbounded and free-form tasks. **BO from partial observations** (Swersky et al., 2014; Klein et al., 2017; Nguyen et al., 2020; Adriaensen et al., 2023; Rakotoarison et al., 2024) infer long-term utility from partial learning curves, which select among a predefined set of tasks, whereas we must generate new tasks online. **BO for Task optimization** optimizes restricted task parameters, such as unknown domain bounds (Ha et al., 2019; Gupta et al., 2020) or objective weights from preference feedback (Abdolshah et al., 2019; Jerry Lin et al., 2022). These are special cases of our setting with fixed parameterized task spaces. **Evolutionary LLMs** (Zelikman et al., 2024; Novikov et al., 2025; Lange et al., 2025; Zhang et al., 2025) perform generative optimization with direct feedback, whereas we must infer long-term value from partial lower-level optimization trajectories.

B. Additional Experiments

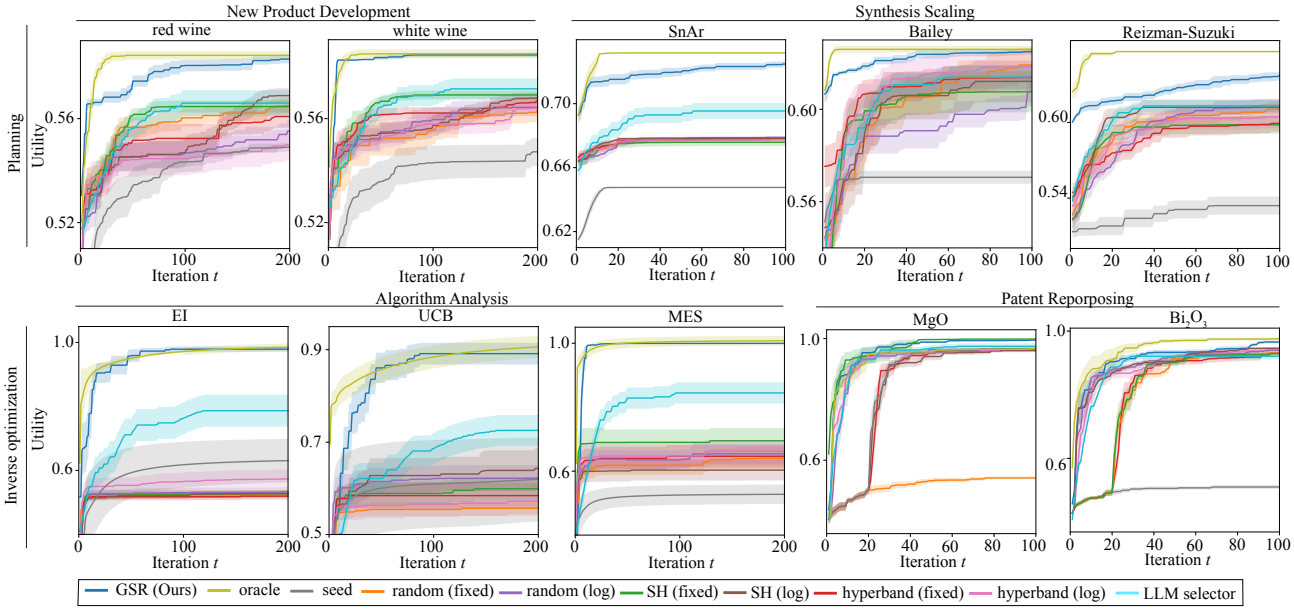

Figure 5. Real-world experiments. Top row: planning tasks—(a) new product development and (b) synthesis scaling. Bottom row: inverse optimization—(c) objective selection. Solid lines and shaded regions indicate the mean ± one standard error over runs. GSR consistently outperforms the baselines.

B.1. Online Experiment 1: Planning

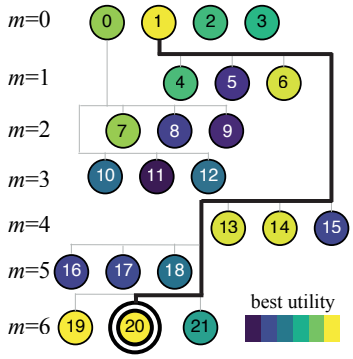

Figure 6. GSR's task evolution tree shows coarse-to-fine refinement across levels m toward the best task (20). Colors indicate final utility, and evaluation counts vary across tasks; task (1) initially lags behind its children (4–6), improves with more BO evaluations, and is ultimately surpassed by task (20).

We first apply GSR to *planning*, where one must decide both *what* to optimize and *how* to evaluate it, starting from a vague seed task that is progressively refined (Fig.5).

New product development. We consider a product manager at a local winery designing a new wine to attract new customers. This requires jointly identifying a target customer segment (task $i^{\star}_{\epsilon_U}$) and its optimal wine (solution $x^{(i^{\star}_{\epsilon_U})}$). We use the Wine Quality dataset (Cortez et al., 2009), where inputs are physicochemical properties (e.g., acidity, sugar, alcohol) and outputs are quality scores. Each task objective combines (i) quality maximization and (ii) taste–intent matching, where natural-language descriptions (e.g., fruity, bold) are instantiated as numerical targets x_{target} via LLMs, and matching is measured by the Mahalanobis distance between x_{target} and x_t. Cross-task utility is defined by majority vote from an LLM committee. We generate diverse consumer personas (e.g., engineers, event hosts) with latent preferences stored as JSON and treat them as new potential customers. For each comparison (i_t, a_t), the committee votes using fixed personas from

the pool, which remain hidden from the task generator. An evolutionary LLM then mutates the current best task (Fig.6), generating $J = 3$ new tasks per resolution ϵ_m^U by refining taste intent, objective weights (quality vs. taste matching), and the search space. Across red and white wines, GSR achieves near-oracle performance and discovers persona-aligned styles (e.g., bold&crisp reds and fruity&smooth whites). See Appendix K for full details, including prompts.

B.2. Online Experiment 2: Inverse Optimization

Table 1. Discovered tasks in the algorithm analysis

policy	objective	dim. d	bounds	noise σ_{noise}
EI	Styblinski-Tang	5	$[-5, 5]$	0.01
UCB	Levy	3	$[0, 1]$	0.1
MES	Levy	2	$[-0.03, 3.78]$	0.01

Next, we apply GSR to *inverse optimization* (IO), which seeks a task under which a given solution x is optimal, rather than optimizing x for a fixed task. While classical IO focuses on exact recovery in linear programming (Chan et al., 2025), we extend IO to black-box, free-form task descriptions using BO and LLMs, and solve it approximately by allowing mild refinement of x. Starting from a seed task i_0 and solution x_0, GSR jointly evolves both until the refined solution $x^\star$ is optimal for the discovered task $i^\star_{\epsilon^U}$.

Algorithm analysis. We apply IO to acquisition function analysis (EI (Mockus et al., 1978), GP-UCB (Srinivas et al., 2010), and MES (Wang & Jegelka, 2017)). Which acquisition function performs best depends strongly on the underlying problem setting. We study the inverse question: *under what task does each acquisition function perform best?* We define utility as the performance gap between the targeted acquisition function and the best-performing alternative, passed through a sigmoid to lie in $[0, 1]$. Each task corresponds to standard test functions available in BoTorch, and an evolutionary LLM generates new tasks by mutating problem settings. The discovered tasks (Table 1) match known behavior: EI performs well on higher-dimensional smooth functions; UCB is more robust under noisy feedback; and MES excels on functions with many low-noise local optima. These findings are consistent with established results: EI is more exploitative (Shahriari et al., 2015), GP-UCB admits formal regret guarantees under noisy observations (Srinivas et al., 2012), and information-theoretic approaches are effective for multimodal objectives (Wang & Jegelka, 2017).

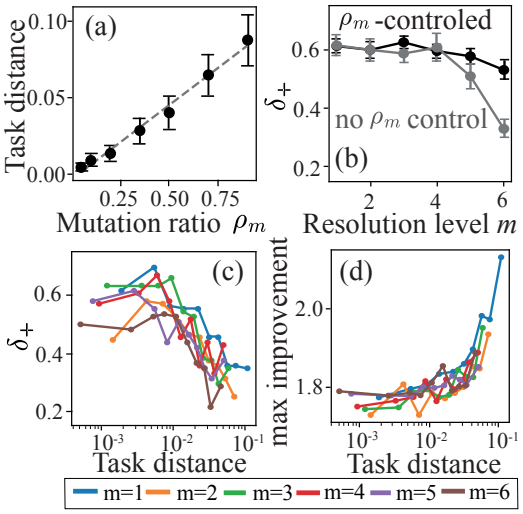

Figure 7. Resolution controllability. (a) Mutation ratio ρ_m correlates with task distance. (b) finer task generation probability δ_+ is non-zero across all resolution levels m. (c) Finer mutations yield higher δ_+. (d) Coarser mutations achieve larger improvement.

Patent Repurposing. In industrial R&D, materials developed for a target application are often abandoned despite substantial sunk costs; repurposing aims to identify alternative applications where such materials can perform well. We use the MATERIALS PROJECT dataset (Jain et al., 2013) and construct input–output pairs, where inputs are crystal structure descriptors encoded by CGCNN (Xie & Grossman, 2018) and outputs consist of six material properties. Each task optimizes

two components: (i) surprisal (Levine, 2009), defined as the negative log-probability of material properties, computed by fitting a kernel density estimator to each simulated property and linearly aggregating their surprisals as a feedback oracle; and (ii) a regularizer measuring the Euclidean distance between the seed material x_0 and the selected material $x_t^{(i_t)}$. An evolutionary LLM refines tasks by mutating surprisal aggregation weights, regularization weights, and the search space, while an LLM committee evaluates scientific and product value via pairwise comparisons. Using MgO and Bi_2O_3 as seeds, GSR identifies plausible additives—$TmCl_3$ and HgF_2 for MgO (shear modulus, band gap), and CuO and $PrTe_2$ for Bi_2O_3 (Poisson ratio, Fermi energy). Based on these candidates, LLMs further propose repurposed applications, including more durable MRAM (Parkin et al., 2004), inorganic photolithography layers, and ion-conducting or active-material coatings for solid oxide fuel cells based on Bi_2O_3. We find no prior reports for these hypotheses in a targeted keyword search. Overall, the results illustrate *data-supported hypothesis generation* for materials repurposing, rather than hypotheses derived solely from LLMs.

B.3. Experimental Analysis, Ablation Study

Human-LLM evaluation alignment. LLM-as-a-judge is powerful, but it risks misalignment with human values. To assess this, we compared its judgments against those of two PhD-level materials scientists on 100 randomly sampled task pairs. Both the human annotators and the LLM were given exactly the same task descriptions and partial optimization histories, and the annotators were blinded to the committee votes. Agreement, measured by Cohen's kappa, was high: 0.905 for synthesis and 0.937 for repurposing on average. See Appendix L.2 for the details.

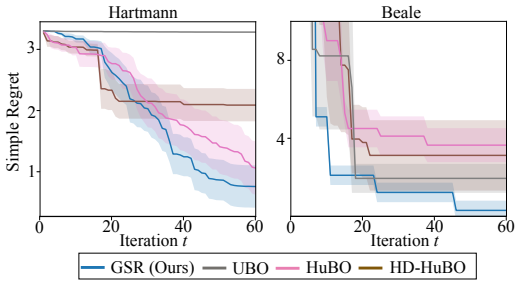

Figure 8. Unknown search space.

Resolution controllability. We analyze the LLM experiments using the white wine task. GSR uses mutation to control task resolution. We generate $J = 40$ child tasks for 4 anchor tasks across mutation ratios ρ_m, resolution levels m, and random seeds (Fig.7). (a) Task distance—Euclidean distance over numerical task parameters—correlates with ρ_m, confirming that JSON mutation controls the effective resolution ϵ_m^U. (b) The success probability δ_+ (Condition E.4) remains non-zero across all m. Without controlling ρ_m, δ_+ decreases at higher m (grey), while a coarse-to-fine ρ_m schedule stabilizes δ_+ (black). (c) δ_+ increases as task distance decreases, further motivating smaller ρ_m at finer resolutions. (d) Coarser mutations (higher ρ_m, lower m) yield larger improvements, supporting an exploration-to-exploitation strategy: explore early with larger ρ_m, then gradually reduce ρ_m to exploit.

Non-LLM task generator. Next, we evaluate GSR with a non-LLM generator in the *unknown search space* setting (Ha et al., 2019), where the true optimum $x^\star$ lies outside the initial domain $\mathcal{X}_0$. The goal is to expand candidate domains to include $x^\star$ while avoiding unnecessary expansion. This is a single-objective, so tasks (domains) are ranked directly by the objective $f(x)$ without utility. We instantiate the generator by doubling the domain at each resolution level m and compare against UBO (Ha et al., 2019), HuBO, and HD-HuBO (Gupta et al., 2020) on two test functions, measuring simple regret $f^\star - \overline{y}_t^{(i_t)}$. Despite being specialized for this setting, these baselines are outperformed by GSR (Fig. 8).

Offline experiments. Finally, we consider an offline setting with no generator: all tasks are fixed a priori, reducing the problem to task selection. Algorithm 1 thus simplifies to pure task-UCB (omitting Lines 8–10). We evaluate six BoTorch benchmark functions, defining task utilities via a Gaussian CDF, $u^{(i)}(\overline{y}_t) := \Phi\Big(\overline{y}_t^{(i)} - \mu^{(i)}/\sigma^{(i)}\Big)$, with task-specific constants $(\mu^{(i)}, \sigma^{(i)})$. We study two variants: (a) fixed task selection, which allocates budget across independent tasks, and (b) objective selection, where all tasks share a common domain $\mathcal{X}^{(i)} = \mathcal{X} = [0,1]^6$. Setting (a) relates to classical budget-allocation baselines (random, successive halving, Hyperband), while (b) connects to preference-based multi-objective BO (MOBO-

PC (Abdolshah et al., 2019) and BOPE (Jerry Lin et al., 2022)[5]. Using best-so-far simple regret $U^\star - \max_{\tau \le t} u^{(i_\tau)}(\overline{y}^{(i_\tau)}_{s_\tau^{(i_\tau)}})$ as the metric, GSR consistently outperforms these baselines (Fig. 9). See Appendix L.1 for additional experiments and Appendix L.4 for extended discussion.

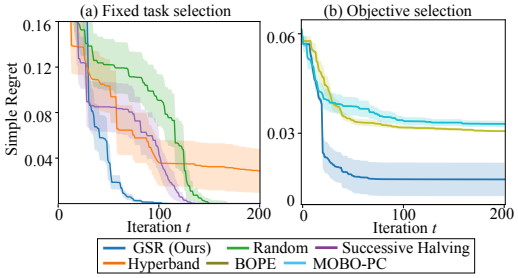

Figure 9. Offline settings: (a) Fixed task selection and (b) objective selection.

[5]With minor adaptations since only one objective $y_t^{(i)}$ is observed at each step whereas MOBO assumes observe all $\{y_t^{(i)}\}_{i=1}^n$.

C. Limitations

We propose GSR, an open-ended BO framework that refines a vague seed task into an ϵ-optimal task with GP-UCB-style regret guarantees, and apply it to planning and inverse optimization. While the theory is principled and the ablations in §B.3 support its assumptions, the current LLM-based task generation remains empirical, a limitation shared with evolutionary LLM methods. A second open challenge is utility specification: although some settings admit objective metrics, such as competitiveness in inverse optimization, and LLM committees show reasonable alignment with human judgments, reliable deployment still requires experimental validation and human oversight. At the same time, this challenge is also an opportunity, since utilities can themselves be learned from data, for example through preference learning with DPO (Rafailov et al., 2023; Fujisawa et al., 2025) or experimental feedback with GRPO (Shao et al., 2024).

D. Notations

Table 2. Notation (global task-rounds and local within-task BO).

Symbol	Meaning
Indices, sets, counters, and shorthand	
$t \in [T]$	Global (meta) round; T is the total evaluation budget.
$i \in \mathbb{N}$	Task index; i_0 is the seed task; i_t is the task selected at round t.
$\mathcal{I}_t$	(Random) set of tasks instantiated and available by round t; $\mathcal{I}_T$ is the set by the horizon.
$s_t^{(i)}$	Local evaluation counter: number of times task i has been selected up to round t.
$m \in \{0, 1, \ldots, \overline{m}_T\}$	Resolution level; $\overline{m}_T$ is the maximum level introduced by time T.
$q_t^{(i)}$	Shorthand for within-task quantities: $q_t^{(i)} := q^{(i)}_{s_t^{(i)}}$ (used throughout the main text).
Task-level BO and observations	
$\mathcal{X}^{(i)} \subset \mathbb{R}^{d^{(i)}}$	Domain of task i; $d^{(i)}$ is its dimension.
$f^{(i)} : \mathcal{X}^{(i)} \to \mathbb{R}$	Unknown black-box objective for task i.
$x_t^{(i_t)}$	Design evaluated at global round t on task i_t (equivalently $x^{(i_t)}_{s_t^{(i_t)}}$ in local time).
$y_t^{(i_t)}$	Noisy objective observation: $y_t^{(i_t)} = f^{(i_t)}(x_t^{(i_t)}) + \xi_t^f$.
$\xi_t^f \sim \mathrm{SubGauss}(\sigma_f^2)$	Sub-Gaussian objective noise with variance proxy σ_f^2.
$\overline{y}_s^{(i)}$	Incumbent after s local evals on task i: $\overline{y}_s^{(i)} := \max_{1 \le r \le s} f^{(i)}(x_r^{(i)})$.
$f^{\star(i)}$	Optimal objective value: $f^{\star(i)} := \max_{x \in \mathcal{X}^{(i)}} f^{(i)}(x)$; $x^{\star(i)} \in \arg\max_x f^{(i)}(x)$.
ϵ_s^f	GP-UCB optimization gap bound after s local evals (Eq. (1)); depends on i via $\beta_s^{(i)}, \gamma_s^{(i)}$.
$\overline{\epsilon}_s^f$	Uniform gap across instantiated tasks: $\overline{\epsilon}_s^f := \max_{i \in \mathcal{I}_T} \epsilon_s^f$.
$\beta_s^{(i)}, \gamma_s^{(i)}$	GP-UCB exploration parameter and information-gain term for task i.
Utility, long-run value, and confidence bounds	
$u^{(i)} : \mathbb{R} \to [0, 1]$	Scale-invariant utility applied to incumbents; monotone and $L^{(i)}$-Lipschitz.
$L^{(i)}$	Lipschitz constant of $u^{(i)}$.
$L^\star$	Global Lipschitz upper bound: $L^\star := \sup_i L^{(i)}$.
$\overline{L}$	Supplied Lipschitz bound used by the algorithm, intended to satisfy $\overline{L} \ge L^\star$.
$\tilde{u}_t^{(i_t)}$	Noisy utility call: $\tilde{u}_t^{(i_t)} = u^{(i_t)}(\overline{y}_t^{(i_t)}) + \xi_t^u$.
$\xi_t^u \sim \mathrm{SubGauss}(\sigma_u^2)$	Sub-Gaussian utility noise (averaging K_t votes reduces proxy to σ_u^2/K_t).
$U^{(i)}$	Long-run task value: $U^{(i)} := u^{(i)}(f^{\star(i)})$; and $U^\star := \sup_i U^{(i)}$.
$\underline{u}_t^{(i)}, \overline{u}_t^{(i)}$	Confidence bounds for $u^{(i)}(\overline{y}_t^{(i)})$.
$\underline{U}_t^{(i)}, \overline{U}_t^{(i)}$	Value envelopes for $U^{(i)}$ (Thm. E.3).
$w_t^{(i)}$	Value interval width: $w_t^{(i)} := \overline{U}_t^{(i)} - \underline{U}_t^{(i)}$.
$\underline{w}_t$	Minimum value-width across tasks: $\underline{w}_t := \min_{j \in \mathcal{I}_t} w_t^{(j)}$.
$\overline{\epsilon}_t^U$	Anchor-resolution threshold: $\overline{\epsilon}_t^U := \max\{c_g\, \epsilon_m^U,\ \underline{w}_t\}$.
w_t	Width of the selected anchor: $w_t := w_t^{(a_t)}$.
K_t	Number of committee votes averaged to form $\tilde{u}_t^{(i_t)}$ (if using vote-based feedback).
Resolution, generation, and regret	
ϵ^U	Utility-scale optimality tolerance ("resolution").
ϵ_m^U	Resolution ladder: $\epsilon_m^U := \epsilon_0^U 2^{-m}$; $m^\star$ is the BO-achievable level (Cond. 2.3).
$\mathcal{I}^\star(\epsilon^U)$	ϵ^U-optimal task set: $\{i : U^\star - U^{(i)} \le \epsilon^U\}$.
a_t	Anchor task used for mutation at round t (Eq. (4)).
$\mathcal{B}_m$	Batch of J mutated tasks at level m: $\mathcal{B}_m \sim \mathsf{Gen}(a_t, m, J)$.
J	Number of tasks generated per generation event (batch size).
δ_+	Per-draw probability that a mutation hits an ϵ_m^U-optimal task (Cond. E.4).
$\rho(i, a_t)$	Mutation ratio between child task i and parent/anchor a_t; schedule $\rho_m = \rho_0 2^{-m}$.
$\mathcal{R}_T$	Meta regret: $\mathcal{R}_T := \sum_{t=1}^T \big(U^\star - u^{(i_t)}(\overline{y}_t^{(i_t)})\big)$.
$R_T^\star$	Uniform single-task oracle scale: $R_T^\star := L^\star \sum_{s=1}^T \overline{\epsilon}_s^f$.
N_T	Number of instantiated tasks by time T: $N_T := \lvert\mathcal{I}_T\rvert$.
c_g	Scheduler/gating constant in Alg. 1 (triggers refinement when $w_t \le c_g \epsilon_m^U$).

E. Theoretical Analysis

E.1. Main regret bound

Confidence envelopes. We now establish the regret bound for GSR. As a first step, we derive CIs for the task-UCB; see Appendix H for the proofs.

Lemma E.1 (Utility CIs)**.** *Let $\delta_t := \delta_u/(\pi^2 t^2)$ and $\phi_t := \sqrt{2\sigma_u^2 \log(2/\delta_t)}$. Then w.p.$\geq 1-\delta_u$, for all $t \in [T]$,*

$$\underline{u}_t^{(i_t)} := \mathit{clip}_{[0,1]}(\tilde{u}_t^{(i_t)} - \phi_t),\ \overline{u}_t^{(i_t)} := \mathit{clip}_{[0,1]}(\tilde{u}_t^{(i_t)} + \phi_t),$$

Remark E.2 (Averaged feedback). Averaging $K_t \geq 1$ independent samples to form $\tilde{u}_t$ is (σ_u^2/K_t)-sub-Gaussian.

Theorem E.3 (Value envelopes)**.** *Under Lemma E.1 and the GP-UCB events (1), if $\overline{L} \geq L^\star$, then for every instantiated task $i \in \mathcal{I}_t$ with local counter $s_t^{(i)}$,*

$$\underline{U}_t^{(i)} := \underline{u}_t^{(i)}, \qquad \overline{U}_t^{(i)} := \mathrm{clip}_{[0,1]}\left(\overline{u}_t^{(i)} + \overline{L}\,\epsilon^f_{s_t^{(i)}}\right),$$

with initialization $[\underline{U}_0^{(i)}, \overline{U}_0^{(i)}] = [0,1]$. If the utility interval width is controlled as $\overline{u}_t^{(i)} - \underline{u}_t^{(i)} \leq c_u\,\overline{L}\,\epsilon^f_{s_t^{(i)}}$ for $c_u > 0$, then $w_t^{(i)} := \overline{U}_t^{(i)} - \underline{U}_t^{(i)} \leq (c_u+1)\overline{L}\,\epsilon^f_{s_t^{(i)}}$.

Condition E.4 (Refinement probability)**.** *Under Condition 2.3, there exists a nonzero probability $\delta_+ \in (0,1]$ such that $\forall m, s, a_t$ that is resolved finer than ϵ_m^U, a single draw $i \sim \mathsf{Gen}(a_t, m, J=1)$ hits an ϵ_m^U-optimal task for $m \geq 1$:* $\Pr\Big(i \in \mathcal{I}^\star(\epsilon_m^U) \,\Big|\, w_t \leq c_g \epsilon_{m-1}^U\Big) \;\geq\; \delta_+$.

This condition is enforced in Line 1 of Alg. 1. Conditioning on a sufficiently solved anchor enables progressively finer task proposals, down to resolution $\epsilon_m^U \lesssim L^\star \overline{\epsilon}_T^f$.

Max depth $\overline{m}_T$ For brevity, set $\Psi_T := \max_{i\in\mathcal{I}_T} \beta_T^{(i)}\gamma_T^{(i)}$. Then, $\overline{m}_T$ is set as:

Lemma E.5 (Reachability)**.** *Under Theorem E.3 and Condition E.4, assume the batch size J is a constant. Set*

$$\overline{m}_T := \left\lfloor \frac{1}{2}\log_2\left(\frac{T}{A_T(\log(eT))^2}\right)\right\rfloor_+,$$

where $A_T := \left(\frac{2C_v\,\overline{L}\sqrt{C_\lambda\,\Psi_T}}{c_g\,\epsilon_0^U}\right)^2$, and $\lfloor x \rfloor_+ = \max\{0, \lfloor x \rfloor\}$. Then, under Condition 2.3 and Algorithm 1, the optimal level $m^\star$ is introduced by time T.

Task regret bound. For brevity, we set $\beta_T := \max_{i\in\mathcal{I}_T} \beta_T^{(i)}$ and $\gamma_T := \max_{i\in\mathcal{I}_T} \gamma_T^{(i)}$.

Theorem E.6 (Regret bound)**.** *Under Theorem E.3 and Lemma E.5, let $R_T^\star := L^\star \sum_{s=1}^T \overline{\epsilon}_s^f$ be the uniform single-task regret bound. Let $N_T := |\mathcal{I}_T|$ be the number of instantiated tasks by time T and let $\Lambda_T := \log(eT)$. Then Algorithm 1 satisfies, with probability at least $1-(\delta_f+\delta_u+\delta_-)$,*

$$\mathcal{R}_T \;\leq\; C\left[\Lambda_T + \left(1+\sqrt{N_T}\right)\frac{\overline{L}}{L^\star}\right] R_T^\star, \tag{5}$$

for $C > 0$. Since $N_T \leq 1 + J\,(\overline{m}_T + 1)$ and $\overline{m}_T = \tilde{\mathcal{O}}(\log T)$ for constant J, we obtain a logarithmic up to overhead relative to the single-task BO when $\overline{L} = \Theta(L^\star)$.

Proof Sketch. Let $U_t^{\max} := \max_{i\in\mathcal{I}_t} U^{(i)}$. On the high-probability envelope and generation-success events, the instantaneous regret can be decomposed to:

$$R_T^\star \leq \underbrace{(U^\star - U_t^{\max})}_{\text{generation}} + \underbrace{(U_t^{\max} - U^{(i_t)})}_{\text{selection}} + \underbrace{\left(U^{(i_t)} - u^{(i_t)}(\overline{y}_t^{(i_t)})\right)}_{\text{within-task}}$$

(a) *Generation:* $U^\star - U_t^{\max} \leq \epsilon^U_{m_t \wedge m^\star}$, so the sum is $\mathcal{O}(\Lambda_T R_T^\star)$ under Condition 2.3.

(b) *Selection:* task-UCB implies $U_t^{\max} - U^{(i_t)} \leq w_{t-1}^{(i_t)} \lesssim \overline{L}\,\epsilon^f_{s_{t-1}^{(i_t)}}$, giving $\tilde{\mathcal{O}}((\overline{L}/L^\star)\sqrt{N_T}\,R_T^\star)$.

(c) *Within-task:* Lipschitzness + GP-UCB bound give $U^{(i)} - u^{(i)}(\overline{y}_s^{(i)}) \leq L^\star \epsilon_s^f$, giving $\tilde{\mathcal{O}}(\sqrt{N_T}\,R_T^\star)$.

Combining yields (5); see Appendix G.2.1 and I.2 for the proof. □

E.2. Discussion of the theoretical results

Q. What if we use an oracle task as the comparator? Let $\underline{R}_T := L^\star \sum_{s=1}^{T} \epsilon_s^{f,(i^\star_{\epsilon^U})}$. For RBF, the extra overhead is $\kappa_T := R_T^\star / \underline{R}_T = \tilde{\mathcal{O}}\big((\log T)^{\overline{d} - d^{(i^\star_{\epsilon^U})}}\big)$, where $\overline{d} := \max_{i \in \mathcal{I}_T} d^{(i)}$; see Lemma I.3.

Q. Isn't $\overline{L}$ hard to estimate? We adapt $\overline{L}$ online via regret-balancing (Ziomek et al., 2024); Alg. 2 in Appendix J, which incurs only a logarithmic overhead; the same idea applies to other hyperparameters. In experiments, GP hyperparameters are fit by marginal likelihood, while remaining constants are fixed ($\epsilon_0^U = 1$, $c_g = 0.5$; see Appendix J). See Figure 12 for ablation study on $\overline{L}$ estimators.

Q. Isn't Condition E.4 strong? This may be considered strong, but without it, the first term in our regret decomposition is irreducible, making sublinear task-regret information-theoretically impossible. Our assumption makes explicit what is often left implicit in practice: evolutionary LLMs already show strong empirical performance in code and proof generation (Novikov et al., 2025; Lange et al., 2025; Assumpção et al., 2025). Appendix G.3 further supports this through a Bayesian in-context learning analysis (Wu et al., 2024; Wakayama & Suzuki, 2025), showing that, conditioned on mutation history, the posterior mass on the correct refinement type concentrates exponentially fast with the number of in-context examples. In particular, Corollary G.15 gives $\delta_+ \gtrsim \rho_m(\epsilon_m^U) - \epsilon_{\text{ICL}}$, where $\rho_m(\epsilon_m^U)$ is the problem-specific, LLM-independent success probability of the true refinement, and ϵ_{ICL} is the Bayes approximation error, controlled by pretraining scale and typically small. This implies a nonzero success probability for *refinable* problems with *informative* mutation histories. Finally, §B.3 empirically confirms that δ_+ remains nonzero across levels m.

F. Preliminary

F.1. Gaussian process regression background

Gaussian process model. Let $f : \mathcal{X} \to \mathbb{R}$ be an unknown function with prior $f \sim \mathcal{GP}(0, k)$ (Stein, 1999; Williams & Rasmussen, 2006). At each local round s, we query $\mathbf{x}_s \in \mathcal{X}$ and observe $y_s = f(\mathbf{x}_s) + \xi_s^f$, where $\xi_s^f \sim \text{SubGauss}(\sigma_f^2)$. In GP regression we model this as Gaussian noise with variance $\lambda > 0$. Let $\mathbf{X}_s = [\mathbf{x}_1, \dots, \mathbf{x}_s]^\top$ and $\mathbf{Y}_s = [y_1, \dots, y_s]^\top$. Conditioned on the data $\mathbf{D}_s = (\mathbf{X}_s, \mathbf{Y}_s)$, the posterior is $f_s \mid \mathbf{D}_s \sim \mathcal{GP}(\mu_s, k_s)$, where

$$\mu_s(\mathbf{x}) = k(\mathbf{x}, \mathbf{X}_s)\,(\mathbf{K}_s + \lambda \mathbf{I})^{-1} \mathbf{Y}_s,$$
$$k_s(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{X}_s)\,(\mathbf{K}_s + \lambda \mathbf{I})^{-1}\, k(\mathbf{X}_s, \mathbf{x}'),$$

with $\mathbf{K}_s = [k(\mathbf{x}_r, \mathbf{x}_{r'})]_{r,r'=1}^{s}$ and $\sigma_s^2(\mathbf{x}) := k_s(\mathbf{x}, \mathbf{x})$.

GP confidence bounds. The f lies in a GP confidence band uniformly over time and space w.h.p.:

Theorem F.1 (Theorem 2 in (Chowdhury & Gopalan, 2017))**.** *Let $f \in \mathcal{H}(k)$ be in the RKHS associated with k with $\|f\|_k \le B$. Let $\lambda > 0$ be the noise variance used in GP regression and define $\beta_s^{1/2} := B + \sigma_{\text{noise}} \sqrt{2\big(\gamma_{s-1} + 1 + \ln(1/\delta_f)\big)}$, where γ_S upper bounds the maximum information gain: $\frac{1}{2}\log\big|\mathbf{I} + \lambda^{-1}\mathbf{K}_S\big| \le \gamma_S$. Then, w.p.$\ge 1 - \delta_f$, for all $\mathbf{x} \in \mathcal{X}$ and all $s \ge 1$,*

$$\big|f(\mathbf{x}) - \mu_{s-1}(\mathbf{x})\big| \le \beta_s^{1/2}\, \sigma_{s-1}(\mathbf{x}).$$

For $s \ge 1$, define the pointwise confidence bounds

$$\underline{f}_s(\mathbf{x}) := \mu_{s-1}(\mathbf{x}) - \beta_s^{1/2} \sigma_{s-1}(\mathbf{x}),$$
$$\overline{f}_s(\mathbf{x}) := \mu_{s-1}(\mathbf{x}) + \beta_s^{1/2} \sigma_{s-1}(\mathbf{x}).$$

Theorem F.2 (Theorem 3 in (Chowdhury & Gopalan, 2017))**.** *Assume the setting of Theorem F.1 and run GP-UCB with decision rule $\boldsymbol{x}_s \in \arg\max_{\boldsymbol{x} \in \mathcal{X}^{(i)}} \overline{f}_s^{(i)}(\mathbf{x})$. With the cumulative regret $R_{cr}(S) := \sum_{s=1}^{S} \big(f^\star - f(\boldsymbol{x}_s)\big)$, we have*

$$R_{cr}(S) \le 2\sqrt{C_\lambda\, S\, \beta_S\, \gamma_S} = S\, \epsilon_S^f,$$

where $C_\lambda := \frac{2}{\log(1+\lambda^{-1})}$ and $\epsilon_s^f := 2\sqrt{\frac{C_\lambda \beta_s \gamma_s}{s}}$..

Lemma F.3 (Uniform GP confidence over adaptively instantiated tasks)**.** *For each task $i \in \mathbb{N}$, run GP regression / GP-UCB using confidence level*

$$\delta_f^{(i)} := \frac{\delta_f}{\pi^2 i^2}.$$

Let $\beta_s^{(i)}$ be defined as in Theorem F.1 but with δ_f replaced by $\delta_f^{(i)}$. Then with probability at least $1-\delta_f$, simultaneously for all tasks $i \in \mathbb{N}$, all $s \geq 1$, and all $\mathbf{x} \in \mathcal{X}^{(i)}$,

$$\left|f^{(i)}(\mathbf{x}) - \mu_{s-1}^{(i)}(\mathbf{x})\right| \leq \left(\beta_s^{(i)}\right)^{1/2} \sigma_{s-1}^{(i)}(\mathbf{x}).$$

Consequently, all within-task optimization gap bounds $f^{\star(i)} - \overline{y}_s^{(i)} \leq \epsilon_s^f$ hold simultaneously for all instantiated tasks.

Proof. Apply Theorem F.1 to each fixed task i with confidence $\delta_f^{(i)}$. Then, by a union bound over $i \in \mathbb{N}$,

$$\Pr\left(\exists i: \text{ GP confidence for task } i \text{ fails}\right) \leq \sum_{i=1}^{\infty} \delta_f^{(i)} = \frac{\delta_f}{\pi^2} \sum_{i=1}^{\infty} \frac{1}{i^2} \leq \delta_f.$$

On the complement event, the GP confidence bands hold for all tasks simultaneously, implying the optimization gap envelopes via the same argument as Lemma H.2. □

G. LLM task generator and evaluator

G.1. Evaluator

Bradley–Terry preference model. We model the evaluator's latent preference using the popular Bradley-Terry (BT) model (Bradley & Terry, 1952):

$$\Pr\big((i,z) \succ (j,z')\big) \;=\; \sigma\Big(\theta^{(i)}(z) - \theta^{(j)}(z')\Big), \tag{6}$$

where $\sigma(x) := {}^1\!/_{1+e^{-x}}$ and $\theta^{(i)}(z) \in \mathbb{R}$ is the preference score over task-incumbent pairs; (i,z) and $(j,\overline{y}')$.

Ordinal to cardinal utility. However, BT model's $\theta^{(i)}$ is *ordinal*, not cardinal. To convert, we assume a fixed *reference* r (the seed task i_0 and its incumbent $\overline{y}_0^{(i_0)}$). We define the utility as a win probability against this fixed reference: $u^{(i)}(z) \;:=\; \Pr\big((i,z) \succ r\big) \in [0,1]$. This yields a bounded *cardinal* utility. We normalize the reference so that $\theta(r) = 0$; i.e., $u^{(i)}(z) = \sigma(\theta^{(i)}(z))$.

Moving-anchor queries. Comparisons to a fixed reference can saturate (voters always prefer the improved task), making feedback uninformative. We instead compare against a moving anchor a_t (the current anchor task) and observe votes $b_{t,k} \in \{0,1\}$, where $b_{t,k} = 1$ iff $(i_t, \overline{y}_t^{(i_t)}) \succ (a_t, \overline{y}_t^{(a_t)})$. Let $p_t := \Pr(b_{t,k} = 1 \mid \mathcal{F}_{t-1})$ be the true win rate and $\hat{p}_t := \frac{1}{K_t}\sum_{k=1}^{K_t} b_{t,k}$ its empirical estimate. Under the BT model, we have:

$$\operatorname{logit} u^{(i_t)}\big(\overline{y}_t^{(i_t)}\big) = \operatorname{logit} u^{(a_t)}\big(\overline{y}_t^{(a_t)}\big) + \operatorname{logit} p_t. \tag{7}$$

See Fig.10; given votes and a cached anchor utility interval, we can estimate cardinal utilities.

G.1.1. Preference feedback and sub-Gaussian concentration.

Our envelope theory only requires the CI interface. In the main paper we implement utility calls using *pairwise* comparisons calibrated via a global BT scale, where the raw feedback is a collection of bounded votes and thus concentrates.

Lemma G.1 (Bounded utility feedback is conditionally sub-Gaussian)**.** *Fix a utility call ℓ and condition on $\mathcal{F}_{t_\ell-1}$ and $\overline{y}_{s_\ell}^{(i_\ell)}$. Assume $\tilde{u}_\ell \in [0,1]$ and $\mathbb{E}[\tilde{u}_\ell \mid \mathcal{F}_{t_\ell-1}, \overline{y}_{s_\ell}^{(i_\ell)}] = u^{(i_\ell)}(\overline{y}_{s_\ell}^{(i_\ell)})$. If $\tilde{u}_\ell$ is formed as an average of $K_\ell \geq 1$ conditionally independent bounded samples in $[0,1]$ (with the same conditional mean), then letting $\xi_\ell := \tilde{u}_\ell - u^{(i_\ell)}(\overline{y}_{s_\ell}^{(i_\ell)})$ we have for all $\lambda \in \mathbb{R}$,*

$$\mathbb{E}\Big[\exp(\lambda\xi_\ell) \mid \mathcal{F}_{t_\ell-1}, \overline{y}_{s_\ell}^{(i_\ell)}\Big] \leq \exp\left(\frac{\lambda^2}{8K_\ell}\right).$$

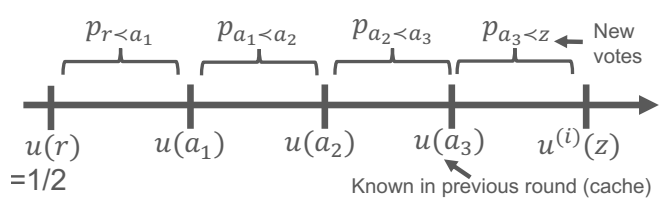

Figure 10. Log-odds chain rule: adding logits and mapping back with sigmoid offers cardinal utility $u^{(i)}(z)$ estimation.

Equivalently, ξ_ℓ is conditionally sub-Gaussian with variance proxy $1/(4K_\ell)$. In particular, for single-shot bounded feedback we recover the case $K_\ell = 1$.

Proof. Fix ℓ and condition on $\mathcal{F}_{t_\ell - 1}$ and $\overline{y} := \overline{y}_{s_\ell}^{(i_\ell)}$. Let $\mu := u^{(i_\ell)}(z)$. Let $\tilde{u}_\ell = \frac{1}{K_\ell}\sum_{k=1}^{K_\ell} b_{\ell,k}$ with $b_{\ell,k} \in [0,1]$ conditionally independent and $\mathbb{E}[b_{\ell,k} \mid \cdot] = \mu$. Then each $b_{\ell,k} - \mu$ is mean-zero and lies in $[-\mu, 1-\mu] \subset [-1,1]$. By Hoeffding's lemma, for all $\lambda \in \mathbb{R}$,

$$\mathbb{E}\big[\exp\big(\lambda(b_{\ell,k} - \mu)\big) \mid \mathcal{F}_{t_\ell - 1}, z\big] \le \exp\left(\frac{\lambda^2}{8}\right).$$

By conditional independence and $\xi_\ell = \tilde{u}_\ell - \mu = \frac{1}{K_\ell}\sum_{k=1}^{K_\ell}(b_{\ell,k} - \mu)$,

$$\mathbb{E}\big[\exp\big(\lambda\xi_\ell\big) \mid \mathcal{F}_{t_\ell - 1}, z\big] = \prod_{k=1}^{K_\ell} \mathbb{E}\left[\exp\Big(\frac{\lambda}{K_\ell}(b_{\ell,k} - \mu)\Big) \mid \mathcal{F}_{t_\ell - 1}, z\right] \le \exp\left(\frac{\lambda^2}{8K_\ell}\right).$$

□

Confidence interval for the pairwise probability. Let $\delta_\ell \in (0,1)$ be the per-call failure budget and define the Hoeffding radius

$$\theta_p(K_\ell, \delta_\ell) := \sqrt{\frac{\log(2/\delta_\ell)}{2K_\ell}}.$$

Then with probability at least $1 - \delta_\ell$ (conditional on the compared incumbents),

$$p_\ell \in \big[\hat{p}_\ell - \theta_p(K_\ell, \delta_\ell),\ \hat{p}_\ell + \theta_p(K_\ell, \delta_\ell)\big].$$

We clip endpoints to $[0,1]$ (and, when actually evaluating logits in code, one may additionally clip to $[\varepsilon_{\text{clip}}, 1 - \varepsilon_{\text{clip}}]$ to avoid overflow):

$$p_\ell^- := \text{clip}_{[0,1]}(\hat{p}_\ell - \theta_p(K_\ell, \delta_\ell)), \qquad p_\ell^+ := \text{clip}_{[0,1]}(\hat{p}_\ell + \theta_p(K_\ell, \delta_\ell)).$$

(For numerical stability in code, one may additionally clip $p_\ell^\pm$ to $[\varepsilon_{\text{clip}}, 1 - \varepsilon_{\text{clip}}]$; see the practical note below.)

Transported utility confidence interval. Assume we already maintain a valid interval for the anchor utility

$$u^{(a_\ell)}(\overline{y}_{s_\ell}^{(a_\ell)}) \in [\underline{u}^{(a_\ell)}, \overline{u}^{(a_\ell)}].$$

(For the fixed reference artifact r, we may take $\underline{u}^{(r)} = \overline{u}^{(r)} = \frac{1}{2}$ since $u(r) = \Pr(r \succ r) = \frac{1}{2}$.) Define the transported interval

$$\underline{u}^{(i_\ell)} := \sigma\Big(\text{logit}(\underline{u}^{(a_\ell)}) + \text{logit}(p_\ell^-)\Big), \qquad \overline{u}^{(i_\ell)} := \sigma\Big(\text{logit}(\overline{u}^{(a_\ell)}) + \text{logit}(p_\ell^+)\Big). \tag{8}$$

Lemma G.2 (Validity of the transported utility interval)**.** *Work on the event that (i) the anchor interval is valid and (ii) $p_\ell \in [p_\ell^-, p_\ell^+]$. Under BT model,*

$$u^{(i_\ell)}(\overline{y}_{s_\ell}^{(i_\ell)}) \in [\underline{u}^{(i_\ell)}, \overline{u}^{(i_\ell)}].$$

Proof. By transport identity,

$$\text{logit}\, u^{(i_\ell)}(\overline{y}_{s_\ell}^{(i_\ell)}) = \text{logit}\, u^{(a_\ell)}(\overline{y}_{s_\ell}^{(a_\ell)}) + \text{logit}\, p_\ell.$$

On the event that $u^{(a_\ell)}(\cdot) \in [\underline{u}^{(a_\ell)}, \overline{u}^{(a_\ell)}]$ and $p_\ell \in [p_\ell^-, p_\ell^+]$, monotonicity of $\text{logit}(\cdot)$ implies

$$\text{logit}\, u^{(i_\ell)}(\cdot) \in \big[\text{logit}(\underline{u}^{(a_\ell)}) + \text{logit}(p_\ell^-),\ \text{logit}(\overline{u}^{(a_\ell)}) + \text{logit}(p_\ell^+)\big].$$

Applying the monotone map $\sigma(\cdot)$ to endpoints gives (8). □

Uniform confidence across all utility calls (with a summable schedule). Let $\delta_u \in (0,1)$ and set $\delta_\ell := \delta_u/(\pi^2\ell^2)$. Let $\mathcal{E}_p$ be the event that $p_\ell \in [p_\ell^-, p_\ell^+]$ holds for all ℓ simultaneously. By a union bound and Hoeffding, $\Pr(\mathcal{E}_p) \ge 1 - \delta_u$. On $\mathcal{E}_p$, validity of transported intervals follows by induction over utility calls using Lemma G.2 (starting from the reference r which has a trivial valid interval).

Committee size (vote complexity). Our envelope analysis only requires that, at each checkpoint, the constructed utility interval width $\overline{u}^{(i)} - \underline{u}^{(i)}$ is below a target tolerance (e.g., $2\eta^{(i)}(s)$). With transported intervals, a practical rule is to start from a small K and keep drawing fresh votes until the width target is met.

The next theorem gives a sufficient condition that ensures a target *utility* half-width, under a mild non-degeneracy condition. (For near-optimal tasks compared against the current best anchor, this condition is typically satisfied.)

Theorem G.3 (Sufficient committee size for a target transported-utility half-width)**.** *Fix a utility call ℓ and let w_a be the* anchor log-odds width

$$w_a := \mathrm{logit}(\overline{u}^{(a_\ell)}) - \mathrm{logit}(\underline{u}^{(a_\ell)}).$$

Assume the true comparison probability satisfies $p_\ell \in [\kappa, 1-\kappa]$ for some $\kappa \in (0, \frac{1}{2}]$, and choose K_ℓ so that

$$K_\ell \;\geq\; \max\left\{\frac{8\log(2/\delta_\ell)}{\kappa^2}, \frac{32\log(2/\delta_\ell)}{\kappa^2\,(8\eta - w_a)^2}\right\}, \qquad \textit{for some target } \eta > 0 \textit{ with } w_a < 8\eta. \tag{9}$$

Then, on the event of Lemma G.2, the transported interval satisfies

$$\overline{u}^{(i_\ell)} - \underline{u}^{(i_\ell)} \leq 2\eta.$$

Proof. Let $\theta_a^- := \mathrm{logit}(\underline{u}^{(a_\ell)})$ and $\theta_a^+ := \mathrm{logit}(\overline{u}^{(a_\ell)})$ so that $w_a = \theta_a^+ - \theta_a^-$. Let $d^- := \mathrm{logit}(p_\ell^-)$ and $d^+ := \mathrm{logit}(p_\ell^+)$. Then the transported log-odds interval for $\theta^{(i_\ell)}$ has width

$$w_i := (\theta_a^+ + d^+) - (\theta_a^- + d^-) = w_a + (d^+ - d^-).$$

Since σ has derivative at most $1/4$, it is $1/4$-Lipschitz, hence

$$\overline{u}^{(i_\ell)} - \underline{u}^{(i_\ell)} \leq \frac{1}{4}w_i = \frac{1}{4}w_a + \frac{1}{4}(d^+ - d^-).$$

It remains to control $d^+ - d^-$. The first term in (9) gives $\theta_p \leq \kappa/4$. On the Hoeffding event, $\hat{p}_\ell \geq p_\ell - \theta_p$ and $\hat{p}_\ell \leq p_\ell + \theta_p$, hence $p_\ell^- \geq p_\ell - 2\theta_p \geq \kappa/2$ and $p_\ell^+ \leq p_\ell + 2\theta_p \leq 1 - \kappa/2$. Thus $p_\ell^\pm \in [\kappa/2, 1-\kappa/2]$. On this interval, $\mathrm{logit}'(p) = 1/(p(1-p)) \leq 4/\kappa$ since $p(1-p) \geq (\kappa/2)(1/2) = \kappa/4$. By the mean value theorem,

$$d^+ - d^- = \mathrm{logit}(p_\ell^+) - \mathrm{logit}(p_\ell^-) \leq \frac{4}{\kappa}(p_\ell^+ - p_\ell^-) \leq \frac{4}{\kappa}\cdot 2\theta_p = \frac{8\theta_p}{\kappa}.$$

Therefore

$$\overline{u}^{(i_\ell)} - \underline{u}^{(i_\ell)} \leq \frac{w_a}{4} + \frac{2\theta_p}{\kappa}.$$

To make this at most 2η, it suffices that

$$\theta_p \leq \frac{\kappa}{2}\Big(2\eta - \frac{w_a}{4}\Big) = \frac{\kappa}{8}(8\eta - w_a).$$

Since $\theta_p = \sqrt{\log(2/\delta_\ell)/(2K_\ell)}$, the inequality above is equivalent to

$$K_\ell \;\geq\; \frac{32\log(2/\delta_\ell)}{\kappa^2\,(8\eta - w_a)^2},$$

which is exactly (9). □

Remark G.4 (Practical committee sizing via doubling). In practice, κ is unknown and may vary across calls. A robust choice is a doubling rule: start from a small K and keep sampling fresh voters until the computed width $\overline{u}^{(i_\ell)} - \underline{u}^{(i_\ell)}$ drops below the target (e.g., $2\eta^{(i)}(s)$). Theorem G.3 implies that when the comparison is informative (i.e., p_ℓ is not extremely close to 0 or 1 for tasks that remain competitive), the required K scales as $\tilde{\mathcal{O}}(\eta^{-2})$.

Remark G.5 (Anchor-width limitation of transported intervals). Transported logit intervals inherit the anchor log-odds width w_a. Therefore increasing the number of votes for the child comparison can shrink only the comparison part of the interval, not the cached anchor uncertainty. If $w_a \geq 8\eta$, the target child half-width η is unattainable from that anchor; the implementation must first refine the anchor interval, use a more certain reference, or fall back to direct bounded utility feedback. The regret analysis uses only the resulting valid CI interface and its width, not the particular transport mechanism.

Choosing K_s explicitly (and why it is not circular). If we want the transported interval to have a *target* utility half-width $\eta_s > 0$ at step s, we can choose K_s using Theorem G.3 as follows. Let the anchor *log-odds width* be

$$w_{a_t} \;:=\; \mathrm{logit}(\overline{u}^{(a_t)}) - \mathrm{logit}(\underline{u}^{(a_t)}),$$

which is known *before* drawing the K_s votes at step s because it depends only on the already-maintained anchor interval. Assume the comparison is not saturated: $p_s \in [\kappa, 1-\kappa]$ for some fixed $\kappa \in (0, \frac{1}{2}]$. If $w_{a_t} < 8\eta_s$, then it suffices to pick

$$K_s \;:=\; \left\lceil \max\left\{ \frac{8\log(2/\delta_u)}{\kappa^2},\; \frac{32\log(2/\delta_u)}{\kappa^2\,(8\eta_s - w_{a_t})^2} \right\} \right\rceil. \tag{10}$$

With this choice, Theorem G.3 guarantees $\overline{u}_s^{(i)} - \underline{u}_s^{(i)} \le 2\eta_s$ on the same event as Lemma G.2. There is no circular dependence: w_{a_t} depends on the *anchor*'s previously computed interval, not on the current K_s.

Practical note on $\varepsilon_{\mathrm{clip}}$. The lemma/proof above does not require any $\varepsilon_{\mathrm{clip}}$ (we allow $\mathrm{logit}(0), \mathrm{logit}(1)$ as $\pm\infty$). In code, to avoid numerical overflow, a safe rule of thumb is to clip only for computation: replace $p_s^{\pm}$ by $\mathrm{clip}_{[\varepsilon_{\mathrm{clip}}, 1-\varepsilon_{\mathrm{clip}}]}(p_s^{\pm})$ with, e.g.,

$$\varepsilon_{\mathrm{clip}} = 10^{-6}\ \ (\text{float32}) \qquad \text{or} \qquad \varepsilon_{\mathrm{clip}} = 10^{-12}\ \ (\text{float64}),$$

and if $p_s^- \le \varepsilon_{\mathrm{clip}}$ (resp. $p_s^+ \ge 1 - \varepsilon_{\mathrm{clip}}$) simply set the transported bound to 0 (resp. 1). There is no clean way to choose $\varepsilon_{\mathrm{clip}}$ purely from δ_u because δ_u controls estimation error around p_s while the true p_s itself can be arbitrarily close to 0 or 1.

A simple vote budget in terms of $\beta_s^{(i)}$ and $\gamma_s^{(i)}$. The envelope theory only needs that, at each utility call, the produced interval $\left[\underline{u}, \overline{u}\right]$ is valid and has width below a target tolerance. A convenient sufficient condition (covering direct bounded scalar ratings or Bernoulli vote averages) is obtained by combining Hoeffding concentration with the GP-UCB optimization gap ϵ_s^f.

Lemma G.6 (Sufficient committee size for width $\propto \epsilon_s^f$)**.** *Fix a utility call ℓ that occurs when task i_ℓ has local counter s_ℓ. Assume the feedback $\tilde{u}_\ell \in [0,1]$ is formed as the average of $K_\ell \ge 1$ conditionally independent bounded samples in $[0,1]$ with conditional mean $u^{(i_\ell)}\big(\overline{y}_{s_\ell}^{(i_\ell)}\big)$. Let the per-call failure budget be $\delta_\ell := \delta_u/(\pi^2\ell^2)$ and define the Hoeffding radius*

$$\phi_\ell := \sqrt{\frac{\log(2/\delta_\ell)}{2K_\ell}}.$$

If K_ℓ satisfies

$$K_\ell \;\ge\; \left\lceil \frac{s_\ell}{2\,c_u^2\,C_\lambda\,\overline{L}^2\,\beta_{s_\ell}^{(i_\ell)}\,\gamma_{s_\ell}^{(i_\ell)}}\;\log\!\Big(\frac{2}{\delta_\ell}\Big) \right\rceil, \tag{11}$$

then on the event of the corresponding Hoeffding bound we have

$$\overline{u}_\ell^{(i_\ell)} - \underline{u}_\ell^{(i_\ell)} \;=\; 2\phi_\ell \;\le\; c_u\,\overline{L}\,\epsilon_{s_\ell}^f, \qquad \epsilon_s^f := 2\sqrt{C_\lambda\,\beta_s^{(i)}\gamma_s^{(i)}/s} \quad \textit{(for the corresponding task } i\textit{)..}$$

Proof. We require $2\phi_\ell \le c_u\,\overline{L}\,\epsilon_{s_\ell}^f$. Since $\phi_\ell = \sqrt{\log(2/\delta_\ell)/(2K_\ell)}$, this is equivalent to

$$K_\ell \;\ge\; \frac{2\log(2/\delta_\ell)}{c_u^2\,\overline{L}^2\,(\epsilon_{s_\ell}^{f,(i_\ell)})^2}.$$

Using $(\epsilon_s^f)^2 = 4C_\lambda\,\beta_s^{(i)}\gamma_s^{(i)}/s$ gives (11). □

G.1.2. Total number of votes under sparse utility calls.

Assume the sparse schedule $\mathcal{S}_u = \{1, 2, 4, \dots\}$ and let $N_i := s_T^{(i)}$ be the number of objective evaluations allocated to task i by time T. Let U_T denote the total number of utility calls made by time T (across all tasks). Then each task i contributes at most $1 + \lfloor \log_2 N_i \rfloor$ calls, hence

$$U_T \;\le\; \sum_{i \in \mathcal{I}_T} \big(1 + \lfloor \log_2 N_i \rfloor\big) \;\le\; N_T\big(1 + \log_2 T\big).$$

Define the total number of votes

$$\mathrm{Votes}(T) \;:=\; \sum_{\ell=1}^{U_T} K_\ell.$$

If we choose K_ℓ according to (11) at every utility call, then using $\log(2/\delta_\ell) \le \log\!\big(2\pi^2 U_T^2/\delta_u\big)$ for all $\ell \le U_T$ yields

$$\mathrm{Votes}(T) \;\le\; \frac{\log\!\big(2\pi^2 U_T^2/\delta_u\big)}{2\,c_u^2\,C_\lambda\,\overline{L}^2} \sum_{i\in\mathcal{I}_T} \sum_{r=0}^{\lfloor \log_2 N_i \rfloor} \frac{2^r}{\beta_{2^r}^{(i)}\,\gamma_{2^r}^{(i)}} \;+\; U_T, \tag{12}$$

where the final $+U_T$ accounts for the ceilings in (11). (When the ceilings are dropped, the bound holds without the $+U_T$ term.)

Corollary G.7 (Big-$\mathcal{O}$ vote complexity under sparse utility calls)**.** *Assume the sparse utility schedule $\mathcal{S}_u = \{1, 2, 4, \dots\}$ and choose the committee sizes K_ℓ according to (11) at every utility call. Recall $N_i := s_T^{(i)}$, $N_T := |\mathcal{I}_T|$, and $U_T \le N_T(1+\log_2 T)$. Then (12) implies the deterministic bound*

$$\mathrm{Votes}(T) \;\le\; \frac{\log\!\left(\frac{2\pi^2 N_T^2 (1+\log_2 T)^2}{\delta_u}\right)}{2\,c_u^2\,C_\lambda\,\overline{L}^2} \sum_{i\in\mathcal{I}_T} \sum_{r=0}^{\lfloor \log_2 N_i \rfloor} \frac{2^r}{\beta_{2^r}^{(i)}\,\gamma_{2^r}^{(i)}} \;+\; N_T(1+\log_2 T). \tag{13}$$

In particular, under Algorithm 1 with constant batch size J we have $N_T \le 1 + J(\overline{m}_T + 1) = \tilde{\mathcal{O}}(\log T)$ (hence $U_T = \tilde{\mathcal{O}}(\log^2 T)$), so the logarithmic prefactor in (13) is $\tilde{\mathcal{O}}(\log\log T)$ and the additive term is $\tilde{\mathcal{O}}(\log^2 T)$.

Moreover, suppose there exists a nondecreasing function $g : \mathbb{N} \to \mathbb{R}_+$ and a constant $c_{\mathrm{ig}} > 0$ such that for all instantiated tasks $i \in \mathcal{I}_T$ and all $s \ge 1$,

$$\beta_s^{(i)}\,\gamma_s^{(i)} \;\ge\; c_{\mathrm{ig}}\,g(s). \tag{14}$$

Then, absorbing the prefactor in (13) into $\tilde{\mathcal{O}}(\cdot)$,

$$\mathrm{Votes}(T) \;\le\; \tilde{\mathcal{O}}\left(\sum_{i\in\mathcal{I}_T} \sum_{r=0}^{\lfloor \log_2 N_i \rfloor} \frac{2^r}{g(2^r)}\right) \;+\; \tilde{\mathcal{O}}\big(N_T \log T\big). \tag{15}$$

Two useful special cases are:

(a) ***Polynomial growth.*** *If $g(s) = s^{2\alpha}$ for some $\alpha > 0$, then letting $(x)_+ := \max\{x, 0\}$,*

$$\mathrm{Votes}(T) \;\le\; \tilde{\mathcal{O}}\big(N_T^{2\alpha}\,T^{(1-2\alpha)_+}\big), \tag{16}$$

and in particular if $N_T = \mathrm{polylog}(T)$ then $\mathrm{Votes}(T) = o(T)$.

(b) ***Polylog growth.*** *If $g(s) = (\log(es))^p$ for some $p > 0$, then*

$$\mathrm{Votes}(T) \;\le\; \tilde{\mathcal{O}}\left(\sum_{i\in\mathcal{I}_T} \frac{N_i}{(\log(eN_i))^p}\right) \;\le\; \tilde{\mathcal{O}}\left(\frac{T}{(\log(eT))^p} + N_T\right) \;=\; o(T) \qquad (\textit{if } N_T = \mathrm{polylog}(T)). \tag{17}$$

For example, for squared-exponential (RBF) kernels on $[0,1]^d$ one may take $p = 2(d+1)$, yielding $\mathrm{Votes}(T) = \tilde{\mathcal{O}}\big(T/(\log T)^{2(d+1)}\big)$.

Proof sketch. Plug $U_T \le N_T(1+\log_2 T)$ into (12) to obtain (13). If (14) holds, then each summand satisfies $\frac{2^r}{\beta_{2^r}^{(i)}\gamma_{2^r}^{(i)}} \le \frac{1}{c_{\mathrm{ig}}}\frac{2^r}{g(2^r)}$, yielding (15). For $g(s) = s^{2\alpha}$, the inner sum is a geometric series in $2^{(1-2\alpha)r}$ and the across-task bound $\sum_i N_i^{1-2\alpha} \le N_T^{2\alpha} T^{1-2\alpha}$ (for $\alpha \in (0, \frac{1}{2})$) gives (16) (up to polylog factors, and with the standard modifications at $\alpha \ge \frac{1}{2}$). For $g(s) = (\log(es))^p$, the sum over r is dominated by the last term, giving the displayed bound in (17). □

G.2. Generator

G.2.1. Proof of Lemma E.5

Proof. Let m_T denote the current ladder level in Algorithm 1. We show that with the stated choice of $\overline{m}_T$, every level $m \le \overline{m}_T$ is introduced by time T.

Step 1: an explicit upper bound on the per-level sample threshold s_m. Fix a level $m \geq 0$ and define

$$s_m \;:=\; \min\Big\{s \in \mathbb{N} :\; C_v \overline{L}\, \overline{\epsilon}^f_s \;\leq\; c_g\, \epsilon^U_m\Big\}.$$

Using the monotonicity $\beta^{(i)}_s \leq \beta^{(i)}_T$ and $\gamma^{(i)}_s \leq \gamma^{(i)}_T$ for $s \leq T$ and the definitions $\beta_T := \max_{i \in \mathcal{I}_T} \beta^{(i)}_T$, $\gamma_T := \max_{i \in \mathcal{I}_T} \gamma^{(i)}_T$, we have for all $s \leq T$,

$$\overline{\epsilon}^f_s = \max_{i \in \mathcal{I}_T} 2\sqrt{\frac{C_\lambda\, \beta^{(i)}_s\, \gamma^{(i)}_s}{s}} \;\leq\; 2\sqrt{\frac{C_\lambda\, \beta_T\, \gamma_T}{s}} =: U(s).$$

Here, if we have the inequality

$$C_v \overline{L}\, U(s) \leq c_g \epsilon^U_m,$$

Then, we can say

$$C_v \overline{L} \overline{\epsilon}^f \leq C_v \overline{L}\, U(s) \leq c_g \epsilon^U_m.$$

i.e.,

$$C_v \overline{L} \cdot 2\sqrt{\frac{C_\lambda\, \beta_T\, \gamma_T}{s}} \leq c_g \epsilon^U_m \quad \Longleftrightarrow \quad s \geq \left(\frac{2 C_v \overline{L} \sqrt{C_\lambda\, \beta_T\, \gamma_T}}{c_g\, \epsilon^U_m}\right)^2.$$

Consequently, the following condition can enforce the first inequality:

$$s_m \;\leq\; \left\lceil \left(\frac{2 C_v \overline{L} \sqrt{C_\lambda\, \beta_T\, \gamma_T}}{c_g\, \epsilon^U_m}\right)^2 \right\rceil.$$

Since $\epsilon^U_m = \epsilon^U_0 2^{-m}$, this yields

$$s_m \;\leq\; \left\lceil \left(\frac{2 C_v \overline{L} \sqrt{C_\lambda\, \beta_T\, \gamma_T}}{c_g\, \epsilon^U_0}\right)^2 4^m \right\rceil \;=\; \lceil A_T\, 4^m \rceil,$$

where $A_T := \left(\frac{2 C_v\, \overline{L} \sqrt{C_\lambda\, \beta_T\, \gamma_T}}{c_g\, \epsilon^U_0}\right)^2$.

Step 2: once level m is active, it must advance within τ_m rounds. Let t_m be the first global round at which the algorithm's current level becomes m. Up to and including level m, the algorithm has instantiated at most $K_m := 1 + J(m{+}1)$ tasks deterministically (one batch of size J per level, including level 0).

Consider the $\tau_m := K_m s_m$ rounds $\{t_m, t_m + 1, \ldots, t_m + \tau_m - 1\}$ during which the current level remains m. Exactly one task is evaluated per round, so by the pigeonhole principle some instantiated task $\tilde{i}$ is evaluated at least s_m times within these τ_m rounds. At the moment $\tilde{i}$ reaches local count s_m, the envelope-width bound on the event of Theorem E.3 implies

$$w^{(\tilde{i})}_{s_m} \;\leq\; C_v \overline{L}\, \overline{\epsilon}^f_{s_m} \;\leq\; c_g\, \epsilon^U_m.$$

Hence at some round $t \leq t_m + \tau_m$ we have $\underline{w}_t := \min_{j \in \mathcal{I}_t} w^{(j)}_t \leq c_g \epsilon^U_m$. By the anchor definition (4), $\overline{\epsilon^U}_t = \max\{c_g \epsilon^U_m,\, \underline{w}_t\} = c_g \epsilon^U_m$, so the anchor a_t is chosen from a nonempty set of tasks satisfying $w^{(j)}_{s^{(j)}_t} \leq \overline{\epsilon^U}_t$. Therefore the anchor width satisfies

$$w_t = w^{(a_t)}_{s^{(a_t)}_t} \;\leq\; \overline{\epsilon^U}_t \;=\; c_g \epsilon^U_m,$$

and the scheduler condition in Algorithm 1 triggers, incrementing the level to $m{+}1$. Thus, level $m{+}1$ is introduced by time at most $t_m + \tau_m$.

Step 3: summing τ_m up to $\overline{m}_T$ fits within the horizon. By iterating the previous argument, level M is introduced by time at most

$$\sum_{m=0}^{M-1} \tau_m.$$

We now bound this sum for $M = \overline{m}_T$.

Using $s_m \le \lceil A_T 4^m \rceil \le A_T 4^m + 1$ and $N_m = J(m{+}1)$,

$$\sum_{m=0}^{M-1} \tau_m \;\le\; J \sum_{m=0}^{M-1} (m{+}1)\,(A_T 4^m + 1) = J A_T \sum_{m=0}^{M-1} (m{+}1) 4^m \;+\; J \sum_{m=0}^{M-1} (m{+}1).$$

For the first sum, use $(m{+}1) \le M$ and $\sum_{m=0}^{M-1} 4^m \le \frac{4^M}{3}$:

$$\sum_{m=0}^{M-1} (m{+}1) 4^m \;\le\; M \sum_{m=0}^{M-1} 4^m \;\le\; \frac{M\,4^M}{3}.$$

For the second sum, $\sum_{m=0}^{M-1} (m{+}1) = \frac{M(M+1)}{2} \le M^2$. Thus

$$\sum_{m=0}^{M-1} \tau_m \;\le\; \frac{J A_T M\,4^M}{3} \;+\; J M^2.$$

Now set $M = \overline{m}_T$. By definition of $\overline{m}_T$ we have $4^M \le \frac{T}{A_T (\log(eT))^2}$, hence

$$\frac{J A_T M\,4^M}{3} \;\le\; \frac{JM}{3(\log(eT))^2}\, T.$$

If J is an absolute constant (or more generally $J \le \log(eT)$), then since $M = \mathcal{O}(\log T)$ we have $\frac{JM}{(\log(eT))^2} = o(1)$, and for all T large enough (or by absorbing finitely many small T into constants)

$$\sum_{m=0}^{M-1} \tau_m \;\le\; T.$$

Therefore every level $m \le \overline{m}_T$ is introduced by time T.

Step 4: concluding $m^\star$ and the order of $\overline{m}_T$. If $m^\star \le \overline{m}_T$ (Condition 2.3), then level $m^\star$ is introduced by time T. Finally,

$$\overline{m}_T = \left\lfloor \tfrac{1}{2} \log_2\!\left(\tfrac{T}{A_T (\log(eT))^2} \right) \right\rfloor_+ = \mathcal{O}(\log T).$$

□

G.3. Bayesian in-context learning interpretation of Condition E.4

G.3.1. Known results

Setup. (Wakayama & Suzuki, 2025) analyze in-context learning in a *finite mixture of task types* (their Def. 1) and prove (a) a Bayes-risk decomposition (Thm. 1), (b) a finite-sample Bayes-gap bound for a uniform-attention Transformer (Thm. 2), and (c) exponential *task-type identification* in mixtures (Thm. 3).

Theorem G.8 (Bayes gap bound; Theorem 2 in (Wakayama & Suzuki, 2025))**.** *(Informal restatement.) Under the prompt-generating process (Def. 1) and boundedness/independence assumptions (Assumptions 1–2), and assuming the Bayes predictor is Hölder smooth, the ERM Transformer $M_{\hat\theta}$ satisfies an upper bound of the form*

$$\mathbb{E}\, R_{\mathrm{BG}}(M_{\hat\theta}) \;\lesssim\; d_{\textit{feature}}^{-2\alpha/d_{\mathrm{eff}}} \;+\; \widetilde{O}\!\left(\frac{d_{\textit{feature}}}{pN} + \frac{1}{N} \right),$$

where p is the pretraining context length, N is the number of pretraining prompts, and $d_{\textit{feature}}$ is the learned feature dimension.

Assumption G.9 (task-type identification conditions; Theorem 3 in (Wakayama & Suzuki, 2025))**.** Fix a (true) task-type index $i^\star \in \{1, \dots, T\}$. For each wrong type $j \neq i^\star$, define the predictive log-likelihood ratio increment

$$Z_{j,t} \;:=\; \log \frac{p_j(y_t \mid x_t, D_{t-1})}{p_{i^\star}(y_t \mid x_t, D_{t-1})}.$$

Assume there exist constants $D_j > 0$ and (ν_j, b_j) such that, under the true type $i^\star$ and for all t,

$$\mathbb{E}[Z_{j,t} \mid \mathcal{G}_{t-1}, I = i^\star] \;\leq\; -D_j, \qquad \mathbb{E}[\exp\{\lambda(Z_{j,t} + D_j)\} \mid \mathcal{G}_{t-1}, I = i^\star] \;\leq\; \exp\left(\frac{\lambda^2 \nu_j^2}{2}\right)$$

for all $|\lambda| \leq 1/b_j$, where $\mathcal{G}_{t-1}$ is the natural filtration. Define

$$D_{\min} := \min_{j \neq i^\star} D_j > 0, \qquad C := \min_{j \neq i^\star} \frac{D_j^2}{8(\nu_j^2 + b_j D_j/2)} > 0.$$

Lemma G.10 (Posterior mass on the true type concentrates exponentially; Theorem 3 in (Wakayama & Suzuki, 2025))**.** *Let $\pi_i(D_k) := \Pr(I = i \mid D_k)$ be the Bayes posterior over the type index. Under Assumption G.9, for every $k \geq 1$,*

$$\mathbb{E}\big[1 - \pi_{i^\star}(D_k) \mid I = i^\star\big] \leq \frac{1 - \alpha_{i^\star}}{\alpha_{i^\star}} e^{-D_{\min} k/2} \;+\; (T-1)e^{-Ck}, \tag{18}$$

$$\Pr\left(\pi_{i^\star}(D_k) \;\geq\; \frac{1}{1 + \frac{1-\alpha_{i^\star}}{\alpha_{i^\star}} e^{-D_{\min} k/2}} \;\middle|\; I = i^\star\right) \geq\; 1 - (T-1)e^{-Ck}. \tag{19}$$

G.3.2. Refinement model

Task generation adjustment. At each refinement level m, interpret the generator's behavior as selecting among finitely many *refinement types* (or modes) $I^{(m)} \in \{1, \dots, T_m\}$ based on in-context evidence (utility-improvement history).

Assumption G.11 (Refinement types form a finite mixture; adapted from Def. 1 in (Wakayama & Suzuki, 2025))**.** Fix a time t and refinement level m. Conditioned on the event $w_t \leq c_g \epsilon_{m-1}^U$, there is a latent refinement type $I^{(m)} \in \{1, \dots, T_m\}$ with prior weights $\alpha_i^{(m)} > 0$ ($\sum_i \alpha_i^{(m)} = 1$). The prompt contains k in-context examples

$$\mathcal{D}_{t,k}^{(m)} = \{s_1, \dots, s_k\},$$

where each s_r encodes (anchor a_t, level m, and feedback such as utility improvements / preferences). The conditional predictive distributions $p_i^{(m)}(\cdot \mid \mathcal{D}_{t,r-1}^{(m)})$ satisfy the identifiability conditions (Assumption G.9) with constants $D_{\min}^{(m)}$, $C^{(m)}$ and mixture size T_m.

Assumption G.12 (Correctness of the true refinement type implies ϵ_m^U-success)**.** Let $\mathcal{I}_m^\star(a_t)$ denote the set of ϵ_m^U-successful refinements. There exists a distinguished (true) refinement type $i_m^\star = i_m^\star(a_t)$ such that, conditioned on $w_t \leq c_g \epsilon_{m-1}^U$, the data $\mathcal{D}_{t,k}^{(m)}$ are generated under $I^{(m)} = i_m^\star$, and moreover a single refinement proposal drawn *given the true type* succeeds with probability at least $1 - \beta_m$:

$$\Pr\Big(\iota \in \mathcal{I}_m^\star(a_t) \;\Big|\; \widehat{I} = i_m^\star,\; a_t, m, \mathcal{D}_{t,k}^{(m)}\Big) \;\geq\; 1 - \beta_m, \qquad \beta_m \in [0, 1).$$

Assumption G.13 (LLM approximately samples the Bayes posterior over refinement types)**.** Let $q_{m,k}(\cdot \mid a_t, \mathcal{D}_{t,k}^{(m)})$ be the LLM-induced distribution over refinement types $\widehat{I} \in \{1, \dots, T_m\}$ (e.g., induced by decoding / sampling). Let $\pi^{(m)}(\cdot \mid a_t, \mathcal{D}_{t,k}^{(m)})$ be the Bayes posterior over types. Assume a (uniform) total-variation approximation error

$$\mathrm{TV}\Big(q_{m,k}(\cdot \mid a_t, \mathcal{D}_{t,k}^{(m)}),\; \pi^{(m)}(\cdot \mid a_t, \mathcal{D}_{t,k}^{(m)})\Big) \;\leq\; \varepsilon_{\mathrm{ICL}}(m, k).$$

G.3.3. Main theorem: Condition 4.4 with explicit $\delta_+(m,k)$

Theorem G.14 (Bayesian ICL $\Rightarrow$ refinement probability)**.** *Fix t, m and suppose $w_t \le c_g \epsilon^U_{m-1}$. Under Assumptions G.11–G.13, a single call $\iota \sim \mathsf{Gen}(a_t, m, J{=}1)$ satisfies*

$$\Pr\big(\iota \in \mathcal{I}^\star_m(a_t) \;\big|\; w_t \le c_g \epsilon^U_{m-1}\big) \;\ge\; \delta_+(m,k),$$

where the (level-dependent) lower bound can be chosen as

$$\delta_+(m,k) \;:=\; \rho_m(\epsilon^U_m)\Bigg[1 - \underbrace{\frac{1-\alpha^{(m)}_{i^\star_m}}{\alpha^{(m)}_{i^\star_m}} e^{-D^{(m)}_{\min}k/2} - (T_m-1)e^{-C^{(m)}k}}_{=:\,\eta_m(k)}\Bigg]_+ \;-\; \rho_m(\epsilon^U_m)\,\varepsilon_{\mathrm{ICL}}(m,k), \tag{20}$$

with $[u]_+ := \max\{u, 0\}$. In particular, for any fixed m, $\delta_+(m,k)$ improves exponentially fast *in k up to the approximation error $\varepsilon_{\mathrm{ICL}}(m,k)$.*

Proof. Write the success event as $S := \{\iota \in \mathcal{I}^\star_m(a_t)\}$. Let $\widehat{I} \in \{1, \dots, T_m\}$ denote the sampled refinement type underlying the proposal. By the law of total probability and Assumption G.12,

$$\Pr(S \mid a_t, m, \mathcal{D}^{(m)}_{t,k}) \;\ge\; \Pr(S \mid \widehat{I} = i^\star_m, a_t, m, \mathcal{D}^{(m)}_{t,k})\,\Pr(\widehat{I} = i^\star_m \mid a_t, m, \mathcal{D}^{(m)}_{t,k}) \;\ge\; (1-\beta_m)\, q_{m,k}(i^\star_m \mid a_t, \mathcal{D}^{(m)}_{t,k}).$$

By total-variation control (Assumption G.13),

$$q_{m,k}(i^\star_m \mid a_t, \mathcal{D}^{(m)}_{t,k}) \;\ge\; \pi^{(m)}(i^\star_m \mid a_t, \mathcal{D}^{(m)}_{t,k}) \;-\; \varepsilon_{\mathrm{ICL}}(m,k).$$

Taking expectations over $\mathcal{D}^{(m)}_{t,k}$ under the true type $I^{(m)} = i^\star_m$, we obtain

$$\Pr(S \mid w_t \le c_g \epsilon^U_{m-1}) \;\ge\; (1-\beta_m)\Big(\mathbb{E}[\pi^{(m)}(i^\star_m \mid a_t, \mathcal{D}^{(m)}_{t,k})] - \varepsilon_{\mathrm{ICL}}(m,k)\Big).$$

Finally, Lemma G.10 (applied at level m with $T = T_m$, $\alpha = \alpha^{(m)}$, and constants $D^{(m)}_{\min}, C^{(m)}$) yields the exponential lower bound on $\mathbb{E}[\pi^{(m)}(i^\star_m \mid a_t, \mathcal{D}^{(m)}_{t,k})]$ via $\mathbb{E}[\pi] = 1 - \mathbb{E}[1-\pi]$ and (18). Combining these inequalities gives (20). □

Shorthand used in the main text. Assumption G.12 lower bounds success *conditional on the true refinement type*. To match the main-text discussion, define the within-type success probability

$$\rho_m(\epsilon^U_m) \;:=\; 1 - \beta_m \;\in (0, 1]. \tag{21}$$

Moreover, we will use a simplified (large-k) Bayes-approximation error proxy

$$\epsilon_{\mathrm{ICL}} \;:=\; \sup_{m \le \overline{m}_T} \;\sup_{k \ge k_0} \;\varepsilon_{\mathrm{ICL}}(m,k), \tag{22}$$

for some reference in-context length k_0 at which the exponential type-identification term becomes negligible (cf. Lemma G.10).

Corollary G.15 (Large-k simplification (main-text form))**.** *Work under the assumptions of Theorem G.14. Fix any rung m and suppose the in-context length k is large enough that the WS type-identification term satisfies $\eta_m(k) \le 1/2$ (e.g., by Lemma G.10). Then the single-draw refinement success probability satisfies*

$$\delta_+(m,k) \;\ge\; \frac{1}{2}\rho_m(\epsilon^U_m) \;-\; \rho_m(\epsilon^U_m)\,\varepsilon_{\mathrm{ICL}}(m,k) \;\ge\; \frac{1}{2}\rho_m(\epsilon^U_m) \;-\; \epsilon_{\mathrm{ICL}}. \tag{23}$$

In particular, up to constant factors (absorbed by $\gtrsim$), this matches the main-text summary $\delta_+ \gtrsim \rho_m(\epsilon^U_m) - \epsilon_{\mathrm{ICL}}$.

Proof. By Theorem G.14 and the definition of $\eta_m(k)$,

$$\delta_+(m,k) \;\ge\; \rho_m(\epsilon^U_m)\big(1 - \eta_m(k)\big) \;-\; \rho_m(\epsilon^U_m)\varepsilon_{\mathrm{ICL}}(m,k).$$

If $\eta_m(k) \le 1/2$, then $\rho_m(\epsilon^U_m)(1-\eta_m(k)) \ge \rho_m(\epsilon^U_m)/2$, giving the first inequality. The second uses $\varepsilon_{\mathrm{ICL}}(m,k) \le \epsilon_{\mathrm{ICL}}$ for $k \ge k_0$ by (22). □

Corollary G.16 (Batching J samples at level m)**.** *Under the conditions of Theorem G.14, if you draw J i.i.d. refinements $\iota^{(1)}, \ldots, \iota^{(J)}$ from $\mathsf{Gen}(a_t, m, J{=}1)$, then*

$$\Pr\Big(\exists j \in \{1, \ldots, J\} : \iota^{(j)} \in \mathcal{I}^\star_m(a_t) \;\Big|\; w_t \le c_g \epsilon^U_{m-1}\Big) \;\ge\; 1 - (1 - \delta_+(m,k))^J \;\ge\; 1 - \exp\{-J\,\delta_+(m,k)\}.$$

Remark G.17 (How to get a clean level-dependent $\delta_+(m)$). If your Appendix B uses a per-level constant $\delta_+(m)$ (no explicit k), you can fix a per-level prompt budget $k = k_m$ and define

$$\delta_+(m) \;:=\; \delta_+(m, k_m)$$

using (20). The exponential terms show that $k_m = O(\log(1/\eta))$ is sufficient to make the mixture-identification loss $\le \eta$ (up to $\varepsilon_{\mathrm{ICL}}(m, k_m)$).

G.3.4. Near-optimal task generation: batch-size selection

This subsection collects the batch-success guarantees used when Algorithm 1 introduces a new rung and calls the generator. We start with a one-shot i.i.d. batching bound that holds for any (possibly unknown) generator success probability δ_+, and then specialize it to (i) a δ_+-calibrated choice that yields a uniform success guarantee across all introduced rungs, and (ii) a δ_+-free choice whose success probability depends explicitly on δ_+.

Single-batch success (arbitrary J, possibly unknown δ_+).

Lemma G.18 (Batch success with arbitrary size)**.** *Assume Condition E.4 holds with some (unknown) $\delta_+ \in (0, 1]$. Consider any generator call at rung $m \ge 1$ for which the anchor task a_t satisfies the precondition of Condition E.4 at the moment of generation (e.g., a_t is ϵ^U_m-solved; Definition 2.2). Let $J \ge 1$ and draw $i_1, \ldots, i_J \overset{\text{i.i.d.}}{\sim} \mathsf{Gen}(a_t, m)$. Then*

$$\Pr\Big(\forall k \in [J] :\; i_k \notin \mathcal{I}^\star(\epsilon^U_m)\Big) \le (1 - \delta_+)^J \le \exp(-\delta_+ J).$$

Equivalently, the batch contains an ϵ^U_m-optimal task with probability at least $1 - \exp(-\delta_+ J)$.

Proof. Under the stated precondition, Condition E.4 implies that a single draw fails with probability at most $1 - \delta_+$. Independence of the J draws yields

$$\Pr\Big(\forall k \in [J] :\; i_k \notin \mathcal{I}^\star(\epsilon^U_m)\Big) \le (1 - \delta_+)^J.$$

Finally, $(1 - \delta_+)^J \le \exp(-\delta_+ J)$ is standard. □

Uniform success across introduced rungs (known δ_+).

Lemma G.19 (Near-optimal task generation)**.** *Assume Condition E.4. Fix $\delta_- \in (0, 1)$. Suppose that whenever Algorithm 1 introduces a new rung m, it draws an i.i.d. batch of size J from $\mathsf{Gen}(a, m)$ where a is the anchor parent at that call.*

Choose

$$J := \left\lceil \frac{\log\big((\overline{m}_T + 1)/\delta_-\big)}{\log\big(1/(1 - \delta_+)\big)} \right\rceil \;\le\; \left\lceil \frac{1}{\delta_+} \log \frac{\overline{m}_T + 1}{\delta_-} \right\rceil.$$

Then with probability at least $1 - \delta_-$, simultaneously for every rung $m \in \{1, \ldots, \overline{m}_T\}$ that is introduced, the corresponding batch contains at least one ϵ^U_m-optimal task: there exists i_m in that batch such that $U^\star - U^{(i_m)} \le \epsilon^U_m$.

Proof. Fix a rung $m \in \{1, \ldots, \overline{m}_T\}$ and consider the moment it is introduced. By Algorithm 1, $\mathsf{Gen}(a, m)$ is called only after the scheduler condition holds at the previous rung, so the anchor a satisfies the generation-gating condition required by Condition E.4 at the moment of generation. Applying Lemma G.18 at rung m gives that the failure probability for this batch is at most $(1 - \delta_+)^J$.

By the choice of J, $(1 - \delta_+)^J \le \delta_-/(\overline{m}_T + 1)$. A union bound over the at most $(\overline{m}_T + 1)$ rung-introductions gives total failure probability at most δ_-. □

Lemma G.20 (Oracle-gap bound for the best instantiated value)**.** *Work on the event of Lemma G.19. Define, at each global round t,*

$$U_t^{\max} \;:=\; \max_{i \in \mathcal{I}_t} U^{(i)}.$$

Let m_t denote the current ladder level in Algorithm 1 at round t. Then for every t,

$$U^\star - U_t^{\max} \;\le\; \epsilon^U_{\min\{m_t,\, m^\star\}}.$$

In particular, before reaching level $m^\star$ we have $U^\star - U_t^{\max} \le \epsilon^U_{m_t}$, and after level $m^\star$ is introduced we have $U^\star - U_t^{\max} \le \epsilon^U_{m^\star}$ for all subsequent rounds.

Proof. Fix t and let $m = m_t$. If $m \le m^\star$, then level m has already been introduced by time t. On the event of Lemma G.19, at the moment level m was introduced, its generation batch contained some task i_m with $U^\star - U^{(i_m)} \le \epsilon^U_m$. Since tasks are only added (never removed), we have $i_m \in \mathcal{I}_t$, hence $U_t^{\max} \ge U^{(i_m)} \ge U^\star - \epsilon^U_m$, i.e. $U^\star - U_t^{\max} \le \epsilon^U_m$.

If $m > m^\star$, then level $m^\star$ has also been introduced by time t, so the same argument gives $U^\star - U_t^{\max} \le \epsilon^U_{m^\star}$. Combining the two cases yields the claim with $\epsilon^U_{\min\{m_t, m^\star\}}$. □

Choosing J without knowing δ_+.

Theorem G.21 (Choosing J without knowing δ_+)**.** *Fix any target failure probability $\delta_- \in (0,1)$ and choose a batch size schedule that does not depend on δ_+, for example*

$$J_T \;:=\; \left\lceil \log^2\!\left(\frac{2T}{\delta_-}\right) \right\rceil.$$

Then, whenever a rung m is introduced and the parent used for generation satisfies the precondition of Condition E.4 at that call (e.g., it is ϵ^U_m-solved; Definition 2.2), Lemma G.18 implies the probability that the batch contains an ϵ^U_m-optimal task is at least

$$1 - \exp(-\delta_+ J_T).$$

In particular, at the moment level $m^\star$ is introduced, the probability of generating an $\epsilon^U_{m^\star}$-optimal task is at least $1 - \exp(-\delta_+ J_T)$, and whenever $\delta_+ \ge \log(1/\delta_-)/J_T$, the failure probability is at most δ_-.

Remark G.22 (Effect on regret via N_T). Using a δ_+-free upper bound such as J_T changes only the number of instantiated tasks: $N_T \le J_T(\lfloor \log_2 g_{m_T} \rfloor + 1)$ deterministically. The regret bound of Theorem E.6 continues to hold on the event that an $\epsilon^U_{m^\star}$-optimal task is instantiated, and the probability of this event is controlled by $1 - \exp(-\delta_+ J_T)$.

Corollary G.23 (Regret overhead with δ_+-free batching)**.** *Assume Condition E.4 holds for some unknown $\delta_+ \in (0,1]$, and use*

$$J_T \;:=\; \left\lceil \left(\log \frac{2T}{\delta_-}\right)^2 \right\rceil$$

whenever a new rung is introduced. If $\overline{m}(T)$ is fixed, then deterministically $N_T \le (\overline{m}(T)+1)J_T$. Moreover, on the joint event that (i) the envelope confidence event of Theorem E.3 holds and (ii) every introduced batch up to level $m^\star$ contains an ϵ^U_m-optimal task, Algorithm 1 satisfies

$$\mathcal{R}(T) \;\le\; \tilde{\mathcal{O}}(\log T)\, R_T^\star.$$

The uniform batch-success event in (ii) has probability at least

$$1 - (m^\star + 1)\exp(-\delta_+ J_T),$$

by Lemma G.18 and a union bound over all introduced levels up to $m^\star$.

Proof. Since GSR introduces each level at most once and generates at most J_T tasks per introduced level, we have $N_T \le (\overline{m}(T)+1)J_T$ deterministically. On the envelope event and the oracle-instantiation event, the proof of Theorem E.6 goes through with the same regret-vs.-N_T dependence, yielding

$$\mathcal{R}(T) \;\le\; C\big(1 + \sqrt{N_T}\big)\, R_T^\star \;\le\; C\Big(1 + \sqrt{\overline{m}(T)+1}\,\sqrt{J_T}\Big)\, R_T^\star.$$

Let $a := \log(2T/\delta_-)$ so that $J_T = \lceil a^2 \rceil$. Then $\sqrt{J_T} \le \sqrt{a^2+1} \le a+1$, hence $1 + \sqrt{\overline{m}(T)+1}\sqrt{J_T} = \tilde{\mathcal{O}}(\log(2T/\delta_-)) = \tilde{\mathcal{O}}(\log T)$ for fixed δ_-. □

G.3.5. Decaying generator success across refinement levels: a logit-linear model

In Condition E.4 we lower bounded the success probability of generating an ϵ_m^U-optimal task by a constant δ_+. Empirically (Fig.7(b)), the success probability may decrease as the refinement level m increases. This subsection analyzes a simple parametric model for this decay and its implication on the overhead factor via N_T.

Level-dependent generation condition. We replace Condition E.4 by the following level-dependent version.

Condition G.24 (ϵ_m^U-optimal task generation with level-dependent probability)**.** *Under Condition 2.3, assume there exists a function $\delta_+(\cdot) : (\mathbb{Z}_{\geq 1} \to (0, 1])$ such that for every rung $m \geq 1$, every global time t, and every anchor task a_t satisfying the generation-gating condition $w_t \leq c_g \epsilon_{m-1}^U$ at the moment of generation, a single i.i.d. draw $i \sim \mathrm{Gen}(a_t, m, J{=}1)$ hits an ϵ_m^U-optimal task with probability at least $\delta_+(m)$:*

$$\Pr\Big(i \in \mathcal{I}^\star(\epsilon_m^U) \,\Big|\, w_t \leq c_g \epsilon_{m-1}^U\Big) \;\geq\; \delta_+(m).$$

Logit-linear decay model. Motivated by boundedness and the approximately smooth monotone decay in Fig.7(b), we consider the log-odds to be linear:

Assumption G.25 (Logit-linear decay)**.** There exist $\alpha \in \mathbb{R}$ and $\beta \geq 0$ such that for all $m \geq 1$,

$$\mathrm{logit}(\delta_+(m)) = \alpha - \beta m, \qquad \text{i.e.} \qquad \delta_+(m) = \sigma(\alpha - \beta m),$$

where $\sigma(x) = 1/(1 + e^{-x})$ and $\mathrm{logit}(p) = \log(p/(1 - p))$.

Main result. We now show how to pick per-rung batch sizes to maintain a high-probability generation-success event, and how the resulting N_T scales under Assumption G.25.

Proposition G.26 (Regret overhead under logit-linear $\delta_+(m)$)**.** *Assume the conditions of Theorem E.6, except that Condition E.4 is replaced by Condition G.24. Fix a target failure probability $\delta_- \in (0, 1)$ and, whenever rung m is introduced, draw an i.i.d. batch of size*

$$J_m \;:=\; \left\lceil \frac{\log\big((m_T + 1)/\delta_-\big)}{\log\big(1/(1 - \delta_+(m))\big)} \right\rceil \;\leq\; \left\lceil \frac{1}{\delta_+(m)} \log\Big(\frac{m_T + 1}{\delta_-}\Big) \right\rceil,$$

where the inequality uses $-\log(1 - x) \geq x$ for $x \in (0, 1)$. Let N_T be the total number of instantiated tasks by time T.

Then:

1. ***(Generation success across rungs).*** *With probability at least $1 - \delta_-$, simultaneously for every rung $m \in \{1, \ldots, m_T\}$ that is introduced, the corresponding batch contains at least one ϵ_m^U-optimal task.*

2. ***(Regret bound with variable batch sizes).*** *On the joint event of Theorem E.3's envelope confidence and the generation-success event in (1), the regret bound of Theorem E.6 continues to hold with the (random) N_T induced by $\{J_m\}$:*

$$R_{\mathrm{val}}(T) \;\leq\; C\big(1 + \sqrt{N_T}\big) \frac{L}{L^\star} R_T^\star,$$

for the same oracle scale $R_T^\star = L^\star \sum_{s=1}^T \bar{\epsilon}_s^f$.

3. ***(Scaling under logit-linear decay).*** *Under Assumption G.25 with $\beta > 0$, $\delta_+(m)$ is nonincreasing in m and hence J_m is nondecreasing. Therefore,*

$$N_T \;\leq\; 1 + \sum_{m=0}^{m_T} J_m \;\leq\; 1 + (m_T + 1) J_{m_T} \;=\; \tilde{O}\Big(\frac{m_T}{\delta_+(m_T)}\Big).$$

Moreover, since $\delta_+(m)^{-1} = 1 + e^{\beta m - \alpha}$, we have $\delta_+(m_T)^{-1} = \tilde{O}\big(e^{\beta m_T}\big)$ and thus

$$\sqrt{N_T} \;=\; \tilde{O}\big(e^{\beta m_T/2}\big).$$

If m_T is chosen as in Lemma E.5 (so $m_T = \Theta(\log T)$), then $e^{\beta m_T} = \tilde{O}\big(T^{\beta/(2\ln 2)}\big)$ and hence

$$\sqrt{N_T} \;=\; \tilde{O}\big(T^{\beta/(4\ln 2)}\big), \qquad \text{so} \qquad \frac{R_{\mathrm{val}}(T)}{R_T^\star} \;=\; \tilde{O}\big(T^{\beta/(4\ln 2)}\big) \quad (\text{when } L = \Theta(L^\star)).$$

In particular, if $R_T^\star = \tilde{O}(\sqrt{T})$ (as for standard GP-UCB regimes), then $R_{\mathrm{val}}(T) = \tilde{O}\big(T^{1/2+\beta/(4\ln 2)}\big)$ is sublinear whenever $\beta < 2\ln 2$.

Proof. (1) Fix a rung $m \in \{1, \dots, m_T\}$ and consider the moment it is introduced. Condition G.24 implies that each i.i.d. draw fails to be ϵ_m^U-optimal with probability at most $1 - \delta_+(m)$. Hence the probability that all J_m draws fail is at most

$$(1 - \delta_+(m))^{J_m} \;\le\; \exp\big(-\delta_+(m) J_m\big).$$

By the choice of J_m, we have $\exp(-\delta_+(m)J_m) \le \delta_-/(m_T + 1)$. Applying a union bound over at most m_T rung introductions gives total failure probability at most δ_-.

(2) The proof of Theorem E.6 uses the generator only through the event that each introduced rung m contains a ϵ_m^U-optimal task (cf. Lemmas G.19 and G.20) and through the induced N_T. On the event established in (1), the analogue of Lemma B.11 holds verbatim: for all t, $U^\star - U_t^{\max} \le \epsilon^U_{\min\{m_t, m^\star\}}$. The remainder of the regret decomposition in Appendix I.2 (selection and within-task terms) is unchanged, except that the bound is expressed in terms of the realized N_T. This yields the displayed bound.

(3) Under Assumption G.25 with $\beta > 0$, $\delta_+(m)$ is nonincreasing, so J_m is nondecreasing and $\sum_{m \le m_T} J_m \le (m_T+1)J_{m_T}$. Using $\delta_+(m)^{-1} = 1 + e^{\beta m - \alpha}$ gives the stated $\tilde{O}(\cdot)$ scaling. Finally, Lemma E.5 implies $m_T = \Theta(\log T)$, so $e^{\beta m_T} = \tilde{O}(T^{\beta/(2\ln 2)})$ and the conclusion follows. □

H. Confidence envelopes of long-run values

This appendix provides the construction and proof of Theorem E.3.

H.1. Uniform sub-Gaussian confidence over utility calls

Lemma H.1 (Uniform confidence over BTL-calibrated utility calls)**.** *Fix $\delta_u \in (0,1)$ and let $\delta_\ell := \delta_u/(\pi^2\ell^2)$. Assume utility calls are implemented via the pairwise-comparison + transport construction, with K_ℓ fresh independent pairwise votes at call ℓ. Then with probability at least $1 - \delta_u$, simultaneously for all utility calls $\ell \ge 1$,*

$$u^{(i_\ell)}(\overline{y}_{s_\ell}^{(i_\ell)}) \in \big[\underline{u}_\ell^{(i_\ell)}, \overline{u}_\ell^{(i_\ell)}\big],$$

where $(\underline{u}_\ell^{(i_\ell)}, \overline{u}_\ell^{(i_\ell)})$ is the transported interval computed at call ℓ (e.g., Eq. (8)).

Proof. Let $\mathcal{E}_\ell$ be the event that the Hoeffding CI for p_ℓ holds at call ℓ. By Hoeffding and the choice of δ_ℓ, we have $\Pr(\mathcal{E}_\ell) \ge 1 - \delta_\ell$ and thus

$$\Pr\Big(\bigcap_{\ell \ge 1} \mathcal{E}_\ell\Big) \;\ge\; 1 - \sum_{\ell=1}^{\infty} \delta_\ell \;\ge\; 1 - \delta_u.$$

Work on $\cap_{\ell \ge 1}\mathcal{E}_\ell$. The reference artifact r has a trivial valid interval ($u(r) = 1/2$). Assume inductively that the anchor interval used at call ℓ is valid. Then Lemma G.2 implies the transported interval computed at call ℓ contains the true utility $u^{(i_\ell)}(\overline{y}_{s_\ell}^{(i_\ell)})$. This closes the induction and proves simultaneous validity over all calls. □

H.2. GP-UCB optimization gap: best-so-far envelope

For a task i, define the within-task optimization gap bound

$$\epsilon_s^f := 2\sqrt{\frac{C_\lambda\, \beta_s^{(i)}\, \gamma_s^{(i)}}{s}}, \qquad C_\lambda := \frac{2}{\log(1 + \lambda^{-1})}.$$

Lemma H.2 (Best-so-far optimization gap under GP-UCB)**.** *On the GP-UCB confidence event, for each task i and all $s \geq 1$,*

$$f^{\star(i)} - \max_{1 \leq r \leq s} f^{(i)}(x_r^{(i)}) \leq \epsilon_s^f.$$

Proof. Fix a task i and suppress superscripts. Let $x_1, \dots, x_s$ be the queried points and let $R_s := \sum_{t=1}^{s}(f^\star - f(x_t))$ be the cumulative regret. Then

$$\max_{1 \leq t \leq s} f(x_t) \;\geq\; \frac{1}{s}\sum_{t=1}^{s} f(x_t) = f^\star - \frac{R_s}{s},$$

so $f^\star - \max_{t \leq s} f(x_t) \leq R_s/s$. Apply Theorem F.2 at horizon s to get $R_s \leq 2\sqrt{C_\lambda s \beta_s \gamma_s}$, hence $f^\star - \max_{t \leq s} f(x_t) \leq 2\sqrt{C_\lambda \beta_s \gamma_s / s} = \epsilon_s^f$. □

H.3. Envelope construction

Utility is queried at local times $s \in \mathcal{S}_u \subseteq \mathbb{N}$. The dense choice in the main text corresponds to $\mathcal{S}_u = \mathbb{N}$ (so the most recent checkpoint is always $\nu_s^{(i)} = s$), while the optional sparse schedule uses $\mathcal{S}_u = \{1, 2, 4, \dots\}$ (Appendix H).

For each task i and local time s, let

$$\nu_s^{(i)} := \max\{r \leq s : \; r \in \mathcal{S}_u\}.$$

be the most recent checkpoint at or before s (if no checkpoint has occurred yet, treat $\nu_s^{(i)}$ as undefined).

At a checkpoint $\nu = \nu_s^{(i)}$, the evaluator returns a valid interval $u^{(i)}(\overline{y}_\nu^{(i)}) \in [\underline{u}_\nu^{(i)}, \overline{u}_\nu^{(i)}]$ on the event of Lemma H.1. For our main instantiation (pairwise BTL with moving anchors), $(\underline{u}_\nu^{(i)}, \overline{u}_\nu^{(i)})$ is computed by the transported interval construction.

Propagate them to the current incumbent using monotonicity + Lipschitzness (Assumption 2.1). Since incumbents are nondecreasing ($\overline{y}_s^{(i)} \geq \overline{y}_\nu^{(i)}$), monotonicity implies $u^{(i)}(\overline{y}_s^{(i)}) \geq u^{(i)}(\overline{y}_\nu^{(i)})$, so the lower bound should never decrease between checkpoints:

$$\overline{u}_s^{(i)}(L) := \min\{1, \; \overline{u}_\nu^{(i)} + L\big(\overline{y}_s^{(i)} - \overline{y}_\nu^{(i)}\big)\}, \tag{24}$$

$$\underline{u}_s^{(i)}(L) := \underline{u}_\nu^{(i)}. \tag{25}$$

(If no checkpoint exists yet, set $\overline{u}_s^{(i)}(L) \equiv 1$ and $\underline{u}_s^{(i)}(L) \equiv 0$.)

Now define the value envelopes:

$$\underline{U}_s^{(i)}(L) := \underline{u}_s^{(i)}(L), \tag{26}$$

$$\overline{U}_s^{(i)}(L) := \min\{1, \; \overline{u}_s^{(i)}(L) + L\epsilon_s^f\}, \tag{27}$$

$$w_s^{(i)}(L) := \overline{U}_s^{(i)}(L) - \underline{U}_s^{(i)}(L). \tag{28}$$

Lemma H.3 (Validity of the envelopes)**.** *Work on the joint event of Lemma H.1 and Theorem F.1. Fix a task i and $L \geq L^{(i)}$. Then for all $s \geq 1$,*

$$\underline{U}_s^{(i)}(L) \;\leq\; U^{(i)} \;\leq\; \overline{U}_s^{(i)}(L).$$

Proof. Fix i, s and let $\nu = \nu_s^{(i)}$. On Lemma H.1, we have $u^{(i)}(\overline{y}_\nu^{(i)}) \in [\underline{u}_\nu^{(i)}, \overline{u}_\nu^{(i)}]$. Since incumbents are nondecreasing ($\overline{y}_s^{(i)} \geq \overline{y}_\nu^{(i)}$), monotonicity yields

$$u^{(i)}(\overline{y}_s^{(i)}) \geq u^{(i)}(\overline{y}_\nu^{(i)}) \geq \underline{u}_\nu^{(i)} = \underline{u}_s^{(i)}(L).$$

Moreover, by Lipschitzness,

$$u^{(i)}(\overline{y}_s^{(i)}) \leq u^{(i)}(\overline{y}_\nu^{(i)}) + L\big(\overline{y}_s^{(i)} - \overline{y}_\nu^{(i)}\big) \leq \overline{u}_\nu^{(i)} + L\big(\overline{y}_s^{(i)} - \overline{y}_\nu^{(i)}\big) = \overline{u}_s^{(i)}(L).$$

Thus $\underline{U}_s^{(i)}(L) = \underline{u}_s^{(i)}(L) \leq u^{(i)}(\overline{y}_s^{(i)})$. Also $u^{(i)}(\overline{y}_s^{(i)}) \leq U^{(i)}$ by monotonicity and $\overline{y}_s^{(i)} \leq f^{\star(i)}$, giving the lower bound on $U^{(i)}$.

For the upper bound, Lipschitzness gives

$$U^{(i)} = u^{(i)}(f^{\star(i)}) \le u^{(i)}(\overline{y}_s^{(i)}) + L\big(f^{\star(i)} - \overline{y}_s^{(i)}\big).$$

On Lemma H.2, $f^{\star(i)} - \overline{y}_s^{(i)} \le \epsilon_s^f$, and on Lemma H.1, $u^{(i)}(\overline{y}_s^{(i)}) \le \overline{u}_s^{(i)}(L)$, hence $U^{(i)} \le \overline{u}_s^{(i)}(L) + L\epsilon_s^f = \overline{U}_s^{(i)}(L)$. □

H.4. Proof of Theorem E.3

Proof of Theorem E.3. The envelope validity is exactly Lemma H.3.

For the width bound, fix task i and time s and let $\nu = \nu_s^{(i)}$ be the most recent checkpoint. By construction (using (24)–(25) and $\overline{y}_s^{(i)} \ge \overline{y}_\nu^{(i)}$),

$$w_s^{(i)}(L) \le \big(\overline{u}_\nu^{(i)} - \underline{u}_\nu^{(i)}\big) + L\big(\overline{y}_s^{(i)} - \overline{y}_\nu^{(i)}\big) + L\epsilon_s^f.$$

At the checkpoint, we ensure the utility interval width satisfies $\overline{u}_\nu^{(i)} - \underline{u}_\nu^{(i)} \le c_u\, L\, \epsilon_\nu^f$ (e.g., via Lemma G.6). Moreover, since $\overline{y}_s^{(i)} \le f^{\star(i)}$,

$$\overline{y}_s^{(i)} - \overline{y}_\nu^{(i)} \le f^{\star(i)} - \overline{y}_\nu^{(i)} \le \epsilon_\nu^f$$

by Lemma H.2. Thus

$$w_s^{(i)}(L) \le (c_u + 1)L\epsilon_\nu^f + L\epsilon_s^f.$$

Finally, for the geometric sparse schedule $\mathcal{S}_u = \{1, 2, 4, \dots\}$ we have $\nu \ge s/2$, and for the dense schedule $\mathcal{S}_u = \mathbb{N}$ we have $\nu = s$. In either case, $\nu \ge s/2$, so using $\epsilon_s^f = 2\sqrt{C_\lambda \beta_s^{(i)} \gamma_s^{(i)}/s}$ and monotonicity of $\beta_s^{(i)}\gamma_s^{(i)}$,

$$\epsilon_\nu^f \le 2\sqrt{\frac{C_\lambda \beta_s^{(i)} \gamma_s^{(i)}}{\nu}} \le 2\sqrt{\frac{C_\lambda \beta_s^{(i)} \gamma_s^{(i)}}{s/2}} = \sqrt{2}\,\epsilon_s^f.$$

Therefore

$$w_s^{(i)}(L) \le \Big(\sqrt{2}(c_u + 1) + 1\Big) L\, \epsilon_s^f.$$

This is the claimed bound with $C_v := \sqrt{2}(c_u + 1) + 1$. □

I. Regret bounds

I.1. Auxiliary inequalities

Lemma I.1 (Square-root harmonic sum)**.** *For any integer $n \ge 1$,*

$$\sum_{s=1}^{n} \frac{1}{\sqrt{s}} \;\le\; 2\sqrt{n}.$$

Proof. For $s \ge 2$, since $x \mapsto x^{-1/2}$ is decreasing we have $\int_{s-1}^{s} x^{-1/2}\,dx \ge s^{-1/2}$. Summing from $s = 2$ to n yields

$$\sum_{s=2}^{n} \frac{1}{\sqrt{s}} \;\le\; \int_1^n x^{-1/2}\,dx \;=\; 2(\sqrt{n} - 1).$$

Adding the $s = 1$ term gives $\sum_{s=1}^{n} s^{-1/2} \le 1 + 2(\sqrt{n} - 1) = 2\sqrt{n} - 1 \le 2\sqrt{n}$. □

I.1.1. TASK-UCB CROSS-TASK REGRET BOUND (UCB-STYLE)

Lemma I.2 (Standard UCB step: gap is controlled by the selected task's envelope width)**.** *Work on the envelope confidence event of Theorem E.3, so that for all instantiated tasks j and all local times s,*

$$\underline{U}_s^{(j)}(\overline{L}) \;\le\; U^{(j)} \;\le\; \overline{U}_s^{(j)}(\overline{L}).$$

Fix a global round t and define the best instantiated value at that time

$$U_t^{\max} \;:=\; \max_{j\in\mathcal{I}_t} U^{(j)}.$$

Let i_t be the task selected by task-UCB at round t and let $s_{t-1}^{(i_t)}$ be its local play-count before *selection. Then*

$$U_t^{\max} - U^{(i_t)} \;\le\; w_{s_{t-1}^{(i_t)}}^{(i_t)}(\overline{L}) := \overline{U}_{s_{t-1}^{(i_t)}}^{(i_t)}(\overline{L}) - \underline{U}_{s_{t-1}^{(i_t)}}^{(i_t)}(\overline{L}).$$

Consequently, writing $N_i := s_T^{(i)}$,

$$\sum_{t=1}^{T}\big(U_t^{\max} - U^{(i_t)}\big) \;\le\; \sum_{i\in\mathcal{I}_T}\sum_{s=1}^{N_i} w_{s-1}^{(i)}(\overline{L}).$$

Proof. Fix t. By the task-UCB selection rule,

$$\overline{U}_{s_{t-1}^{(i_t)}}^{(i_t)}(\overline{L}) \;=\; \max_{j\in\mathcal{I}_t} \overline{U}_{s_{t-1}^{(j)}}^{(j)}(\overline{L}).$$

On the confidence event, $\overline{U}_{s_{t-1}^{(j)}}^{(j)}(\overline{L}) \ge U^{(j)}$ for all instantiated $j \in \mathcal{I}_t$, hence the maximum is at least $U_t^{\max}$:

$$\overline{U}_{s_{t-1}^{(i_t)}}^{(i_t)}(\overline{L}) \;\ge\; U_t^{\max}.$$

Also on the same event, $U^{(i_t)} \ge \underline{U}_{s_{t-1}^{(i_t)}}^{(i_t)}(\overline{L})$. Subtracting yields

$$U_t^{\max} - U^{(i_t)} \;\le\; \overline{U}_{s_{t-1}^{(i_t)}}^{(i_t)}(\overline{L}) - \underline{U}_{s_{t-1}^{(i_t)}}^{(i_t)}(\overline{L}) \;=\; w_{s_{t-1}^{(i_t)}}^{(i_t)}(\overline{L}).$$

Summing over $t = 1, \dots, T$ and regrouping by tasks gives the final inequality. □

I.1.2. Generator

I.2. Proof of Theorem E.6

Proof. Fix horizon T. Let $\mathcal{I}_T$ be the set of tasks instantiated by time T, with $N_T := |\mathcal{I}_T|$. For each $i \in \mathcal{I}_T$, define the number of objective evaluations allocated to task i up to time T as

$$N_i \;:=\; s_T^{(i)} \;=\; \sum_{t=1}^{T} \mathbf{1}\{i_t = i\}, \qquad \text{so that} \qquad \sum_{i\in\mathcal{I}_T} N_i = T.$$

For each global round t, define the best instantiated long-run value

$$U_t^{\max} \;:=\; \max_{i\in\mathcal{I}_t} U^{(i)}.$$

Events. Let $\mathcal{E}_{\text{env}}$ denote the confidence event of Theorem E.3; it holds with probability at least $1 - (\delta_f + \delta_u)$ and implies: (i) the GP-UCB optimization gap bound $f^{\star(i)} - \overline{y}_s^{(i)} \le \epsilon_s^f$ for all instantiated tasks and all $s \le T$, and (ii) validity and the width bound for the value envelopes. Let $\mathcal{E}_{\text{gen}}$ denote the batch-success event of Lemma G.19; it holds with probability at least $1 - \delta_-$. Work on $\mathcal{E}_{\text{env}} \cap \mathcal{E}_{\text{gen}}$.

Step 1: regret decomposition. For each round t, add and subtract $U_t^{\max}$ and $U^{(i_t)}$:

$$U^\star - u^{(i_t)}\big(\overline{y}_t^{(i_t)}\big) = \underbrace{(U^\star - U_t^{\max})}_{\text{generation}} + \underbrace{(U_t^{\max} - U^{(i_t)})}_{\text{selection}} + \underbrace{\Big(U^{(i_t)} - u^{(i_t)}(\overline{y}_t^{(i_t)})\Big)}_{\text{within-task}}. \tag{29}$$

Summing over $t \in [T]$ yields

$$\mathcal{R}(T) \;\le\; \sum_{t=1}^{T}(U^\star - U_t^{\max}) + \sum_{t=1}^{T}\big(U_t^{\max} - U^{(i_t)}\big) + \sum_{t=1}^{T}\Big(U^{(i_t)} - u^{(i_t)}(\overline{y}_t^{(i_t)})\Big). \tag{30}$$

Step 2: bound the generation term $\sum_t (U^\star - U_t^{\max})$. On $\mathcal{E}_{\text{gen}}$, Lemma G.20 gives

$$U^\star - U_t^{\max} \;\le\; \epsilon^U_{m_t \wedge m^\star} \qquad \forall t.$$

Let t_m be the first global round at which the current ladder level becomes m (with the convention $t_0 = 1$), and note m_t is nondecreasing. Then

$$\sum_{t=1}^{T}(U^\star - U_t^{\max}) \;\le\; \sum_{m=0}^{m^\star-1}(t_{m+1} - t_m)\,\epsilon^U_m \;+\; (T - t_{m^\star})\,\epsilon^U_{m^\star} \;\le\; \sum_{m=0}^{m^\star-1}(t_{m+1} - t_m)\,\epsilon^U_m \;+\; T\,\epsilon^U_{m^\star}.$$

By Lemma E.5, once level m is active it advances within at most $\tau_m := K_m s_m$ rounds, where $K_m := J(m+1)$ and

$$s_m := \min\Big\{s \in \mathbb{N} :\; C_v \overline{L}\,\overline{\epsilon}^f_s \le c_g \epsilon^U_m\Big\}.$$

Hence $t_{m+1} - t_m \le \tau_m$ for all $m < m^\star$, so

$$\sum_{t=1}^{T}(U^\star - U_t^{\max}) \;\le\; \sum_{m=0}^{m^\star-1}\tau_m\,\epsilon^U_m \;+\; T\,\epsilon^U_{m^\star}. \tag{31}$$

Moreover, using $\overline{\epsilon}^f_s \le 2\sqrt{C_\lambda \beta_T \gamma_T / s}$ and $\epsilon^U_m = \epsilon^U_0 2^{-m}$ gives $s_m \le \lceil A_T 4^m \rceil$ with $A_T = \left(\frac{2C_v\,\overline{L}\sqrt{C_\lambda \beta_T \gamma_T}}{c_g \epsilon^U_0}\right)^2$, as in the proof of Lemma E.5. With $m^\star \le \overline{m}_T$ and $K_m \le N_T$, For the pre-$m^\star$ sum, use Cauchy–Schwarz:

$$\sum_{m=0}^{m^\star-1}\tau_m \epsilon^U_m \le \left(\sum_{m=0}^{m^\star-1}\tau_m\right)^{1/2}\left(\sum_{m=0}^{m^\star-1}\tau_m(\epsilon^U_m)^2\right)^{1/2}.$$

The first factor is at most $\sqrt{T}$ by Lemma E.5. For the second, $\tau_m = K_m s_m$, $s_m \le A_T 4^m + 1$, and ${\epsilon^U_m}^2 = {\epsilon^U_0}^2 4^{-m}$ imply

$$\sum_{m=0}^{m^\star-1}\tau_m(\epsilon^U_m)^2 \le \sum_{m=0}^{m^\star-1} K_m(A_T 4^m + 1){\epsilon^U_0}^2 4^{-m} \le \tilde{\mathcal{O}}\Big(N_T A_T {\epsilon^U_0}^2\Big),$$

because $K_m \le N_T$ and $m^\star \le \overline{m}_T = \tilde{\mathcal{O}}(\log T)$. Therefore

$$\sum_{m=0}^{m^\star-1}\tau_m \epsilon^U_m \le \tilde{\mathcal{O}}\Big(\overline{L}\sqrt{C_\lambda \Psi_T}\sqrt{N_T T}\Big).$$

Step 3: bound the selection term $\sum_t (U_t^{\max} - U^{(i_t)})$. By Lemma I.2,

$$\sum_{t=1}^{T}\big(U_t^{\max} - U^{(i_t)}\big) \;\le\; \sum_{i \in \mathcal{I}_T}\sum_{s=1}^{N_i} w^{(i)}_{s-1}(\overline{L}).$$

Using the envelope width bound from Theorem E.3 (for $s \ge 1$, and absorbing $w^{(i)}_0 \le 1$ into constants), we have $w^{(i)}_{s-1}(\overline{L}) \le C_v\,\overline{L}\,\epsilon^f_{s-1}$, so

$$\sum_{t=1}^{T}\big(U_t^{\max} - U^{(i_t)}\big) \;\le\; \tilde{\mathcal{O}}\Big(\overline{L}\sum_{i \in \mathcal{I}_T}\sum_{s=1}^{N_i}\epsilon^f_s\Big).$$

Since $\epsilon^f_s \le 2\sqrt{C_\lambda \beta_T \gamma_T / s}$ for all $s \le T$ and $i \in \mathcal{I}_T$, Lemma I.1 yields $\sum_{s=1}^{N_i}\epsilon^f_s \le 4\sqrt{C_\lambda \beta_T \gamma_T}\,\sqrt{N_i}$, hence

$$\sum_{t=1}^{T}\big(U_t^{\max} - U^{(i_t)}\big) \;\le\; \tilde{\mathcal{O}}\Big(\overline{L}\sqrt{C_\lambda \beta_T \gamma_T}\,\sqrt{N_T T}\Big). \tag{32}$$

Step 4: bound the within-task term. For any task i and local time s, Lipschitzness and monotonicity of $u^{(i)}$ give

$$U^{(i)} - u^{(i)}(\overline{y}_s^{(i)}) \le L^\star\big(f^{\star(i)} - \overline{y}_s^{(i)}\big) \le L^\star\,\epsilon_s^f.$$

Therefore

$$\sum_{t=1}^{T}\Big(U^{(i_t)} - u^{(i_t)}(\overline{y}_t^{(i_t)})\Big) = \sum_{i\in\mathcal{I}_T}\sum_{s=1}^{N_i}\Big(U^{(i)} - u^{(i)}(\overline{y}_s^{(i)})\Big) \le L^\star \sum_{i\in\mathcal{I}_T}\sum_{s=1}^{N_i}\epsilon_s^f.$$

Using the same $\epsilon_s^f \le 2\sqrt{C_\lambda\beta_T\gamma_T/s}$ and Lemma I.1 argument,

$$\sum_{t=1}^{T}\Big(U^{(i_t)} - u^{(i_t)}(\overline{y}_t^{(i_t)})\Big) \;\le\; \tilde{\mathcal{O}}\Big(L^\star\sqrt{C_\lambda\beta_T\gamma_T}\,\sqrt{N_T T}\Big). \tag{33}$$

Step 5: combine. Plugging (31), (32), and (33) into (30) yields

$$\mathcal{R}(T) \;\le\; T\,\epsilon_{m^\star}^U + \tilde{\mathcal{O}}\Big((\overline{L} + L^\star)\sqrt{C_\lambda\beta_T\gamma_T}\,\sqrt{N_T T}\Big).$$

Since $\overline{L} \ge L^\star$, this simplifies to

$$\mathcal{R}(T) \;\le\; T\,\epsilon_{m^\star}^U + \tilde{\mathcal{O}}\Big(\overline{L}\sqrt{C_\lambda\beta_T\gamma_T}\,\sqrt{N_T T}\Big). \tag{34}$$

Step 6: relate to $R_T^\star$. Recall $R_T^\star := L^\star\sum_{s=1}^T \overline{\epsilon}_s^f$ and $\overline{\epsilon}_s^f$ is nonincreasing, so $\sum_{s=1}^T \overline{\epsilon}_s^f \ge T\overline{\epsilon}_T^f$. Condition 2.3 implies

$$T\epsilon_{m^\star}^U \le c_\epsilon\Lambda_T\,L^\star T\overline{\epsilon}_T^f \le c_\epsilon\Lambda_T\,R_T^\star.$$

Moreover, with $\Psi_T := \max_{i\in\mathcal{I}_T}\beta_T^{(i)}\gamma_T^{(i)}$,

$$\overline{\epsilon}_T^f = 2\sqrt{C_\lambda\Psi_T/T}, \qquad R_T^\star \ge L^\star T\overline{\epsilon}_T^f = 2L^\star\sqrt{C_\lambda\Psi_T T}.$$

Hence

$$\overline{L}\sqrt{C_\lambda\Psi_T}\sqrt{N_T T} \le \frac{\overline{L}}{2L^\star}\sqrt{N_T}\,R_T^\star.$$

Applying these displays to (34) gives

$$\mathcal{R}(T) \le C\left[\Lambda_T + \Big(1+\sqrt{N_T}\Big)\frac{\overline{L}}{L^\star}\right]R_T^\star.$$

□

I.3. Oracle scale comparison

Lemma I.3 (Bounding κ_T for RBF and Matérn kernels)**.** *Fix a horizon T and let $\mathcal{I}_T$ denote the set of instantiated tasks by time T. For each task $i \in \mathcal{I}_T$, let $d^{(i)}$ be its input dimension, and let $\epsilon_s^f = 2\sqrt{C_\lambda\beta_s^{(i)}\gamma_s^{(i)}/s}$ be the within-task gap bound after s evaluations, with $\gamma_s^{(i)}$ the maximum information gain and $\beta_s^{(i)}$ the corresponding confidence parameter (e.g., IGP-UCB). Define $\overline{\epsilon}_s^f := \max_{i\in\mathcal{I}_T}\epsilon_s^f$ and*

$$\kappa_T := \frac{\sum_{s=1}^T \overline{\epsilon}_s^f}{\sum_{s=1}^T \epsilon_s^{f,(i^\star_{\epsilon_U})}}.$$

(a) (Squared-exponential / RBF) If each task uses a squared-exponential kernel, then

$$\kappa_T = \tilde{\mathcal{O}}\big((\log T)^{d_{\max}-d_\star}\big), \quad d_{\max} := \max_{i\in\mathcal{I}_T} d^{(i)}, \;\; d_\star := d^{(i^\star_{\epsilon_U})}.$$

(b) (Matérn-ν) If each task uses a Matérn kernel with smoothness $\nu_i > 0$, then

$$\kappa_T = \tilde{\mathcal{O}}\big(T^{\alpha_{\max}-\alpha_\star}\big), \quad \alpha_{\max} := \max_{i\in\mathcal{I}_T}\alpha(d^{(i)},\nu_i), \;\; \alpha_\star := \alpha(d_\star,\nu_\star),$$

where $\alpha(d,\nu) := \frac{d(d+1)}{2\nu+d(d+1)}$ and $\nu_\star := \nu_{i^\star_{\epsilon_U}}$.

Proof. We use the standard IGP-UCB choice of confidence parameter, for which (up to notation) $\beta_s^{(i)}$ scales as a constant plus $\gamma_{s-1}^{(i)}$ and $\log(1/\delta_i)$. Therefore $\beta_s^{(i)} = \tilde{\mathcal{O}}(\gamma_s^{(i)})$ under the normalization assumptions, and hence

$$\epsilon_s^f = 2\sqrt{\frac{C_\lambda\, \beta_s^{(i)}\, \gamma_s^{(i)}}{s}} = \tilde{\mathcal{O}}\left(\sqrt{\frac{(\gamma_s^{(i)})^2}{s}}\right) = \tilde{\mathcal{O}}\left(\frac{\gamma_s^{(i)}}{\sqrt{s}}\right).$$

(a) RBF. For a squared-exponential kernel on a compact subset of $\mathbb{R}^{d^{(i)}}$ we have $\gamma_s^{(i)} = \mathcal{O}((\log s)^{d^{(i)}+1})$. Thus $\epsilon_s^f = \tilde{\mathcal{O}}((\log s)^{d^{(i)}+1}/\sqrt{s})$ and hence $\overline{\epsilon}_s^f = \tilde{\mathcal{O}}((\log s)^{d_{\max}+1}/\sqrt{s})$. Summing and comparing to the same bound for $i^\star_{\epsilon^U}$ yields

$$\kappa_T \;=\; \frac{\tilde{\mathcal{O}}\Big(\sum_{s=1}^T (\log s)^{d_{\max}+1} s^{-1/2}\Big)}{\tilde{\mathcal{O}}\Big(\sum_{s=1}^T (\log s)^{d_\star+1} s^{-1/2}\Big)} \;=\; \tilde{\mathcal{O}}\big((\log T)^{d_{\max}-d_\star}\big),$$

where we used $\sum_{s=1}^T s^{-1/2}(\log s)^p = \Theta(\sqrt{T}(\log T)^p)$.

(b) Matérn. For a Matérn-ν_i kernel on a compact subset of $\mathbb{R}^{d^{(i)}}$ we have $\gamma_s^{(i)} = \mathcal{O}(s^{\alpha(d^{(i)},\nu_i)} \log s)$ with $\alpha(d,\nu) = \frac{d(d+1)}{2\nu+d(d+1)}$. Hence $\epsilon_s^f = \tilde{\mathcal{O}}(s^{\alpha(d^{(i)},\nu_i)-1/2})$ and $\overline{\epsilon}_s^f = \tilde{\mathcal{O}}(s^{\alpha_{\max}-1/2})$. Summing gives $\sum_{s=1}^T s^{\alpha-1/2} = \Theta(T^{\alpha+1/2})$ for $\alpha > -1/2$, and therefore

$$\kappa_T = \frac{\tilde{\mathcal{O}}(T^{\alpha_{\max}+1/2})}{\tilde{\mathcal{O}}(T^{\alpha_\star+1/2})} = \tilde{\mathcal{O}}\big(T^{\alpha_{\max}-\alpha_\star}\big).$$

This proves both claims. □

I.4. Sparse vote case

Corollary I.4 (Explicit β_T, γ_T form)**.** *Under the conditions of Theorem E.6, recall* $\beta_T := \max_{i\in\mathcal{I}_T} \beta_T^{(i)}$ *and* $\gamma_T := \max_{i\in\mathcal{I}_T} \gamma_T^{(i)}$*. Then for all* $s \le T$*,*

$$\overline{\epsilon}_s^f = \max_{i\in\mathcal{I}_T} 2\sqrt{C_\lambda\, \beta_s^{(i)}\gamma_s^{(i)}/s} \;\le\; 2\sqrt{C_\lambda\, \beta_T\, \gamma_T/s}.$$

Consequently,

$$R_T^\star = L^\star \sum_{s=1}^T \overline{\epsilon}_s^f \;\le\; 4\, L^\star \sqrt{C_\lambda\, \beta_T\, \gamma_T\, T},$$

and the regret bound (5) implies

$$\mathcal{R}(T) \;\le\; 4C\left(1+\sqrt{N_T}\right) \overline{L}\, \sqrt{C_\lambda\, \beta_T\, \gamma_T\, T}. \tag{35}$$

If K_s is controlled so that $\overline{u}_\nu^{(i)} - \underline{u}_\nu^{(i)} \le c_u\, \overline{L}\, \epsilon_\nu^f$ at every checkpoint $\nu \in \mathcal{S}_u$, then for all s,

$$w_s^{(i)}(\overline{L}) \le C_v\, \overline{L}\, \epsilon_s^f,$$

where $C_v = c_u + 1$ for the dense schedule $\mathcal{S}_u = \mathbb{N}$, and $C_v := \sqrt{2}(c_u+1)+1$ for $\mathcal{S}_u = \{1,2,4,\dots\}$ (or any valid constant from Appendix H).

J. Hyperparameters

J.1. Hyperparameter Adaptation by Balance-Eliminate

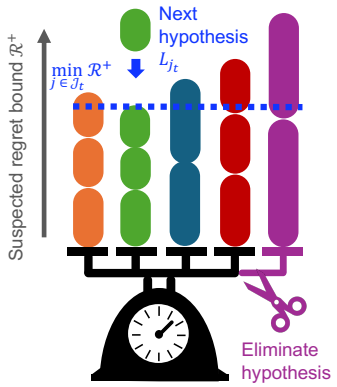

Figure 11. Alg. 2: We hold multiple candidate Lipschitz bounds L_j as hypotheses. Balance decides which hypothesis to test by balancing suspected regret bound amongst hypotheses, and eliminates any hypothesis whose bound is falsified by observation.

Algorithm 2 Evidence Balancing–Hypothesis Elimination

Input: base L_0, growth function $g_L(\cdot)$, failure budget δ_{be}.
Hypotheses: $L_j := q_L(j) = 2^j L_0$.
Init: $j_{\max} \leftarrow 0$; $\mathcal{A}_1 \leftarrow \{0\}$; and $\forall j \geq 0, S_0^j \leftarrow \emptyset$.
for $t \in [T]$
  Choose $j_t \in \arg\min_{j \in \mathcal{A}_t} \mathcal{R}^+(n_{t-1}^j + 1, L_j)$.
  $(i_t, \overline{y}_t, \tilde{u}_t) \leftarrow$ RunOneStepAlg.1($\overline{L} = L_{j_t}$, state S^{j_t}).
  For j_t, $S_t^{j_t} \leftarrow S_{t-1}^{j_t} \cup \{t\}$ and $\forall j \neq j_t$, $S_t^j \leftarrow S_{t-1}^j$.
  $\mathcal{A}_{t+1} \leftarrow \mathcal{A}_t$.
  if $\min_{j \in \mathcal{A}_t} n_t^j > 0$
    Set M_t, χ_t, $\underline{U}_t(j)$, and $\overline{U}_t(j)$ for all $j \in \mathcal{A}_t$.
    $\mathcal{A}_{t+1} \leftarrow \left\{ j \in \mathcal{A}_t : \text{Condition (38)} \right\}$.
  end if
  while $L_{j_{\max}+1} \leq L_0 g_L(t)$
    $j_{\max} \leftarrow j_{\max} + 1$; $\mathcal{A}_{t+1} \leftarrow \mathcal{A}_{t+1} \cup \{j_{\max}\}$.
  end while
end for

Theorem E.6 assumes we can supply a valid Lipschitz bound $\overline{L} \geq L^\star$. In practice, however, $L^\star$ is unknown and tightly coupled to the (unknown) ϵ^U-optimal task. We therefore perform *hyperparameter adaptation* (not point estimation) via a regret balancing wrapper (Alg. 2, Fig.11): we maintain multiple candidate bounds L_j (hypotheses), balance evidence across them using regret certificates, and eliminate candidates that are *falsified* by observed utility $\tilde{u}_s$.

State used in the balancing wrapper. For the theoretical guarantee, each hypothesis j maintains its own state S^j of Algorithm 1, including its task set, envelopes, generation level, and GP states. At round t, only the selected hypothesis j_t is advanced by one physical evaluation. This independent-state interpretation prevents an invalid hypothesis from triggering task-generation events that contaminate the state of a valid hypothesis. A shared-state implementation is a practical heuristic and should be reported separately from the formal guarantee.

Hypotheses. Fix $L_0 > 0$ and a growth function $g_L(T) \geq 1$ such that $L^\star \leq L_0 g_L(T)$. We consider the hypotheses generator $L_j := q_L(j) = 2^j L_0$ and maintain an active set $\mathcal{A}_t$ of hypotheses. For each j, let

$$S_t^j := \{\tau \leq t : j_\tau = j\}, \qquad n_t^j := |S_t^j|$$

be the set on which hypothesis j was selected so far.

Evidence Balancer. When a hypothesis is *valid* ($L_j \geq L^\star$), Theorem E.6 implies that running Alg. 1 for n rounds with $\overline{L} = L_j$ incurs value-gap regret at most (Theorem E.6)

$$\mathcal{R}^+(n, L_j) := C\,(1 + \sqrt{N_T})\, L_j \sum_{s=1}^{n} \overline{\epsilon}_s^f, \tag{36}$$

For online computability, the certificate used by the master is evaluated with deterministic upper bounds available at round t:

$$N_t^+ := 1 + J(\overline{m}_t + 1), \qquad \epsilon_s^{f^+}(t) := 2\sqrt{C_\lambda \Psi_t^+ / s},$$

where Ψ_t^+ is the current monotone upper bound on $\max_{i \in \mathcal{I}_t} \beta_s^{(i)} \gamma_s^{(i)}$ for $s \leq t$. Thus the causal certificate is

$$\mathcal{R}_t^+(n, L_j) := C(1 + \sqrt{N_t^+}) L_j \sum_{s=1}^{n} \epsilon^{f^+}_s(t).$$

In practice, constants common across hypotheses are dropped in the argmin, but the elimination test uses the same causal certificate.

Balancer selects which hypothesis to test next by Line 2:

$$j_t \in \arg\min_{j \in \mathcal{A}_t} \mathcal{R}^+(n_{t-1}^j + 1, L_j).$$

In practice, constant factors are irrelevant in minimization; we set $\mathcal{R}^+(n, L_j) = L_j \sum_{s=1}^{n} \overline{\epsilon}_s^f$. Intuitively, a hypothesis that has been played little has a small regret, so it gets selected until its regret "catches up" to the others. This prevents over-committing to any single hypothesis before the alternatives have accrued comparable evidence.

Hypotheses Elimination. For each hypothesis j with $n_t^j > 0$, define the empirical mean utility $\widehat{U}_t(j) := \frac{1}{n_t^j} \sum_{\tau \in S_t^j} \tilde{u}_\tau$. Similarly to Lemma E.1, we form CIs

$$\underline{U}_t(j) := \widehat{U}_t(j) - \sqrt{\chi_t / n_t^j}, \quad \overline{U}_t(j) := \widehat{U}_t(j) + \sqrt{\chi_t / n_t^j}, \tag{37}$$

with $\chi_t := 2\sigma_u^2 \log\!\left(\frac{4\pi^2 t^2 M_t}{\delta_{\mathrm{be}}}\right)$, and $M_t := 1 + \max \mathcal{A}_t$[6]. We eliminate hypothesis j if it does not satisfy:

$$\overline{U}_t(j) + \mathcal{R}^+(n_t^j, L_j)/n_t^j \;\geq\; \max_{k \in \mathcal{A}_t} \underline{U}_t(k). \tag{38}$$

Intuitively, this rule compares the most optimistic bound on hypothesis j (LHS) with the best pessimistic bound among all hypotheses (RHS). If even the weakest consistency condition—the upper bound exceeds the lower bound—is violated, then hypothesis j cannot be valid and is eliminated. Valid hypotheses are never removed (Lemma J.2).

Lemma J.1 (Uniform concentration for per-rung utility averages)**.** *Fix $\delta_{\mathrm{be}} \in (0, 1)$ and define χ_t as above. Then with probability at least $1 - \delta_{\mathrm{be}}$, simultaneously for all $t \leq T$ and all rungs j with $n_t^j > 0$,*

$$\left| \widehat{U}_t(j) - \frac{1}{n_t^j} \sum_{\tau \in S_t^j} u_\tau \right| \;\leq\; \sqrt{\frac{\chi_t}{n_t^j}}.$$

Proof. For fixed (t, j), $\sum_{\tau \in S_t^j} \xi_\tau$ is sub-Gaussian with variance proxy $n_t^j \sigma_u^2$. Thus,

$$\Pr\left(\left| \sum_{\tau \in S_t^j} \xi_\tau \right| > \sqrt{n_t^j \chi_t} \right) \leq 2 \exp\!\left(-\frac{\chi_t}{2\sigma_u^2} \right) = \frac{\delta_{\mathrm{be}}}{2\pi^2 t^2 M_t}.$$

Union bound over $t \in [T]$ and $j \in [M_T]$ and use $\sum_{t=1}^{\infty} t^{-2} = \pi^2/6$. □

[6] $\max \mathcal{A}_t$ denotes the largest *index* in the active rung set (not its cardinality $|\mathcal{A}_t|$).

Lemma J.2 (No false elimination of valid Lipschitz rungs)**.** *Work on the intersection of Lemma J.1 and the confidence event of Theorem E.6. Then Algorithm 2 never eliminates any rung j with $L_j \geq L^\star$. In particular, the smallest valid rung $j^\star$ remains in $\mathcal{A}_t$ for all t.*

Proof. Fix t and a valid rung j with $n_t^j > 0$. Let $\hat{u}_t(j) := \frac{1}{n_t^j}\sum_{\tau\in S_t^j} u_\tau$ denote the average noise-free utility on plays of rung j. Since $u_\tau \leq U^\star$ for all τ, we have $\hat{u}_t(j) \leq U^\star$, hence $\underline{U}_t(j) \leq \hat{u}_t(j) \leq U^\star$ on Lemma J.1, and therefore $\max_{k\in\mathcal{A}_t} \underline{\hat{u}}_t(k) \leq U^\star$.

For a valid rung j, Theorem E.6 implies (up to the constant absorbed in $\mathcal{R}^+$) that

$$\sum_{\tau\in S_t^j} \big(U^\star - u_\tau\big) \;\leq\; \mathcal{R}^+(n_t^j, L_j).$$

Rearranging gives

$$U^\star \;\leq\; \hat{u}_t(j) + \frac{\mathcal{R}^+(n_t^j, L_j)}{n_t^j}.$$

On Lemma J.1, $\hat{u}_t(j) \leq \overline{U}_t(j)$, hence

$$U^\star \;\leq\; \overline{U}_t(j) + \frac{\mathcal{R}^+(n_t^j, L_j)}{n_t^j}.$$

Combining the two displays yields

$$\overline{U}_t(j) + \frac{\mathcal{R}^+(n_t^j, L_j)}{n_t^j} \;\geq\; U^\star \;\geq\; \max_{k\in\mathcal{A}_t} \underline{U}_t(k),$$

so the falsification condition (38) cannot hold for any valid rung. □

Theorem J.3 (Lipschitz-balancing overhead)**.** *Assume the conditions of Theorem E.6 and $L^\star \leq L_0 g_L(T)$. Let $M_T := 1 + \lceil \log_2 g_L(T) \rceil$. Then, with probability at least $1 - (\delta_f + \delta_u + \delta_- + \delta_{\mathrm{be}})$,*

$$\mathcal{R}_{\textit{val, GSR}}(T) \;\leq\; \tilde{\mathcal{O}}\big(M_T\big)\, \mathcal{R}_{\textit{val, GSR}^{(j^\star)}}(T) \;\leq\; \tilde{\mathcal{O}}\big(\log g_L(T)\big) \left(1 + \sqrt{N_T}\right) R_T^\star,$$

where $\mathrm{GSR}^{(j^\star)}$ denotes Alg. 1 run with $\overline{L} = L_{j^\star}$ and $R_T^\star = L^\star \sum_{s=1}^T \overline{\epsilon}_s^f$ is the single-task oracle scale from Theorem E.6.

Proof sketch. Algorithm 2 is an instance of regret-bound balancing with elimination (Ziomek et al., 2024) applied to the ladder $\{L_j\}_{j=0}^{M_T-1}$, using the candidate bounds $\mathcal{R}^+(\cdot, L_j)$. Lemma J.2 ensures that a well-specified rung (in particular $j^\star$) is never eliminated. Standard RBLE analysis then yields a master regret bounded by $\tilde{\mathcal{O}}(M_T)$ times the regret of the best well-specified rung at horizon T. Finally, $L_{j^\star} \leq 2L^\star$ and Theorem E.6 imply $\mathcal{R}_{\text{val, GSR}^{(j^\star)}}(T) = \tilde{\mathcal{O}}((1+\sqrt{N_T})R_T^\star)$, giving the claim. □

J.2. Other hyperparameters

Other hyperparameters can also be estimated from regret balancing wrapper.

Theorem J.4 (Overhead factors under unknown hyperparameters)**.** *Under Theorem E.6 with the parameter adaptation with Algorithm 2, we obtain:*

Unknown(s)	$R(T)/R_T^\star$
Lipschitz bound $\overline{L}$	$\tilde{\mathcal{O}}(\log g_L(T))$
Regret scale C	$\tilde{\mathcal{O}}(\log g_C(T))$
GP hyperparameters	$\tilde{\mathcal{O}}(\log g_k(T))$

where $g(T)$ are geometric generators for each parameters.

J.2.1. UNKNOWN WITHIN-TASK GP HYPERPARAMETERS: REGRET BALANCING OVER A KERNEL LADDER

This is straightforward application of (Ziomek et al., 2024).

J.2.2. UNKNOWN REGRET SCALE

The elimination rule uses the regret certificate $\mathcal{R}^+(\cdot, L)$, which hides an absolute multiplicative constant. In the main text we denote this constant by C and treat it as a user-specified slack. Here we show that C can itself be adapted by the same regret-balancing + falsification mechanism.

Certificate-scale ladder. Assume there exists an (unknown) constant $C^\star < \infty$ such that, whenever $\overline{L} \geq L^\star$, Algorithm 1 satisfies the value-gap regret bound

$$\sum_{t=1}^{n}\Big(U^\star - u^{(i_t)}(\overline{y}_t^{(i_t)})\Big) \;\leq\; C^\star\,(1+\sqrt{N_T})\,\overline{L}\sum_{s=1}^{n}\overline{\epsilon}_s^f, \qquad \forall n \leq T,$$

on the same confidence event as Theorem E.6. We introduce a geometric ladder

$$c_r := 2^r c_0, \qquad r \in \{0, 1, \ldots, R_C\}, \qquad R_C := \lceil \log_2 g_C(T) \rceil\,,$$

where $c_0 > 0$ and $g_C(T) \geq 1$ satisfy $C^\star \leq c_0 g_C(T)$.

Two-dimensional hypotheses. To adapt both the unknown Lipschitz bound and the unknown certificate scale, treat each hypothesis as a pair $h = (j, r)$ with (L_j, c_r). Define the hypothesis-specific certificate

$$\mathcal{R}^+(n; h) \;:=\; c_r\,(1+\sqrt{N_T})\,L_j\sum_{s=1}^{n}\overline{\epsilon}_s^f.$$

Run the same balance–eliminate wrapper as Algorithm 2 over hypotheses h (where selecting $h = (j, r)$ executes Algorithm 1 with $\overline{L} = L_j$ and uses $\mathcal{R}^+(\cdot; h)$ in the master).

Lemma J.5 (No false elimination of well-specified (L_j, c_r))**.** *Let $h = (j, r)$ be* well-specified*, i.e. $L_j \geq L^\star$ and $c_r \geq C^\star$. Work on the intersection of: (i) the envelope confidence event required by Theorem E.6, and (ii) the uniform concentration event for per-hypothesis utility averages (analogous to Lemma J.1). Then the balance–eliminate master never eliminates hypothesis h.*

Proof. The proof is identical to Lemma J.2, replacing $\mathcal{R}^+(\cdot, L_j)$ by $\mathcal{R}^+(\cdot; h)$. Well-specified hypotheses satisfy the regret guarantee with the corresponding certificate, so their certificate-relaxed upper bound always dominates the best lower bound and cannot be falsified. □

Theorem J.6 (Overhead from adapting the certificate scale)**.** *Assume $L^\star \leq L_0 g_L(T)$ and $C^\star \leq c_0 g_C(T)$ and the conditions of Theorem E.6. Let $M_L := 1 + \lceil \log_2 g_L(T) \rceil$ and $M_C := 1 + \lceil \log_2 g_C(T) \rceil$. Running the balance–eliminate master on the combined ladder incurs an overhead $\tilde{\mathcal{O}}(M_L M_C)$ relative to a single well-specified hypothesis. Equivalently, relative to the Lipschitz-only ladder of Theorem J.3, adapting the certificate scale adds a multiplicative factor $\tilde{\mathcal{O}}(M_C) = \tilde{\mathcal{O}}(\log g_C(T))$.*

Proof sketch. There are at most $M_L M_C$ hypotheses (j, r). By Lemma J.5, at least one well-specified hypothesis survives forever. Applying the standard regret-balancing + elimination master guarantee (the same black-box result used for the Lipschitz ladder) bounds the master regret by a factor $\tilde{\mathcal{O}}(\sqrt{M_L M_C})$ times the regret of the best well-specified hypothesis. Relative to the Lipschitz-only ladder (which contributes $\tilde{\mathcal{O}}(\sqrt{M_L})$), this introduces an extra factor $\tilde{\mathcal{O}}(\sqrt{M_C})$. □

K. Experimental Setting

We performed experiments on MacBook Pro 2019, 2.4 GHz 8-Core Intel Core i9, 64 GB 2667 MHz DDR4.

K.1. LLM interfaces: task mutation generator and evaluator

This subsection specifies the **exact LLM interfaces** used in our experiments for (i) task mutation generation $\mathsf{Gen}(\cdot)$ and (ii) cross-task evaluation (pairwise "quick tasting"). We implement both via chat-completions in **strict JSON mode** to ensure parseable outputs.

LLM API and decoding. We use the OpenAI chat-completions interface with:

- **Model:** `gpt-4o-mini`.
- **Decoding:** `temperature=0.0` (deterministic given a fixed seed), `max_tokens=800`.
- **Structured output:** `response_format={type: "json_object"}` (JSON-only responses).
- **Robustness:** if parsing fails or the JSON schema is violated, we retry up to `MAX_RETRIES=2` times; if still invalid, we fall back to a conservative default (e.g., random vote).

In the committee evaluator, we also randomize the A/B ordering per vote and map the returned winner back to the original candidate/anchor to reduce positional bias.

Task specification as JSON. Across all domains, each task i is represented as a structured JSON object with a fixed schema. The schema is *domain-dependent* (e.g., wine planning vs. acquisition-function analysis), but the LLM interface is shared: the generator outputs a list of valid task-spec JSONs; the evaluator compares two task-spec JSONs plus their current incumbents / confidence summaries.

K.1.1. Task mutation generator: actor–critic prompting

We implement $\mathsf{Gen}(a, m, J)$ as a two-stage LLM pipeline: **Actor** proposes J mutated children from an anchor task a (at rung/level m), and **Critic** reviews and repairs the list to satisfy schema constraints and remove duplicates.

Generator I/O format and verification pipeline. We implement task mutation as an Actor–Critic generator with an additional deterministic verifier. The Actor proposes candidate child task specifications by editing an anchor task JSON under a rung-dependent edit budget ρ_m. A Critic then audits and repairs the proposals (schema, edit-budget adherence, feasibility, and duplicates), after which a rule-based verifier enforces hard constraints before admission. This layered design is intended to prevent pathological cases where a small number of edits (small ρ_m) causes a large semantic shift.

Input to Actor/Critic:

- the anchor task specification (JSON), injected into the prompt as `{{TASK_SPEC}}`,
- the current rung/level m and target mutation ratio ρ_m (or an equivalent edit budget), injected as `{{LEVEL_M}}` and `{{RHO_M}}`,
- the full mutation/evaluation history up to the current generation call—i.e., all previously generated tasks and, for each past global round τ, the tuple $(i_\tau, x_\tau, y_\tau, \tilde{u}_\tau)$— injected as `{{MUTATION_HISTORY_JSON}}`,
- the requested batch size J,
- domain constraints (allowed categorical choices; bounds format; dimensionality consistency), plus any rung-dependent safeguards (e.g., tighter bound-change allowances as ρ_m shrinks).

Output: a JSON object with key `"task_list"` containing exactly J mutated child task specifications that share the same top-level schema as the anchor.

Mutation history JSON (conditioning signal). We provide the generator a machine-readable log that contains prior information needed for evidence-based mutations and for duplicate avoidance. Concretely, `{{MUTATION_HISTORY_JSON}}` is a JSON object that includes: (i) a per-round evaluation trace (at each global round t: selected task, evaluated design, observed outcome, and utility), and (ii) a registry of all generated tasks (including parent/anchor lineage and best-so-far summaries). A minimal schema is:

```
{
  "t_now": int,
  "eval_trace": [
    {
      "t": int,
      "task_spec": { ... },      // the task evaluated at round t
      "x": [ ... ],              // design used at round t (local-to-task design)
      "y": float,                // observed objective
      "u": float                 // utility (or proxy) at round t
    }
    // one entry per past round
  ],
  "task_registry": [
    {
      "task_spec": { ... },
      "parent_spec": { ... } or null,
      "m": int,                  // rung/level at which this task was generated
      "best_x": [ ... ] or null, // best-so-far design for this task
      "best_y": float or null,
      "best_u": float or null
    }
    // one entry per instantiated/generated task
  ]
}
```

We pass the full history when it fits; otherwise we pass a compact summary that preserves the information needed for conditioning and filtering (e.g., lineage and per-task best-so-far statistics), so the generator can still ground mutations in prior evidence.

Exact prompt templates (generator). Tables 3, 4 are the templates used in code. The placeholders `{{TASK_SPEC}}`, `{{LEVEL_M}}`, `{{RHO_M}}`, `{{MUTATION_HISTORY_JSON}}`, and `{J}` are filled programmatically. In addition to enforcing ρ_m, the prompts explicitly instruct the models to avoid near-duplicates and to use the history to steer mutations away from previously low-utility edit patterns.

Table 3. Generator prompts (Actor)

Actor: system	You are an assistant that proposes new BO tasks (problem settings) by mutating an existing task JSON. You will be given the full mutation/evaluation history (generated tasks and past evaluations with x, y, and u). Use this history as in-context evidence: learn which mutations improved utility and avoid repeating failures. Prefer *local*, budget-consistent edits when `rho_m` is small; avoid semantic drift from the anchor. Follow the schema strictly and output only JSON.

Actor: user	You are given:
	(1) Current rung level and target mutation ratio: `m={{LEVEL_M}}`, `rho_m={{RHO_M}}`.
	(2) Anchor task specification as JSON:
	
	{{TASK_SPEC}}
	
	(3) Mutation/evaluation history as JSON (all past rounds + all generated tasks):
	
	{{MUTATION_HISTORY_JSON}}
	
	Generate {J} mutated child task specifications.
	Requirements:
	- Keep the same top-level schema keys (do not add/remove keys).
	- Respect the target mutation ratio `rho_m`: change approximately a `rho_m` fraction of editable fields (about `round(rho_m * #editable_fields)` fields); keep the rest identical to the anchor.
	- Use the history: prefer edits correlated with higher observed utility `u`; avoid repeating edit patterns that correlate with lower `u` or clearly worse outcomes.
	- Do not introduce large semantic shifts for small `rho_m` (e.g., drastic scale changes); keep mutations local.
	- Do not duplicate the anchor, any previously instantiated task in `task_registry`, or each other (treat near-duplicates as duplicates).
	- Ensure bounds are valid, consistent with dim, and not degenerate.
	- Output ONLY JSON with the schema:
	{
	"task_list": [
	{
	"acquisition": "...",
	"objective": "...",
	"dim": int,
	"bounds": [[float, float], ...],
	"noise_sigma": float,
	"notes": "..."
	},
	...
	]
	}

Table 4. Generator prompts (critic)

Critic: system	Review the proposed tasks using the anchor, rung (m, rho_m), and the full mutation history. Fix schema violations, enforce the mutation-ratio constraint, remove duplicates (including with history), and ensure feasibility. When `rho_m` is small, repair proposals that cause disproportionate semantic drift from the anchor. Return exactly the same JSON schema with "task_list" of length {J}.

Critic: user	Review the following proposed tasks.
	Fix schema violations, enforce the target mutation ratio, remove duplicates (including with history), and ensure feasibility.
	Current rung and ratio: `m={{LEVEL_M}}`, `rho_m={{RHO_M}}`.
	Anchor task JSON:
	
	{{TASK_SPEC}}
	
	Mutation/evaluation history JSON:
	
	{{MUTATION_HISTORY_JSON}}
	
	Proposed tasks JSON:
	
	{{PROPOSED_TASKS_JSON}}
	
	Return ONLY:
	{
	"task_list": [... exactly {J} tasks ...]
	}

Deterministic (rule-based) verification and regeneration. After the Critic returns `"task_list"`, we apply a rule-based verifier before any task is admitted. This verifier enforces hard constraints and filters out near-duplicates against both the anchor and the full history. If any task fails verification, it is rejected and replaced via critic-driven regeneration until a valid batch of size J is obtained.

Practical validation checks (hard constraints). After parsing the JSON, we enforce: (i) all required keys exist and types are valid; (ii) `dim` matches `len(bounds)`; (iii) each bound satisfies `lower < upper`; (iv) `noise_sigma > 0`; (v) the mutation-ratio constraint is satisfied within tolerance by counting edited (editable) fields relative to the anchor; (vi) near-duplicate tasks (within a small tolerance on numeric fields) are rejected if they duplicate the anchor, each other, *or any task already present in the mutation history* (e.g., `task_registry`). Only tasks that pass both the Critic audit and these deterministic checks are evaluated downstream.

K.1.2. Evaluator: pairwise judge prompt (committee votes)

We implement the cross-task utility oracle as a **pairwise judge** that compares two task–incumbent summaries and returns a binary preference (A vs. B). To reduce variance, we use a committee of K independent votes and average them.

Evaluator I/O format. **Input:** two JSON blobs (A and B), each containing:

- `task_spec`: the task JSON (domain-specific schema),
- `g`: the current estimated utility / score (higher is better),
- `gbar`: an optimistic score (e.g., an upper confidence summary),
- `metrics`: optional diagnostic fields (noise, dim, etc.).

Output: a single JSON object

```
{ "winner": "A" or "B", "notes": "..." }
```

We parse `winner` and map it back to the original candidate/anchor ordering if A/B were swapped.

Exact prompt templates (evaluator). Table 5 is the exact templates used in code.

Table 5. Evaluator prompts (random personas)

Judge: system	You are a careful judge that decides which BO task should get the next evaluation. You must output only JSON.
Judge: user	You will be given two candidate tasks A and B. Each is a JSON describing a BO task setting with: - task_spec: the task specification JSON. - g: the current best utility/score achieved so far (higher is better). - gbar: an optimistic estimate / upper confidence summary for the long-run value. - metrics: optional extra diagnostics. Your job: choose which task should receive the next expensive BO evaluation. Decision guidelines: - Prefer larger gbar (more promising under uncertainty), but consider g (exploitation). - If similar, prefer settings that are more learnable / reliable (e.g., lower noise, simpler search space), and prefer tasks that are not obviously dominated. - If both are weak or unclear, choose the one with larger gbar. Return ONLY JSON with schema: { "winner": "A" or "B", "notes": "one sentence" } Task A: {{TASK_A_JSON}} Task B: {{TASK_B_JSON}}

K.1.3. Persona-conditioned evaluator prompt, loading, and vote aggregation

Why personas. To reduce "single-judge idiosyncrasy" and to better emulate heterogeneous stakeholder preferences, we evaluate each pairwise comparison using a *committee* of persona-conditioned judges. Each committee member is the same base model but with a different persona injected into the system prompt.

Persona bank format and loading. We represent each persona as a single JSON object and store a persona bank as JSONL (one persona per line). At runtime, we load the bank once and then sample personas for each query:

$$\texttt{personas} \leftarrow \texttt{[json.loads(line) for line in open("personas.jsonl")]}.$$

Each vote draws either (i) K personas i.i.d. with replacement, or (ii) K personas without replacement when $|\texttt{personas}| \geq K$.

Persona schema (JSON).

```
{
  "persona_id": "string",
  "role": "string",                % e.g., "risk-averse engineer",
```

Table 6. Evaluator prompt (pooled personas)

SYSTEM:	You are simulating the following persona (JSON): {persona_json} You will compare two candidates for THIS persona. Focus on the persona's goals and preferences. Return ONLY a valid JSON object with keys: winner, confidence, notes.
USER	Candidate A (JSON): {candidate_A_json} Candidate B (JSON): {candidate_B_json} Question: Which candidate is better for the persona? Pick A or B. Output JSON only: {"winner": "...", "confidence": ..., "notes": "..."}

```
  "goals": ["...","..."],          % short bullet goals
  "preferences": { ... },          % domain-specific knobs (weights, constraints, etc.)
  "avoid": ["...","..."]           % optional hard dislikes / constraints
}
```

Evaluator input. Each comparison query provides two candidates (e.g., task i vs. anchor task a), including the task spec and the current incumbent performance summary (the "best-so-far" outcome for each task). We pass these inputs as JSON blocks.

Evaluator output schema (JSON). The judge returns a single JSON object:

```
{
  "winner": "A" | "B",
  "confidence": 0.0-1.0,
  "notes": "short rationale"
}
```

We discard any malformed outputs (rare under JSON mode), and in case of ties or ambiguity the judge is instructed to pick a winner with `confidence` near 0.5.

Prompt template (persona-conditioned judge). The system message injects the persona; the user message supplies the two candidates and reiterates the output schema. (Our reference implementation uses JSON mode, i.e., the model is required to emit a valid JSON object.) Table 6 shows the exact prompt for persona-conditioned evaluator.

K.2. New product development: wine planning benchmark

This subsection fully specifies the wine benchmark used in §4 (planning). The goal is to model a realistic "product design" loop in which (i) within-task BO proposes wine recipes/chemistries, and (ii) a cross-task evaluator (simulated or LLM committee) judges which *target product brief* is more promising.

K.2.1. Dataset and feature normalization

We use the Wine Quality dataset (Cortez et al., 2009), separately for red and white wines. Each datapoint contains 11 physicochemical features and a quality label. Let $x_{\text{raw}} \in \mathbb{R}^{11}$ denote the raw features and y_{raw} the integer quality label.

Robust bounds and unit-cube mapping. To avoid extrapolation to extreme outliers, we compute per-feature lower/upper bounds using empirical quantiles: for each feature dimension k, let ℓ_k be the 2% quantile and u_k the 98% quantile over the dataset. We map raw features to the unit cube via

$$x_k \;=\; \frac{x_{\text{raw},k} - \ell_k}{u_k - \ell_k} \quad \in [0,1], \qquad x = (x_1, \ldots, x_{11}) \in [0,1]^{11}. \tag{39}$$

All task domains $\mathcal{X}^{(i)}$ are axis-aligned boxes in the unit cube.

K.2.2. Quality surrogate and bounded tasting score

We define a reusable, fast environment by fitting a regression surrogate $\widehat{g}(x_{\text{raw}})$ that predicts the quality label from physicochemical measurements. In our released implementation we use histogram gradient boosting regression and normalize to $[0,1]$:

$$q(x) \;=\; \text{clip}_{[0,1]}\left(\frac{\widehat{g}(x_{\text{raw}}) - y_{\min}}{y_{\max} - y_{\min}}\right) \in [0,1], \tag{40}$$

where $y_{\min}, y_{\max}$ are the minimum/maximum labels in the dataset split used to fit $\widehat{g}$. This quantity $q(x)$ plays the role of a normalized "intrinsic wine quality."

K.2.3. Task specification: product briefs as structured constraints

A task corresponds to a *product brief* (target customer/taste intent) and is represented as a structured specification. Concretely, each task i is a tuple

$$\mathcal{T}_i = \Big(\mathcal{X}^{(i)} \subseteq [0,1]^{11},\; w_i,\; t_i,\; \sigma_i\Big), \tag{41}$$

where:

- $\mathcal{X}^{(i)} = \prod_{k=1}^{11} [L_{i,k}, U_{i,k}]$ is a box constraint in the normalized space;
- $w_i = (w_{q,i}, w_{m,i})$ with $w_{q,i} \in [0,1]$ and $w_{m,i} = 1 - w_{q,i}$ controls the trade-off between quality and style-match;
- t_i specifies a target *style profile* on a small subset of features (e.g., residual sugar, alcohol, acidity, sulphates), expressed in unit coordinates;
- σ_i specifies per-feature tolerances (style "bandwidths") for the same subset.

Style-match score. Let $\mathcal{F}$ denote the subset of style-relevant feature indices. For each $k \in \mathcal{F}$, the task defines a target $t_{i,k} \in [0,1]$ and tolerance $\sigma_{i,k} > 0$. We define the style match as a Gaussian-shaped preference:

$$m_i(x) \;=\; \exp\left(-\frac{1}{2}\sum_{k\in\mathcal{F}}\left(\frac{x_k - t_{i,k}}{\sigma_{i,k}}\right)^2\right) \in (0,1]. \tag{42}$$

Within-task objective (bounded). Given a task $\mathcal{T}_i$ and a design $x \in \mathcal{X}^{(i)}$, the environment returns a bounded score

$$f^{(i)}(x) \;=\; w_{q,i}\, q(x) + w_{m,i}\, m_i(x) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}}), \tag{43}$$

clipped to $[0,1]$ for numerical stability. In our code, σ_{noise} is a small constant that emulates measurement/estimation noise in tasting outcomes.

K.2.4. Within-task BO details (wine)

Within each task i, we run GP-based BO to maximize $f^{(i)}$ over $\mathcal{X}^{(i)}$. We use a single-task GP surrogate and UCB-style acquisition:

- **Initialization:** the first n_{init} evaluations for a task are uniform random points in $\mathcal{X}^{(i)}$.
- **Model:** a single-task GP (Matérn-$5/2$ kernel in our implementation).
- **Acquisition:** q-UCB (with $q = 1$), optimized by multi-start local search.
- **Incumbent:** $\overline{y}_s^{(i)} = \max_{r \le s} y_r^{(i)}$.

All tasks use the same BO hyperparameters so that differences in performance come from task-level task selection/generation.

K.2.5. Cross-task utility: committee comparisons and transported CIs

To compare tasks with potentially different scales and constraints, we score task progress through a bounded utility $u^{(i)}(\overline{y}_s^{(i)}) \in [0, 1]$. In the wine benchmark we use pairwise comparisons against an anchor task a_t and recover cardinal utilities using Bradley–Terry transport (Appendix G.1).

Pairwise vote model. At global round t, the evaluator compares the pairs $(i_t, \overline{y}_t^{(i_t)})$ vs. $(a_t, \overline{y}_t^{(a_t)})$ and returns K_t binary votes. The empirical win-rate $\hat{p}_t$ estimates $p_t = \Pr(i_t \succ a_t)$, and a Hoeffding CI yields $p_t \in [p_t^-, p_t^+]$ with high probability.

Transport to cardinal utility. Given an anchor utility interval $u^{(a_t)} \in [\underline{u}^{(a_t)}, \overline{u}^{(a_t)}]$, we update the selected task's utility interval by the logit-add identity:

$$\operatorname{logit} u^{(i_t)} \;=\; \operatorname{logit} u^{(a_t)} + \operatorname{logit} p_t, \tag{44}$$

implemented by transporting endpoints. This provides $\underline{u}^{(i_t)}$ and $\overline{u}^{(i_t)}$, which are then lifted to value envelopes $\underline{U}, \overline{U}$ using the within-task optimization-gap headroom as in Theorem E.3.

Simulated vs. LLM committees. For reproducibility and ablations, we provide a simulated committee that follows a Bradley–Terry/logistic preference model. In LLM experiments, the same interface is used but each vote is produced by an LLM judge (committee members differ by prompts/personas), and K_t is fixed or increased until the target CI width is reached (Appendix G.1.2).

K.2.6. Task generation: coarse-to-fine mutations of product briefs

We represent tasks as structured specs (e.g., JSON) and generate new tasks by mutating editable fields. Generation is coupled to the resolution ladder ϵ_m^U via a decreasing mutation schedule.

Seed tasks (level $m = 0$). We start from a small set of human-written product briefs (archetypes), which play the role of the initial seed pool. Each archetype specifies: (i) a name (e.g., "Dry & Crisp"), (ii) a weight w_q balancing quality vs. style, and (iii) style targets on $\mathcal{F}$ (Eq. 42). All begin with the full domain $\mathcal{X} = [0, 1]^{11}$.

Mutation schedule and edited fields. At ladder level m, we target a mutation ratio $\rho_m = \rho_0 2^{-m}$: higher m corresponds to smaller edits. A child task is generated by applying a small number of edits drawn from:

- **Weight mutation:** $w_q \leftarrow \operatorname{clip}(w_q + \Delta_w(m)\xi, 0, 1)$ with $\xi \sim \operatorname{Unif}[-1, 1]$ and $\Delta_w(m) = \Delta_w^{(0)} 2^{-m}$.
- **Style-target mutation:** for $k \in \mathcal{F}$, $t_k \leftarrow \operatorname{clip}(t_k + \Delta_t(m)\xi_k, 0, 1)$ with $\Delta_t(m) = \Delta_t^{(0)} 2^{-m}$.
- **Domain refinement (optional):** shrink a subset of bounds $[L_k, U_k]$ around the current anchor best design to focus on a more specific style region, while enforcing a minimum per-dimension width.

We discard duplicates (up to tolerance) and invalid/degenerate domains.

Table 7. Default hyperparameters for the wine planning benchmark.

Component	Value (default)
Wine type	red and white (separate runs)
Global horizon	$T = 200$
Within-task BO init	n_{init} random points per task (e.g., 6)
Within-task GP	single-task GP, Matérn-$5/2$
Acquisition	q-UCB with $q = 1$
Acqf optimization	multi-start, fixed restarts/raw samples
Obs. noise (simulated tasting)	small Gaussian noise, clipped to $[0, 1]$
Utility oracle	simulated or LLM committee (pairwise votes)
Votes per utility query	$K = 64$ (chosen by Eq. 10)
Generator batch size	$J = 3$ (chosen by Lemma G.19)
Mutation schedule	$\rho_m = \rho_0 2^{-m}$, step sizes $\propto 2^{-m}$

K.2.7. Wine benchmark hyperparameters

Table 7 summarizes the default hyperparameters used in the released wine benchmark implementation.

K.3. Synthesis Scaling Experiments

K.3.1. Benchmark environment and decision variables

We use the Summit SnAr benchmark. Each evaluation corresponds to selecting a 4D reaction condition vector $x = (\tau, \text{equiv}, \text{conc}, \text{temp})$, with global bounds:

$$\tau \in [0.5, 2.0], \quad \text{equiv} \in [1.0, 5.0], \quad \text{conc} \in [0.1, 0.5], \quad \text{temp} \in [30, 120].$$

The environment returns two primary outcomes: space-time yield (STY; higher is better) and E-factor (lower is better). We normalize them using fixed global bounds ($\texttt{STY_BOUNDS} = (0, 8000)$ and $\texttt{E_BOUNDS} = (0, 20)$ to obtain $\phi_{\text{sty}} \in [0, 1]$ and $\phi_{\text{ef}} \in [0, 1]$. We also define the "good" E-factor utility as $\phi_{\text{ef,good}} = 1 - \phi_{\text{ef}}$.

K.3.2. Task specification (reaction plans)

A task i corresponds to a *reaction plan* parameterized by: (i) a local search region (box constraints) expressed in normalized coordinates $\texttt{bounds_unit} \subseteq [0, 1]^4$, and (ii) a scalarization weight vector over normalized objectives.

$$\texttt{spec}_i = \big(\texttt{task_id}_i,\ \texttt{bounds_unit}_i,\ \texttt{weights}_i\big), \quad \texttt{weights}_i = (w_{\text{sty}}, w_{\text{ef}}),\ \ w_{\text{sty}} + w_{\text{ef}} = 1.$$

Given a design x, the task-level scalarized score is

$$g_i(x) = w_{\text{sty}}\phi_{\text{sty}}(x) + w_{\text{ef}}\phi_{\text{ef,good}}(x), \qquad g_i(x) \in [0, 1].$$

Each task instance maintains its incumbent best score $\bar{g}_i = \max_{s \le t} g_i(x_{i,s})$ and incumbent best design $x_i^\star$.

K.3.3. Within-task optimizer

For each task, we run BO:

- **Initialization.** We draw $n_{\text{init}} = 4$ i.i.d. designs uniformly from the task box constraints, evaluate the simulator, and seed a dataset (X, Y).
- **Surrogate model.** We fit a single-task GP with an Matérn-$5/2$ kernel and standard output normalization. Inputs are normalized to $[0, 1]^d$.
- **Acquisition.** We use an upper-confidence bound (UCB) acquisition $\mu(x) + \beta\sigma(x)$.
- **Acquisition optimization.** We optimize UCB using multi-start gradient-based search with `n_restarts`=10 and `raw_samples`=512, returning the best candidate.

Each global meta step allocates exactly *one* new simulator call to exactly one task.

K.3.4. TASK MUTATION GENERATOR

We generate new reaction plans. At a generation event, we pick an *anchor* plan and propose a batch of mutated tasks by:

- **Weight mutation.** Add Gaussian noise to w_{sty} with a resolution-dependent step size, then clip to $[0.05, 0.95]$ and set $w_{\text{ef}} = 1 - w_{\text{sty}}$.
- **Bound mutation.** Shrink and re-center each dimension's interval around the anchor's incumbent $x^\star$ (with a small random jitter), while enforcing a minimum span of `min_span_frac`=0.15 times the global span. The shrink factor decreases with the resolution level m via `shrink_decay`.
- **Deduplication.** Reject candidate tasks whose (weights,bounds) are within a small ℓ_2 tolerance `duplication_eps` of any existing task.

Unless the global task budget is reached, accepted tasks are instantiated with fresh within-task BO state and added to the active set.

K.4. Algorithm Analysis Experiments: Acquisition-Function Inverse Optimization

This appendix provides implementation-level details for the *algorithm analysis* experiments, where we use inverse optimization to discover black-box optimization settings ("tasks") under which a *target* Bayesian optimization acquisition function outperforms strong alternatives. Concretely, each task is a parameterized synthetic benchmark derived from standard BoTorch test functions, and the task utility is computed by running a *BO race* between acquisition functions and converting their performance gap into a bounded utility in $[0, 1]$.

K.4.1. TASK PARAMETERIZATION AND SEARCH SPACE

Task schema. Each task is represented as a JSON-serializable structure:

```
TaskSpec {
  base_function: str,
  dim: int,
  bounds: List[[low_i, high_i]] for i=1..dim,
  noise_std: float,
  input_transform: {shift: List[float], scale: List[float]} or null,
  output_transform: {scale: float, offset: float} or null,
  negate: bool,
  metadata: {target_acq: str, rationale: str, m: int}
}
```

Base function catalog. We draw and mutate tasks over the following BoTorch synthetic objectives: `Ackley`, `Rosenbrock`, `Griewank`, `Levy`, `StyblinskiTang`, `Branin`, `Beale`, `SixHumpCamel`, and `Hartmann`.

Domain and transforms. Each task is evaluated over the unit hypercube $\mathbf{x} \in [0, 1]^d$ and then mapped to the task bounds. Let $\mathbf{b}_i = [\ell_i, u_i]$ denote the task bounds for dimension i. The wrapper used in the BO race applies (i) an optional affine transform in unit space, (ii) a bounds map to $\mathbf{b}_i$, and (iii) an optional output affine transform:

$$\tilde{\mathbf{x}} = \text{clip}\big(\mathbf{x} \odot \text{scale} + \text{shift},\, 0, 1\big), \qquad \mathbf{x}_{\text{task},i} = \ell_i + \tilde{x}_i(u_i - \ell_i),$$

$$y = f_{\text{base}}(\mathbf{x}_{\text{task}}), \quad y \leftarrow (-y) \text{ if negate}, \quad y \leftarrow y \cdot s_{\text{out}} + o_{\text{out}} \text{ if output_transform enabled.}$$

We add i.i.d. Gaussian observation noise only to the value fed into the surrogate model: $y_{\text{obs}} = y + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2)$, where `noise_std`$= \sigma$.

Validity checks. Generated tasks are validated before use: the base function must be in the catalog; `dim` must be within a configured range; `bounds` must have length `dim` and satisfy $\ell_i < u_i$; `noise_std` ≥ 0; and input/output transform shapes must be consistent.

K.4.2. Within-Task Evaluator: BO Race and Ground-Truth Utility

Normalized regret. To normalize across heterogeneous objectives/scales, we use a known global optimum $f^\star$ in BoTorch test functions. We define a robust scale

$$\text{robust_scale} = \text{Quantile}_q(f^\star - f(\mathbf{x}_j)), \qquad q = 0.10,$$

and set the denominator

$$\text{denom} = \max\left(f^\star - y_{\text{init}}^{\max},\ \text{robust_scale}\right),$$

where $y_{\text{init}}^{\max}$ is the best noiseless value amongst the initial points. At each step s, an agent's (simple) normalized regret is

$$r_a(s) = \frac{f^\star - \overline{y}_t}{\text{denom}},$$

where $\overline{y}_t$ is the incumbent. We also track cumulative regret $R_a(s) = \sum_{t=1}^{s} r_a(t)$.

Gap and utility. Let t denote the target acquisition and let $\mathcal{A} \setminus \{t\}$ be its competitors. We convert cumulative regret into a higher-is-better "score" by using negative average cumulative regret:

$$\text{score}_a(s) \equiv -\frac{R_a(s)}{s}.$$

We then compute the target advantage (gap) against the strongest competitor:

$$\bar{g}(s) = \gamma\left(\text{score}_t(s) - \max_{o \neq t} \text{score}_o(s)\right),$$

where `gap_scale`$= \gamma$ (default $\gamma = 1$). Thus $\bar{g}(s) > 0$ indicates the target acquisition has lower average cumulative regret.

Finally, we map this gap to a bounded task utility using a sigmoid:

$$u(s) = \sigma\left(\frac{\bar{g}(s) - \kappa}{\tau}\right), \qquad \sigma(z) = \frac{1}{1 + e^{-z}},$$

with `sigmoid_kappa`$= \kappa$ (default $\kappa = 0$) and `sigmoid_tau_slope`$= \tau$ (default $\tau = 0.25$). To maintain a non-decreasing "incumbent" task utility over time, we store $\bar{g}_{\max}(s) = \max_{s' \leq s} \bar{g}(s')$ and set $u(s) = \sigma\big((\bar{g}_{\max}(s) - \kappa)/\tau\big)$.

K.4.3. LLM Task Mutation Generator for Algorithm Analysis

We generate new benchmark tasks by mutating existing tasks. For algorithm analysis, the goal of mutation is directional: produce children that make the *target* acquisition function look better than its strongest alternative.

Actor–critic generation loop. Given a parent task, we: (i) sample J candidate children from an LLM "actor", (ii) optionally score each child with an LLM "critic", (iii) retain the top-K candidates (and apply a diversity filter if enabled), and (iv) insert the selected children into the task pool.

LLM I/O format. The actor is required to output strict JSON with a single key `child_task` containing the fields of `TaskSpec` (excluding `metadata`, which is filled/updated by the runner). A minimal example is:

```
{
  "child_task": {
    "base_function": "Ackley",
    "dim": 6,
    "bounds": [[-5, 5], [-5, 5], [-5, 5], [-5, 5], [-5, 5], [-5, 5]],
```

Table 8. Default hyperparameters for the Algorithm Analysis.

Component	Key hyperparameters (defaults)
BO race init	$n_0 = 4$ shared random initial points
GP surrogate	`SingleTaskGP` + `Standardize(1)`; exact MLL fit
Acq opt.	`raw_samples`=256, `num_restarts`=8
UCB	β_t is computed by (Hong et al., 2023)
MES	candidate set = 1024; fantasies = 16; MV samples = 10; Y samples = 128
Calibration	$N_{\text{cal}} = 1024$; quantile $q = 0.10$
Gap/utility	`gap_scale` = 1.0; $\kappa = 0$; $\tau = 0.25$; strongest competitor via `max`
LLM generator	$J = 8$ proposals; keep $K = 10$; diversity filter on
Actor	temp = 1.0; max tokens = 500; model `gpt-4o-mini`
Critic	temp = 0.0; max tokens = 300; model `gpt-4o-mini`

```
    "noise_std": 0.05,
    "input_transform": {"shift": [0,0,0,0,0,0], "scale": [1,1,1,1,1,1]},
    "output_transform": {"scale": 1.0, "offset": 0.0},
    "negate": true,
    "rationale": "Slightly higher noise + mild input rescaling to stress exploration."
  }
}
```

The critic outputs:

```
{"score": 0.82, "short_reason": "Adds noise and mild rescaling; likely helps UCB vs EI/MES."}
```

K.4.4. Hyperparameters and Reproducibility Checklist

Table 8 lists the default hyperparameters used in the evaluator and generator implementations.

K.5. Patent repurposing

Goal. We instantiate our framework on a *patent repurposing* proxy task based on the Materials Project (MP): starting from a *seed* material (e.g., MgO or Bi_2O_3), we search for candidate materials that are (i) *structurally similar* to the seed (capturing the intuition of "repurposing" with minimal design changes), yet (ii) *unusual* along some target combination of materials properties (capturing novelty / potential for new use-cases). In our implementation, the candidate set is a finite subset of MP entries, and both within-task optimization and task discovery operate on this fixed set.

Dataset construction (MP subset). We query MP via `mp-api` (requires an `MP_API_KEY`) and build a cached subset of N candidate materials (typical runs use $N \approx 2{,}000$, configurable). We retain entries with valid values for a fixed list of scalar properties (examples used in our configs include `formation_energy_per_atom`, `energy_per_atom`, `band_gap`, `efermi`, `poisson_ratio`, and elastic surrogates such as `k_vrh` / `g_vrh` (or `bulk_modulus` / `shear_modulus`)), dropping entries with missing values. We cache: MP material IDs, reduced formulas, the chosen property matrix, and structural embeddings (below) (e.g., `mp_subset.npz` / `mp_subset.csv`; see `prepare_mp_subset.py`).

Structural representation (CGCNN → PCA). Each material is mapped to a learned structure embedding using a pretrained crystal-graph convolutional network (CGCNN). We extract a 128-dimensional penultimate-layer embedding $\phi(m) \in \mathbb{R}^{128}$ for each material m and reduce it to $d = 6$ dimensions using PCA, followed by standardization:

$$x(m) \;=\; \mathrm{Std}(\mathrm{PCA}_6(\phi(m))) \in \mathbb{R}^6.$$

The within-task domain is the *discrete* set $\mathcal{X} = \{x(m)\}_{m=1}^{N}$.

Property surprisal features (per-property KDE). Let $p_j(m)$ denote property $j \in \{1, \ldots, P\}$ for material m. We z-score each property across the dataset and fit an independent 1D Gaussian KDE $\hat{p}_j(\cdot)$ on the standardized values. We then define the *surprisal* feature

$$s_j(m) \;=\; -\log \hat{p}_j\big(\overline{y}_j(m)\big),$$

clipping the log-density from below for numerical stability (implementation: `mp_dataset.compute_surprisal`). This yields a surprisal vector $s(m) \in \mathbb{R}^P$ per material.

Task parameterization and objective. A task corresponds to a particular "repurposing direction" specified by: (i) a convex weight vector $w \in \Delta^{P-1}$ selecting which properties should be *surprising*, and (ii) a nonnegative regularization strength $\lambda \geq 0$ encouraging proximity to the seed in embedding space. Fixing a seed material index m_{ref} (and $x_{\text{ref}} = x(m_{\text{ref}})$), the task objective is

$$f_{w,\lambda}(m) \;=\; w^\top s(m) \;-\; \lambda \, \|x(m) - x_{\text{ref}}\|_2. \tag{45}$$

Thus, maximizing $f_{w,\lambda}$ favors candidates that are *rare* in the selected properties while remaining *close* to the seed in embedding space. (Implementation: `mp_task.PatentTaskSpec.objective`.)

Within-task optimization: discrete GP-UCB over the MP subset. For each task (w, λ), we run Bayesian optimization on the discrete set $\mathcal{X}$. We fit an exact GP with an RBF kernel (Gaussian likelihood with small noise), update it on the evaluated points, and choose the next candidate by maximizing a UCB acquisition on the full discrete pool:

$$m_{t+1} \in \arg\max_{m \in \mathcal{X} \setminus \mathcal{D}_t} \; \mu_t\big(x(m)\big) \;+\; \beta_t^{1/2} \, \sigma_t\big(x(m)\big).$$

Each task is initialized with a small random design (e.g., 4 points) before UCB selection begins. (Implementation: `mp_bo.MPTaskBO` and `mp_bo.DiscreteGPUCB`.)

Task mutation generator. Task generation is *heuristic* (no LLM) and operates directly in parameter space. Given an "anchor" task (w, λ) at resolution level m, we sample J children by Gaussian perturbations with level-dependent step sizes:

$$w' \leftarrow \Pi_\Delta\big(w + \sigma_w(m)\, \xi\big), \quad \xi \sim \mathcal{N}(0, I), \qquad \lambda' \leftarrow \text{clip}\big(\lambda + \sigma_\lambda(m)\, \eta,\ [\lambda_{\min}, \lambda_{\max}]\big),$$

where Π_Δ denotes projection to the simplex and $\sigma_w(m), \sigma_\lambda(m)$ decrease geometrically with m (e.g., $\sigma(m) = \sigma_0 \cdot 2^{-m}$), implementing a coarse-to-fine search over task space.

task-level selection and generation. We run our main meta-algorithms (task-UCB / GSBE) on the evolving task set. Each global step advances exactly one task by one within-task BO evaluation, producing an updated incumbent utility estimate. Generation events add J mutated tasks around the current anchor when the anchor's uncertainty width falls below a rung-dependent threshold, progressively refining the search. For this benchmark, we use a simple decreasing optimization-gap schedule $\varepsilon_f(s) = \varepsilon_0/\sqrt{s}$ as a conservative surrogate for the within-task optimization error after s evaluations.

LLM-based "repurposability" evaluator. An LLM-as-a-judge evaluator compares two candidate task outcomes and returns a binary preference vote. Concretely, each vote is a call to a chat model with: (i) a *persona* system prompt (e.g., senior materials scientist / product engineer / patent strategist) and (ii) a JSON user message containing option A and B (each includes a brief task description and the current best candidate material). The model is constrained to return JSON of the form `{"winner":"A"}` or `{"winner":"B"}`. A committee repeats this with multiple personas / samples and aggregates by majority vote.

Default hyperparameters. The following values are typical defaults used in our provided scripts (all configurable):

Quantity	Typical value
Candidate set size	$N \approx 2{,}000$ (MP subset; configurable)
Embedding dimension	$d = 6$ (PCA-reduced CGCNN embedding)
# properties	$P = 6$–7 (configurable list)
Initial BO design per task	$n_0 = 4$
Global budget	$T = 100$ total BO evaluations
Children per generation	$J \in [3, 8]$
Mutation scales	$\sigma_{w,0} \approx 0.25$–$0.30$, $\sigma_{\lambda,0} \approx 0.20$–$0.25$, scaled by 2^{-m}
λ range	$[0, 2]$ (clipped)
Within-task kernel	RBF (exact GP), $\sigma_{\text{noise}} = 0.01$

K.5.1. Experimental analysis and ablation study details

This appendix provides implementation-level details for the three diagnostic studies reported in §B.3: (i) *resolution controllability*, (ii) the *mutation scheduler* ablation, and (iii) *Lipschitz constant adaptation*. Unless otherwise stated, we use the same planning benchmark, within-task BO implementation, and LLM interfaces as in Appendix K.2 and Appendix K.1. All plots report the mean and $\pm$ one standard error over random seeds (same aggregation protocol as the main experiments).

What is held fixed across all ablations. Across all ablation variants we keep fixed: (i) the seed task(s) i_0 (and the initial seed pool if multiple seeds are used), (ii) the total global budget T (objective evaluations), (iii) the within-task BO stack (GP family, acquisition rule, optimization settings; Appendix K.2), (iv) the utility elicitation protocol (committee voting / simulated voting and its CI target; Appendix G.1), and (v) the generator interface and JSON schema (Appendix K.1.1). Each ablation modifies only the component stated in its title.

Notation (used in the plots). At rung/level m, the target mutation ratio is ρ_m and the resolution threshold is $\epsilon_m^U = \epsilon_0^U 2^{-m}$. A generation event proposes a batch of J child tasks $\mathcal{B}_m \sim \mathsf{Gen}(a_t, m, J)$ from the current anchor a_t. For each task i, we evaluate it by its best-so-far (incumbent) utility $u^{(i)}(\overline{y}_s^{(i)}) \in [0, 1]$, and we use the long-run-value notation $U^{(i)} = u^{(i)}(f^{\star(i)})$ from §2.3.

K.5.2. Resolution controllability (Fig.7)

The goal of this experiment is to verify that the *effective task step size* induced by our JSON mutation operator is controlled by the target mutation ratio ρ_m, and to empirically characterize how often mutations produce near-optimal children as refinement proceeds.

Anchor tasks and sampling protocol. We select a small set of representative *anchor tasks* $\{a^{(1)}, \ldots, a^{(A)}\}$ (we use $A = 4$ in the paper) from independent runs of the LLM-powered wine planning setting. Anchors are chosen to span multiple refinement levels and utility ranges (early/mid/late in the run). For each anchor a and each rung m considered, we generate a batch of J children (using the same generator Gen as in the main experiment; Appendix K.1.1). In the plots we use $J = 40$ children per condition, and repeat the procedure over multiple random seeds (for both the generator sampling and the within-task BO randomness) to estimate variability.

Task distance (Fig.7a). Each task is represented as a structured specification (JSON), but the controllability plot requires a numeric distance. Let $\theta^{(i)} \in \mathbb{R}^p$ denote the vector obtained by concatenating the *numeric* fields of the task (e.g., continuous weights, target style parameters, tolerances, and/or numeric domain parameters; we exclude purely textual fields). We rescale each coordinate to $[0, 1]$ using its natural bounds from the task schema (or global min/max if appropriate), yielding $\tilde{\theta}^{(i)} \in [0, 1]^p$. We define the task distance between a child i and its parent anchor a as

$$d(i, a) \;:=\; \left\| \tilde{\theta}^{(i)} - \tilde{\theta}^{(a)} \right\|_2 . \tag{46}$$

We then correlate $d(i, a)$ with the requested mutation ratio ρ_m used when generating i. The plot in Fig.7(a) shows that larger ρ_m leads to larger distances on average, confirming that the mutation ratio provides a practical knob for controlling resolution.

Empirical near-optimal generation probability $\widehat{\delta}_+(m)$ (Fig.7b–c). Condition E.4 is stated in terms of the (unknown) optimal value $U^\star$. To estimate the probability of producing a near-optimal child at rung m in a reproducible way, we use a proxy *within the sampled pool* at that rung: for each anchor a and rung m, we evaluate each generated child $i \in \mathcal{B}_m$ with a fixed probe budget of within-task BO steps (identical across children), producing an estimated value $\widehat{U}^{(i)}$ (e.g., the final value-envelope midpoint or the final best-so-far utility; the choice is fixed across all children). We then define the best observed value within that batch as

$$\widehat{U}_m^\star(a) \;:=\; \max_{j \in \{a\} \cup \mathcal{B}_m} \widehat{U}^{(j)}.$$

A child i is counted as a success at rung m if it is within ϵ_m^U of the batch best:

$$\mathbf{1}\{i \text{ successful at rung } m\} \;:=\; \mathbf{1}\Big\{ \widehat{U}_m^\star(a) - \widehat{U}^{(i)} \le \epsilon_m^U \Big\}. \tag{47}$$

Finally, we estimate the generation success probability by averaging over anchors, random seeds, and children:

$$\widehat{\delta}_+(m) \;:=\; \mathbb{E}\Big[\,\mathbf{1}\{i \text{ successful at rung } m\}\,\Big]. \tag{48}$$

Fig.7(b) plots $\widehat{\delta}_+(m)$ across refinement levels and shows that it remains non-zero across all measured rungs, while typically decreasing as m increases. Fig.7(c) further stratifies success by the task distance $d(i, a)$ in (46), illustrating how coarse mutations (larger distance) can increase the chance of finding a stronger child when improvement is still possible.

Improvement magnitude (Fig.7d). To quantify how much a child improves over its anchor, we compute

$$\Delta(i; a) \;:=\; \widehat{U}^{(i)} - \widehat{U}^{(a)}, \tag{49}$$

with $\widehat{U}^{(\cdot)}$ obtained under the same probe budget and evaluation protocol as above. Fig.7(d) reports the distribution (or mean) of $\Delta(i; a)$ across rungs and confirms the expected exploration-to-exploitation pattern: coarse edits tend to produce larger improvements when improvements are available, while finer edits are better suited once the anchor is already strong.

K.5.3. Mutation scheduler ablation (Fig.12)

This ablation isolates the effect of coupling the mutation ratio schedule ρ_m with the *width-gated* generation rule in Algorithm 1 (Line 1).

Default scheduler (GSR). Our default generator uses the rung-dependent schedule

$$\rho_m \;=\; \rho_0 2^{-m},$$

and we *only* advance the rung m and generate a new batch when the current anchor is sufficiently resolved:

$$w_t \;\le\; c_g \epsilon_m^U,$$

where w_t is the anchor width from Eq. (4). Intuitively, this ensures that coarse edits are used when uncertainty is still high, while finer edits are used only after the anchor task has been evaluated enough for BO feedback to become informative.

Ablation variants. We compare the default scheduler against three replacements, keeping all other components unchanged (task selector, generator prompt format/schema, within-task BO, and the evaluator):

1. **Constant mutation ratio.** Use $\rho_m \equiv \rho_0$ for all rungs, while keeping the same width-gated generation trigger $w_t \le c_g \epsilon_m^U$.
2. **Time-based rung updates (with gating).** Increase the rung deterministically every fixed number of global rounds (e.g., every 20 iterations in our implementation), but still only *generate* children when $w_t \le c_g \epsilon_m^U$.
3. **No gating (time-only).** Increase the rung deterministically every fixed number of global rounds and generate children at those times regardless of the width condition (i.e., ignore w_t).

Fig.12 shows that removing either the rung-dependent ρ_m schedule or the width gating degrades performance, supporting the design choice that the refinement schedule should be *evidence-driven* (i.e., coupled to the task-level uncertainty summarized by w_t).

What is plotted. We plot the same headline metric as in the main wine experiments: the best-so-far achieved utility over global rounds, aggregated over seeds (mean $\pm$ SEM). All variants receive the same global budget T of objective evaluations.

K.5.4. Lipschitz constant adaptation ablation (Fig.12)

The envelope construction and the task-UCB rule use an upper bound $\overline{L} \ge L^\star$ (Theorem E.3), but $L^\star$ is unknown in practice. We therefore adapt $\overline{L}$ online using the Balance–Eliminate mechanism in Algorithm 2 (Appendix J).

Table 9. Default hyperparameters for the unknown search space.

Benchmark	d	$\mathcal{X}_{\text{true}}$	$\mathcal{X}_0$	Budget T
Beale	2	$[-4.5,\ 4.5]^2$	$[-1.0,\ 0.0]^2$	75
Hartmann6	6	$[0.0,\ 1.0]^6$	$[0.0,\ 0.5]^6$	150

Candidate ladder and update rule. We maintain a geometric ladder of candidate bounds

$$L_j \;=\; L_0 2^j, \qquad j = 0, 1, \ldots, J_L,$$

and treat each candidate L_j as a hypothesis about a valid global Lipschitz constant. At each global round, Algorithm 2: (i) chooses which hypothesis to test next by balancing the suspected regret certificate (Line 2), and (ii) eliminates hypotheses whose certificate is falsified by observed utility (Line 2). The active set $\mathcal{A}_t$ thus shrinks over time while (with high probability) retaining at least one valid hypothesis.

Ablation variants. We compare the adaptive procedure to three simple alternatives:

1. **Fixed-L (underestimated).** Always use the smallest candidate $\overline{L} = L_0$ (no adaptation).
2. **Max-candidate.** Always use $\overline{L} = \max_{j \in \mathcal{A}_t} L_j$ among the remaining candidates. This is conservative and typically maintains $\overline{L} \geq L^\star$ once a sufficiently large rung is introduced.
3. **Min-candidate.** Always use $\overline{L} = \min_{j \in \mathcal{A}_t} L_j$ among the remaining candidates. This is aggressive and can become overconfident if the smallest surviving rung still underestimates $L^\star$.

All variants share the same ladder, the same elimination test, and the same within-rung task-UCB logic; they differ only in how $\overline{L}$ is chosen from the active set at each time. Fig.12 shows that the adaptive procedure performs best overall, while the max-candidate heuristic is competitive, consistent with the theory that validity ($\overline{L} \geq L^\star$) is crucial for maintaining correct envelopes.

Reproducibility note. For all Lipschitz ablations, we use identical random seeds and the same LLM committee / simulated committee protocol as the main run. This ensures that changes in performance are attributable to the choice of $\overline{L}$ (and not to different generator/evaluator randomness).

K.6. Non-LLM Task Generator Experiment: Unknown Search Spaces

This experiment isolates the *multi-task search and selection* components of our method by replacing the LLM-based task generator with a simple, fully algorithmic (non-LLM) generator. We instantiate tasks as *candidate search-space bounds* and study optimization when the true search bounds are unknown and the initial bounds do not contain the global optimum.

K.6.1. Problem setup and metrics

We consider black-box maximization of a scalar objective $f : \mathbb{R}^d \to \mathbb{R}$. The optimizer is given an *initial* axis-aligned box $\mathcal{X}_0 = [\ell_0, u_0] \subset \mathbb{R}^d$, while the true domain $\mathcal{X}_{\text{true}} = [\ell_{\text{true}}, u_{\text{true}}]$ (and thus the optimizer $x^\star$) may lie outside $\mathcal{X}_0$. The algorithm has access to point evaluations $y = f(x)$ (noise-free in our synthetic study).

We report performance using **best-so-far objective** and the equivalent **simple regret**

$$r_t \;=\; f(x^\star) \;-\; \max_{s \leq t} f(x_s),$$

where $x^\star$ is the known global maximizer of the benchmark (used only for evaluation).

K.6.2. Benchmarks

We use standard continuous test functions (implemented as maximization by negating the canonical minimization form): $f(x) = -f_{\text{std}}(x)$. The initial box $\mathcal{X}_0$ is a strict subset of $\mathcal{X}_{\text{true}}$ and excludes $x^\star$.

Figure 8 displays the first 60 common objective evaluations for visual comparability. The full runs use the budgets listed in Table 9; the truncation is for plotting only.

K.6.3. NON-LLM TASK GENERATOR: DOMAIN EXPANSION OPERATOR

Concretely, given current bounds $[\ell, u]$, expansion factor $\rho > 1$, and an anchor point $a \in \mathbb{R}^d$, we define

$$w = u - \ell, \qquad \ell' = a - \frac{\rho}{2}w, \qquad u' = a + \frac{\rho}{2}w.$$

In the experiments, a is the current incumbent best design of the parent task. If the parent has not yet been evaluated, we use the center $(\ell + u)/2$. The expanded box is clipped to $\mathcal{X}_{\text{true}}$ only for benchmark feasibility, not for selecting a.

Warm-starting new tasks. When a new task/domain is created, we immediately collect a small initial design of $n_{\text{init}} = 4$ points sampled uniformly in that box to fit an initial surrogate. (The same n_{init} is used for the initial task on $\mathcal{X}_0$.)

K.6.4. SURROGATE MODEL AND ACQUISITION OPTIMIZATION

Each task m maintains its own Gaussian process surrogate trained on evaluations collected while querying within $\mathcal{X}^{(m)}$. We use an exact single-task GP (BoTorch/GPyTorch) trained by maximizing the exact marginal log-likelihood, and select points using an upper confidence bound (UCB) acquisition.

K.6.5. OUR METHOD WITHOUT AN LLM: GSBE WITH DETERMINISTIC EXPANSION

We evaluate a non-LLM instantiation of our method by setting tasks to be candidate boxes and using the deterministic generator in §K.6.3. At a high level, the method alternates between: (i) selecting *which* task/box to query next using *global* confidence bounds over each box, (ii) performing a standard within-box BO step, and (iii) occasionally expanding the search space by generating a new box.

K.6.6. BASELINES FOR UNKNOWN SEARCH SPACES

We compare against representative specialized methods for unknown search bounds:

UBO (Unknown Bounds Optimization). We implement a GP-UCB-based UBO-style expansion rule in which the algorithm expands a domain when a confidence-based boundary proximity statistic exceeds a threshold $d_{\max} = 1.0$. Expansion uses the same doubling operator ($\rho = 2$) and the same cap $m_{\max} = 10$. The UBO confidence parameter is computed via the standard β-schedule used in the code (`tau`=1.0, `delta`=0.1) and per-step updates based on the model and current domain.

HuBO and HD-HuBO. We also include HuBO-style and HD-HuBO-style schedules that expand the domain according to a predefined growth rule (volume-based growth for HuBO, radius-based growth with an exponential waiting schedule for HD-HuBO), while selecting points via GP-UCB within the currently active domain(s). All methods share the same initial design size ($n_{\text{init}} = 4$), within-task confidence ($\beta_{\text{bo}} = 5.0$), and the same maximum number of expansions ($m_{\max} = 10$).

K.7. Offline Experiment: Fixed Task Selection

Setting and goal. This experiment removes the LLM task generator entirely and studies *fixed task selection*: a finite set of K tasks is given upfront and the learner must decide *which task to spend the next evaluation on* under a global evaluation budget. At each global step $t = 1, \ldots, T$, the algorithm selects a task index $i_t \in \{1, \ldots, K\}$ and performs *one* black-box function evaluation on that task using a within-task Bayesian optimizer (described below). The objective is to maximize the best achieved task utility (and equivalently minimize regret w.r.t. the best achievable utility) under the same total budget T.

Benchmark suite (fixed tasks). We use $K = 6$ standard continuous optimization benchmarks implemented via BoTorch test functions. Following the code, we treat each benchmark as a *maximization* problem by negating the standard minimization form, i.e., for each task i we optimize $f_i(x) = -y_i(x)$ where y_i is the usual BoTorch test function. All tasks use the standard box constraints listed in Table 10. We also store each task's known optimum value $f_i^\star$ (Table 10), which is used only for regret-based utility variants (not the default).

Noisy function evaluations. When querying task i, we observe a noisy outcome $y = f_i(x) + \epsilon, \;\; \epsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$ with $\sigma_{\text{noise}} = 0.01$ (default). The within-task GP model is fit to these noisy observations, while the experiment state also tracks an incumbent $\overline{y}_i$ as the best achieved value so far on task i.

Table 10. Fixed-task benchmark suite.

Task i	Dim. d	Domain $\mathcal{X}_i$	$f_i^\star$
Ackley-2D	2	$[-5,5]^2$	0
Beale-2D	2	$[-4.5,4.5]^2$	0
Branin-2D	2	$[-5,10]\times[0,15]$	-0.397887
Hartmann-6D	6	$[0,1]^6$	3.322368
Levy-2D	2	$[-10,10]^2$	0
Rosenbrock-4D	4	$[-2,2]^4$	0

Utility normalization (comparable utilities across tasks). Because raw objective scales differ across tasks, we map incumbents to a comparable $[0,1]$ utility. The default utility model in the code is

$$u_i(z) \;=\; \Phi\left(\frac{z-\mu_i}{\sigma_i}\right), \tag{50}$$

where Φ is the standard normal CDF and (μ_i,σ_i) are task-specific location/scale parameters. By default, (μ_i,σ_i) are *estimated* per task using Monte Carlo sampling: we draw $S = 20000$ points uniformly from $\mathcal{X}_i$, evaluate $f_i(\cdot)$, and set μ_i and σ_i to the sample mean and standard deviation (robust estimation is disabled by default).

Task-selection methods compared. All methods operate under the same global budget $T = 200$ evaluations.

- **GSBE-MetaUCB (ours, no generator).** We use the fixed-task variant implemented in `gsbe_meta_ucb.py`. At each step, select the task maximizing a task-UCB score based on the oracle upper confidence bound and an exploration "headroom" term that decays with the number of pulls s_i of task i: $\text{score}_i \;=\; \text{UCB}_i \;+\; \bar{L}\cdot\varepsilon(s_i)$, where $\varepsilon(s) = c/\sqrt{s}$ with default `headroom_coef` $c = 0.5$ and $\bar{L}$ is the running estimate used by the implementation.
- **Round-robin.** Cycle through tasks deterministically.
- **Uniform random.** Sample a task uniformly at random each step.
- **Successive halving.** Use $\eta = 3$ and repeatedly allocate equal per-task budgets, then prune the bottom $1 - 1/\eta$ fraction of tasks based on the observed utility estimate $\hat{u}_i$.
- **Hyperband.** Use Hyperband with $\eta = 3$ and maximum per-bracket budget $R = T$, running successive-halving brackets with different initial task counts/budgets.

The global budget cap is enforced for all baselines. For Hyperband, $R = T$ is the maximum resource parameter used to construct brackets, but execution is stopped as soon as the cumulative number of black-box evaluations reaches T; initialization evaluations are included in this count.

K.8. Offline Experiment: objective selection

This subsection provides implementation-level details for the **offline** experiment in Fig.9(b), where the task pool is fixed *a priori* and the task-learner must decide *which objective (task) to allocate each evaluation to*. This experiment isolates the *task-selection* component of GSR, i.e., Algorithm 1 without task generation.

Task pool and shared domain. We fix a set of K objectives (tasks) $\{f^{(i)}\}_{i=1}^K$ drawn from the BoTorch synthetic benchmark suite. In the **objective-selection** variant, all tasks share the *same* design domain

$$\mathcal{X}^{(i)} \equiv \mathcal{X} = [0,1]^d, \qquad d = 6,$$

so the only difference across tasks is the objective function. At each global round $t \in \{1,\dots,T\}$, a method chooses an objective index $i_t \in [K]$ and a design $x_t \in \mathcal{X}$, observes one scalar outcome

$$y_t^{(i_t)} = f^{(i_t)}(x_t) + \xi_t^f,$$

and updates the incumbent $\overline{y}_s^{(i)} = \max_{r \le s} y_r^{(i)}$ for the selected objective only. Importantly, *only one objective is observed per round*, which matches a "decoupled evaluation" scenario where measuring different objectives has separate costs.

The six objectives used in Fig. 9(b) are: Ackley-6D, Griewank-6D, Levy-6D, Rosenbrock-6D, Styblinski–Tang-6D, and Hartmann-6D. All are evaluated on the shared unit cube after the standard BoTorch input transform.

Utility normalization (cross-task comparability). Because each $f^{(i)}$ can have a different scale and offset, we map incumbents to a common $[0, 1]$ utility scale using a task-wise Gaussian CDF:

$$u^{(i)}(z) \;:=\; \Phi\Big(\frac{z - \mu^{(i)}}{\sigma^{(i)}}\Big) \in [0, 1], \tag{51}$$

where $\Phi(\cdot)$ is the standard normal CDF and $(\mu^{(i)}, \sigma^{(i)})$ are *fixed per task*. In our implementation, $(\mu^{(i)}, \sigma^{(i)})$ are computed once from a calibration set $\mathcal{C} = \{x_j\}_{j=1}^{N_{\text{cal}}} \subset [0, 1]^d$ (Sobol points):

$$\mu^{(i)} := \frac{1}{N_{\text{cal}}} \sum_{x \in \mathcal{C}} f^{(i)}(x), \qquad (\sigma^{(i)})^2 := \frac{1}{N_{\text{cal}} - 1} \sum_{x \in \mathcal{C}} \big(f^{(i)}(x) - \mu^{(i)}\big)^2,$$

with a small floor $\sigma^{(i)} \leftarrow \max\{\sigma^{(i)}, \sigma_{\min}\}$ for numerical stability. This monotone normalization allows task selection to be evaluated in a scale-invariant manner.

Ground-truth values (for evaluation only). For plotting, we use the best utility found up to round t:

$$\text{simple-regret}(t) := U^\star - \max_{\tau \le t} u^{(i_\tau)}\Big(\overline{y}_{s_\tau^{(i_\tau)}}^{(i_\tau)}\Big).$$

This is why the curves in Fig. 9(b) are monotone nonincreasing.

Within-task Bayesian optimization (shared across all methods). Each objective i maintains its own GP surrogate fit to its observed data $\{(x_r, y_r^{(i)})\}_{r=1}^{s}$. Unless otherwise noted, we use the same BO stack for all objectives and all methods:

- **Surrogate:** `SingleTaskGP` (BoTorch/GPyTorch) with an Matérn-$5/2$ kernel, input normalization to $[0, 1]^d$, and `Standardize` outcome transform.
- **Noise:** synthetic objectives are treated as noiseless; we fit with a small jitter/noise floor.
- **Acquisition:** UCB-style acquisition (e.g., `qUpperConfidenceBound` with q=1).
- **Acqf optimization:** multi-start gradient-based optimization (fixed `raw_samples` and `num_restarts`) over $\mathcal{X}$.

When an objective i is selected for the first time, it is initialized with a small number of random points; these initialization evaluations are counted toward the global budget T (i.e., there is no "free" warm start).

Baselines. We compare against three families of baselines, each using the *same* within-task BO module above: Preference-based MOBO baselines (MOBO-PC / BOPE): we adapt these methods to the *decoupled-observation* setting where only one objective is observed per round. Concretely, we maintain separate GP surrogates for each objective, compute the MOBO acquisition using the current posterior over the *full* objective vector (integrating over unobserved objectives via GP posteriors), propose a candidate design x_t, and then choose which objective to evaluate at x_t using an uncertainty-driven rule (evaluate the objective with the largest posterior uncertainty at x_t). The resulting single observation is then used to update only that objective's GP.

Table 11. Default hyperparameters for the objective selection benchmark.

Item	Value / description
# objectives (tasks)	$K = 6$ BoTorch test functions
Domain	$\mathcal{X} = [0, 1]^6$
Global budget	$T = 200$
Utility map	$u^{(i)}(z) = \Phi((z - \mu^{(i)})/\sigma^{(i)})$
Within-task GP	Matérn-$5/2$, `SingleTaskGP` + `Standardize`

L. Additional Experiments and Discussion

L.1. Additional experiments

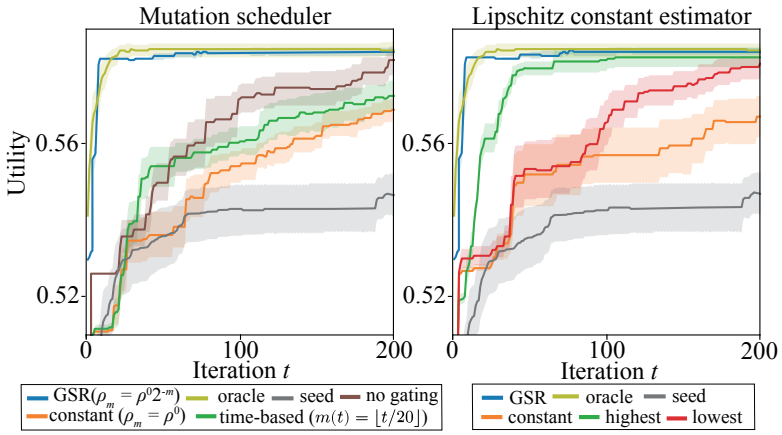

Figure 12. Ablation study. Replacing the mutation scheduler ρ_m or the Lipschitz constant adaptation causes a performance drop.

Lipschitz constant adaptation. We evaluate adaptation of the Lipschitz constant $\overline{L}$ via regret balancing (Appendix J). The method maintains candidates $L_j = L_0 2^j$ and discards them when falsified. We compare against three variants: (i) constant $L_j = L_0$; (ii) highest $\max_j L_j$, selecting the largest remaining candidate; (iii) lowest $\min_j L_j$, selecting the smallest. Adaptive selection performs best overall. Notably, the largest-candidate variant also performs well, as it guarantees $\overline{L} \geq L^\star$, supporting our theoretical analysis.

Mutation scheduler ρ_m. We further study the mutation scheduler $\rho_m = \rho_0 2^{-m}$ (Fig.12) by replacing it with three variants: (i) constant $\rho_m = \rho_0$; (ii) time-based scheduling, where m increases every 20 iterations and child tasks are generated when $w_t \leq c_g \epsilon_m^U$; and (iii) no gating, which ignores the w_t condition and updates m every 20 iterations. The results show that coupling ρ_m with the w_t-based generation rule is crucial for efficiency.

L.2. LLM-human evaluation alignment

To test whether the committee is merely exploiting idiosyncratic LLM preferences, we ran a blinded human–LLM alignment study on two representative scientific settings: synthesis scaling (SnAr) and patent repurposing (MgO). Two PhD-level materials scientists compared 100 randomly sampled task pairs using exactly the same task descriptions and partial optimization histories shown to the committee; the annotators were blinded to the committee votes. We measured agreement using Cohen's kappa, since it measures agreement beyond chance and avoids comparing raw vote counts across groups of different size (64 committee votes vs. 2 human votes).

As Table 12 shows, agreement between each human annotator and the committee majority vote is high. We interpret this as evidence that the committee is not obviously pathological under the shared-information protocol, while not claiming full validation of scientific judgment. GSR can also be instantiated with human feedback directly, at higher annotation cost; Appendix C.1.2 studies a feedback-efficient variant. For transparency, the anonymous repository now includes raw evaluator traces for SnAr, including the compared tasks, rationale, and vote.

Table 12. Human-LLM alignment using Cohen's kappa.

Voter	synthesis	repurposing
human A	0.898	0.926
human B	0.912	0.947

Table 13. LLM variants for task generation

alg.	utility	latency	API cost	average raw proposals per trigger
random	0.5602	N/A	N/A	N/A
SH	0.5646	N/A	N/A	N/A
hyperband	0.5524	N/A	N/A	N/A
gpt-4o-mini	0.5804	21 s	$2.00	6.6
gpt-5.4	0.6184	60 s	$40.00	4.8
gpt-5.4-mini	0.6079	28 s	$12.00	5.2
gpt-5.4-nano	0.5980	12 s	$3.27	5.5
gpt-5.4-nano-reasoning-low	0.6002	15 s	$3.60	5.3
gpt-5.4-nano-reasoning-high	0.6047	24 s	$4.90	5.2

Annotator instructions, interface, and compensation. The annotators were shown a spreadsheet-style interface with two anonymized task–incumbent summaries, labeled A and B, plus the same partial optimization history available to the LLM committee. They were given the following written instruction:

> For each row, compare Candidate A and Candidate B as scientific or product repurposing hypotheses. Use only the task description, current best candidate, and partial optimization history shown in the row. Select the candidate that you would prioritize for the next expensive evaluation. If both are close, choose the one that appears more scientifically plausible, safer, or more actionable. Record only A or B; optional comments may be added for quality control.

No personally identifying information was collected. The study used unpaid internal collaborator annotators; compensation was not paid. We follow the existing protocols in our institutions.

L.3. Computational cost

Model choice We chose GPT-4o-mini because this instantiation makes many structured generation and committee-evaluation calls, so both latency and cost matter. To test model sensitivity, we varied only the task-generation model on the red-wine benchmark while keeping the evaluator fixed at GPT-4o-mini (Table 13). We isolate generator sensitivity because changing the evaluator would change the utility definition itself, making runs no longer directly comparable. The committee uses 64 votes per step run in parallel; in this benchmark, that adds about 15 s wall-clock latency and about $3.81 total committee-evaluation cost over 100 steps. Task generation is triggered only intermittently (6.2 times per 100 steps on average); with GPT-4o-mini this contributes about 21 s latency per trigger and about $2.00 total generation cost over 100 steps.

In our current implementation, the task-generation prompt grows with the mutation history, so generation latency does increase over time. Empirically, we observe an approximately linear increase in generation latency (about +2.93 s per additional mutation history in the red-wine benchmark). The main context-growth cost is therefore in the generation call; the hard validation layer itself is relatively lightweight, since it mainly consists of structured parsing and rule-based feasibility checks on the current proposal. Also, because task generation is triggered only intermittently (6.2 times per 100 steps on average in our experiments), the overall wall-clock overhead remained manageable in our setting. For substantially longer runs, a natural extension would be to summarize or truncate early mutation history to cap prompt length.

Prompt effects. To test prompt sensitivity, we evaluated four semantically equivalent task-generation prompts on red-wine while keeping the evaluator model and evaluator prompt fixed. Mean utility changed only modestly (0.5798–0.5973 versus 0.5804 for the original prompt), and all variants remained above Random/SH/HB. For transparency, the anonymous repository `https://anonymous.4open.science/r/Generate-Select-Refine-2D48/` includes the task-generation

Table 14. Existing assets and licenses.

Asset	Use in this paper	License / terms
Wine Quality dataset (Cortez et al., 2009)	Wine planning benchmark	CC BY 4.0
SUMMIT simulator (Felton et al., 2021)	Synthesis scaling benchmark	MIT License
Materials Project data (Jain et al., 2013)	Materials repurposing benchmark	CC BY 4.0
CGCNN (Xie & Grossman, 2018)	Crystal-structure embeddings	MIT License
BoTorch (Balandat et al., 2020)	BO implementation	MIT License
GPyTorch (Gardner et al., 2018)	GP implementation	MIT License

prompts for all four variants.

L.4. Discussion

Why not preferential BO? Preferential BO (PBO; (González et al., 2017)) optimizes a preferred *input* x; GSR optimizes a preferred *task* i using LLMs. Our framework also supports non-preference utilities (see §B.2-B.3).
Can kernels vary across tasks? Yes, within kernel families with tractable information gain (see (Lee & Javidi, 2025)).
Why not use a single GP over $(x_\tau, \tilde{u}_\tau)_{\tau=1}^t$? Utility observations are nonlinear expectations, so standard GP assumptions do not apply (see (Astudillo & Frazier, 2019)).
Aren't LLM evaluations unreliable? Both the generator and the utility feedback can be replaced by humans.
LLMs/human feedback is costly. Utility queries can be sparse (e.g., at $\{1, 2, 4, 8, \dots\}$) while keeping $\mathcal{O}(\sqrt{\log T})$ regret with a sublinear number of votes (Appendix G.1.2).
What if small edits (ρ_m) cause large semantic shifts? We mitigate this with two safeguards: (i) an actor–critic generator, where a critic LLM verifies desiderata, and (ii) a rule-based verifier enforcing hard feasibility constraints (Appendix K.1); Fig.7 confirms their effectiveness.

L.5. Societal impact

This work could be especially useful for self-driving laboratories, where planning is increasingly becoming the bottleneck, and faster application discovery could accelerate the industrial adoption of new materials. On the other hand, greater autonomy also increases the risk of misalignment with human values. Our framework is explicitly controlled by the utility function, so users must carefully consider the consequences of how that utility is specified. This is a general risk shared by highly automated decision-making systems.

L.6. Asset licenses

Table 14 lists the existing assets used in the experiments and their licenses.