OUTCOME-RANKED PROPOSAL TUNING: TRAINING LLMS TO TRANSFER EXPERIENCE ACROSS BAYESIAN OPTIMIZATION TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian optimization (BO) is widely used for expensive black-box optimization. When related optimization problems arise repeatedly, experience from previous tasks can improve optimization on new ones. We aim to train a large language model (LLM) to transfer the knowledge from the prior tasks to new ones through BO initialization. We introduce **Outcome-Ranked Proposal Tuning (ORPT)**, which trains an LLM to propose initialization points based on how they affect the subsequent BO process. ORPT trains the LLM to favor initialization candidates based on their effect on subsequent BO, rather than their standalone objective values. We further mathematically show that these learned preferences are linearly connected to the expected outcome of downstream BO. Experiments on synthetic Branin optimization and antimicrobial peptide design show that ORPT outperforms a broad range of single-task, multi-task, and LLM-based BO baselines, yielding stronger initializations and better downstream optimization. These gains persist through subsequent BO, while ablations confirm that downstream BO outcomes provide a stronger training signal than candidate objective values alone. Overall, our results show that learned initialization is an effective mechanism for transferring experience across related optimization tasks.

1 INTRODUCTION

Bayesian optimization (BO) provides a sample-efficient framework for optimizing expensive black-box objectives under a limited evaluation budget. In many applications, related optimization problems are encountered repeatedly, and multi-task BO extends it to the subsequent BO scenarios with multiple relevant tasks by leveraging experience from previously optimized tasks (Swersky et al., 2013). We focus on a scalable regime in which related tasks share a search space and an increasing number of completed BO trajectories obtained from past completed tasks becomes available over time. Here, scalability refers to learning from an increasing number of completed related tasks.

Many multi-task BO methods transfer information through shared surrogates or representations learned across tasks (Perrone et al., 2018; Patacchiola et al., 2020; Wistuba & Grabocka, 2021b; Hakhamaneshi et al., 2021). Recent work has also explored language models as components of BO (Chen et al., 2022; Liu et al., 2024). Among these approaches, BOLT (Zeng et al., 2026) uses an LLM specifically for initialization. It trains a task-conditioned LLM proposal policy on high-performing candidates from completed BO trajectories. Given the context of a new task, the policy generates candidates that form its initialization pool, and the subsequent BO procedure is conducted. With sufficient training across completed BO tasks, BOLT learns to propose initialization points with strong standalone objective values (Zeng et al., 2026). However, BOLT trains its proposal policy using candidates selected by their standalone objective values. This leaves two questions open.

1. Does a candidate's standalone objective value reflect how much it improves downstream BO?
2. Can a better training signal be found to improve candidate-generation efficiency as completed-task experience grows?

The first question concerns the training signal itself. Downstream BO starts from the full initialization pool, so the value of a candidate depends not only on its own objective value but also on how

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

its inclusion affects the BO procedure that follows. The second question then asks whether a better training signal translates into more effective candidate generation on new tasks.

To address the first question, we introduce ***O**utcome-**R**anked LLM **P**roposal **T**uning* (ORPT). Rather than defining initialization quality through the standalone objective value of an individual candidate, ORPT evaluates it using a downstream BO rollout with horizon H. The resulting H-rollout BO utility measures the quality of the solution obtained after running BO for H acquisition rounds from an initialization pool, for example through the best objective value found. Because downstream BO starts from multiple initial observations, this utility is naturally defined on the full initialization pool.

Directly optimizing preferences between entire pools, however, would require the proposal policy to represent and score complete candidate sets rather than individual candidates. This would couple the generation interface to the initialization pool size used during training and increase the sequence length and memory footprint of each training example as the pool size grows. We therefore reduce the pool-level comparison to candidate-level supervision. Specifically, ORPT compares *matched initialization pools* that differ in exactly one candidate. Their rollout outcomes induce a preference between the differing candidates, which can be used to tune the task-conditioned LLM proposal policy with a direct preference objective (Rafailov et al., 2023). This pool-to-candidate reduction preserves the candidate-level generation interface and allows the tuned proposal policy to construct an initialization pool by repeatedly generating individual candidates.

The pool-to-candidate reduction is practical, but it raises an important theoretical question. Does optimizing preferences between individual candidates remain aligned with the pool-level H-rollout BO utility from which those preferences were derived? In Section 3.5, we show that candidate-level preference tuning is locally aligned with improving the expected pool-level H-rollout BO utility. This result connects the candidate-level training objective used by ORPT to the pool-level downstream BO objective that motivates it.

We address the second question empirically on synthetic Branin optimization and antimicrobial peptide design (Wan et al., 2024). Across both domains, ORPT generates stronger candidates than BOLT at matched generation budgets and produces stronger initialization pools as more completed-task experience becomes available. These gains generally carry into downstream BO, with the largest improvements on peptide design appearing at intermediate training scales. Finally, an $H = 0$ versus $H \geq 1$ ablation shows that BO rollout supervision improves both initialization and downstream optimization over training from standalone candidate values alone.

Our main contributions are as follows.

- We introduce ORPT, which uses downstream BO rollouts to train the proposal policy. By comparing matched initialization pools, ORPT converts rollout outcomes into candidate-level preferences for direct preference tuning.
- We theoretically justify this pool-to-candidate reduction. We show that candidate-level preference tuning is locally aligned with improving the expected pool-level H-rollout BO utility.
- We evaluate ORPT on synthetic Branin optimization and antimicrobial peptide design. ORPT generates stronger candidates with few generations and improves initialization as more training tasks become available. An $H = 0$ versus $H \geq 1$ ablation further demonstrates the benefit of BO rollout-based training over standalone candidate values.

2 Related Work

Bayesian optimization. Bayesian optimization (BO) is a black-box optimization method for optimizing when the objective function f is expensive. Rather than exhaustively exploring the search space, BO maintains a probabilistic surrogate model (e.g., Gaussian process (GP)), that approximates the relationship between candidate inputs and their observed objective values (Jones et al., 1998; Shahriari et al., 2015). The surrogate provides both a prediction of the objective and an estimate of predictive uncertainty. BO proceeds sequentially by repeating three main steps: (1) fitting or updating the surrogate model using the observations collected so far, (2) computing an acquisition function based on the surrogate posterior, and (3) observing the objective at the candidate selected by optimizing the acquisition function. Recently, Latent-space Bayesian optimization extends BO to structured or high-dimensional domains by optimizing over learned low-dimensional represen-

tations rather than directly in the original input space (Wilson et al., 2016; Maus et al., 2022a; Gómez-Bombarelli et al., 2018; Picard et al., 2024). Iteratively incorporating newly observed objective values into the surrogate, BO progressively refines its estimate of the objective landscape and directs evaluations toward promising candidates.

Scalable multi-task Bayesian optimization. Multi-task Bayesian Optimization (MTBO) extends BO to settings that multiple related tasks are available. It leverages observations from related tasks to accelerate optimization on a new task. Early approaches model cross-task correlations with multi-task Gaussian processes (Swersky et al., 2013), while later methods learn shared representations or transferable surrogate priors (Perrone et al., 2018; Wistuba & Grabocka, 2021a; Hakhamaneshi et al., 2021). There also exists methods utilizing multiple Gaussian process experts (Schilling et al., 2016), achieving per-task specialization through multiple experts. These approaches primarily encode cross-task information in the surrogate or its prior. In contrast, ORPT transfers information through a task-conditioned proposal policy while keeping the downstream BO procedure task-specific.

Language models for Bayesian optimization. Recent works have explored transformers, particularly large language models (LLMs), as flexible models for black-box optimization and BO (Chen et al., 2022; Ramos et al.; Zeng et al., 2026). OptFormer (Chen et al., 2022) learns from optimization trajectories, jointly modeling optimization policies and function values to predict promising evaluations. LLAMBO Liu et al. (2024) uses an LLM across multiple components of the BO pipeline, including warm-start initialization, candidate sampling, and surrogate modeling. Rather than training the LLM on optimization trajectories, LLAMBO prompts the model with the observed trajectory and uses its contextual predictions to guide subsequent evaluations. BOLT (Zeng et al., 2026) finetunes an LLM on high-performing solutions from completed BO trajectories and uses the resulting model to generate initialization points for unseen tasks. This shifts the role of the LLM to model the surrogate model toward transferring optimization information through learned initialization. ORPT follows this initialization-based paradigm but differs in its training signal: Rather than learning to directly propose high-performing solutions, ORPT learns an initialization policy that generates candidate sets leading to better final BO outcomes.

3 METHOD

We propose ***O**utcome **R**anked LLM **P**roposal **T**uning* (ORPT), which learns a task-conditioned LLM proposal policy for initializing Bayesian optimization. Rather than training the proposal policy only from the standalone objective values of individual candidates, ORPT uses downstream BO rollouts to evaluate initialization quality. At test time, the tuned LLM generates an initialization pool for an unseen task, after which a standard task-specific BO procedure proceeds unchanged.

To formalize this idea, we first define the *H-rollout BO utility* of an initialization pool from the best objective value observed after running downstream BO for H acquisition rounds. We then construct candidate preferences using *matched initialization pools* that differ in exactly one candidate, so that their utility difference provides a candidate-level preference signal. We use these preferences to tune the task-conditioned LLM proposal policy. Under an idealized population model, we then show that candidate-level preference tuning is locally aligned with improving the expected pool-level H-rollout BO utility. The full training procedure is given in Appendix A.

3.1 PROBLEM SETUP

We consider a family of related black-box optimization tasks

$$\{(C_t, f_t)\}_{t=1}^{T}, \qquad f_t : \mathcal{X} \to \mathbb{R},$$

defined over a shared search space $\mathcal{X}$, where C_t denotes the context that identifies task t. We treat all objectives as minimization problems and write f_C for the objective associated with a generic context C.

For each training task, we assume access to a completed BO trajectory

$$D_t = \{(x_{t,i}, y_{t,i})\}_{i=1}^{n_t}, \qquad y_{t,i} = f_t(x_{t,i}).$$

These completed BO trajectories provide evaluated candidates for learning a task-conditioned LLM proposal policy $\pi_\theta(x \mid C)$.

3.2 H-ROLLOUT BO UTILITY

To model the impact of an initialization on the optimization process, we define the H-rollout BO utility for horizons $H \geq 0$. For $H \geq 1$, downstream BO starts from an initialization pool, so its H-rollout BO utility depends on the pool rather than on an individual candidate in isolation. We therefore define the utility on an initialization pool for $H \geq 1$, while treating $H = 0$ as the no-BO special case.

For a candidate x under context C, the zero-rollout utility is its standalone objective value with the sign reversed:

$$U_0(C, x) = -f_C(x). \tag{1}$$

Thus, at $H = 0$, candidates are compared directly by their standalone objective values, without constructing an initialization pool or running downstream BO.

For $H \geq 1$, let $\mathcal{O}$ denote a fixed BO operator, including the surrogate model, its fitting procedure, the acquisition function, the acquisition optimizer, and the rule used to select the next evaluation. The operator is chosen to match the downstream BO procedure.

Given an initialization pool S for context C, define

$$D_0^{\mathcal{O}}(C, S) = \{(x, f_C(x)) : x \in S\}.$$

For $h = 1, \ldots, H$, one BO acquisition round produces

$$X^{(h)} = \mathcal{O}\big(D_{h-1}^{\mathcal{O}}(C, S)\big), \tag{2}$$

where $X^{(h)}$ contains the candidate or acquisition batch selected by the downstream BO procedure. We evaluate the selected candidates with the task oracle and form

$$D_h^{\mathcal{O}}(C, S) = D_{h-1}^{\mathcal{O}}(C, S) \cup \{(x, f_C(x)) : x \in X^{(h)}\}.$$

For a minimization task, the H-rollout BO utility for $H \geq 1$ is

$$U_H^{\mathcal{O}}(C, S) = -\min_{(x,y) \in D_H^{\mathcal{O}}(C,S)} y. \tag{3}$$

Higher utility therefore corresponds to a lower best objective among the initialization observations and the observations collected during the H BO acquisition rounds. In our experiments, we instantiate $\mathcal{O}$ as the GP-based BO procedure.

3.3 MATCHED CANDIDATE PREFERENCES FROM H-ROLLOUT BO

For $H \geq 1$, $U_H^{\mathcal{O}}$ evaluates an initialization pool rather than an individual candidate. Directly learning preferences over entire initialization pools would require the proposal policy to represent and score complete candidate sets. This would couple the learned generation interface to the pool size used during training and increase training cost as the pool size grows, since multiple candidates must be represented jointly in each preference example. We therefore reduce the pool-level comparison to candidate-level supervision by comparing *matched initialization pools* that differ in exactly one candidate while holding all other candidates fixed. Their difference in H-rollout BO utility provides a candidate-level preference signal.

For each task t, let $\mathcal{E}_t$ denote the eligible (e.g., deduplicated, feasible) candidates from its completed BO trajectory, and let q_t denote a distribution over $\mathcal{E}_t$ used to construct comparisons. Let m_{pref} denote the size of a matched initialization pool. For a candidate x, define its marginal H-rollout utility as

$$Q_{H,q_t}(C_t, x) = \mathbb{E}_A \left[U_H^{\mathcal{O}}(C_t, A \cup \{x\}) \right], \tag{4}$$

where A contains $m_{\text{pref}} - 1$ valid unique candidates sampled from q_t, excluding x. The exact construction of q_t is given in Appendix C.3.1.

For a candidate pair (x_i, x_j), we sample M shared base sets $A_1, \ldots, A_M$ and form

$$S_{r,i} = A_r \cup \{x_i\}, \qquad S_{r,j} = A_r \cup \{x_j\}.$$

We evaluate both matched initialization pools with the same downstream BO procedure and compute the average paired utility difference

$$\widehat{\Delta}_H(C_t, x_i, x_j) = \frac{1}{M} \sum_{r=1}^{M} \left[U_H^{\mathcal{O}}(C_t, S_{r,i}) - U_H^{\mathcal{O}}(C_t, S_{r,j}) \right]. \tag{5}$$

Reliable comparisons are retained, with the candidate having the larger estimated H-rollout BO utility labeled as preferred. The reliability criterion and full finite-sample construction are given in Appendix C.3. The shared base sets and BO rollouts are used only to construct preference labels and are not provided to the LLM.

3.4 Bayesian Optimization Rollout-Guided LLM Proposal Tuning (ORPT)

Given the candidate preferences constructed above, ORPT tunes the task-conditioned LLM proposal policy with a direct preference objective relative to a supervised reference proposal. We first train π_{ref} on the top K_{sft} candidates from each completed BO trajectory and then keep it fixed during preference optimization.

Define the reference-relative candidate score

$$v_\theta(C, x) = \log \pi_\theta(x \mid C) - \log \pi_{\mathrm{ref}}(x \mid C). \tag{6}$$

Assuming candidates within an initialization pool are generated independently by the proposal policy, the probability of a pool factorizes over its candidates. Because matched initialization pools differ in exactly one candidate, the shared candidates cancel from the reference-relative likelihood ratio between the two pools. The resulting comparison therefore depends only on $v_\theta(C, x_i) - v_\theta(C, x_j)$. The full cancellation is given in Appendix B.2.

Let $\mathcal{R}$ denote the resulting preference dataset of triples (C, x^+, x^-). We optimize

$$\mathcal{L}_{\mathrm{ORPT}}(\theta) = -\mathbb{E}_{(C,x^+,x^-)\sim\mathcal{R}} \Big[\log \sigma\big(\beta[v_\theta(C, x^+) - v_\theta(C, x^-)]\big) \Big], \tag{7}$$

where σ is the logistic sigmoid and $\beta > 0$ is the preference scale. This direct preference objective has the algebraic form of Direct Preference Optimization (Rafailov et al., 2023), but its preference labels are determined by matched H-rollout BO utility comparisons. The BO rollouts are used only to construct preference labels; no gradient is propagated through the downstream BO procedure.

3.5 Interpretation of ORPT as Marginal H-Rollout Utility Tuning

We now explain what the candidate-level preference objective encourages the proposal policy to learn, and how this relates to the quality of the full initialization pool. Recall that $Q_{H,q}(C, x)$ measures the expected H-rollout BO utility when candidate x is included in a pool whose remaining candidates are sampled from q.

Interpretation. Suppose that, for two candidates, the one with larger marginal H-rollout utility is more likely to be preferred. We model this with the standard pairwise logistic model

$$\Pr(x_i \succ x_j \mid C) = \sigma(\kappa \left[Q_{H,q}(C, x_i) - Q_{H,q}(C, x_j) \right]), \qquad \kappa > 0.$$

Under this model, the optimum of the preference objective has the form

$$\pi_{\theta^\star}(x \mid C) \propto \pi_{\mathrm{ref}}(x \mid C) \exp(\eta Q_{H,q}(C, x)), \qquad \eta > 0.$$

Thus, preference tuning increases the relative probability of candidates with larger marginal H-rollout utility.

We next ask whether this candidate-level reweighting also improves the utility of the full initialization pool. For a proposal distribution π, let

$$J_H(\pi; C) = \mathbb{E}_{x_{1:m_{\mathrm{pref}}} \sim \pi^{m_{\mathrm{pref}}}(\cdot|C)} \left[U_H^{\mathcal{O}}(C, \{x_1, \ldots, x_{m_{\mathrm{pref}}}\}) \right]$$

denote the expected H-rollout BO utility of a pool sampled from π. When $q = \pi_{\mathrm{ref}}(\cdot \mid C)$, consider moving slightly from the reference proposal toward candidates with larger marginal utility:

$$\pi_\lambda(x \mid C) \propto \pi_{\mathrm{ref}}(x \mid C) \exp(\lambda Q_{H,\pi_{\mathrm{ref}}}(C, x)).$$

At the reference proposal, $\lambda = 0$, this change satisfies

$$\frac{d}{d\lambda} J_H(\pi_\lambda; C)\Big|_{\lambda=0} = m_{\text{pref}} \ \text{Var}_{x\sim\pi_{\text{ref}}(\cdot|C)} \left[Q_{H,\pi_{\text{ref}}}(C, x)\right] \geq 0.$$

Therefore, increasing the probability of candidates with larger marginal rollout utility is locally aligned with improving the expected utility of a pool. The assumptions and full derivation are given in Appendix B.

Proof sketch. Suppose population preferences follow a Bradley–Terry model whose log-odds are proportional to differences in marginal H-rollout utility. At the population optimum of equation 7, the reference-relative score therefore satisfies, up to a context-dependent additive constant,

$$v_{\theta^\star}(C, x) = a(C) + \eta Q_{H,q}(C, x), \qquad \eta > 0.$$

Since $v_\theta(C, x) = \log \pi_\theta(x \mid C) - \log \pi_{\text{ref}}(x \mid C)$, the corresponding proposal has the exponential-tilt form above. Differentiating J_H along this tilt at $\lambda = 0$ gives $m_{\text{pref}} \ \text{Cov}\left(U_H^{\mathcal{O}}, Q_{H,\pi_{\text{ref}}}(C, x_1)\right)$. By the definition of marginal utility, $Q_{H,\pi_{\text{ref}}}(C, x_1) = \mathbb{E}[U_H^{\mathcal{O}} \mid x_1]$, so this covariance equals

$$\text{Var}[Q_{H,\pi_{\text{ref}}}(C, x_1)],$$

which yields the result above: $\frac{d}{d\lambda} J_H(\pi_\lambda; C)\big|_{\lambda=0} \geq 0$.

4 Experiments

4.1 Experimental Setup

Domains. We evaluate on two domains: synthetic function optimization based on the Branin function and antimicrobial peptide design. For the synthetic domain, following prior work (Volpp et al., 2019; Perrone et al., 2018), we construct related tasks by varying the Branin task parameter t. We use 50 training tasks and 20 held-out test tasks, and report the best Branin score (↑). For each synthetic training task, we collect a BO trajectory with an oracle budget of 50 evaluations. For peptide design, the objective is predicted minimum inhibitory concentration (MIC; ↓), subject to at least 0.75 sequence similarity to the reference peptide. We use 900 training tasks and a fixed 50-task held-out split. For each peptide training task, we collect a BO trajectory with an oracle budget of 20,000 evaluations. Full domain construction and implementation details are provided in Appendix C.

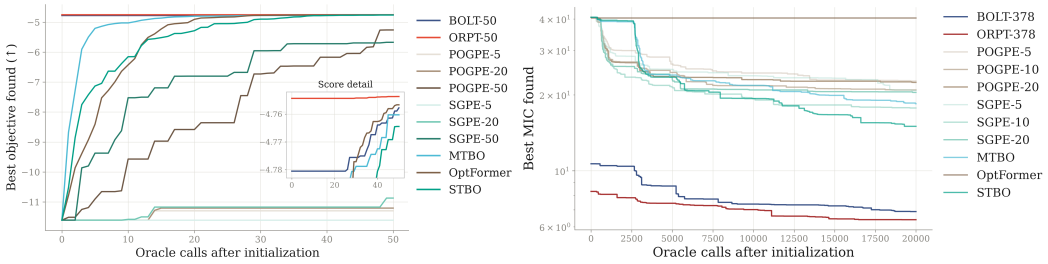
Methods and evaluation protocol. We compare ORPT with BOLT (Zeng et al., 2026), single-task BO (STBO) (Maus et al., 2022b), multi-task BO (MTBO) (Patacchiola et al., 2020; Perrone et al., 2018; Wistuba & Grabocka, 2021b), GP-expert baselines (POGPE and SGPE) (Schilling et al., 2016), OptFormer (Chen et al., 2022), and LLAMBO (Liu et al., 2024). For downstream BO evaluation, synthetic runs start from 5 evaluated candidates and use 50 additional oracle calls, while peptide runs start from 1,000 evaluated candidates and use 20,000 additional oracle calls. We report initialization quality and the best observed objective as downstream BO proceeds. Section 4.2 reports downstream BO performance, Section 4.3 studies performance across training-task checkpoints, and Section 4.4 evaluates proposal quality before downstream BO.

Training and baseline configuration. For methods trained across completed tasks, we use *training-task checkpoint* to denote the number of completed training tasks available for training. We evaluate checkpoints $\{2, 5, 10, 20, 30, 40, 50\}$ for the synthetic domain and $\{126, 252, 378, 630, 900\}$ for the peptide domain. BOLT, ORPT, and OptFormer use Qwen2.5-3B-Instruct (Qwen et al., 2024). BOLT and ORPT use LoRA rank 16 for synthetic tasks and rank 8 for peptide tasks. BOLT and ORPT initialize BO with task-conditioned LLM proposals, whereas the remaining BO methods use the common domain-specific initialization. For BOLT and ORPT, the generated initialization is followed by the same task-specific LOLBO (Maus et al., 2022b) procedure without further LLM queries. For peptide experiments, all BO-based methods use the same pretrained VAE representation used by BOLT (Zeng et al., 2026). MTBO uses a shared PPGPR (Parametric Predictive Gaussian Process Regressor) surrogate, while POGPE and SGPE use per-task PPGPR experts; STBO performs task-specific BO without cross-task training, and LLAMBO

Table 1: **Initialization and final BO performance on held-out synthetic and peptide tasks.** BOLT and ORPT use matched training checkpoints of 30 tasks for Branin and 378 tasks for peptide design. Values are mean ± standard deviation across held-out tasks. Higher is better for Branin; lower is better for peptide design.

		Synthetic: Branin ↑		Peptide Design ↓	
Category	**Method**	**Initialization**	**Final BO**	**Initialization**	**Final BO**
Single-task BO	STBO	-11.61±6.17	-4.76±2.95	40.59±38.47	15.00±14.51
Multi-task BO	MTBO	-11.61±6.17	-4.76±2.96	40.59±38.47	18.45±21.80
	POGPE	-11.61±6.17	-5.26±2.98	40.59±38.47	22.41±24.50
	SGPE	-11.61±6.17	-5.67±4.15	40.59±38.47	20.45±25.14
LM-based BO	OptFormer	-11.61±6.17	-4.76±2.96	40.59±38.47	40.33±37.97
	LLAMBO*	-11.61±6.17	-7.15±4.85	–	–
	BOLT	-4.78±2.94	-4.76±2.96	10.44±9.84	7.01±4.39
	ORPT	**-4.75**±2.95	**-4.75**±2.96	**8.73**±5.69	**6.58**±3.81

**LLAMBO is not reported for peptide because it reached the 10M-token budget before completing evaluation.*



(a) Synthetic: best Branin score (↑).

(b) Peptide: best MIC (↑).

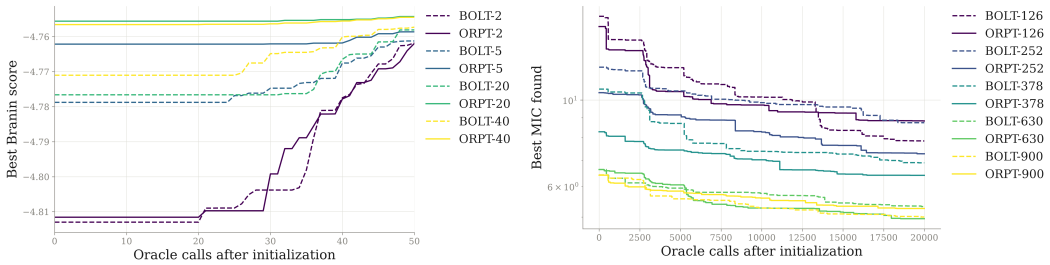
Figure 1: **Downstream BO performance on held-out tasks. (a):** synthetic Branin optimization, reporting the best objective found so far (higher is better). **(b):** antimicrobial peptide design, reporting the best predicted MIC found so far (lower is better). The x-axis counts oracle calls made after initialization. STBO, MTBO, POGPE, SGPE, OptFormer, and LLAMBO start from the common random initialization, whereas BOLT and ORPT initialize BO with task-conditioned LLM proposals. After initialization, BOLT and ORPT use the same task-specific BO procedure and do not query the LLM again.

uses the frozen Qwen2.5-3B-Instruct model. POGPE and SGPE additionally vary the number of pretrained experts, $n \in \{5, 10, 20\}$, independently of the training-task checkpoints. Unless otherwise noted, we construct preferences using one-rollout BO outcomes ($H = 1$) for synthetic Branin optimization and peptide design. Full training, preference-construction, and baseline implementation details are provided in Appendix C.

4.2 Downstream BO from Learned Proposals

Figure 1 shows the best observed objective as downstream BO proceeds on held-out tasks, and Table 1 summarizes performance at initialization and at the end of the BO budget. On the synthetic Branin tasks, ORPT already provides the strongest initialization. At initialization, ORPT achieves a best score of -4.75, compared with -4.78 for BOLT and -11.61 for methods using the common random initialization. Although the absolute differences among the strongest methods become numerically small in this score range, the zoomed view in Figure 1(a) shows a clear separation between ORPT and the baselines throughout downstream BO. Moreover, ORPT continues to improve slightly as additional oracle calls are made, reaching -4.75 at the end of the budget, compared with approximately -4.76 for BOLT, OptFormer, MTBO, and STBO. Thus, the advantage of ORPT is not limited to initialization: the initialization advantage remains throughout downstream BO.

On the peptide design tasks, ORPT shows a clear advantage at the 378-task checkpoint, both before and after downstream BO. It provides the strongest initialization among the compared methods, achieving a mean MIC of 8.73, compared with 10.44 for BOLT-378 and 40.59 for the common random initialization. This advantage persists through the full downstream BO budget: after 20,000



(a) Synthetic: best Branin score (↑).

(b) Peptide: best MIC (↑).

Figure 2: **Scaling with the number of completed training tasks. (a):** synthetic Branin optimization at training-task checkpoints $\{2, 5, 20, 40\}$, reporting the best objective found so far (higher is better). **(b):** antimicrobial peptide design at checkpoints $\{126, 252, 378, 630, 900\}$, reporting the best predicted MIC found so far (lower is better). Dashed and solid lines denote BOLT and ORPT, respectively, with colors indicating the training-task checkpoint; the x-axis counts oracle calls made after initialization.

additional oracle calls, ORPT-378 reaches a mean MIC of 6.58, outperforming BOLT-378 at 7.01 as well as STBO (15.00), MTBO (18.45), POGPE (22.41), and SGPE (20.45).

4.3 Scaling with the Number of Training Tasks

We next examine how BOLT and ORPT scales as the number of completed training tasks grows. Since the other baselines do not explicitly leverage completed tasks to learn an initialization policy, this scaling analysis focuses on BOLT and ORPT. Figure 2 compares the two methods at matched training-task checkpoints, using the same held-out tasks and downstream BO procedure at each checkpoint.

On the synthetic Branin tasks, Figure 2(a) shows that ORPT generally outperforms BOLT at matched training-task checkpoints. For each checkpoint, the ORPT trajectory remains above the corresponding BOLT trajectory over most of the downstream BO budget, particularly during the first 40 of 50 oracle calls. The gap narrows near the end of optimization as both methods approach similar high-scoring regions, indicating that ORPT primarily improves how quickly strong solutions are reached.

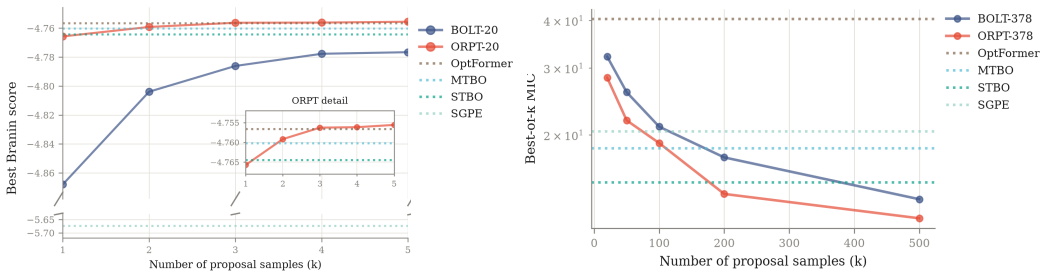
On the peptide tasks, Figure 2(b) demonstrates that ORPT achieves better initialization quality than BOLT at every evaluated training-task checkpoint. The downstream BO advantage is most pronounced at intermediate checkpoints at 252, 378, and 630 tasks, ORPT reaches final mean MIC values of 7.89, 6.58, and 5.55, compared with 8.39, 7.01, and 6.57 for BOLT. The largest separation occurs at the 630-task checkpoint. At the smallest and largest checkpoints, the two methods reach similar final BO performance, with BOLT slightly better at 126 and 900 tasks.

Overall, ORPT provides stronger learned initializations across most training-task checkpoints, while its downstream BO advantage is most evident at intermediate training scales.

4.4 Proposal Quality Before Downstream BO

In this section, we evaluate initialization proposal quality in isolation from downstream BO performance, focusing solely on the objective value of the initial solution. Figure 3 reports the best objective as the number of valid, unique proposals increases, using the maximum Branin score for the synthetic tasks and the minimum MIC for the peptide tasks. The resulting curves therefore measure the quality of the generated candidates without any BO refinement. For reference, the horizontal lines show the final performance of OptFormer, MTBO, STBO, and SGPE under the full BO budget, although these results are not compute-matched.

Across both domains, ORPT finds stronger candidates than BOLT at the same proposal budget. On Branin, ORPT reaches the narrow performance range of the full-budget BO baselines within only a few proposals ($\sim$5), whereas BOLT approaches the same range more gradually. On peptide design,



(a) Synthetic: best Branin score (↑). (b) Peptide: best MIC (↑).

Figure 3: **Proposal quality before downstream BO. (a):** synthetic Branin optimization, reporting the best objective among the first k valid unique LLM proposals (higher is better). **(b):** antimicrobial peptide design, reporting the best predicted MIC among the first k valid unique LLM proposals (lower is better). The x-axis is the proposal budget k, and no downstream BO steps are performed in this evaluation. BOLT and ORPT are evaluated by increasing the number of task-conditioned proposals on held-out tasks. Horizontal lines show the final outcomes of full-budget downstream BO baselines and are included only as reference points for proposal quality, not as matched-compute comparisons.

the difference is more pronounced: as the proposal budget increases, ORPT reaches and surpasses the full-budget MTBO, SGPE, and STBO reference levels with substantially fewer proposals than BOLT. i.e., by 200 proposals, ORPT already achieves a lower MIC than these full-budget BO baselines, while BOLT remains above the STBO reference at the same proposal budget. Overall, ORPT reaches strong objective values with fewer generated candidates than BOLT, providing a higher-quality initialization before downstream optimization begins.

4.5 Sensitivity to the BO Horizon H

We ablate the BO horizon H used to construct ORPT preferences. At $H = 0$, candidates are compared directly by their objective values, without running BO. Using H-rollout BO evaluation ($H \geq 1$) yields better held-out performance at the 50-task training checkpoint. Among the evaluated horizons, $H = 3$ achieves the highest mean initialization score (-4.7519, compared with -4.7707 for $H = 0$) and the highest mean score after 50 additional oracle calls (-4.7513, compared with -4.7593). These results suggest that constructing preferences from subsequent BO outcomes can provide a more useful training signal than comparing candidates by their standalone objective values alone.

Table 2: **Sensitivity to the BO horizon H on synthetic Branin tasks.** Results are reported at the 50-task training checkpoint over 20 held-out tasks. Values are mean best score $\pm$ standard deviation; higher is better. Final BO performance is measured after 50 additional oracle calls.

Horizon	Initialization	Final BO
$H = 0$	-4.7707 ± 2.95	-4.7593 ± 2.96
$H = 1$	-4.7543 ± 2.96	-4.7537 ± 2.96
$H = 2$	-4.7548 ± 2.96	-4.7540 ± 2.96
$H = 3$	$\mathbf{-4.7519} \pm 2.96$	$\mathbf{-4.7513} \pm 2.96$

5 Conclusion

We introduce ORPT, a framework for learning transferable initialization policies for families of related black-box optimization tasks. Rather than defining a good initialization solely by its immediate objective value, ORPT evaluates initializations by their downstream utility after subsequent BO steps and learns their relative quality through pairwise ranking. This directly aligns the learned proposal policy with the role of initialization in the overall optimization process.

Across synthetic and peptide design tasks, ORPT consistently provides strong initial solutions and improves final BO performance, with its advantage becoming more pronounced as additional training tasks become available. Our results suggest that completed optimization trajectories can be used not only to model promising regions of the search space, but also to learn how to initialize future optimization tasks more effectively.

AI USE STATEMENT

In this work, we used generative AI tools for providing suggestions on methods and experiments, formulate mathematical claims, assist in the writing of proofs, translation. We have not used generative AI tools to generate synthetic data sets, develop the core theoretical models or conceptual frameworks. All AI-assisted text was reviewed and revised by the authors, and AI-assisted code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have explained the general method at Sec. 3, including H-rollout BO utility, matched candidate preference construction, LLM proposal Tuning. We additionally provide the algorithm for overall process at Appendix A. We specified the experiment setup for both synthetic function optimization and peptide design at Sec. 4.1, including the number of training and held-out tasks, oracle budget. More detailed implementation details are provided at Appendix C, including baseline implementation, training and preference construction, BO evaluation, candidate sampling, and pool initialization evaluation.

REFERENCES

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Qiuyi Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'Aurelio Ranzato, Sagi Perel, and Nando de Freitas. Towards learning universal hyperparameter optimizers with transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Kourosh Hakhamaneshi, Pieter Abbeel, Vladimir Stojanovic, and Aditya Grover. Jumbo: Scalable multi-task bayesian optimization using offline data. *arXiv preprint arXiv:2106.00942*, 2021.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *International Conference on Learning Representations (ICLR)*, 2024.

Natalie Maus, Haydn Jones, Juston Moore, Matt J Kusner, John Bradshaw, and Jacob Gardner. Local latent space bayesian optimization over structured inputs. *Advances in neural information processing systems*, 35:34505–34518, 2022a.

Natalie Maus, Haydn Jones, Juston Moore, Matt J. Kusner, John Bradshaw, and Jacob Gardner. Local latent space Bayesian optimization over structured inputs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 34505–34518, 2022b. doi: 10.48550/arXiv.2201.11872.

Massimiliano Patacchiola, Jack Turner, Elliot J. Crowley, Michael O'Boyle, and Amos J. Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2020.

Valerio Perrone, Rodolphe Jenatton, Matthias W. Seeger, and Cedric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc., 2018.

Cyril Picard, Faez Ahmed, et al. Fast and accurate bayesian optimization with pre-trained transformers for constrained engineering problems. *arXiv preprint arXiv:2404.04495*, 2024.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

MC Ramos, SS Michtavy, MD Porosoff, and AD White. Bayesian optimization of catalysis with in-context learning. arxiv 2023, doi: 10.48550. *arXiv preprint arXiv.2304.05341*.

Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Scalable hyperparameter optimization with products of Gaussian process experts. In Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken (eds.), *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pp. 33–48, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46128-1.

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

Kevin Swersky, Jasper Snoek, and Ryan Adams. Multi-task bayesian optimization. *Advances in neural information processing systems (NeurIPS)*, 26, 2013.

Michael Volpp, Lukas P Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-learning acquisition functions for transfer learning in bayesian optimization. *arXiv preprint arXiv:1904.02642*, 2019.

Fangping Wan, Marcelo D. T. Torres, Jacqueline Peng, and Cesar de la Fuente-Nunez. Deep-learning-enabled antibiotic discovery through molecular de-extinction. *Nature Biomedical Engineering*, 8(7):854–871, 2024. doi: 10.1038/s41551-024-01201-x.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

Martin Wistuba and Josif Grabocka. Few-shot bayesian optimization with deep kernel surrogates. *arXiv preprint arXiv:2101.07667*, 2021a.

Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. *arXiv preprint arXiv:2101.07667*, 2021b.

Yimeng Zeng, Natalie Maus, Haydn Thomas Jones, Jeffrey Tao, Fangping Wan, Marcelo Der Torossian Torres, Cesar de la Fuente-Nunez, Ryan Marcus, Osbert Bastani, and Jacob R. Gardner. Scaling multi-task bayesian optimization with large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

A ALGORITHM

Algorithm 1 Outcome Ranked LLM Proposal Tuning (ORPT)

Require: Training trajectories $\{D_t\}_{t=1}^{T}$, contexts $\{C_t\}_{t=1}^{T}$, initial model π_0, supervised cutoff K_{sft}, preference-pool size m_{pref}, candidate pairs per task K, shared base sets per comparison M, horizon $H \geq 1$, BO operator $\mathcal{O}$, minimum utility difference δ_t, reliability threshold $z_{\min}$, numerical stabilizer ϵ, DPO scale β

Ensure: Tuned proposal policy π_θ

-tune π_0 on the top K_{sft} candidates from each completed training trajectory to obtain π_{ref}

Initialize $\mathcal{R} \leftarrow \emptyset$
for each training task t **do**
  Construct the eligible trajectory candidate set $\mathcal{E}_t$
  Construct the reference-weighted candidate distribution q_t
  for $k = 1, \ldots, K$ **do**
    Select a candidate pair (x_i, x_j) from $\mathcal{E}_t$
    for $r = 1, \ldots, M$ **do**
      Sample a shared base set A_r of $m_{\mathrm{pref}} - 1$ valid unique candidates from q_t, excluding x_i and x_j
      $S_{r,i} \leftarrow A_r \cup \{x_i\}$
      $S_{r,j} \leftarrow A_r \cup \{x_j\}$
      Evaluate $R_{r,i} \leftarrow U_H^{\mathcal{O}}(C_t, S_{r,i})$ using H BO acquisition rounds
      Evaluate $R_{r,j} \leftarrow U_H^{\mathcal{O}}(C_t, S_{r,j})$ using H BO acquisition rounds
      $d_r \leftarrow R_{r,i} - R_{r,j}$
    end for
    Compute $\widehat{\Delta}_H$ and $\widehat{\mathrm{SE}}(\widehat{\Delta}_H)$ using equation 18–equation 19
    if $|\widehat{\Delta}_H| \geq \delta_t$ and $\frac{|\widehat{\Delta}_H|}{\widehat{\mathrm{SE}}(\widehat{\Delta}_H)+\epsilon} \geq z_{\min}$ **then**
      if $\widehat{\Delta}_H > 0$ **then**
        $(x^+, x^-) \leftarrow (x_i, x_j)$
      else
        $(x^+, x^-) \leftarrow (x_j, x_i)$
      end if
      Add (C_t, x^+, x^-) to $\mathcal{R}$
    end if
  end for
end for
Initialize $\pi_\theta \leftarrow \pi_{\mathrm{ref}}$
Train π_θ on $\mathcal{R}$ by minimizing $\mathcal{L}_{\mathrm{ORPT}}$
return π_θ

B INTERPRETATION OF ORPT AS MARGINAL H-ROLLOUT UTILITY TUNING

This appendix expands the analysis in Section 3.5. We analyze an idealized population counterpart of the finite matched-pool preference construction. Fix a task context C and a connected comparison support on which $\pi_{\mathrm{ref}}(x \mid C) > 0$; all candidate distributions in this section are understood on this support. Unlike the finite construction, which samples valid unique shared candidates without replacement, the population analysis uses i.i.d. draws from a fixed candidate distribution q. This isolates the relation between candidate-level preference tuning and the expected pool-level H-rollout BO utility.

The analysis proceeds in four steps. First, we define the marginal H-rollout utility of a candidate under a fixed candidate distribution. Second, we show that, under factorized candidate generation, the shared candidates in matched initialization pools cancel from the pool-level reference-relative likelihood ratio. Third, under a Bradley–Terry population model, we derive the pairwise score relationship and the resulting exponential reweighting of the reference proposal according to marginal

H-rollout utility. Finally, when q is the reference proposal, we show that the utility-tilt direction underlying this reweighting is first-order aligned with improving the expected pool-level H-rollout BO utility.

B.1 Marginal H-Rollout Utility under a Fixed Candidate Distribution

Fix a task context C and preference-pool size m_{pref}. In the population analysis, a shared base set is formed from i.i.d. draws from a fixed candidate distribution q:

$$A = (x_1, \dots, x_{m_{\text{pref}}-1}) \sim q^{m_{\text{pref}}-1}.$$

The first-order alignment result below specializes to $q = \pi_{\text{ref}}(\cdot \mid C)$.

Let $\mathcal{O}$ be the fixed BO operator defined in Section 3.2. For candidate x, define

$$Q_{H,q}(C, x) = \mathbb{E}_{A \sim q^{m_{\text{pref}}-1}} \left[U_H^{\mathcal{O}}(C, A \cup \{x\}) \right]. \tag{8}$$

This quantity is not the standalone objective value of x. It is the expected H-rollout BO utility when x occupies one position in the initialization pool, the remaining $m_{\text{pref}} - 1$ candidates are drawn i.i.d. from q, and the BO operator executes H acquisition rounds.

For the following analysis, we assume that candidate preference probabilities follow a Bradley–Terry model,

$$\Pr(x_i \succ x_j \mid C) = \sigma\left(\kappa \left[Q_{H,q}(C, x_i) - Q_{H,q}(C, x_j)\right]\right), \qquad \kappa > 0. \tag{9}$$

The finite construction is the matched-pool analogue of this population quantity: it uses shared, pair-excluded base sets sampled without replacement and retains only sufficiently large and reliable empirical utility differences, as described in Appendix C.3.2. The derivation below concerns the i.i.d. population model above.

Positive reference probability. The reference proposal satisfies

$$\pi_{\text{ref}}(x \mid C) > 0$$

for every candidate that can occur in a comparison.

Model capacity. The proposal policy family can represent the required pairwise reference-relative candidate score differences.

Connected comparison support. The comparison graph is connected over the candidates under consideration, so that pairwise score differences determine candidate scores up to a context-dependent additive constant.

B.2 Matched-Pool Cancellation under a Factorized Pool Policy

Define the reference-relative candidate score

$$v_\theta(C, x) = \log \pi_\theta(x \mid C) - \log \pi_{\text{ref}}(x \mid C).$$

Under the factorized pool policy,

$$\Pi_\theta(A \cup \{x\} \mid C) = \left(\prod_{a \in A} \pi_\theta(a \mid C) \right) \pi_\theta(x \mid C).$$

Therefore,

$$\begin{aligned} &\log \Pi_\theta(A \cup \{x\} \mid C) - \log \Pi_{\text{ref}}(A \cup \{x\} \mid C) \\ &= \sum_{a \in A} v_\theta(C, a) + v_\theta(C, x). \end{aligned}$$

For two matched initialization pools that share the same base set A,

$$\begin{aligned} &[\log \Pi_\theta(A \cup \{x_i\} \mid C) - \log \Pi_{\text{ref}}(A \cup \{x_i\} \mid C)] \\ &\quad - [\log \Pi_\theta(A \cup \{x_j\} \mid C) - \log \Pi_{\text{ref}}(A \cup \{x_j\} \mid C)] \\ &= \left[\sum_{a \in A} v_\theta(C, a) + v_\theta(C, x_i) \right] - \left[\sum_{a \in A} v_\theta(C, a) + v_\theta(C, x_j) \right] \\ &= v_\theta(C, x_i) - v_\theta(C, x_j). \end{aligned} \tag{10}$$

Thus, all reference-relative score terms associated with the shared base set cancel exactly. Under the factorized pool policy, the pool-level comparison between matched initialization pools therefore reduces to the candidate-level score difference between the two candidates being compared.

B.3 Pairwise Score Relationship from Marginal H-Rollout Utility

Using the candidate score defined above, the preference probability predicted by the DPO model is

$$\Pr_\theta(x_i \succ x_j \mid C) = \sigma\left(\beta\left[v_\theta(C, x_i) - v_\theta(C, x_j)\right]\right). \tag{11}$$

For a fixed unordered pair (x_i, x_j), let

$$q_{ij} = \Pr(x_i \succ x_j \mid C).$$

The expected binary cross-entropy for this pair is

$$\ell_{ij}(p) = -q_{ij}\log p - (1 - q_{ij})\log(1 - p), \qquad p \in (0, 1).$$

Its derivative is

$$\frac{d\ell_{ij}}{dp} = -\frac{q_{ij}}{p} + \frac{1 - q_{ij}}{1 - p},$$

which vanishes only at $p = q_{ij}$. Strict convexity therefore gives the unique minimum at $p = q_{ij}$.

Under the model-capacity assumption, the population loss for the pair is minimized when

$$\sigma\left(\beta\left[v_{\theta^\star}(C, x_i) - v_{\theta^\star}(C, x_j)\right]\right) = \sigma\left(\kappa\left[Q_{H,q}(C, x_i) - Q_{H,q}(C, x_j)\right]\right).$$

Since the sigmoid is strictly monotone,

$$v_{\theta^\star}(C, x_i) - v_{\theta^\star}(C, x_j) = \eta\left[Q_{H,q}(C, x_i) - Q_{H,q}(C, x_j)\right], \qquad \eta = \frac{\kappa}{\beta} > 0. \tag{12}$$

Rearranging equation 12 gives

$$v_{\theta^\star}(C, x_i) - \eta Q_{H,q}(C, x_i) = v_{\theta^\star}(C, x_j) - \eta Q_{H,q}(C, x_j).$$

Because the comparison support is connected, this quantity must be constant across candidates. Hence, for some context-dependent scalar $a(C)$,

$$v_{\theta^\star}(C, x) = a(C) + \eta Q_{H,q}(C, x). \tag{13}$$

Thus, at the population optimum, the reference-relative candidate score is an affine function of marginal H-rollout BO utility.

B.4 Candidate-Level Exponential Reweighting

Substituting equation 13 into the definition of $v_{\theta^\star}$ gives

$$\log \pi_{\theta^\star}(x \mid C) = \log \pi_{\mathrm{ref}}(x \mid C) + a(C) + \eta Q_{H,q}(C, x).$$

Exponentiating,

$$\pi_{\theta^\star}(x \mid C) = \pi_{\mathrm{ref}}(x \mid C)\exp(a(C))\exp(\eta Q_{H,q}(C, x)).$$

Normalization determines $\exp(a(C))$, yielding

$$\pi_{\theta^\star}(x \mid C) = \frac{\pi_{\mathrm{ref}}(x \mid C)\exp(\eta Q_{H,q}(C, x))}{Z_\eta(C)}, \tag{14}$$

where

$$Z_\eta(C) = \mathbb{E}_{z \sim \pi_{\mathrm{ref}}(\cdot \mid C)}\left[\exp(\eta Q_{H,q}(C, z))\right].$$

Therefore, under the population assumptions above, the candidate-level preference objective reweights the reference proposal toward candidates with larger marginal H-rollout BO utility.

B.5 First-Order Alignment with Expected Pool-Level H-Rollout BO Utility

We now connect the candidate-level exponential reweighting derived above to the expected pool-level H-rollout BO utility. For this result, we specialize to

$$q = \pi_{\mathrm{ref}}(\cdot \mid C).$$

For a proposal π, define its expected pool-level H-rollout BO utility as

$$J_H(\pi; C) = \mathbb{E}_{x_{1:m_{\mathrm{pref}}} \sim \pi^{m_{\mathrm{pref}}}(\cdot \mid C)} \left[U_H^{\mathcal{O}} \left(C, \{x_1, \ldots, x_{m_{\mathrm{pref}}}\} \right) \right].$$

Throughout this subsection, $Q_{H,\pi_{\mathrm{ref}}}$ denotes the marginal utility in equation 8 with $q = \pi_{\mathrm{ref}}(\cdot \mid C)$.

To study the local direction underlying equation 14, introduce a separate reweighting parameter λ and define

$$\pi_\lambda(x \mid C) = \frac{\pi_{\mathrm{ref}}(x \mid C) \exp(\lambda Q_{H,\pi_{\mathrm{ref}}}(C, x))}{Z_\lambda(C)},$$

where

$$Z_\lambda(C) = \mathbb{E}_{z \sim \pi_{\mathrm{ref}}(\cdot \mid C)} \left[\exp(\lambda Q_{H,\pi_{\mathrm{ref}}}(C, z)) \right].$$

At $\lambda = 0$,

$$\pi_0(\cdot \mid C) = \pi_{\mathrm{ref}}(\cdot \mid C).$$

Moreover, in the reference case considered here, $\lambda = \eta$ recovers the population reweighting in equation 14.

Taking logs,

$$\log \pi_\lambda(x \mid C) = \log \pi_{\mathrm{ref}}(x \mid C) + \lambda Q_{H,\pi_{\mathrm{ref}}}(C, x) - \log Z_\lambda(C).$$

Therefore,

$$\frac{d}{d\lambda} \log \pi_\lambda(x \mid C) = Q_{H,\pi_{\mathrm{ref}}}(C, x) - \frac{d}{d\lambda} \log Z_\lambda(C).$$

Also,

$$\begin{aligned} \frac{d}{d\lambda} \log Z_\lambda(C) &= \frac{\frac{d}{d\lambda} Z_\lambda(C)}{Z_\lambda(C)} \\ &= \frac{\mathbb{E}_{z \sim \pi_{\mathrm{ref}}(\cdot \mid C)} \left[Q_{H,\pi_{\mathrm{ref}}}(C, z) \exp(\lambda Q_{H,\pi_{\mathrm{ref}}}(C, z)) \right]}{Z_\lambda(C)} \\ &= \mathbb{E}_{z \sim \pi_\lambda(\cdot \mid C)} \left[Q_{H,\pi_{\mathrm{ref}}}(C, z) \right]. \end{aligned}$$

Hence,

$$\frac{d}{d\lambda} \log \pi_\lambda(x \mid C) = Q_{H,\pi_{\mathrm{ref}}}(C, x) - \mathbb{E}_{z \sim \pi_\lambda(\cdot \mid C)} \left[Q_{H,\pi_{\mathrm{ref}}}(C, z) \right]. \tag{15}$$

We now differentiate $J_H(\pi_\lambda; C)$. Using the log-derivative identity,

$$\frac{d}{d\lambda} \pi_\lambda(x \mid C) = \pi_\lambda(x \mid C) \frac{d}{d\lambda} \log \pi_\lambda(x \mid C),$$

and applying it to the product distribution of the i.i.d. pool gives

$$\frac{d}{d\lambda} J_H(\pi_\lambda; C) = \mathbb{E}_{x_{1:m_{\mathrm{pref}}} \sim \pi_\lambda^{m_{\mathrm{pref}}}(\cdot \mid C)} \left[U_H^{\mathcal{O}} \left(C, \{x_1, \ldots, x_{m_{\mathrm{pref}}}\} \right) \cdot \sum_{i=1}^{m_{\mathrm{pref}}} \frac{d}{d\lambda} \log \pi_\lambda(x_i \mid C) \right].$$

Evaluating at $\lambda = 0$ and using equation 15,

$$\begin{aligned} \frac{d}{d\lambda} J_H(\pi_\lambda; C) \Big|_{\lambda=0} = \mathbb{E}_{x_{1:m_{\mathrm{pref}}} \sim \pi_{\mathrm{ref}}^{m_{\mathrm{pref}}}(\cdot \mid C)} \Bigg[& U_H^{\mathcal{O}} \left(C, \{x_1, \ldots, x_{m_{\mathrm{pref}}}\} \right) \\ & \cdot \sum_{i=1}^{m_{\mathrm{pref}}} \left(Q_{H,\pi_{\mathrm{ref}}}(C, x_i) - \mathbb{E}_{z \sim \pi_{\mathrm{ref}}(\cdot \mid C)} \left[Q_{H,\pi_{\mathrm{ref}}}(C, z) \right] \right) \Bigg]. \end{aligned}$$

Because the m_{pref} candidates are i.i.d. and $U_H^{\mathcal{O}}$ is permutation-invariant in the initialization pool, each summand has the same expectation. Therefore,

$$\frac{d}{d\lambda} J_H(\pi_\lambda; C)\Big|_{\lambda=0} = m_{\text{pref}}\, \mathbb{E}_{x_{1:m_{\text{pref}}} \sim \pi_{\text{ref}}^{m_{\text{pref}}}(\cdot|C)} \Big[U_H^{\mathcal{O}}\left(C, \{x_1, \ldots, x_{m_{\text{pref}}}\}\right) \cdot \left(Q_{H,\pi_{\text{ref}}}(C, x_1) - \mathbb{E}_{z \sim \pi_{\text{ref}}(\cdot|C)}\left[Q_{H,\pi_{\text{ref}}}(C, z)\right]\right) \Big].$$

By the definition of $Q_{H,\pi_{\text{ref}}}$,

$$\mathbb{E}_{x_{1:m_{\text{pref}}} \sim \pi_{\text{ref}}^{m_{\text{pref}}}(\cdot|C)} \left[U_H^{\mathcal{O}}\left(C, \{x_1, \ldots, x_{m_{\text{pref}}}\}\right) Q_{H,\pi_{\text{ref}}}(C, x_1) \right] = \mathbb{E}_{x_1 \sim \pi_{\text{ref}}(\cdot|C)} \left[Q_{H,\pi_{\text{ref}}}(C, x_1)^2 \right],$$

and

$$\mathbb{E}_{x_{1:m_{\text{pref}}} \sim \pi_{\text{ref}}^{m_{\text{pref}}}(\cdot|C)} \left[U_H^{\mathcal{O}}\left(C, \{x_1, \ldots, x_{m_{\text{pref}}}\}\right) \right] = \mathbb{E}_{x \sim \pi_{\text{ref}}(\cdot|C)} \left[Q_{H,\pi_{\text{ref}}}(C, x) \right].$$

Combining these identities gives

$$\begin{aligned} \frac{d}{d\lambda} J_H(\pi_\lambda; C)\Big|_{\lambda=0} &= m_{\text{pref}} \left(\mathbb{E}_{x \sim \pi_{\text{ref}}(\cdot|C)} \left[Q_{H,\pi_{\text{ref}}}(C, x)^2 \right] \right. \\ &\qquad \left. - \left(\mathbb{E}_{x \sim \pi_{\text{ref}}(\cdot|C)} \left[Q_{H,\pi_{\text{ref}}}(C, x) \right] \right)^2 \right) \\ &= m_{\text{pref}}\, \operatorname{Var}_{x \sim \pi_{\text{ref}}(\cdot|C)} \left[Q_{H,\pi_{\text{ref}}}(C, x) \right] \geq 0. \end{aligned} \tag{16}$$

Thus, the utility-tilt direction induced by marginal H-rollout utility is a first-order ascent direction for the expected pool-level H-rollout BO utility at the reference proposal. If $Q_{H,\pi_{\text{ref}}}(C, x)$ is nonconstant under the reference proposal, the derivative is strictly positive.

C Implementation Details

C.1 Domain and Evaluation Setup

Synthetic Branin tasks. We use the Branin function

$$f_t(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s,$$

following prior work (Volpp et al., 2019; Perrone et al., 2018). We set $a = 1$, $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $r = 6$, and $s = 10$, and vary the task parameter t across tasks. We use 50 training tasks and 20 held-out test tasks, with t evenly sampled from $[0, 1]$ for each split. For each training task, we collect a BO trajectory with a budget of 50 oracle evaluations. At evaluation time, downstream BO starts from 5 evaluated candidates and uses 50 additional oracle calls. Methods trained across completed tasks use training-task checkpoints $\{2, 5, 10, 20, 30, 40, 50\}$. Figure 2 displays a subset of these checkpoints.

Antimicrobial peptide design. Each task is defined by a reference peptide, and the objective is predicted minimum inhibitory concentration (MIC), with lower values indicating better antimicrobial activity. Candidates are required to have sequence similarity of at least 0.75 to the reference peptide. We use 900 training tasks and a fixed set of 50 held-out test tasks. For each training task, we collect a BO trajectory with a budget of 20,000 oracle evaluations. For held-out evaluation, downstream BO starts from 1,000 evaluated candidates and uses 20,000 additional oracle calls. Methods trained across completed tasks use training-task checkpoints $\{126, 252, 378, 630, 900\}$. BO-based peptide methods operate in the latent representation defined by the pretrained VAE checkpoint used by BOLT (Zeng et al., 2026).

C.2 ORPT Training and Preference Construction

BOLT and ORPT use Qwen2.5-3B-Instruct (Qwen et al., 2024). For synthetic Branin optimization, we use rank-16 LoRA, while for peptide design, we use rank-8 LoRA.

For ORPT, we first train the supervised reference proposal for 5 epochs using AdamW with a learning rate of 3×10^{-4}. The resulting model is frozen as the reference proposal π_{ref}. Starting from the same supervised checkpoint, we perform preference optimization for 1 epoch using AdamW with a learning rate of 2×10^{-5}. All remaining optimizer settings use the default AdamW configuration.

For the synthetic tasks, matched preference pools have size $m_{\text{pref}} = 5$, and each candidate pair is evaluated using $M = 3$ shared base sets. For peptide design, matched preference pools have size $m_{\text{pref}} = 20$, with $M = 8$ shared base sets per candidate pair. In both settings, we construct up to 30 candidate comparisons per training task and use $\beta = 0.25$ in the ORPT preference objective.

C.3 Matched H-Rollout BO Evaluation

This section describes the finite-data construction of the matched H-rollout BO preferences defined in Section 3.3. We specify how shared base sets are sampled from completed trajectories and how the resulting matched initialization pools are evaluated.

C.3.1 Reference-Weighted Candidate Sampling

The eligible candidate set $\mathcal{E}_t$ is defined as in Section 3.3. For each $x \in \mathcal{E}_t$, let

$$\ell_{\text{ref}}(C_t, x) = \log \pi_{\text{ref}}(x \mid C_t).$$

We instantiate the reference-weighted candidate distribution as

$$q_t(x) = \frac{\exp(\ell_{\text{ref}}(C_t, x)/\tau_q)}{\sum_{z \in \mathcal{E}_t} \exp(\ell_{\text{ref}}(C_t, z)/\tau_q)}. \tag{17}$$

Here, $\tau_q > 0$ controls how strongly sampling favors candidates with larger reference likelihood. For a candidate pair (x_i, x_j), each shared base set A_r is sampled without replacement from q_t after excluding x_i and x_j, so that all candidates in the resulting initialization pools are unique. We use $\tau_q = 1$ for synthetic Branin optimization and peptide design.

C.3.2 Evaluating Matched Initialization Pools

For each shared base set A_r, we evaluate the matched initialization pools $S_{r,i}$ and $S_{r,j}$ defined in Section 3.3 using the H-rollout BO utility $U_H^{\mathcal{O}}$ from Section 3.2. The objective values of candidates in the initialization pools are taken from the completed training trajectory. A surrogate is fit separately to each pool, and each BO run then executes H acquisition rounds.

For synthetic Branin optimization and peptide design, we instantiate $\mathcal{O}$ using the same LOLBO procedure used for downstream BO, including its surrogate fitting and acquisition optimization. Candidates selected during the H acquisition rounds are evaluated using the corresponding task oracle.

The same A_r is used on both sides of a comparison, so the two pools differ only in x_i versus x_j. For each shared base set, define the paired utility difference

$$d_r = U_H^{\mathcal{O}}(C_t, S_{r,i}) - U_H^{\mathcal{O}}(C_t, S_{r,j}).$$

The average paired utility difference is

$$\widehat{\Delta}_H(C_t, x_i, x_j) = \frac{1}{M} \sum_{r=1}^{M} d_r, \tag{18}$$

with standard error

$$\widehat{\mathrm{SE}}(\widehat{\Delta}_H) = \sqrt{\frac{1}{M(M-1)} \sum_{r=1}^{M} (d_r - \widehat{\Delta}_H)^2}. \tag{19}$$

We retain a comparison when

$$|\widehat{\Delta}_H| \geq \delta_t \quad \text{and} \quad \frac{|\widehat{\Delta}_H|}{\widehat{\mathrm{SE}}(\widehat{\Delta}_H) + \epsilon} \geq z_{\min}. \tag{20}$$

For synthetic Branin optimization and peptide design, we use $\delta_t = 0$, $z_{\min} = 1$, and $\epsilon = 0$. If the criterion holds, x_i is preferred to x_j when $\widehat{\Delta}_H > 0$, and the preference is reversed when $\widehat{\Delta}_H < 0$. The shared base sets and BO traces are used only to construct the preference label and are not provided to the LLM.

C.4 Baseline Implementation Details

We provide implementation details for the baselines compared in Section 4. The task splits, initialization procedures, training trajectories, oracle functions, evaluation budgets, and training-task checkpoints for synthetic Branin optimization and peptide design are described in Appendix C.1. Unless otherwise stated, we use the same baseline architectures and training hyperparameters across the two settings.

BOLT (Zeng et al., 2026). We implement BOLT using Qwen2.5-3B-Instruct (Qwen et al., 2024) with LoRA. We use rank-16 LoRA for synthetic Branin optimization and rank-8 LoRA for peptide design. At each training-task checkpoint, BOLT is fine-tuned on high-quality solutions collected from completed BO trajectories. We train the supervised proposal for 5 epochs using AdamW with a learning rate of 3×10^{-4} and otherwise use the default AdamW configuration. At test time, the fine-tuned model generates the initialization pool for the held-out task. The generated candidates are evaluated to construct the initial BO dataset, after which task-specific BO proceeds without further LLM queries.

STBO (Maus et al., 2022b). STBO performs task-specific BO without an LLM and without information transferred across tasks. We use the same LOLBO procedure used for downstream optimization in the corresponding domain. STBO starts from the domain-specific initialization described in Appendix C.1, and all surrogate fitting and BO updates use observations from the held-out task only.

MTBO (Patacchiola et al., 2020; Perrone et al., 2018; Wistuba & Grabocka, 2021b). MTBO transfers information across tasks through a shared surrogate rather than through an LLM proposal model. We use a shared PPGPR surrogate trained on observations from the completed training trajectories available at each training-task checkpoint. The surrogate is trained with a learning rate of 0.01 for 20 epochs using 1,024 inducing points. At test time, the learned surrogate parameters are transferred to the held-out task, and BO proceeds under the domain-specific initialization and evaluation protocol.

POGPE and SGPE (Schilling et al., 2016). POGPE and SGPE transfer information using ensembles of GP experts. Each pretrained expert is fit to data from a single completed training task using the same PPGPR architecture and fitting hyperparameters as MTBO. We evaluate $n \in \{5, 10, 20\}$ pretrained experts. The expert count is independent of the training-task checkpoints used for methods trained on increasing numbers of completed tasks. POGPE combines the pretrained experts with equal weights using a product of experts. SGPE additionally fits an expert to observations from the held-out task and assigns this target-task expert a weight equal to the sum of the weights of the pretrained experts.

OptFormer (Chen et al., 2022). We implement OptFormer using Qwen2.5-3B-Instruct with LoRA. At each training-task checkpoint, the model is fine-tuned for one epoch on history-conditioned windows constructed from the available optimization trajectories. Training uses a context length of 100 trials, 50 windows per task, and 100 discretized objective-value bins. At test time, OptFormer iteratively proposes a candidate conditioned on the recent optimization history, evaluates the proposal, and appends the resulting observation to its context before generating subsequent candidates.

LLAMBO (Liu et al., 2024). We use the frozen Qwen2.5-3B-Instruct checkpoint without task-specific fine-tuning. Following the end-to-end LLAMBO procedure, the LLM is used as an unconditioned proposer, a target-conditioned proposer, and an in-context surrogate. For the surrogate, we average $K = 10$ Monte Carlo predictions. Expected improvement is computed in a normalized $[0, 1]$ objective space with exploration parameter $\alpha = 0.1$. At each round, LLAMBO generates $m = 20$ candidates and evaluates one selected candidate. We cap generation at 10^7 input tokens.

Table 3: **Initialization and final BO performance across training-task checkpoints.** Each row reports matched synthetic/peptide checkpoints, shown as Synthetic / Peptide in the first column. Values are mean ± standard deviation across held-out tasks. Higher is better for Branin; lower is better for peptide design.

		Synthetic: Branin ↑		Peptide ↓	
Training-task checkpoint (Synthetic / Peptide)	**Method**	**Initialization**	**Final BO**	**Initialization**	**Final BO**
2 / 126	BOLT	**-4.81±2.93**	**-4.76±2.96**	20.42±27.74	**8.80±6.71**
	ORPT	-4.81±2.94	-4.76±2.96	**16.02±19.09**	9.33±8.58
10 / 252	BOLT	**-4.77±2.95**	**-4.76±2.95**	11.80±12.36	8.39±8.83
	ORPT	-4.77±2.95	-4.76±2.96	**11.51±9.83**	**7.89±5.11**
20 / 378	BOLT	-4.78±2.95	-4.76±2.95	10.44±9.84	7.01±4.39
	ORPT	**-4.76±2.96**	**-4.75±2.96**	**8.73±5.69**	**6.58±3.81**
30 / 630	BOLT	-4.78±2.95	-4.76±2.96	8.13±4.44	6.57±4.10
	ORPT	**-4.76±2.95**	**-4.75±2.96**	**7.25±3.74**	**5.55±3.48**
50 / 900	BOLT	-4.78±2.94	**-4.76±2.95**	7.39±5.32	**5.68±3.41**
	ORPT	**-4.75±2.96**	-4.76±2.96	**7.04±4.30**	5.81±4.01

C.5 Additional Results

C.5.1 Scaling with the Number of Training Tasks

Table 3 reports initialization and final BO performance across training-task checkpoints. In both synthetic function optimization and peptide design, the learned initializations generally improve as more completed training tasks become available, demonstrating the benefit of scaling task-level optimization experience. On the synthetic tasks, ORPT increasingly provides stronger initialization performance and achieves comparable or improved final BO performance relative to BOLT. The advantage is more pronounced in peptide design: ORPT consistently improves initialization quality across the evaluated checkpoints and achieves better final BO performance at most checkpoints. The largest gains appear at intermediate training scales, while the gap narrows at the largest checkpoint, where the two methods achieve similar performance. Overall, these results indicate that outcome-ranked proposal tuning effectively benefits from additional completed-task experience, with its gains most clearly reflected in the quality of the learned initialization.