Direct Regret Optimization in Bayesian Optimization

Fengxue Zhang
Department of Computer Science
University of Chicago
Chicago, IL 60637
zhangfx@uchicago.edu

Yuxin Chen
Department of Computer Science
University of Chicago
Chicago, IL 60637
chenyuxin@uchicago.edu

Abstract

Bayesian optimization (BO) is a powerful paradigm for optimizing expensive black-box functions. Traditional BO methods typically rely on separate hand-crafted acquisition functions and surrogate models for the underlying function, and often operate in a myopic manner. In this paper, we propose a novel *direct regret optimization* approach that jointly learns the optimal model and non-myopic acquisition by distilling from a set of candidate models and acquisitions, and explicitly targets minimizing the multi-step regret. Our framework leverages an *ensemble of Gaussian Processes (GPs)* with varying hyperparameters to generate simulated BO trajectories, each guided by an acquisition function chosen from a pool of conventional choices, until a *Bayesian early stop criterion* is met. These simulated trajectories, capturing multi-step exploration strategies, are used to train an end-to-end decision transformer that directly learns to select next query points aimed at improving the ultimate objective. We further adopt a *dense training–sparse learning* paradigm: The decision transformer is trained offline with abundant simulated data sampled from ensemble GPs and acquisitions, while a limited number of real evaluations refine the GPs online. Experimental results on synthetic and real-world benchmarks suggest that our method consistently outperforms BO baselines, achieving lower simple regret and demonstrating more robust exploration in high-dimensional or noisy settings.

1 Introduction

Bayesian optimization (BO) has emerged as a powerful framework for optimizing expensive black-box functions under limited evaluation budgets. In many applications, such as hyperparameter tuning in machine learning [3], engineering design [20], and robotic control [5], the objective function is expensive or time-consuming to evaluate. Therefore, classical optimization techniques that rely on dense sampling or gradient information become impractical. Instead, BO constructs a probabilistic surrogate model—commonly a Gaussian Process (GP)—over the objective and uses an *acquisition function* (e.g., Expected Improvement (EI), or Upper Confidence Bound (UCB)) to decide where to sample next. This strategy has proven effective in balancing exploration and exploitation, often achieving promising results with relatively few function evaluations [35].

Despite BO's success, traditional methods typically rely on separately hand-crafted acquisition functions and surrogate models (e.g., GPs, deep models) and often operate in a myopic manner. This separation and the heuristic nature of acquisition functions mean they only *indirectly* target the primary goal of minimizing the *simple regret*—the difference between the global maximum and the best solution found. Such an approach can be suboptimal for achieving optimal multi-step performance, especially in high-dimensional settings. Furthermore, tuning these components and their hyperparameters for complex problems can be non-trivial and may fail to capture intricate objective structures, thereby hindering overall optimization performance.

First Exploration in AI Today Workshop at ICML (EXAIT at ICML 2025).

Direct Regret Optimization. Motivated by these challenges, we propose *Direct Regret Optimization (DRO)*[1], a novel approach that **jointly learns an optimal decision-making model and a non-myopic acquisition policy**, explicitly targeting the minimization of multi-step regret (Fig. 1). Our framework achieves this by **distilling knowledge from a diverse set of candidate models and acquisition strategies**. Specifically, DRO leverages an *ensemble of GPs* with varying hyperparameters to generate simulated BO trajectories. Each trajectory, guided by an acquisition function from a pool of conventional choices, is strategically constrained within an identified *region of interest (ROI)*—a subset of the domain likely to contain the optimum—and is managed by an early-stopping criterion. These simulated trajectories, capturing rich multi-step exploration strategies, are then used to train a *decision transformer* end-to-end. This transformer directly learns to select the next queries to improve the objective, effectively embodying the learned non-myopic acquisition policy. This paradigm shifts BO from relying on separately designed, often myopic heuristics to an integrated sequence modeling approach that directly optimizes for long-term outcomes.

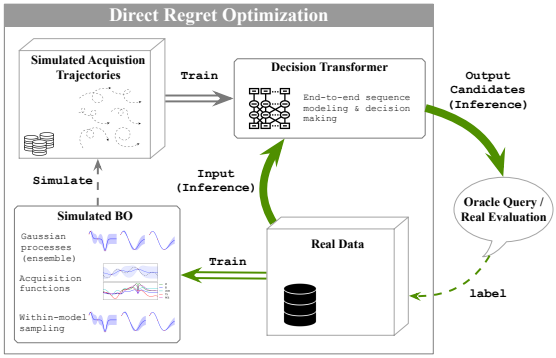


Figure 1: The DRO framework. Gray arrows indicates processes involving simulated data, while green arrows correspond to real data. The dense training (gray) and sparse learning (green) steps are depicted in hollow arrows.

Dense Training and Sparse Learning. A critical aspect of our framework is a novel learning paradigm which we refer to as *dense training–sparse learning*. *Dense training* refers to the offline phase where we collect abundant simulated trajectory data from the GP ensemble. The decision transformer is then trained extensively in this synthetic environment, learning to anticipate multi-step outcomes without incurring the high cost of real evaluations. Upon deployment, the algorithm shifts to *sparse learning* where each actual function evaluation provides limited but crucial real-world feedback. This feedback anchors the learned policy in reality and allows for online refinement, addressing potential model mismatch from simulation and enhancing robustness.

Contributions. Our primary contributions include the introduction of a **direct regret optimization** framework that jointly learns a decision-making model and a non-myopic acquisition policy by distilling from candidate strategies, explicitly minimizing multi-step regret. We also propose a novel use of **ensemble-based rollout simulations**, constrained within a **Region of Interest (ROI)** and guided by early stopping, to generate rich trajectory data for training a *decision transformer* end-to-end. Furthermore, we present a novel **dense training–sparse learning** paradigm that enables robust multi-step planning using offline simulated data, while efficiently incorporating limited real evaluations for online refinement. Finally, we conduct experiments on a range of synthetic and real-world benchmarks, demonstrating that our method consistently outperforms traditional Bayesian optimization baselines in terms of final simple regret and robust exploration in high-dimensional or noisy settings.

2 Related Work

Recent Bayesian Optimization (BO) advancements leverage machine learning: reinforcement learning (RL) offers adaptive, non-myopic sampling policies (e.g., EARL-BO [10]), while deep learning, particularly transformers, enables flexible surrogates, amortized inference (PFNs4BO [29]), and universal optimizers (OptFormer [7]), though these methods often build on specific modeling assumptions or extensive meta-training for generalization. A distinct challenge, surrogate hyperparameter sensitivity in BO, is commonly tackled via fully Bayesian techniques, ensembling, or implicitly by meta-trained models like PFNs. Our work contrasts with these by using a decision transformer to

[1] We note that the acronym DRO is also commonly used for Distributionally Robust Optimization, an unrelated field.

directly learn a non-myopic, sequential query policy from problem-specific simulated BO rollouts. This policy is optimized to minimize ultimate regret without relying on prior meta-data or aiming for universal applicability. We address hyperparameter sensitivity by using an ensemble of Gaussian Processes with varied settings to generate diverse simulation trajectories, thereby embedding robustness into the data that trains our decision-making policy.

3 Problem Formulation

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a d-dimensional design (or input) space, and let $f : \mathcal{X} \to \mathbb{R}$ be an unknown objective function that is typically expensive to evaluate. The goal of *Bayesian optimization* is to find a design $\mathbf{x}^* \in \mathcal{X}$ that *maximizes* this objective, i.e., $\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\ f(\mathbf{x})$.

The conventional Bayesian optimization process is iterative. At each iteration t, a probabilistic surrogate model, commonly a Gaussian process, is updated based on the accumulated data $\mathcal{D}_{t-1} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{t-1}$, where $y_i = f(\mathbf{x}_i) + \epsilon_i$ are (potentially noisy) observations with $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ representing i.i.d. Gaussian noise. This Gaussian process yields a posterior distribution $p(f(\mathbf{x}) \mid \mathcal{D}_{t-1})$ for any point $\mathbf{x} \in \mathcal{X}$, characterized by its mean $\mu_{t-1}(\mathbf{x})$ and variance $\sigma_{t-1}^2(\mathbf{x})$ (with standard deviation $\sigma_{t-1}(\mathbf{x})$). An acquisition function $\alpha(\mathbf{x}; \mathcal{D}_{t-1})$, leveraging $\mu_{t-1}(\mathbf{x})$ and $\sigma_{t-1}(\mathbf{x})$, is then maximized to select the next point $\mathbf{x}_t = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\ \alpha(\mathbf{x}; \mathcal{D}_{t-1})$. The objective function is subsequently evaluated at $\mathbf{x}_t$ to obtain y_t, and the dataset is augmented, $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$, with this cycle repeating for a total of T iterations within a given budget.

Objective and Regret. The primary objective is to identify a near-optimal point $\mathbf{x}^*$ efficiently. Performance is often measured by how quickly the quality of the recommended point improves. Let $f^* = \max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ be the true maximum value of the objective function. After T evaluations, let $\mathbf{x}_T^+$ be the point among $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ that has yielded the highest true function value, i.e., $f(\mathbf{x}_T^+) = \max_{1\le t\le T} f(\mathbf{x}_t)$. The *simple regret* at iteration T is then defined as $\mathrm{Regret}_{\text{simple}}(T) = f^* - f(\mathbf{x}_T^+)$. Alternatively, the *cumulative regret* considers the sum of regrets at each step: $\mathrm{Regret}_{\text{cumulative}}(T) = \sum_{t=1}^{T}\big(f^* - f(\mathbf{x}_t)\big)$. Bayesian optimization aims to minimize these regrets by strategically balancing *exploration* (sampling in uncertain regions to improve the global model) and *exploitation* (sampling near current promising values to refine the optimum). However, the standard approach of greedily maximizing a myopic (one-step lookahead) acquisition function may not optimally reduce the regret over the entire optimization horizon, particularly when considering multi-step interactions and the final simple regret.

4 Method

Direct Regret Optimization (DRO) directly targets minimizing final optimization regret, diverging from conventional BO. Its core strategy simulates diverse BO trajectories using an ensemble of GP models with varying characteristics. These simulations, constrained within an adaptively identified Region of Interest (ROI), train a decision transformer to propose points predicted to yield the lowest simple regret at the optimization horizon's end.

4.1 Ensemble of Gaussian Processes

At each optimization step t, DRO maintains an ensemble of M GP surrogate models, denoted as $\{\mathrm{GP}_1, \mathrm{GP}_2, \ldots, \mathrm{GP}_M\}$. Each GP_m has distinct hyperparameters (e.g., kernel parameters, variances), chosen via methods like predefined grids or sampling. All ensemble GPs are conditioned on the shared historical dataset $\mathcal{D}_{t-1} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{t-1}$, where y_i is $f(\mathbf{x}_i)$ possibly with noise. This captures diverse beliefs about the objective function.

4.2 Adaptive Region of Interest (ROI) Filtering

To focus computations on promising search space areas, DRO uses an adaptive ROI filtering mechanism, inspired by methods in [12, 13, 43, 46, 26]. Each step t, after GP updates, GP_m

identifies its ROI $\hat{\mathcal{X}}_{m,t} \subseteq \mathcal{X}$ to guide its rollouts based on $\mathcal{D}_{t-1}$. For each GP_m, providing a posterior over f conditioned on $\mathcal{D}_{t-1}$, we compute its Upper Confidence Bound (UCB) as $\mathrm{UCB}_{m,t}(\mathbf{x}) = \mu_{m,t-1}(\mathbf{x}) + \beta_t^{1/2}\sigma_{m,t-1}(\mathbf{x})$, and its Lower Confidence Bound (LCB) as $\mathrm{LCB}_{m,t}(\mathbf{x}) = \mu_{m,t-1}(\mathbf{x}) - \beta_t^{1/2}\sigma_{m,t-1}(\mathbf{x})$. Here, $\mu_{m,t-1}(\mathbf{x})$ and $\sigma_{m,t-1}(\mathbf{x})$ are the posterior mean and standard deviation from GP_m given $\mathcal{D}_{t-1}$, and β_t is an exploration-exploitation trade-off parameter. The ROI for GP_m, $\hat{\mathcal{X}}_{m,t}$, is then defined as $\hat{\mathcal{X}}_{m,t} = \{\mathbf{x} \in \mathcal{X} \mid \mathrm{UCB}_{m,t}(\mathbf{x}) \geq \max_{\mathbf{x}' \in \mathcal{X}} \mathrm{LCB}_{m,t}(\mathbf{x}')\}$. This dynamic ROI, $\hat{\mathcal{X}}_{m,t}$, constrains rollout simulations for GP_m, concentrating its simulated efforts on regions with high confidence of containing the global optimum.

4.3 Within-Model Sampling (Rollout Generation) within ROI

Each GP_m generates rollouts by iteratively selecting points within its ROI $\hat{\mathcal{X}}_{m,t}$ using a conventional acquisition function $\alpha(\cdot; \mathrm{GP}_m)$ (e.g., EI, UCB) on its posterior $p_m(f \mid \mathcal{D}_{t-1})$. Simulated observations are drawn from GP_m's posterior predictive distribution. This continues until a Bayesian early stop (§4.4). This yields diverse, ROI-focused simulated trajectories. An ensemble of acquisition functions can add further diversity. Formally, at simulation step τ, the point $\mathbf{x}_\tau^{(m)}$ is selected as $\mathbf{x}_\tau^{(m)} = \underset{\mathbf{x} \in \hat{\mathcal{X}}_{m,t}}{\operatorname{argmax}}\, \alpha\big(\mathbf{x}; \mathrm{GP}_m, \mathcal{D}_{t-1}^{(\tau,m)}\big)$, where $\mathcal{D}_{t-1}^{(\tau,m)}$ is data accumulated up to simulation step $\tau - 1$ for that rollout under GP_m.

4.4 Bayesian Early Stop

A Bayesian early stop (BES) mechanism ensures computational tractability and focuses rollouts on informative sequences. After each simulated query τ for GP_m, a stopping criterion, such as one based on Expected Improvement (EI), is evaluated. Let $f_{m,\text{best}}^{(\tau)}$ be the best simulated value found by GP_m up to simulation step τ. The rollout terminates if $\max_{\mathbf{x} \in \hat{\mathcal{X}}_{m,t}} \mathbb{E}_{p_m(f(\mathbf{x}) \mid \mathcal{D}_{t-1}^{(\tau,m)})}\big[\max(0, f(\mathbf{x}) - f_{m,\text{best}}^{(\tau)})\big] < \delta$. The threshold δ (constant or dynamic) prevents overly long rollouts into regions of predicted negligible improvement, ensuring simulations capture plausible scenarios within the ROI.

4.5 Training and Inference of Decision Transformer

Simulated trajectories train a decision transformer [6]. Trajectories become (state, action, return-to-go) tuples: states cover GP HPs and history; actions are ROI-selected points; returns are simulated simple regret. The transformer learns to predict actions for desired returns. In BO, it proposes $\mathbf{x}_t$ given the real state and target regret (derived from known optima or simulation-based estimates).

4.6 Overall DRO Procedure

DRO's iterative cycle (Algorithm 1): 1) Update GP ensemble with $\mathcal{D}_{t-1}$. 2) Each GP_m identifies its ROI $\hat{\mathcal{X}}_{m,t}$. 3) Simulate BO trajectories per GP_m within its ROI using acquisition functions and early stopping. 4) Train/fine-tune decision transformer with these trajectories to minimize final regret. 5) Transformer proposes $\mathbf{x}_t$. 6) Query $y_t = f(\mathbf{x}_t)$, augment $\mathcal{D}_t$. Repeat for T evaluations.

Training Objective for DRO. The decision transformer aims to minimize final simple regret $R_{T_{real}}$. Unlike one-step proxies, DRO uses simulated multi-step trajectories (in ROIs) and a learned policy to *explicitly* target low final regret. This synthesis of GP ensemble diversity, ROI filtering, rollouts, early stopping, and transformer modeling targets robust optimization, validated in §5.

Dense Training vs. Sparse Learning. DRO leverages **dense training** on simulated data and **sparse learning** from online evaluations. *Dense training*: The transformer learns from numerous offline-generated GP ensemble trajectories within ROIs, internalizing strategies for low final simple regret from these "imagined" experiences. *Sparse learning*: Limited true evaluations $f(\mathbf{x}_t)$ update the GP ensemble, ground the model, refine ROIs, and can fine-tune the transformer.

Advantages of the Dense–Sparse Framework. This ROI-augmented dual approach offers: 1) Enhanced sample efficiency by learning *non-myopic behaviors* from inexpensive, ROI-focused sim-

Algorithm 1 Direct Regret Optimization with ROI Filtering

Require: Initial dataset $\mathcal{D}_0$, ensemble size M, max real iterations T_{real}, early stop threshold δ, num_rollouts_per_gp K, ROI parameters (e.g., β_t)

for $t = 1$ to T_{real} **do**
  Fit/Update GPs: Update GP_m for $m = 1 \dots M$ using $\mathcal{D}_{t-1}$.
  Identify ROI: Determine $\hat{\mathcal{X}}_{m,t} \leftarrow \{\mathbf{x} \in \mathcal{X} \mid \mathrm{UCB}_{m,t}(\mathbf{x}) \geq \max_{\mathbf{x}' \in \mathcal{X}} \mathrm{LCB}_{m,t}(\mathbf{x}')\}$ using GP_m.
  Initialize simulation buffer $\mathcal{B}_{sim} \leftarrow \{\}$.
  for $m = 1$ to M **do**
    for $k = 1$ to K **do**
      Simulate rollout $\mathcal{T}_{m,k} = \{(\mathbf{s}_\tau, \mathbf{a}_\tau, R_\tau)\}_{\tau=0}^{L_{m,k}-1}$ using GP_m,
      with acquisition α maximized over $\hat{\mathcal{X}}_{m,t}$, and Bayesian early stopping.
      Add $\mathcal{T}_{m,k}$ to $\mathcal{B}_{sim}$.
    end for
  end for
  Train/Update Decision Transformer using trajectories in $\mathcal{B}_{sim}$ to predict actions that minimize final regret.
  Infer Next Candidate $\mathbf{x}_t \leftarrow \mathrm{DecisionTransformer}(\text{current real state } \mathbf{s}_t, \text{target return } R_{\text{target}})$.
  Evaluate $y_t = f(\mathbf{x}_t)$ and augment real data $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$.
end for

ulations. 2) Effective *non-myopic exploration* in ROIs via dense simulation data. 3) *Robustness* to model misspecification via GP ensemble rollouts and ROI focusing. DRO's dense-sparse paradigm with ROI filtering thus learns an end-to-end, non-myopic policy for minimal final simple regret.

4.7 Theoretical Justifications

Full theoretical analysis of DRO is future work. Here, we provide insights into key components like the ROI mechanism and EI behavior as regret converges (proofs in Appendix B).

Regret Guarantees with ROI-Constrained Base Acquisition. Constraining a base acquisition function with known cumulative regret guarantees to the adaptive ROI (defined using the UCB-LCB criterion as detailed in Proposition 1, cf. [12, 13, 26, 43]) should largely preserve these guarantees. Performance scales with ROI traits, with a small penalty if the optimum is missed. ROI filtering can thus boost efficiency without sacrificing theoretical soundness of the base strategy.

Convergence of Expected Improvement with Converging Simple Regret. If simple regret converges to zero with high probability, maximum EI across the search space should also converge to zero. As the optimum is approached, potential gain diminishes, reflected by EI with a consistent GP model. This supports using EI-based early stopping in simulated rollouts.

5 Experiments

We evaluate DRO against established baselines and conduct ablation studies on its key components. This section details the benchmarks, comparison algorithms, and experimental setup, with results and discussions following in §6.

5.1 Experimental Setup

Benchmarks. Our evaluation uses synthetic functions, Hyperparameter Optimization (HPO), and complex simulation. The Ackley Function [14] (2D-20D, Fig. 2) tests scalability (output shifted $+10$, 0.1 std Gaussian noise); Ackley 10D is for ablations (Fig. 4). HPO tasks include XGBoost (from YAHPO Gym [33]) and ADAM optimizer tuning for Neural Networks on UCI datasets (Bayesmark [29] setups, Fig. 3). The *LunarLander-v3* [40] control problem serves as the complex simulation (Fig. 4).

General Setup. Inputs are normalized to $[0, 1]^d$. Runs initialize with $N_{init} = 5$ Sobol points. Evaluation budgets T are 30-40 (HPO), up to 200 (LunarLander), and 500 (Ackley). Performance, averaged over $\geq$10 trials, is simple regret (HPO) or best observed value (LunarLander, Ackley); plot shadings show ± 1 standard error. Implementations use BoTorch [2].

DRO Configuration. Default DRO uses an ensemble of $M = 10$ GPs with **RBF kernels**, ROI filtering, and Bayesian early stopping. The decision transformer (128 embedding dimension, 4 attention heads, 2-4 layers) is Adam-trained. For ROI ablation, 'DRO ROI' is this default; 'DRO GLOBAL' disables ROI filtering for rollouts.

Baselines. For HPO and LunarLander tasks, DRO is compared against: standard GP-BO with logEI (labeled "BO") [1]; TuRBO [12], a SOTA trust-region local BO method; PFNs4BO [29], using a transformer surrogate; and SCoreBO [19], which addresses model misspecification by incorporating hyperparameter uncertainty in acquisition.

6 Results and Discussion

We now present the empirical performance of DRO against baselines and discuss insights from its ablation studies and dimensionality scaling.

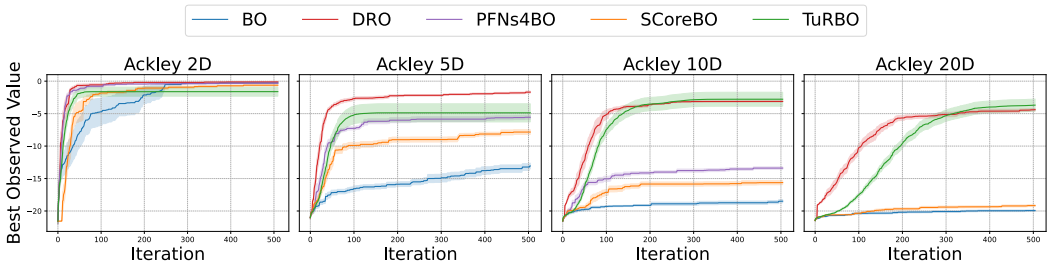


Figure 2: Performance on Ackley Function across dimensions (2D, 5D, 10D, 20D - Best Objective Value Found). Higher values are better. PFNs4BO is omitted from Ackley 20D due to scalability.

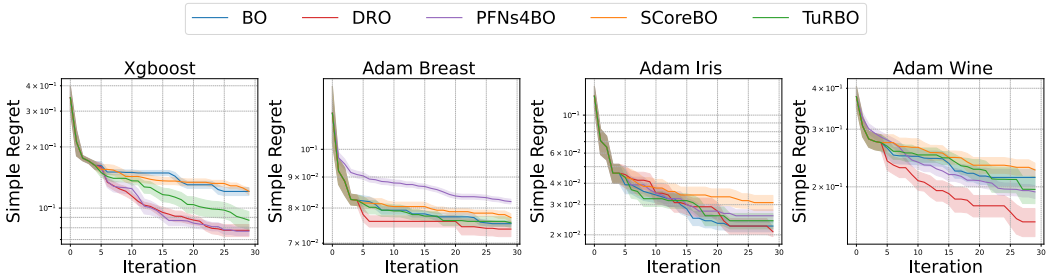
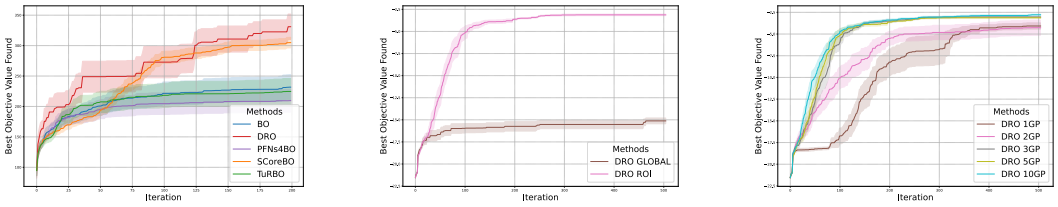


Figure 3: Performance on HPO Tasks (Simple Regret). Lower values are better.

Ackley Function Scalability. Fig. 2 presents results on the Ackley function across dimensions 2D, 5D, 10D, and 20D. DRO consistently demonstrates strong performance, achieving the best or near-best objective values across all tested dimensions when compared to standard BO, PFNs4BO (up to 10D), SCoreBO, and TuRBO. Notably, in the 20D Ackley task, where PFNs4BO is absent due to its reported scaling limitations beyond 18 dimensions, DRO maintains robust performance and clearly outperforms the remaining baselines (BO, SCoreBO, TuRBO). This indicates DRO's favorable scalability to higher-dimensional problems.

Hyperparameter Optimization Tasks. On the suite of HPO tasks (Fig. 3), DRO consistently performs well. For XGBoost tuning, DRO converges to a lower simple regret faster and more reliably than other methods. A similar leading trend is observed for the Adam Wine HPO task. On Adam Iris and Adam Breast Cancer HPO, DRO remains highly competitive, consistently ranking among the top-performing methods. These results suggest its effectiveness for practical HPO problems.



(a) LunarLander Controller (b) Effect of ROI Filtering (c) Effect of Ensemble Size (M)

Figure 4: Performance on LunarLander and Ablation studies for DRO on Ackley 10D. Higher values are better (Best Objective Value Found). All DRO variants in ablations run for 10 trials.

Fig. 4 shows DRO's performance on LunarLander and insights from ablation studies on Ackley 10D.

Complex Simulation (LunarLander). For the LunarLander control task (Fig. 4a), DRO again shows strong and competitive performance. It achieves higher best objective values more rapidly than standard BO, PFNs4BO, and SCoreBO. DRO also demonstrates a clear advantage over TuRBO throughout the optimization process, underscoring its capability in complex sequential decision-making scenarios.

Effect of ROI Filtering. Fig. 4b compares DRO with its ROI filtering mechanism ('DRO ROI', representing the standard DRO configuration) against a variant performing rollouts over the entire global search space ('DRO GLOBAL'). The results unequivocally demonstrate the substantial benefit of ROI filtering: 'DRO ROI' achieves significantly better performance, converging faster and to a much higher objective value (closer to 0, the optimum for Ackley). This confirms that adaptively constraining simulations to promising regions is crucial for DRO's efficiency and effectiveness.

Effect of Ensemble Size (M). The influence of the number of GP models (M) in the ensemble is depicted in Fig. 4c. Performance generally improves with larger ensemble sizes. 'DRO 10GP' (M=10) attains the best results, closely followed by 'DRO 5GP' (M=5). Even smaller ensembles like 'DRO 2GP' and 'DRO 3GP' significantly outperform a single GP ('DRO 1GP'). This highlights the value of model diversity from the ensemble for generating rich simulation data, crucial for training a robust decision-making policy. An ensemble size of $M = 5$ or $M = 10$ appears effective.

6.1 Discussion Summary

The empirical evaluations highlight DRO as a potent and scalable method. It frequently shows superior or highly competitive performance against specialized baselines across synthetic functions (up to 20D Ackley), complex simulations (LunarLander), and various HPO tasks. Its strong showing in higher dimensions, where methods like PFNs4BO may be limited, is particularly noteworthy. The direct learning of a non-myopic policy for final regret minimization, using diverse trajectories from an ROI-focused GP ensemble with RBF kernels, underpins its success. Ablation studies on Ackley 10D confirm that ROI filtering is critical for performance and that larger GP ensembles (e.g., $M \geq 5$) enhance effectiveness. These findings position DRO as a robust framework for sample-efficient black-box optimization, especially in challenging and higher-dimensional settings.

7 Conclusion

This paper introduces DRO, a novel perspective on Bayesian optimization. Unlike conventional methods reliant on hand-crafted acquisition functions and surrogate models, DRO directly minimizes final regret through an end-to-end learned policy. Central to DRO is an ensemble of GPs that simulate BO rollouts within regions of interest, guided by a Bayesian early stopping criterion. These simulations provide dense supervisory signals for training a decision transformer to select query points. A key advantage is its *dense training–sparse learning* paradigm: offline simulated data enables the decision transformer to learn non-myopic exploration strategies from numerous hypothetical trajectories, while sparse updates with true function evaluations prevent overfitting and ground the policy in real-world observations.

References

[1] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, 2023.

[2] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. In *Journal of Machine Learning Research*, volume 13, pages 281–305, 2012.

[4] Eric Brochu, Vlad M. Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical Report TR-2010-23, Dept. of Computer Science, University of British Columbia (UBC), 2010. Often cited tutorial; relevant for discussing integrated acquisition functions like Integrated Expected Improvement (mentioned as potentially in Chapter 7).

[5] Roberto Calandra, Andre Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for policy search on robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 270–277. IEEE, 2016.

[6] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[7] Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'Aurelio Ranzato, et al. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems*, 35:32053–32068, 2022.

[8] Zhongxiang Chen, Yunchuan Zhang, Chen Shi, Lina Zhao, and James T Kwok. Meta-learning acquisition functions for bayesian optimization. In *International Conference on Learning Representations (ICLR 2022)*, 2022. Explicitly addresses meta-learning acquisition functions for BO.

[9] Mujin Cheon, Haeun Byeon, and Jay Hyung Lee. Reinforcement learning based multi-step look-ahead bayesian optimization. In *Proc. 13th IFAC Symposium on Dynamics and Control of Process Systems (DYCOPS)*, pages 100–105, 2022.

[10] Mujin Cheon, Jay H. Lee, Dong-Yeun Koh, and Calvin Tsay. EARL-BO: Reinforcement learning for multi-step lookahead, high-dimensional bayesian optimization. *arXiv preprint arXiv:2411.00171*, 2024.

[11] Mark Deutel, Georgios Kontes, Christopher Mutschler, and Jürgen Teich. Combining multi-objective bayesian optimization with reinforcement learning for tinyml. *ACM Trans. Evolutionary Learning and Optimization*, 2025. to appear.

[12] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

[13] David Eriksson and Matthias Poloczek. Scalable constrained bayesian optimization. In *International conference on artificial intelligence and statistics*, pages 730–738. PMLR, 2021.

[14] P Adorio Ernesto and UP Diliman. Mvf–multivariate test functions library in c for unconstrained global optimization. *University of the Philippines Diliman, Quezon City*, 2005.

[15] Benedikt Philipp Vinzent Flick and Patrick Van Der Smagt. Integrating parameter uncertainty into bayesian optimization. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020)*, volume 108 of *Proceedings of Machine Learning Research*, pages 1779–1788. PMLR, 2020. Focuses specifically on MCMC methods for marginalizing GP hyperparameters in BO.

[16] Marta Garnelo, Johannes Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Conditional neural processes. In *ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018. Introduced Conditional Neural Processes (CNPs).

[17] Bing-Jing Hsieh, Ping-Chun Hsieh, and Xi Liu. Reinforced few-shot acquisition function learning for bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[18] Hailong Huang, Xiubo Liang, Quanwei Zhang, Hongzhi Wang, and Xiangdong Li. RLBOF: Reinforcement learning from bayesian optimization feedback. In *Proc. International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024.

[19] Carl Hvarfner, Erik Hellsten, Frank Hutter, and Luigi Nardi. Self-correcting bayesian optimization through bayesian active learning. *Advances in Neural Information Processing Systems*, 36:79173–79199, 2023.

[20] 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.

[21] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B. Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process based approaches for multi-fidelity optimization. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, volume 48 of *Proceedings of Machine Learning Research*, pages 2961–2969. PMLR, 2016. Introduces key MFBO methods like MF-GP-UCB and BOCA.

[22] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity bayesian optimisation with continuous approximations. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1794–1803. PMLR, 2017. Extends MFBO work, particularly relevant for continuous fidelity levels.

[23] Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000. Key paper on modeling discrepancy between fidelities using GPs (auto-regressive model).

[24] Hyunjik Kim, Andriy Mnih, Johannes Schwarz, Marta Garnelo, Ali Eslami, Yee Whye Teh, and Dan Rosenbaum. Attentive neural processes. In *International Conference on Learning Representations (ICLR 2019)*, 2019. Introduced Attentive Neural Processes (ANPs).

[25] Samuel Kim, Peter Y. Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljačić. Deep learning for bayesian optimization of scientific problems with high-dimensional structure, 2021. arXiv preprint arXiv:2104.11667.

[26] Zihan Li and Jonathan Scarlett. Gaussian process bandit optimization with few batches. In *International Conference on Artificial Intelligence and Statistics*, pages 92–107. PMLR, 2022.

[27] Zijing Liu, Xiyao Qu, Xuejun Liu, and Hongqiang Lyu. Robust bayesian optimization with reinforcement learned acquisition functions. *arXiv preprint arXiv:2210.00476*, 2022.

[28] Alexandre Maraval, Matthieu Zimmer, Antoine Grosnit, and Haitham Bou Ammar. End-to-end meta-bayesian optimisation with transformer neural processes. *Advances in Neural Information Processing Systems*, 36:11246–11260, 2023.

[29] Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. Pfns4bo: In-context learning for bayesian optimization. In *International Conference on Machine Learning*, pages 25444–25470. PMLR, 2023.

[30] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

[31] Thomas Nagler. Statistical foundations of prior-data fitted networks. In *International Conference on Machine Learning*, pages 25660–25676. PMLR, 2023.

[32] Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. Regret for expected improvement over the best-observed value and stopping condition. In *Asian conference on machine learning*, pages 279–294. PMLR, 2017.

[33] Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. Yahpo gym-an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *International Conference on Automated Machine Learning*, pages 3–1. PMLR, 2022.

[34] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006. The standard reference for Gaussian Processes, relevant for surrogate modeling and hyperparameter estimation (MLE/MAP).

[35] 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, 2016.

[36] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pages 2951–2959. Curran Associates, Inc., 2012. Seminal paper on applying BO to hyperparameter tuning; discusses practical aspects including handling GP hyperparameters (e.g., MCMC integration mentioned).

[37] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. of the 27th International Conference on Machine Learning (ICML)*, pages 1015–1022, 2010.

[38] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Amortized bayesian optimization over discrete spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5940–5947, 2020. Example of using learned models (related to NPs) for amortized/meta-BO.

[39] Shion Takeno, Carl Hvarfner, Thomas Gärtner, Cédric Archambeau, and Philipp Hennig. Multi-fidelity active learning with max-value entropy search. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 21237–21249. Curran Associates, Inc., 2020. Introduces the Multi-Fidelity Entropy Search acquisition function.

[40] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[41] Alexander Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33. Curran Associates, Inc., 2020.

[42] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable gaussian process-based bayesian optimization ensembles. In *Proceedings of the 2018 SIAM International Conference on Data Mining (SDM 2018)*, pages 513–521. SIAM, 2018. Proposes using ensembles of GPs with different hyperparameters for robustness.

[43] Wenjie Xu, Yuning Jiang, Bratislav Svetozarevic, and Colin Jones. Constrained efficient global optimization of expensive black-box functions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38485–38498. PMLR, 23–29 Jul 2023.

[44] Fengxue Zhang, Thomas Desautels, and Yuxin Chen. Robust multi-fidelity bayesian optimization with deep kernel and partition. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

[45] Fengxue Zhang, Jialin Song, James C. Bowden, Alexander Ladd, Yisong Yue, Thomas Desautels, and Yuxin Chen. Learning regions of interest for bayesian optimization with adaptive level-set estimation. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*. PMLR, 2023.

[46] Fengxue Zhang, Zejie Zhu, and Yuxin Chen. Finding interior optimum of black-box constrained objective with bayesian optimization. In *NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty*, 2024.

A Detailed Discussion on Literature

A.1 Reinforcement Learning in Bayesian Optimization

Traditional Bayesian optimization (BO) frameworks rely on fixed, hand-crafted acquisition functions to determine query points, balancing exploration and exploitation of the unknown objective [20, 37]. Classical acquisition functions such as Expected Improvement or upper confidence bounds are designed a priori and remain static during the optimization. Recent work has proposed integrating reinforcement learning (RL) into BO to learn a sampling strategy dynamically rather than using a fixed rule. By formulating the selection of sample points as a sequential decision-making problem, an RL agent can adaptively improve the sampling policy based on feedback from previous observations. This approach aims to boost efficiency and outcomes by learning to trade off exploration and exploitation in an adaptive, data-driven manner.

RL-Enhanced Acquisition Function Strategies One line of research uses RL to augment or replace the acquisition function itself, enabling dynamic selection of where to sample next. Hsieh et al. [17] introduced a reinforced few-shot acquisition function learning (FSAF) method that treats the acquisition function as a Q-value function. They train a deep Q-network (DQN) to act as a surrogate acquisition function, using meta-learning to allow quick adaptation to new tasks. By employing a Bayesian variant of DQN (to maintain uncertainty estimates) and fine-tuning with limited data, FSAF achieved comparable or better performance than state-of-the-art hand-crafted acquisitions.

Similarly, Liu et al. [27] proposed an RL-assisted BO framework (RLABO) that formalizes acquisition function selection as a Markov decision process. In RLABO, an RL agent is trained to choose among multiple acquisition functions at each BO iteration based on a reward signal (e.g. improvement in objective value). This learned policy adaptively balances exploration vs. exploitation in real time, outperforming any single fixed heuristic across several benchmark optimization tasks.

RL for Non-Myopic (Multi-Step) Bayesian Optimization Another avenue is using RL to extend BO beyond myopic, one-step decisions by planning multi-step lookahead strategies. Cheon et al. [9] provided early evidence that an RL-driven policy can outperform the greedy one-step Expected Improvement strategy by optimizing over a multi-step horizon. Their approach treated BO as a sequential decision process and showed that a learned policy can yield better long-term outcomes than myopic acquisition functions.

Building on this idea, Cheon et al. [10] recently introduced EARL-BO (Encoder-Augmented RL for BO), an RL framework for multi-step lookahead in high-dimensional optimization problems. EARL-BO represents the state of the BO (e.g. the surrogate model posterior and remaining budget) with an attention-based encoder, and it uses off-policy RL to efficiently learn a near-optimal sampling strategy over multiple future steps. This method significantly improved performance compared to traditional one-step BO and earlier limited lookahead (rollout) approaches, as demonstrated on synthetic benchmarks and real-world hyperparameter optimization tasks.

RL-Integrated Surrogate Modeling and Meta-BO Beyond guiding point selection, researchers have also integrated RL into surrogate modeling and meta-optimization aspects of BO. Huang et al. [18] propose RLBOF (Reinforcement Learning from Bayesian Optimization Feedback), where a reinforcement learning agent (based on proximal policy optimization) continuously adjusts the surrogate model using feedback from the BO process. In this approach, the RL agent learns to tune the surrogate model's parameters or training procedure to better fit the observations and improve prediction of the objective function. By optimizing the surrogate model itself via RL (in tandem with the standard BO loop), this method enhances the overall BO performance. Such integration of RL at the meta-level (sometimes called meta-BO) essentially allows the BO algorithm to learn how to learn – optimizing not only the decisions of where to sample, but also refining the modeling of the objective based on experience.

Applications and Multi-Objective Extensions The combination of RL and BO has been applied in various domains and extended to multi-objective optimization problems. For example, Deutel et al. [11] develop an RL-assisted multi-objective BO approach for TinyML neural architecture search. Their method combines multi-objective BO with an ensemble of RL policies (trained via augmented random search) to efficiently explore trade-offs between model accuracy and resource

constraints. This RL-augmented strategy was able to find better Pareto-optimal neural network architectures (balancing accuracy, memory, and latency) than conventional multi-objective BO techniques. Similarly, RL-integrated BO frameworks have been explored for automated hyperparameter tuning, engineering design optimization, and other real-world scenarios. In these applications, the adaptive exploration-exploitation control provided by RL often yields faster convergence and more robust solutions compared to static BO policies.

Key Takeaways In summary, integrating reinforcement learning into Bayesian optimization enables several key advancements over traditional BO. First, RL allows the acquisition function or strategy to be adaptive rather than fixed, selecting sampling actions based on learned experience. Second, RL-based policies can consider multi-step outcomes, mitigating the greedy, myopic nature of standard one-step BO by planning further ahead. Third, RL can be integrated into different parts of the BO pipeline (acquisition selection, surrogate model tuning, or meta-learning), effectively allowing the optimizer to improve itself over time. Across various studies and applications, RL-enhanced BO methods consistently outperform fixed heuristic strategies, demonstrating improved efficiency in finding optimal solutions for both single-objective and multi-objective problems.

A.2 Deep Learning Approaches in Bayesian Optimization

Recent years have seen a surge of interest in integrating deep learning techniques into Bayesian optimization to improve flexibility, scalability, and the capacity to handle more complex objective landscapes. These approaches often replace or augment the traditional Gaussian Process surrogate and hand-crafted acquisition functions with neural network models that can learn from data and past optimization experience. Below, we highlight several representative methods that combine neural networks with BO and compare them to our decision transformer-based strategy.

Optimization in Latent Spaces via Weighted Retraining. Tripp et al. [41] explore Bayesian optimization in the latent space of deep generative models. Here, a generative network (e.g., a variational autoencoder) learns a lower-dimensional embedding of the input space. BO is then conducted *within this latent space*, where the landscape is often smoother or easier to navigate, before mapping latent representations back to the original domain. To keep the latent space relevant to high-performing solutions, they employ a *weighted retraining* scheme that prioritizes observed samples with superior objective values. Compared to our decision transformer-based approach, which focuses on learning a policy to make direct query decisions, Tripp et al. focus on learning a *representation* of the search space. Both methods harness deep learning for higher-level capabilities—either representation learning or policy learning—beyond classical GP-based BO pipelines.

Deep Surrogate Models for High-Dimensional BO. Other recent work highlights how neural networks can be leveraged to improve surrogate modeling in high-dimensional or structured scientific problems. For instance, Kim et al. [25] propose a deep learning framework tailored for Bayesian optimization tasks with high-dimensional structure, using neural networks to capture complex correlations and accelerate convergence in scientific applications. Their approach demonstrates how deep models can scale to large design spaces that would challenge traditional GPs. Similarly, Zhang et al. [45] introduce a neural network-based adaptive level-set estimation method for BO, termed "Learning Regions of Interest." By identifying and focusing on promising sub-regions of the search space, this framework refines the surrogate model in a targeted manner, effectively allocating sampling budget to areas with higher potential for improvement. These methods align with our emphasis on harnessing deep models to handle challenging search landscapes and guide the BO process.

PFNs4BO and Prior-Data Fitted Networks. One notable recent direction is the use of *Prior-Data Fitted Networks (PFNs)* for BO, exemplified by PFNs4BO [29]. PFNs leverage transformers to *amortize* Bayesian inference by meta-training on synthetic tasks drawn from a chosen prior distribution [31]. In PFNs4BO, a transformer is pre-trained (offline) to approximate the posterior predictive distribution (PPD) of the objective function in-context; at runtime, this single model produces posterior predictions for any new BO dataset in one forward pass [31, 30]. This yields extremely fast surrogate updates — the PFN surrogate does **not** require retraining at each BO iteration, unlike a Gaussian Process which must be refit or updated as data accumulates. Moreover, by training across many sampled functions (e.g., varying GP hyperparameters or even Bayesian neural network priors), the PFN inherently marginalizes over surrogate uncertainty, effectively performing Bayesian

model averaging under the specified prior. Empirically, a single PFN can mimic a GP's posterior almost exactly while attaining orders-of-magnitude speedups, and it flexibly accommodates richer prior knowledge than classical kernels (for instance, one can embed hints of likely optima or ignore irrelevant dimensions). Notably, PFNs4BO can even learn the acquisition function as part of its training (enabling non-myopic policies) by incorporating the selection strategy into the prior/task simulation [29].

Meta-Learning Surrogates and Acquisition Functions with Neural Processes. Beyond PFNs, another line of work leverages meta-learning principles, often employing neural processes (NPs) and their variants (e.g., Conditional NPs, Attentive NPs) [16, 24], to accelerate BO. NPs are deep models trained to approximate distributions over functions, conditioning on context sets (observed data points). In the meta-BO setting [38], NPs can be pre-trained on data from related optimization tasks. When deployed on a new task, the NP surrogate can rapidly adapt its predictions based on the initial samples, providing a warm start compared to training a GP or standard neural network from scratch. Some approaches jointly learn the surrogate model (like an NP or transformer-based model) and an acquisition function (or an ensemble of them) in the meta-training phase [8]. End-to-end meta-BO frameworks like OptFormer [7] and Neural Acquisition Process (NAP) [28] also leverage transformers to learn universal optimizers that generalize across tasks. This allows the BO system to learn not just how to model functions typical of the meta-training distribution, but also how to effectively explore them, transferring knowledge about both the function structure and optimal search strategies across tasks. This contrasts with methods that only meta-learn the surrogate or only meta-learn the policy.

Comparison of Transformer-Based BO Approaches. Although both PFNs4BO, Meta-BO with transformers/NPs (including OptFormer and NAP), and our decision transformer-based strategy employ transformer architectures or related deep sequence models, they target different stages or aspects of the BO process. PFNs4BO operates at the surrogate modeling level by using a transformer to approximate the posterior over objective functions through amortized Bayesian inference—effectively replacing the Gaussian Process for posterior estimation and enabling rapid, retraining-free updates. Meta-BO approaches using NPs or transformers, such as OptFormer and NAP, focus on learning transferable surrogate models and acquisition strategies by training across multiple related tasks to create universal optimizers. In contrast, our decision transformer approach learns the acquisition policy directly from problem-specific simulated trajectories by modeling the sequence of decisions, mapping the history of observations to the next query suggestion, potentially leveraging offline RL paradigms. While PFNs4BO and Meta-NPs/universal optimizers remain closer to the Bayesian paradigm by explicitly modeling uncertainty (or distributions over functions) and leveraging inter-task meta-training, our method bypasses an explicit probabilistic surrogate for decision-making in favor of directly learning sequential query strategies from intra-task simulations. Collectively, these approaches underscore the broad potential of deep neural networks in enhancing BO—whether by constructing latent representations, refining surrogate models via meta-learning, amortizing inference, or directly learning sequential decision-making policies.

A.3 Approximation-Aware and Multi-Fidelity Bayesian Optimization

Classical BO often assumes access to exact evaluations of the objective function, potentially corrupted by uniform noise. However, in many real-world scenarios, particularly those involving complex simulations or physical experiments, function evaluations may come from approximations or models with varying fidelity and cost. Approximation-aware BO methods explicitly account for these complexities.

A prominent area is Multi-Fidelity Bayesian Optimization (MFBO) [21, 22]. MFBO techniques leverage cheap, low-fidelity approximations (e.g., coarse simulations, simpler models) to guide the optimization towards promising regions where expensive, high-fidelity evaluations should be performed. Methods like BOCA [21] and MF-GP-UCB [21] model the relationship between different fidelity levels (e.g., using auto-regressive GP models) and design acquisition functions (like Multi-Fidelity Entropy Search [39]) that optimally trade off information gain and cost across fidelities. Other approaches might deal with approximations by explicitly modeling the discrepancy between the approximate model and the true objective [23] or by incorporating evaluation cost into the acquisition function more directly [36]. These methods aim for end-to-end optimization under realistic

evaluation constraints, acknowledging that the function oracle itself might be an approximation with controllable accuracy and cost, a setting distinct from assuming a perfect black-box.

A.4 Handling Unknown Hyperparameters in Bayesian Optimization

The performance of BO heavily relies on the surrogate model, typically a Gaussian process, whose behavior is governed by hyperparameters (e.g., kernel lengthscales, signal variance, noise variance). Standard BO practice often involves fixing these hyperparameters based on heuristics or estimating them via MLE or MAP estimation on the observed data [34]. However, fixing hyperparameters can lead to suboptimal performance if the initial choice is poor, and MLE/MAP estimates can be unreliable with scarce data early in the optimization.

To address this, several approaches treat hyperparameters as nuisance parameters to be marginalized out, adopting a more fully Bayesian perspective. This involves defining priors over the hyperparameters and integrating the acquisition function over the posterior distribution of these hyperparameters [36]. Techniques like MCMC sampling can be used to approximate the integral [15], leading to acquisition functions like the Integrated Expected Improvement [4]. While computationally more intensive, these methods avoid committing to a single hyperparameter setting and tend to be more robust, especially in the early stages of BO when data is limited. Ensemble methods, where predictions or acquisition functions from models with different hyperparameter settings are averaged [42], offer a practical alternative. Recent developments, such as those utilizing PFNs [29], can implicitly perform marginalization by training on a distribution over hyperparameter settings drawn from the prior. This relies on extensive pre-training and does not scale well with respect to dimensionality. Another line of work incorporate uncertainty in GP hyperparameters and devise acquisition accounting for both hyperparameter uncertainty and conventional max-value entropy [19]. In contrast to our work, they stick to conventional GP surrogate model and information criteria acquisitions, which pose computation challenge.

B Theoretical Analysis

The theoretical underpinnings of the DRO framework, particularly concerning its convergence and regret bounds, present a rich area for future investigation. In the following, we provide theoretical insight behind the design of Bayesian early stopping mechanism that balances the efficiency of the rollout and effectiveness of the sampled trajectory, as well as the theoretical property of the region of interest identification mechanism (based on the UCB $\geq$ max LCB criterion, as detailed for Proposition 1 below) that aims to preserve regret guarantees of a base acquisition.

Below, we provide two theoretical results concerning specific components relevant to DRO's design.

Proposition 1 (Regret Guarantee with ROI-Constrained Base Acquisition)**.** *Assume the standard conditions for Gaussian Process-based Bayesian optimization (e.g., smoothness of f, properties of the kernel $k(\cdot,\cdot)$). Consider a GP model providing a posterior mean $\mu_{t-1}(\mathbf{x})$ and variance $\sigma^2_{t-1}(\mathbf{x})$. Let a Region of Interest (ROI) $\hat{\mathcal{X}}_t$ be defined at iteration t as $\hat{\mathcal{X}}_t = \{\mathbf{x} \in \mathcal{X} \mid \mathrm{UCB}_t(\mathbf{x}) \geq \max_{\mathbf{x}' \in \mathcal{X}} \mathrm{LCB}_t(\mathbf{x}')\}$ based on this GP, such that the true optimum $\mathbf{x}^*$ is contained in $\hat{\mathcal{X}}_t$ with high probability, i.e., $\mathbb{P}[\mathbf{x}^* \in \hat{\mathcal{X}}_t] \geq 1 - \delta_t$ for some small $\delta_t > 0$ (akin to Lemma 1 in Zhang et al. (2024) [46]). If a base acquisition function α_{base}, when optimized over the full domain $\mathcal{X}$, achieves an expected cumulative regret $\mathbb{E}[R_{cumulative}(T)] \leq \mathcal{G}(T, |\mathcal{X}|, \gamma_T)$, where γ_T is the maximum information gain over $\mathcal{X}$. Then, the same acquisition function α_{base}, when optimized at each step t only within such an identified ROI $\hat{\mathcal{X}}_t$, i.e., $\mathbf{x}_t = \mathrm{argmax}_{\mathbf{x} \in \hat{\mathcal{X}}_t} \alpha_{base}(\mathbf{x}; \mathcal{D}_{t-1})$, achieves an expected cumulative regret:*

$$\mathbb{E}[R'_{cumulative}(T)] \leq \mathcal{G}(T, |\hat{\mathcal{X}}_{max}|, \hat{\gamma}_T) + C \sum_{t=1}^{T} \delta_t$$

where $|\hat{\mathcal{X}}_{max}|$ is related to the maximum size or complexity of the ROIs encountered, $\hat{\gamma}_T$ is the maximum information gain within these ROIs, and C is a problem-dependent constant (e.g., related to the range of f).

Insight for Proposition 1: This proposition, which builds upon and extends the principles outlined in Lemma 1 of Zhang et al. [46] concerning ROI-based optimization, suggests that constraining the

search of a sound base acquisition function to a high-probability ROI (formed using the UCB-LCB mechanism as defined in Proposition 1) preserves its regret guarantee. It modifies terms related to the search space size and information gain to reflect the smaller ROI. The additional term accounts for the small probability that the optimum is outside the ROI. This highlights that the original guarantee's structure is largely maintained, with parameters scaled to the ROI and an additive term for $\mathbf{x}^* \notin \hat{\mathcal{X}}_t$. This justifies the use of ROI filtering in DRO for focusing simulated rollouts without fundamentally undermining theoretical soundness of underlying acquisition strategies used in simulation.

Proposition 2 (Convergence of Expected Improvement under Converging Simple Regret)**.** *Consider a Bayesian optimization algorithm using a Gaussian Process surrogate that accurately reflects uncertainty (e.g., posterior variance $\sigma_t^2(\mathbf{x})$ does not prematurely collapse to zero for unexplored suboptimal points). Suppose the algorithm generates a sequence of evaluation points $\{\mathbf{x}_t\}_{t=1}^T$ such that its simple regret converges to zero with high probability as $T \to \infty$, i.e., $f(\mathbf{x}_T^+) \to f^*$ with probability at least $1 - \zeta_T$ where $\zeta_T \to 0$. Let $\mathrm{EI}_t(\mathbf{x} \mid \mathcal{D}_t) = \mathbb{E}_{p(f(\mathbf{x})|\mathcal{D}_t)}[\max(0, f(\mathbf{x}) - f(\mathbf{x}_t^+))]$ be the Expected Improvement at iteration $t+1$ based on data $\mathcal{D}_t$ and current best $f(\mathbf{x}_t^+)$. Then, the maximum Expected Improvement achievable across the search space also converges to zero with high probability:*

$$\sup_{\mathbf{x} \in \mathcal{X}} \mathrm{EI}_t(\mathbf{x} \mid \mathcal{D}_t) \to 0 \quad \text{as } t \to \infty, \text{ with high probability.}$$

Insight for Proposition 2: This proposition, which is a direct extension of Lemma 2 presented by Nguyen et al. [32], implies that if an optimization policy (like the one DRO aims to learn) is successful in driving the simple regret to zero, then a common metric used to guide exploration and exploitation—Expected Improvement—will naturally diminish across the entire search space. As the algorithm identifies points increasingly close to the true optimum f^*, the "room" for further improvement $f(\mathbf{x}) - f(\mathbf{x}_t^+)$ shrinks. If the GP posterior also converges appropriately (i.e., $\mu_t(\mathbf{x}) \to f(\mathbf{x})$ and $\sigma_t(\mathbf{x})$ reflects remaining uncertainty), then the EI, which depends on both this gap and the uncertainty, will also tend to zero. This provides a consistency check: a successful algorithm should eventually see that its own measure of potential gain (EI) becomes negligible everywhere. This is relevant for understanding the behavior of the Bayesian early stopping criterion used in DRO's simulated rollouts, as it relies on expected improvement.

C Extensibility of the DRO Framework: Constrained and Multi-Fidelity Optimization

A significant benefit of the DRO framework is its inherent flexibility and potential for extension to more complex optimization scenarios. The core principle of learning a non-myopic decision-making policy from simulated trajectories can be adapted by appropriately modifying the simulation environment, state representation, and action space. This section outlines conceptual extensions of DRO to two common and important settings: Bayesian optimization with unknown constraints and multi-fidelity Bayesian optimization.

C.1 DRO for Bayesian Optimization with Unknown Constraints

When optimizing an objective $f(\mathbf{x})$ subject to M unknown black-box constraints $\mathcal{C}_m(\mathbf{x}) \geq 0$, DRO can be adapted by primarily modifying the simulation phase and state representation. The objective remains to train a decision transformer to directly minimize the simple regret of the true (constrained) optimum. Key considerations include:

1. **Ensemble GP Models for Objective and Constraints:** Independent ensembles of Gaussian Process (GP) models would be maintained for the objective function and for each of the M constraints. This allows for capturing model uncertainty for all unknown functions. All GPs would be conditioned on all available real observations (objective and constraint values).
2. **Constrained Region of Interest for Simulations:** During the simulation (rollout generation) for a specific set of sampled models (one objective GP and one GP for each constraint from their respective ensembles), a **constrained ROI** must be identified. This can

be achieved by adapting ROI identification strategies from constrained Bayesian optimization (CBO) literature, such as the approach in Zhang et al. [46]. This involves:

- Defining an ROI for each constraint $\mathcal{C}_m$ (based on its current GP model $\mathrm{GP}_{\mathcal{C}_m}$) to include regions likely to satisfy the constraint (e.g., where $\mathrm{UCB}_{\mathcal{C}_m}(\mathbf{x}) \geq 0$).
- Forming a joint feasible ROI by intersecting these individual constraint ROIs.
- Defining an objective ROI based on the current objective GP model (GP_f), potentially using a threshold derived from its LCB within the high-confidence jointly feasible region.
- The final ROI for simulation is the intersection of the objective ROI and the joint feasible ROI. This ensures simulated rollouts explore regions deemed both promising for the objective and likely feasible by the set of models defining that particular simulated world.

3. **Constraint-Aware Rollout Generation:** Simulated trajectories are generated by iteratively selecting points within the dynamically identified constrained ROI. The acquisition function used within the simulation must be constraint-aware (e.g., by multiplying a standard acquisition function with the joint probability of feasibility estimated from the constraint GPs, or by using specialized CBO acquisition functions). Observations for both the objective and constraints are sampled from their respective GP posterior predictive distributions.
4. **Augmented State Representation:** The state $\mathbf{s}_\tau$ for the decision transformer must include information about the constraints, such as hyperparameters of the constraint GPs, historical constraint observations, and potentially features describing the estimated feasible region.
5. **Training Objective:** The decision transformer is trained to predict actions that minimize the final simple regret with respect to the true *constrained* optimum, using return-to-go signals calculated based on the best *feasible* objective value found in simulated trajectories.

This conceptual extension enables DRO to learn a non-myopic policy that navigates the trade-offs inherent in constrained optimization.

C.2 DRO for Multi-Fidelity Bayesian Optimization (MFBO)

DRO can also be extended to MFBO settings, where the objective function can be evaluated at different fidelity levels $f \in \mathcal{F}$, each with an associated cost $c(f)$. The goal is to find the high-fidelity optimum while minimizing total evaluation cost. This extension builds upon the core DRO machinery with the following key adaptations:

1. **Fidelity-Specific GP Ensembles:** Independent ensembles of GPs are maintained for each fidelity level, i.e., $\{\mathrm{GP}_m^{(f)}\}_{m=1}^{M_f}$ for each $f \in \mathcal{F}$. Each $\mathrm{GP}_m^{(f)}$ is trained on data observed at fidelity f. Relationships between fidelities (e.g., via auto-regressive models or deep kernels as in Zhang et al. [44]) could also be incorporated into these GP models if desired, though simpler independent models are a first step.
2. **Fidelity-Aware and Cost-Aware ROI for Simulations:** When generating rollouts, the ROI identification can be made fidelity-aware. For instance, drawing inspiration from methods like those in Zhang et al. [44] that identify robust partitions or regions of agreement across fidelities, the simulation environment for DRO could define ROIs based on information from one or more fidelity-specific GPs. The decision of which fidelity's GP(s) to use for ROI definition within a simulation could be part of the diversity generation.
3. **Joint Action Space (Point and Fidelity):** The decision transformer's action becomes a pair $(\mathbf{x}_t, f_t)$, selecting both the point to evaluate and the fidelity level at which to perform the evaluation.
4. **Augmented State Representation:** The state $\mathbf{s}_\tau$ must include fidelity-related information:
   - Historical data, including the fidelity level and cost of each past evaluation.
   - Current cumulative cost and remaining budget (if applicable).
   - Hyperparameters or summary statistics from the fidelity-specific GP ensembles.

5. **Cost-Sensitive Rollout Simulation and Policy Learning:** Simulated trajectories now involve selecting fidelities for simulated queries, incurring simulated costs. The Bayesian early stopping criterion might be adapted to consider the cost-benefit of continuing a rollout. The return-to-go signal for training the decision transformer would still relate to the final (high-fidelity) simple regret, but the policy must learn to achieve this efficiently with respect to the costs associated with different fidelities.

This extension would allow DRO to learn a non-myopic policy that strategically chooses both query locations and fidelity levels to optimize the high-fidelity objective under budget or cost constraints.

These two examples illustrate that the DRO paradigm—leveraging ensemble-based simulations to train a decision transformer for direct regret minimization—provides a flexible foundation. By appropriately designing the components of the simulated environment (surrogate models, ROI definitions, state-action spaces) and the information fed to the decision transformer, DRO has the potential to be tailored to a variety of complex Bayesian optimization settings beyond the standard unconstrained case.

D Additional Implementation Details

The following subsections provide further details on key hyperparameters and configurations used for the DRO framework components. These correspond to the settings in the provided configuration file and are reflected in the implementation.

D.1 Decision Transformer

The collection of simulated trajectories from all GPs in the ensemble forms an offline dataset for training a *decision transformer* [6]. This model is chosen for its ability to handle sequential decision-making problems by framing them as sequence modeling tasks. Each trajectory $(\mathbf{s}_0, \mathbf{a}_0, r_0, \mathbf{s}_1, \mathbf{a}_1, r_1, \dots)$ is processed into sequences of *(state, action, return-to-go)* tuples suitable for the decision transformer:

- **State ($\mathbf{s}_\tau$)**: An embedding representing the state of the BO process at simulation step τ. This can include features derived from the current GP_m (used for that rollout), the history of simulated observations $(\mathbf{x}_i^{(m)}, y_i^{(m)})$ within that rollout (e.g., best value found, number of steps taken), and context from the real BO process (e.g., current real iteration t, real historical data $\mathcal{D}_{t-1}$).
- **Action ($\mathbf{a}_\tau$)**: The point $\mathbf{x}_\tau^{(m)}$ selected by the conventional acquisition function (within $\hat{\mathcal{X}}_{m,t}$) during the simulation step τ for GP_m.
- **Return-to-Go (R_τ)**: A crucial signal for the decision transformer, representing the cumulative future reward from step τ to the end of the trajectory. It is calculated based on the *final simple regret* of the entire simulated trajectory under GP_m. For a simulated trajectory of length $L_{m,k}$ ending with best point $\mathbf{x}^{(m,k)}_{\text{best\_sim},L_{m,k}}$, the return-to-go for step τ could be $f(\mathbf{x}^{(m,k)}_{\text{best\_sim},L_{m,k}}) - \tilde{f}^*_m$, where $\tilde{f}^*_m$ is an optimistic estimate of the true optimum from GP_m's perspective or a normalized target.

The decision transformer is trained using this offline data to predict an action $\mathbf{a}_\tau$ given a history of states, actions, and a target return-to-go. During the *real* BO procedure, the trained decision transformer is conditioned on the current real BO state (constructed similarly, using the ensemble and real data $\mathcal{D}_{t-1}$) and a high target return (i.e., low target regret) to propose the next query point $\mathbf{x}_t$.

- **Architecture**:
  - **Hidden Size**: The embedding dimension or hidden size of the transformer is 128.
  - **Number of Layers**: The transformer has 4 layers.
  - **Number of Heads**: 4 attention heads are used in the multi-head attention mechanisms.

  - **Dropout**: A dropout rate of 0.1 is applied.
- **Training**:
  - **Learning Rate**: The decision transformer is trained with a learning rate of 1×10^{-4}.
  - **Weight Decay**: A weight decay of 1×10^{-5} is used.
  - **Batch Size**: The batch size for training is 32.
  - **Number of Epochs**: The transformer is trained for 100 epochs on the simulated trajectory data collected in each BO iteration.
- **Sequence Length**: The maximum sequence length processed by the transformer is 20.
- **State Dimension**: The state dimension for the transformer input is dynamically calculated. It includes features for each GP in the ensemble (e.g., key hyperparameters like lengthscale and outputscale, typically 2 parameters per GP), the best observed objective value so far, the current iteration number (or a normalized version), and the coordinates of the best-known points.
- **Action Dimension**: This corresponds to the input dimension of the black-box function being optimized.
- **Return-to-Go Calculation**: During training, the return-to-go for a step in a simulated trajectory is the sum of future rewards (improvements achieved in the simulation in terms of simple regret) until the end of that trajectory. For inference (proposing the next real query point), a high target return-to-go (e.g., 1.0, assuming rewards are normalized or scaled appropriately, or equivalently zero simple regret) is used to prompt the transformer to generate an action aimed at achieving this high return.

D.2 Gaussian Process Ensemble

- **Number of Models (M)**: The ensemble consists of $M = 10$ Gaussian Process (GP) models.
- **Kernel**: The GPs utilize an RBF (Radial Basis Function) kernel by default. The configuration allows for other kernels like Matern or RQ, though RBF is the specified default.
- **Kernel Hyperparameters**:
  - **Lengthscale**: Initial lengthscales for the ensemble models are sampled from a range between a minimum of 0.1 and a maximum of 10.0.
  - **ARD (Automatic Relevance Determination)**: ARD is not enabled by default ('ard: False').
- **Likelihood Noise**: The Gaussian likelihood for each GP has a noise constraint, with the lower bound set to 1×10^{-4}.
- **Training**:
  - **Retraining**: GPs are not retrained from scratch at each BO iteration ('retrain: False'); their hyperparameters are updated based on new data.
  - **Optimizer**: GP hyperparameters are optimized using an Adam optimizer.
  - **Learning Rate**: The learning rate for the GP hyperparameter optimization is 0.1.
  - **Training Iterations**: Each GP model is trained for 50 iterations when updated.

D.3 Bayesian Early Stopping (BES)

- **Activation**: The Bayesian early stopping mechanism is active during the simulation rollouts.
- **Threshold (δ)**: A rollout is terminated if the improvement (e.g., maximum Expected Improvement within the ROI for the respective GP model) falls below a threshold $\delta = 1 \times 10^{-4}$.

D.4 Acquisition Functions for Simulation

- **Strategy**: During the simulation phase for generating trajectories, DRO employs a strategy of rotating through different acquisition functions.
- **Candidate Functions**: The pool of acquisition functions for rotation includes Expected Improvement (EI), Upper Confidence Bound (UCB), Probability of Improvement (PI), and Max-value Entropy Search (MES).
- **Hyperparameters**:
  - **UCB κ (or β)**: The κ parameter for UCB is set to 2.0.
  - **EI/PI ξ**: The trade-off parameter ξ for EI and PI is 0.01.
  - **MES Samples**: For Max-value Entropy Search, the number of max-value samples is 10.
- **ROI Constraint during Optimization**: When optimizing these acquisition functions within the simulation rollouts, the search space can be constrained by the UCB $\geq$ max(LCB) criterion. The κ value used for defining this UCB/LCB constraint is 6.0.

Effect of Simulated Acquisition Strategy. To evaluate the impact of the acquisition function strategy used within the simulation rollouts, we compared DRO variants using fixed acquisition functions (Expected Improvement - logEI, Max-value Entropy Search - MES, Upper Confidence Bound - UCB) against our default strategy of rotating through these candidates ('DRO ROTATE'). Fig. 5 presents these results on the Ackley 10D function. It is well known that there is no silver bullet acquisition function across all scenarios. The 'DRO ROTATE' strategy demonstrates robust performance, achieving competitive results comparable to or no worse than using a single fixed acquisition function throughout the optimization. This supports its choice as a means to diversify the simulated trajectories.

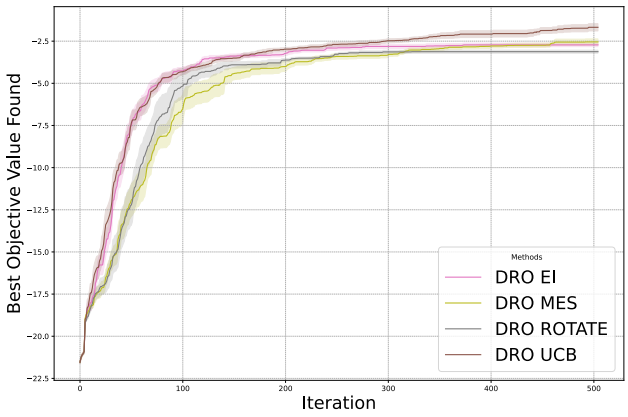


Figure 5: Performance comparison of different acquisition function strategies used within DRO's simulation rollouts on the Ackley 10D function (Best Objective Value Found). 'DRO ROTATE' refers to the strategy of cycling through EI, UCB, PI, and MES for generating simulated trajectories. Higher values are better.