PRINCIPLED DATA GENERATION FOR METABBO VIA ACTIVE TASK SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Meta-Black-Box Optimization (MetaBBO) aims to acquire generalized optimization strategies by training on extensive sets of functions. While traditional approaches predominantly rely on fixed benchmarks, recent efforts have shifted towards maximizing landscape diversity to broaden training distribution coverage. However, we identify a critical *Diversity-Quality Gap*: Simply increasing landscape variety often yields tasks that are misaligned with the agent's evolving capabilities. To address this, we propose *Hierarchical Active Task Selection (HATS)*, a principled framework that constructs an automated curriculum based on the agent's potential for improvement. HATS introduces a regret-based utility metric measuring the performance gap between the current policy and a competent baseline to guide a bi-level selection process. Specifically, it combines a multi-armed bandit for dynamic function class weighting with an active sampler for parameter replay. Experiments demonstrate that HATS significantly outperforms diversity-driven baselines and achieves superior generalization on real-world tasks with fewer training steps, highlighting the value of quality-based data generation in MetaBBO.

1 INTRODUCTION

Black-Box Optimization (BBO), which involves optimizing objective functions where analytical forms and gradient information are inaccessible, is a crucial challenge in real-world complex optimization problem, ranging from hyperparameter tuning (Arango et al., 2021), molecule discovery (Shin et al., 2025) and chip placement (Shi et al., 2023; Xue et al., 2025). Driven by the need to automate these processes, the field of BBO is undergoing a paradigm shift from manually designed heuristics to data-driven automation (Song et al., 2024; Ma et al., 2025d). Meta-Black-Box Optimization (MetaBBO) aims to learn generalized optimization strategies by training parameterized policies on extensive distributions of problem instances (Ma et al., 2023; 2025c). Unlike traditional BBO optimizers defined by static rules, MetaBBO agents learn search patterns through interactions with training functions, making their performance heavily dependent on the quality and structure of the training distribution (Wang et al., 2026).

To prevent overfitting to static instances, current approaches in MetaBBO predominantly generate data via open-loop diversity maximization. Domain Randomization (DR) (Tobin et al., 2017), a technique of randomly sampling function instances, is often employed to augment the training set (Lange et al., 2023; Chen et al., 2024; Hansen et al., 2021). A recent pioneering work, DiverseBBO (Wang et al., 2026), incorporates Exploratory Landscape Analysis (ELA) features (Mersmann et al., 2011) to generate a set of diverse functions. These approaches assume that a training set spanning a wide range of landscape features guarantees a robust learned policy. However, in this paper, we challenge this assumption. Through a quantitative analysis of instance quality, we identify a Diversity-Quality Gap that simply maximizing landscape diversity is misaligned with the agent's capabilities. This observation underscores the necessity of a principled data generation framework for MetaBBO that accounts for the utility of each function instance for the current agent.

To address this, we propose defining task utility by the agent's dynamic potential for improvement. Recognizing that global optima are often intractable in generated functions, we introduce a regret-based utility metric that measures the performance gap between the current MetaBBO policy and a competent baseline (e.g., MadDE (Biswas et al., 2021)). This metric serves as a precise learning

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.

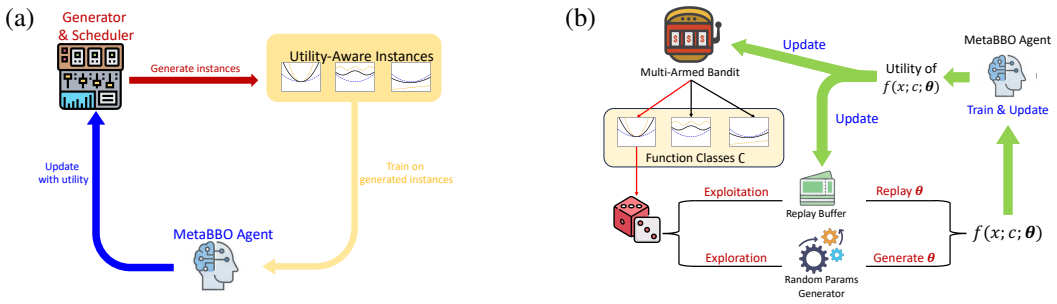


Figure 1: (a) Illustration of our proposed utility-aware framework for MetaBBO, where the generator module interacts with the agent by generating utility-aware instances for agent training and receiving utility feedback from the agent. (b) Illustration of our proposed HATS framework.

signal, guiding the generator to construct a curriculum of tasks that are challenging yet solvable relative to the teacher. Using the utility metric, we reformulate the data generation framework for MetaBBO by scheduling instances based on their utilities, as shown in Figure 1(a). This framework views the generation of instances as a closed-loop process, thereby enabling automated curriculum learning for MetaBBO.

Practically, we instantiate this principle as Hierarchical Active Task Selection (HATS), a framework that leverages the inherent structure of BBO problems. HATS treats the instance space as a hierarchy $\mathcal{G} = \langle \mathcal{C}, \Theta \rangle$, separating global topology (e.g., funnel structures and multi-modality) from local geometric distortions (e.g., rotation and shifting) (Mersmann et al., 2011; Hansen et al., 2021). Specifically, the framework employs a bi-level active selection strategy: 1) A robust multi-armed bandit dynamically allocates the training budget across function classes to ensure topological coverage and prevent catastrophic forgetting; 2) An active sampler prioritizes distortion parameters that maximize the regret-based utility, ensuring the agent continuously trains on the most informative instances for its current stage (Jiang et al., 2021b;a).

Empirically, we evaluate HATS with SYMBOL (Chen et al., 2024) on MetaBox (Ma et al., 2023; 2025c), showing consistent gains over diversity-based data generation and adaptive function-class dynamics throughout training. Our contributions are summarized as follows:

- We empirically demonstrate the Diversity-Quality Gap in MetaBBO, showing that feature diversity correlates poorly with training efficiency.
- We propose HATS, a data generation framework that constructs an adaptive curriculum by actively selecting function class and parameters based on the performance gap towards a baseline.
- We validate HATS across multiple MetaBBO backbones on unseen instances from benchmark suites (CEC and BBOB) and real-world applications including Protein Docking and UAV path planning, showing significant improvements in generalization over state-of-the-art diversity-driven methods.

2 Preliminaries and Related Works

2.1 BBO and MetaBBO

BBO. A standard BBO problem involves minimizing a black-box objective function $f : \Omega \mapsto \mathbb{R}$, where $\Omega \subseteq \mathbb{R}^D$ is the search space of dimension D. The goal is to find $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Omega} f(\mathbf{x})$. In BBO, the analytical form and gradient information of f are inaccessible, thus a BBO optimizer $\mathcal{A}$ interacts with f solely through queries, receiving a response $y = f(\mathbf{x})$ for each query $\mathbf{x}$, constrained by a maximum budget T of function evaluations.

MetaBBO. Leveraging meta-learning techniques (Thrun & Pratt, 1998; Finn et al., 2017), MetaBBO aims to automate the design of BBO algorithms by learning a policy that generalizes across a distribution of problem instances (Ma et al., 2025d), showing superior applicability to real-world complex

optimization problems (Lu et al., 2026). This process can typically be reformulated by a bi-level optimization problem:

- **Inner level**: The optimizer $\mathcal{A}$, controlled by a parameterized policy π_ϕ (where ϕ denotes the meta-parameters, e.g., weights of a neural network), executes the optimization process on a specific problem instance f. This generates a trajectory $\tau_f = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$.
- **Meta level**: The objective of MetaBBO is to find the optimal meta-parameters ϕ^* that maximize the expected performance metric $\mathcal{M}$, e.g., negative regret, over a target problem distribution $p(f)$. Formally, it can be written by:

$$\phi^* = \arg\max_{\phi} \mathbb{E}_{f\sim p(f)}[\mathcal{M}(\mathcal{A}(\pi_\phi, f))].$$

Targeting at different parameterized perspectives, existing MetaBBO methods can be categorized into three lines (Ma et al., 2025d). The first category is algorithm selection, which maintains a pool of algorithms and fits a selector for optimal algorithms (Guo et al., 2024; Cenikj et al., 2024). The second category is algorithm configuration (Hutter et al., 2009; Biedenkapp et al., 2020), which dynamically adjusts parameter values for a fixed optimizer (Xue et al., 2022; Lu et al., 2025; Guo et al., 2025a). Due to the efficiency of leveraging offline data (Ma et al., 2025a) and interpretability (Chen et al., 2024), algorithm generation recently has emerged as a novel solution for MetaBBO.

2.2 Data Generation Issues in MetaBBO

A key limitation of current MetaBBO methods is their reliance on a largely static training distribution $p_{\text{train}}(f)$. Training on fixed benchmark functions or static augmentations can lead to overfitting to specific landscape features, limiting generalization to unseen problems (Wichrowska et al., 2017; Lange et al., 2023). To broaden the training distribution, existing methods commonly apply Domain Randomization (DR) (Tobin et al., 2017) through transformations such as random shifts and rotations (Chen et al., 2022; Lange et al., 2023; Chen et al., 2024; Song et al., 2025), or explicitly maximize landscape diversity based on ELA features (Mersmann et al., 2011; Ma et al., 2025b), as in Diverse-BBO (Wang et al., 2026).

However, more diverse problem instances are not necessarily more useful for training the current MetaBBO agent, as we show in Section 3. Since many MetaBBO agents are trained with RL, the most useful training tasks may change as the agent's capability evolves (Portelas et al., 2020). This motivates an adaptive, utility-aware training distribution rather than static diversity maximization.

2.3 Automated Curriculum Learning

Automated curriculum learning adjusts training tasks as the agent improves, aiming to improve generalization and sample efficiency (Portelas et al., 2020). DR (Tobin et al., 2017) samples tasks uniformly, which can lead to sample inefficiency; Unsupervised Environment Design (UED) instead formulates curriculum generation as a regret-maximization game (Dennis et al., 2020). Prioritized Level Replay (PLR) (Jiang et al., 2021b) prioritizes levels with high value loss for replay, while later work suggests focusing on levels within the agent's learnable zone (Wang et al., 2022). These ideas motivate HATS to select training tasks based on their utility to the current MetaBBO agent; more background is provided in Appendix A.1.

3 Measuring Data Efficiency for MetaBBO

In this section, we re-examine data efficiency for MetaBBO. Section 3.1 challenges the common practice of maximizing instance feature diversity and reveals the poor correlation between instance diversity and quality. To efficiently quantify an instance's quality for an agent, Section 3.2 defines performance gap-based utility, while Section 3.3 discusses the instance structures of BBO functions that motivate a principled data generation pipeline for MetaBBO.

3.1 Does Higher Data Diversity Lead to Better Training Efficiency?

As discussed in Section 2.2, recent works prioritize maximizing the feature diversity of training set to prevent overfitting. These approaches operate under the implicit assumption that a training distri-

bution with sufficient diversity, which spans a broad range of landscape characteristics, inherently guarantees a robust and generalizable MetaBBO policy (Wang et al., 2026). However, this raises a fundamental yet overlooked question: *Does maximizing feature diversity necessarily lead to better training efficiency for the MetaBBO agents?* We conduct an illustrative experiment to answer this question by analyzing the correlation between feature diversity and training efficiency.

To quantify the diversity contribution of each individual instance, following (Wang et al., 2026), we first obtain a 2-dimensional representation for each function by extracting the ELA features of each function using the *pflacco* package (Prager & Trautmann, 2023) and applying them to a pre-trained autoencoder provided by (Wang et al., 2026) [1] to derive a hidden representation $\mathbf{z}$. Then, we compute the averaged Euclidean distance from each problem instance to its k nearest neighbors in the standardized representation space as the diversity score. Formally, the diversity score D_i for the i-th problem instance is calculated by:

$$D_i = \frac{1}{k} \sum_{j \in \mathcal{N}_i} \|\mathbf{z}_i - \mathbf{z}_j\|_2,$$

where $\mathbf{z}_i$ is the normalized feature vector of the i-th instance, and $\mathcal{N}_i$ represents the set of indices of its k nearest neighbors. A higher score D_i indicates that the instance resides in a sparse region of the feature space, thereby contributing more to the overall diversity.

Additionally, to quantify a problem instance f's quality for MetaBBO, we measure its marginal contribution to the learned policy π_ϕ's generalization capability. Given a finite training set $\mathcal{D}_{\text{train}} = \{f_1, \ldots, f_N\}$ and a held-out test set $\mathcal{D}_{\text{test}}$, we adopt a Leave-One-Out (LOO) valuation protocol. For each instance $f_i \in \mathcal{D}_{\text{train}}$, we train a MetaBBO agent on the subset $\mathcal{D}_{-i} = \mathcal{D}_{\text{train}} \setminus \{f_i\}$ to obtain the marginal optimal meta-parameters ϕ^*_{-i}. The quality score Q_i is defined as the performance gap on $\mathcal{D}_{\text{test}}$ between the policy trained on the full dataset and the policy trained without f_i:

$$Q_i = \mathbb{E}_{f \sim \mathcal{D}_{\text{test}}} \left[\mathcal{M}(\mathcal{A}(\pi_{\phi^*_{\text{all}}}, f)) \right] - \mathbb{E}_{f \sim \mathcal{D}_{\text{test}}} \left[\mathcal{M}(\mathcal{A}(\pi_{\phi^*_{-i}}, f)) \right],$$

where ϕ^*_{all} denotes the optimal meta-parameters learned from the full training set $\mathcal{D}_{\text{train}}$. A higher value of Q_i indicates that instance f_i contains critical features that are essential for the optimizer $\mathcal{A}$ to generalize to unseen problems, thereby signifying high quality.

In practice, we use the function instances set constructed by (Wang et al., 2026) [2], which contains 256 diverse instances and encompasses a wide coverage of the ELA features, to train SYMBOL (Chen et al., 2024). We evaluate the model on the held-out test set from (Wang et al., 2026), which consists of unseen synthetic functions and realistic optimization problems including HPO-B (Arango et al., 2021), protein (Tsaban et al., 2022) and UAV (Shehadeh & Kudela, 2025).

In Figure 2(a), we visualize the scatter plots of the instances' diversity and quality and calculate their Spearman correlation coefficient, where the data points are uniformly distributed across the entire space, occupying all four quadrants. Besides, the extremely low Spearman correlation (0.03) reveals a decoupled nature between these two metrics. This observation underscores the necessity of explicitly incorporating quality-aware guidance into the training instances generation, rather than relying solely on diversity maximization, motivating us to propose a novel quality-oriented data generation framework for MetaBBO.

3.2 Performance Gap-based Utility of Function Instances for MetaBBO

The empirical analysis in Section 3.1 establishes that high feature diversity does not guarantee high training quality (Q_i). Ideally, an effective data generator should prioritize instances with high Q_i. However, computing the marginal contribution Q_i via the LOO protocol is computationally prohibitive during the training process, as it requires retraining the model from scratch for every candidate instance. To make quality-aware generation feasible in online MetaBBO agent training, we replace this expensive global LOO objective with a simple proxy: prioritizing instances that have

[1] https://github.com/MetaEvo/Diverse-BBO/blob/main/models/autoencoder_epoch_300.pth

[2] https://github.com/MetaEvo/Diverse-BBO/blob/main/save_gp_functions/all_256_programs.pickle

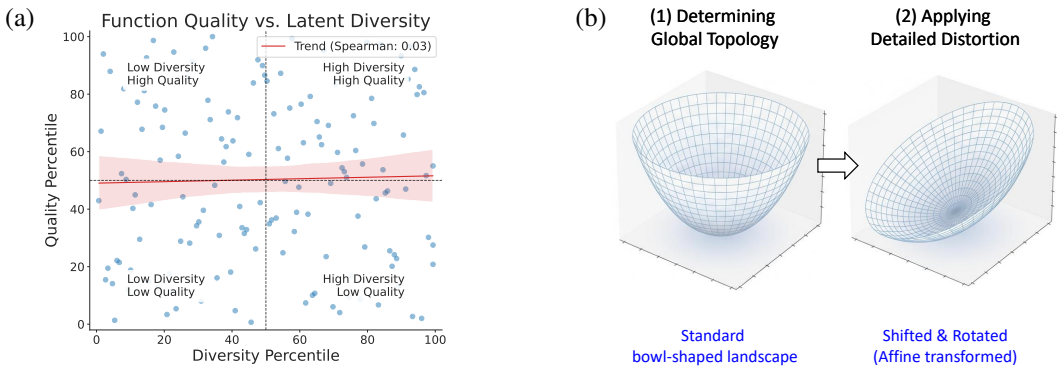


Figure 2: (a) Scatter plot showing the correlation between function quality and its latent diversity. Each point corresponds to one function instance, with diversity measured in the latent space of ELA features and quality quantified by its LOO contribution. The near-zero Spearman correlation (0.03) indicates a weak relationship between function quality and diversity. (b) Illustration of the hierarchical generation of a bowl-shaped instance.

high utility for the current agent, following common practices in RL curriculum design (Portelas et al., 2020; Bengio et al., 2009; Schaul et al., 2015).

In general RL, training-environment utility is typically characterized by *regret* (Jiang et al., 2021b; Dennis et al., 2020), which measures the shortfall of the agent's policy against a theoretical optimal policy. However, in BBO, relying on the theoretical global optimum is flawed. Effective optimization requires underlying landscape structure (Jones & Forrest, 1995; Mersmann et al., 2011); yet, randomly generated functions often exhibit chaotic or deceptive landscapes, effectively acting as unoptimizable noise where the global optimum is practically indistinguishable from random chance (Wolpert & Macready, 1997). In such intractable scenarios, a large global regret reflects the problem's inherent difficulty rather than the agent's incompetence. Consequently, defining utility by the global optimum creates a misalignment, encouraging the agent to chase unsolvable noise.

To mitigate the effects of problem difficulty, we instead assess the agent's *relative potential for improvement* against a competent baseline (e.g., CMA-ES (Hansen et al., 2003) or MadDE (Biswas et al., 2021)). Intuitively, an instance is most informative when the agent underperforms compared to the teacher baseline, indicating that the instance is solvable but not yet mastered by the current MetaBBO policy π_ϕ. Formally, we define the utility as the *Regret Gap* for MetaBBO by:

$$U(f, \phi) = \log \mathcal{M}(\mathcal{A}_{\text{baseline}}(f)) - \log \mathcal{M}(\mathcal{A}(\pi_\phi, f)). \tag{1}$$

This gap serves as a precise learning signal. A near-zero or negative U suggests the task is already learned (low utility), whereas an excessively large U implies the task might be too difficult with respect to the agent. In practice, we employ the concept of "zone of proximal development" (Vygotsky, 1978; Wang et al., 2022) to prioritize instances within a constructable range $[u_{\min}, u_{\max}]$. This reformulation casts data generation as a *dynamic curriculum learning* problem: Rather than training on a static or random sequence of functions, we construct a curriculum that balances the exploration of high-utility new tasks with the replay of critical historical instances to prevent forgetting.

To facilitate this learning process, a well-constructed structure for BBO functions is required, since navigating the unstructured space of all possible mathematical functions is inefficient. In the next subsection, we will introduce a structured parameterization that allows us to effectively select and schedule candidate instances.

3.3 Instance Structures of BBO Functions

To construct a structured curriculum, we organize BBO instances by the landscape properties that determine their optimization difficulty. Following prior landscape analysis (Hansen et al., 2021; Mersmann et al., 2011), these properties can be broadly divided into two factors. *Global topology* describes the overall shape of the objective landscape, such as convexity, multimodality, and funnel structure, and determines the type of search behavior required (Mersmann et al., 2011; Lunacek

et al., 2008). In contrast, *geometric distortion* modifies a fixed topology through transformations such as shifting, rotation, scaling, and conditioning, testing an optimizer's invariance and adaptation (Hansen et al., 2021; Song et al., 2022). Hence, an effective curriculum should cover both different topologies and different distortions within each topology.

Based on this distinction, we model the BBO instance space as a hierarchical generative space $\mathcal{G} = \langle \mathcal{C}, \Theta \rangle$. The discrete class space $\mathcal{C} = \{c_1, \ldots, c_K\}$ contains base function classes (e.g., Sphere, Rastrigin, and Ackley), where the choice of c determines the global topology. Given a class c, the continuous parameter space Θ controls its geometric distortion; for example, $\boldsymbol{\theta} \in \Theta$ may specify rotation, scaling, and shifting. As illustrated in Figure 2(b), a concrete instance is generated by first selecting a base class and then applying a geometric transformation:

$$f(\mathbf{x}; c; \boldsymbol{\theta}) = g_c\big(\mathcal{T}(\mathbf{x}; \boldsymbol{\theta})\big),$$

where g_c denotes the canonical base function and $\mathcal{T}$ the transformation parameterized by $\boldsymbol{\theta}$. This hierarchy naturally motivates HATS to schedule function classes and instance parameters separately, as introduced in the next section.

4 An Efficient Data Generation Framework via Hierarchical Active Task Selection

In this section, we present our framework, Hierarchical Active Task Selection (HATS), which incorporates a bi-level strategy motivated based on the hierarchical generative space discussed in Section 3.3. As shown in Figure 1(b), HATS comprises: 1) a bandit-based mechanism to allocate attention across different function classes $\mathcal{C}$, and 2) an active selection mechanism combining exploration and replay exploitation to schedule the parameters $\boldsymbol{\theta}$ efficiently. One key characteristic of HATS is that the generator is updated *online* using the agent's current competence, rather than optimizing a static diversity objective.

4.1 Multi-Armed Bandit for Function Class Selection

HATS first employs a Multi-Armed Bandit (MAB) mechanism to dynamically allocate the training budget across K distinct function classes, where we treat each function class $c_k \in \mathcal{C}$ (where $K = |\mathcal{C}|$) as an arm. We maintain a logit vector $\boldsymbol{\ell}^{(t)} = [\ell_1^{(t)}, \ldots, \ell_K^{(t)}]$ to track the time-varying utility of each class. Unlike standard stationary bandits, the utility here depends on the agent's evolving competence, showing more robust response to the high stochasticity of the feedback signal.

At each training step t, we determine the selection probability $p_k^{(t)}$ for each class c_k. To prevent starvation of temporarily difficult classes and ensure continuous exploration, we employ a mixture policy that combines a Softmax distribution with uniform exploration. The probability is defined as:

$$p_k^{(t)} = (1 - K\epsilon) \cdot \frac{\exp(\ell_k^{(t)})}{\sum_{j=1}^{K} \exp(\ell_j^{(t)})} + \epsilon,$$

where $\epsilon = 0.05$ is a small constant ensuring a non-zero floor probability for every class. Based on $\boldsymbol{p}^{(t)}$, a batch of task categories is sampled and subsequently populated with concrete instances by the instance generator.

However, empirically, we find that a vanilla MAB may suffer from probability collapse and catastrophic forgetting; detailed analysis is provided in Appendix D.1. To mitigate this, we employ two techniques, collectively termed Robust MAB:

- **Rank-Based Utility Feedback.** After the agent trains on the sampled batch, we quantify the value of each class via *Rank Utility*. For a batch of N tasks, we calculate the utility of each instance i using Equation (1), rank them within the batch, and assign a normalized score $v_i \in [0, 1]$ based on the ranking. The feedback $\bar{v}_k^{(t)}$ for class c_k is the mean utility of all instances belonging to that class in the current batch.
- **Robust Update Mechanism.** To enable a stable update of the logits, we employ a centered decaying update rule, which is given by:

$$\ell_k^{(t+1)} = \text{clip}\left(\gamma \cdot \ell_k^{(t)} + \eta \cdot (\bar{v}_k^{(t)} - 0.5),\ -L,\ L\right).$$

Algorithm 1 UAR for Instance Parameters

1: **Input:** Function class c, global buffer $\mathcal{B}$, policy π_ϕ
2: Sample replay decision $d \sim \text{Bernoulli}(\rho)$
3: **if** $d = 0$ **or** no valid tasks for c in $\mathcal{B}$ **then**
4: // Exploration: Generate new task, no gradient update
5: Sample parameters $\boldsymbol{\theta}$ from generator for class c
6: Collect trajectory τ on $\boldsymbol{\theta}$ with *stop-gradient* $\phi_\perp$
7: **else**
8: // Replay: Sample based on Score S (Equation (2))
9: Sample $\boldsymbol{\theta} \sim \mathcal{B}$ restricted to class c using prob. $\boldsymbol{p}$
10: Collect trajectory τ on $\boldsymbol{\theta}$ and update ϕ with rewards $R(\tau)$
11: **end if**
12: Compute utility $u = U(\tau, \pi)$ via Equation (1)
13: Update global buffer $\mathcal{B}$ with instance $\boldsymbol{\theta}$ and utility u

Here, we center the feedback to 0 by subtracting 0.5, ensuring that classes performing below the median are penalized while those above are rewarded. The decay factor $\gamma \in (0, 1)$ induces a forgetting mechanism for outdated priors, allowing the scheduler to adapt to the agent's new capabilities.

4.2 Active Parameter Sampling via Utility-Aware Replay

Once the MAB allocates the training budget to a specific function class c, we employ a Utility-Aware Replay (UAR) strategy to determine the specific parameters $\boldsymbol{\theta}$. While building upon the Prioritized Level Replay (PLR) framework (Jiang et al., 2021b), our mechanism is tailored to select instances from a global buffer $\mathcal{B}$ that maximize training efficiency for the chosen class. The overall procedure is outlined in Algorithm 1.

Prioritized Sampling Strategy. To select the most effective training tasks from the global buffer $\mathcal{B}$, we calculate a sampling score S_i for each instance i belonging to class c. Let u_i denote the utility score derived from Equation (1) and t_i denote the staleness (number of steps since the last replay). The score is computed as:

$$S_i = \frac{u_i - u_{\min}}{u_{\max} - u_{\min}} + \alpha \cdot \frac{t_i - t_{\min}}{t_{\max} - t_{\min}},$$

where values are normalized over the valid subset of the buffer. The hyperparameter α balances the trade-off between utility and staleness. The probability of replaying instance i is then determined by a Softmax distribution with temperature τ:

$$p_i = \frac{\exp(S_i/\tau)}{\sum_{j \in \mathcal{B}_c} \exp(S_j/\tau)}, \tag{2}$$

where $\mathcal{B}_c$ denotes the subset of the global buffer containing instances of class c, and τ is set to 1.0. This ensures that the agent focuses on tasks with high training utility while periodically revisiting older instances to prevent catastrophic forgetting (Jiang et al., 2021b).

Additionally, as recommended by (Jiang et al., 2021a), UAR employs a *stop-gradient* operation $\phi_\perp$ to exploration trajectories, updating the policy π only on verified replay instances. As detailed in Algorithm 1, when exploring new parameters (Line 4), we generate $\boldsymbol{\theta}$ from the function class c but strictly suppress the policy update. Gradient updates are performed only during the replay phase (Line 8), ensuring the agent learns only from tasks with high utility.

5 Experiments

We evaluate HATS from multiple perspectives. Section 5.1 describes the experimental setup and tasks, while Section 5.2 presents results organized around several Research Questions (RQs).

Table 1: Normalized performance on CEC, BBOB, Protein Docking and UAV using SYMBOL as the MetaBBO backbone, where the best and runner-up on each task are **blue** and **violet**. The results are scaled to the range $[0, 1]$, and lower values indicate better performance.

Base Function	Scheduler Type	CEC	BBOB	Protein Docking	UAV	Avg. Score	Avg. Rank
MA-BBOB	Static	0.249 ± 0.204	0.236 ± 0.189	0.703 ± 0.099	0.427 ± 0.261	0.404	6.50
Diverse-BBO	Static	0.312 ± 0.195	0.374 ± 0.156	0.330 ± 0.069	0.478 ± 0.259	0.373	7.00
BBOB	Static	0.260 ± 0.172	0.268 ± 0.131	0.440 ± 0.107	0.483 ± 0.278	0.363	7.00
	DR	0.122 ± 0.106	0.145 ± 0.112	0.189 ± 0.061	0.331 ± 0.219	0.197	3.75
	HATS (Ours)	**0.041 ± 0.053**	**0.068 ± 0.129**	0.150 ± 0.069	**0.066 ± 0.156**	**0.081**	**2.25**
CEC	Static	0.181 ± 0.127	0.173 ± 0.109	0.569 ± 0.114	0.279 ± 0.220	0.300	5.25
	DR	0.122 ± 0.102	0.158 ± 0.131	**0.112 ± 0.045**	0.259 ± 0.248	0.163	3.00
	HATS (Ours)	**0.034 ± 0.040**	**0.051 ± 0.048**	**0.116 ± 0.056**	**0.057 ± 0.151**	**0.065**	**1.25**

5.1 Experimental Setup and Selected Tasks

MetaBBO Backbones. We use SYMBOL (Chen et al., 2024) as the primary backbone, and further evaluate HATS with LDE (Sun et al., 2021), B2Opt (Li et al., 2025), and GLHF (Li et al., 2024) to examine its applicability across different MetaBBO agents. For SYMBOL, we replace its original DR-based data generation with HATS while keeping the remaining components unchanged. Details of SYMBOL are provided in Appendix B.1.

Compared Baselines. We compare HATS against two categories of baselines: (1) **Static training**: Models trained on fixed function sets without augmentation or curriculum mechanisms. This includes two standard benchmarks, BBOB (Hansen et al., 2021) and CEC (Mohamed et al., 2021), as well as fixed function sets generated by recent methods, MA-BBOB (Vermetten et al., 2025) and Diverse-BBO (Wang et al., 2026); (2) **DR** (Tobin et al., 2017): A widely adopted baseline that applies random shifts and rotations to augment the base functions (i.e., functions from BBOB and CEC). Implementation details and HATS hyperparameters can be found in Appendix C.

Tasks. We evaluate all methods on two types of tasks: unseen synthetic instances constructed from CEC (Mohamed et al., 2021) and BBOB (Hansen et al., 2021) with unseen shifts and rotations, and realistic optimization tasks from MetaBox (Ma et al., 2023; 2025c) for evaluation, including Protein Docking (Tsaban et al., 2022) and UAV (Shehadeh & Kudela, 2025) [3]. These selected tasks cover a wide range of 1,424 distinct instances to assess all methods' performance.

5.2 Experiment Results

In this section, we present our experimental findings, aiming to answer the following RQs.

RQ1: Does HATS improve generalization of the learned MetaBBO agent? Table 1 shows that: (1) HATS achieves state-of-the-art performance by obtaining an average rank of 2.25 and 1.25 when trained on BBOB and CEC functions, respectively, outperforming DR except in one case, showing that HATS improves the generalization of the MetaBBO agent. (2) DR consistently outperforms static training, demonstrating the benefit of instance augmentation.

RQ2: Does HATS generalize across different MetaBBO backbones? We further evaluate HATS with LDE, B2Opt, and GLHF in addition to SYMBOL. Figure 3 shows that HATS achieves lower average ranks than DR and static training in all seven evaluated backbone-task settings, demonstrating broad applicability across MetaBBO agents.

RQ3: Does HATS improve training efficiency? Figure 4 (a) shows that SYMBOL combined with HATS converges faster and achieves better final performance than that with DR on CEC. In particular, HATS reaches a low normalized test performance within fewer training episodes and maintains lower test performance than DR throughout training. We also provide a case study of these two agents in Appendix D.4.

RQ4: How does the curriculum evolve under HATS? Figure 4 (b) shows that HATS dynamically adapts the sampling distribution over CEC function classes during training. The probability of the

[3] We exclude HPO-B (Arango et al., 2021) tasks from MetaBox due to HPO-B's unstable surrogate evaluation, as demonstrated by (Müller et al., 2023).

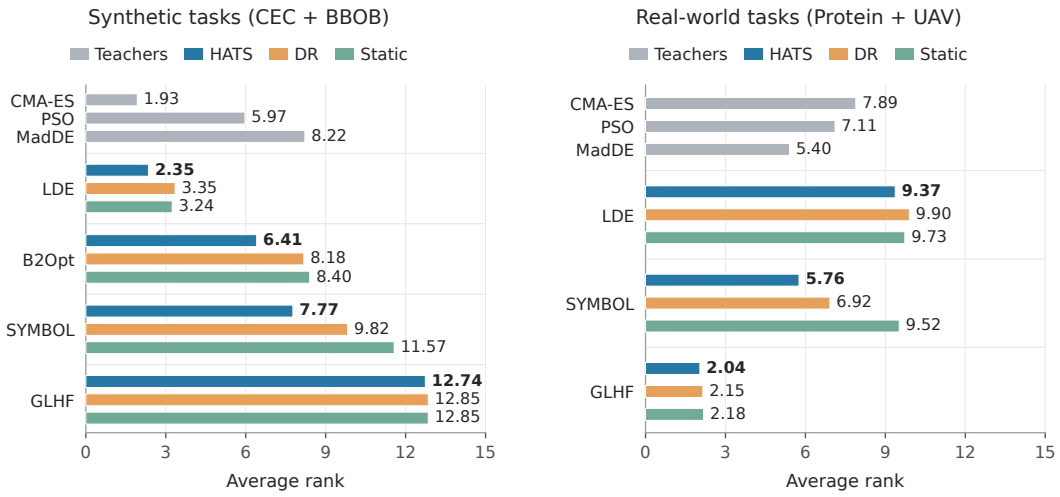


Figure 3: Average rank of HATS, DR, and static training across MetaBBO backbones on synthetic (left) and real-world tasks (right; lower is better). Gray bars denote teacher baselines. B2Opt is omitted from real-world tasks because it is a dimension-dependent model trained under 10D, whereas Protein Docking and UAV are not 10D problems.

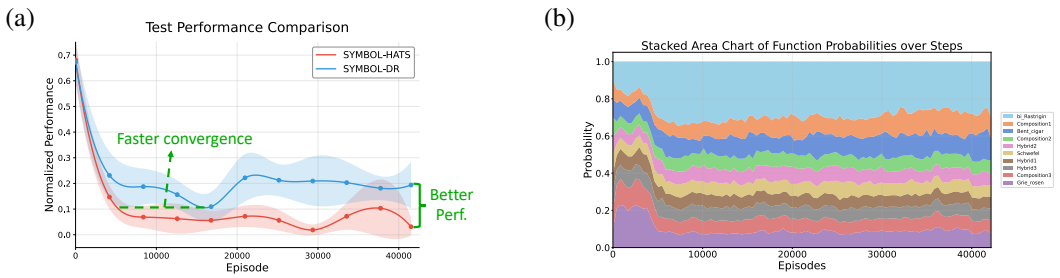


Figure 4: (a) Normalized performance (minimization) comparison between SYMBOL-HATS (red) and SYMBOL-DR (blue). (b) Stacked area chart of function class probability distribution of CEC functions during training.

relatively simple `Grie_rosen` first increases and then decreases after about 5000 episodes, suggesting that the agent has gradually learned to handle this class and therefore requires less training on it. Meanwhile, sampling shifts toward the harder multimodal `bi_Rastrigin`, showing that HATS changes its training focus as the agent's capability evolves.

Additional analysis. We ablate MAB and UAR in Appendix D.2 to validate each component, and further examine HATS's applicability to few-shot offline BBO with ExPT in Appendix D.3. In Appendix D.4 and D.5, we provide a case study comparing HATS and DR, and report the optimization curves on different types of tasks, respectively.

6 Conclusion

In this work, we focus on the data generation issue for MetaBBO. We identify the Diversity-Quality Gap in MetaBBO and propose HATS, a framework that bridges this gap by dynamically selecting a curriculum based on a regret-based utility metric. Across multiple MetaBBO backbones, HATS achieves lower average ranks than DR and static training on both synthetic and real-world tasks, while also showing faster convergence on CEC. These results highlight the benefit of adapting training tasks to the agent's evolving capabilities rather than maximizing diversity alone for more effective and efficient MetaBBO training. Future work may extend HATS to a generator that not only schedules instances but also creates effective data iteratively, e.g., via generating symbolic functions (Meidani et al., 2024; Wang et al., 2025).

AI USE STATEMENT

We used generative AI tools to assist with language polishing, including grammar and punctuation, as well as manuscript layout and formatting.

REFERENCES

Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. ROBEL: Robotics benchmarks for learning with low-cost robots. In *Proceedings of the 4th Conference on Robot Learning*, pp. 1300–1313, Virtual, 2020.

Sebastian Pineda Arango, Hadi Samer Jomaa, Martin Wistuba, and Josif Grabocka. HPO-B: A large-scale reproducible benchmark for black-box HPO based on OpenML. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, Virtual, 2021.

Luis A. Barrera, Anastasia Vedenko, Jesse V. Kurland, Julia M. Rogers, Stephen S. Gisselbrecht, Elizabeth J. Rossin, et al. Survey of variation in human transcription factors reveals prevalent DNA binding changes. *Science*, 351(6280):1450–1454, 2016.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the International Conference on Machine Learning*, pp. 41–48, Montreal, Canada, 2009.

Varun Bhatt, Bryon Tjanaka, Matthew C. Fontaine, and Stefanos Nikolaidis. Deep surrogate assisted generation of environments. In *Advances in Neural Information Processing Systems 36*, pp. 37762–37777, New Orleans, LA, 2022.

A. Biedenkapp, H. F. Bozkurt, T. Eimer, F. Hutter, and M. Lindauer. Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework. In *Proceedings of the 24th European Conference on Artificial Intelligence*, 2020.

Subhodip Biswas, Debanjan Saha, Shuvodeep De, Adam D Cobb, Swagatam Das, and Brian A Jalaian. Improving differential evolution through Bayesian hyperparameter optimization. In *Proceedings of the IEEE Congress of Evolutionary Computation*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540*, 2016.

Gjorgjina Cenikj, Gašper Petelin, and Tome Eftimov. TransOptAS: Transformer-based algorithm selection for single-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 403–406, Melbourne, Australia, 2024.

Jiacheng Chen, Zeyuan Ma, Hongshu Guo, Yining Ma, Jie Zhang, and Yue-Jiao Gong. SYMBOL: Generating flexible black-box optimizers through symbolic equation learning. In *Proceedings of the 12th International Conference on Learning Representations*, Vienna, Austria, 2024.

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 36*, pp. 32053–32068, New Orleans, LA, 2022.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems 34*, Vancouver, Canada, 2020.

Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, Ofer M. Shir, and Thomas Bäck. Benchmarking discrete optimization heuristics with IOHprofiler. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1798–1806, Prague, Czech Republic, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1126–1135, Sydney, Australia, 2017.

Hongshu Guo, Yining Ma, Zeyuan Ma, Jiacheng Chen, Xinglin Zhang, Zhiguang Cao, Jun Zhang, and Yue-Jiao Gong. Deep reinforcement learning for dynamic algorithm selection: A proof-of-principle study on differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(7):4247–4259, 2024.

Hongshu Guo, Zeyuan Ma, Jiacheng Chen, Yining Ma, Zhiguang Cao, Xinglin Zhang, and Yue-Jiao Gong. ConfigX: modular configuration for evolutionary algorithms via multitask reinforcement learning. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, pp. 26982–26990, Philadelphia, PA, 2025a.

Jiacheng Guo, Ling Yang, Peter Chen, Qixin Xiao, Yinjie Wang, Xinzhe Juan, Jiahao Qiu, Ke Shen, and Mengdi Wang. GenEnv: Difficulty-aligned co-evolution between LLM agents and environment simulators. *arXiv:2512.19682*, 2025b.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11:1–18, 2003.

Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, pp. 267–306, 2009.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In *Advances in Neural Information Processing Systems 35*, pp. 1884–1897, Virtual, 2021a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 4940–4950, Virtual, 2021b.

Terry Jones and Stephanie Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 184–192, San Francisco, CA, 1995.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, CA, 2015.

Robert Tjarko Lange, Tom Schaul, Yutian Chen, Tom Zahavy, Valentin Dalibard, Chris Lu, Satinder Singh, and Sebastian Flennerhag. Discovering evolution strategies via meta-black-box optimization. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Rwanda, 2023.

Xiaobin Li, Kai Wu, Yujian Betterrest Li, Xiaoyu Zhang, Handing Wang, and Jing Liu. Pretrained optimization model for zero-shot black box optimization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 14283–14324, Vancouver, Canada, 2024.

Xiaobin Li, Kai Wu, Xiaoyu Zhang, and Handing Wang. B2Opt: Learning to optimize black-box optimization with little budget. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, pp. 18502–18510, Philadelphia, PA, 2025.

Chen Lu, Ke Xue, Lei Yuan, Yao Wang, Yaoyuan Wang, Sheng Fu, and Chao Qian. Sequential multi-agent dynamic algorithm configuration. In *Advances in Neural Information Processing Systems 39*, Vancouver, Canada, 2025.

Chen Lu, Ke Xue, Ruo-Tong Chen, Yunqi Shi, Siyuan Xu, Mingxuan Yuan, Chao Qian, and Zhi-Hua Zhou. Dynamic algorithm configuration for global placement. In *Proceedings of the 2026 Design, Automation & Test in Europe Conference & Exhibition*, Verona, Italy, 2026.

Monte Lunacek, Darrell Whitley, and Andrew Sutton. The impact of global structure on search. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature*, pp. 498–507, Berlin, Heidelberg, 2008.

Zeyuan Ma, Hongshu Guo, Jiacheng Chen, Zhenrui Li, Guojun Peng, Yue-Jiao Gong, Yining Ma, and Zhiguang Cao. MetaBox: A benchmark platform for meta-black-box optimization with reinforcement learning. In *Advances in Neural Information Processing Systems 37*, 2023.

Zeyuan Ma, Zhiguang Cao, Zhou Jiang, Hongshu Guo, and Yue-Jiao Gong. Meta-black-box-optimization through offline Q-function learning. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 41807–41826, Vancouver, Canada, 2025a.

Zeyuan Ma, Jiacheng Chen, Hongshu Guo, and Yue-Jiao Gong. Neural exploratory landscape analysis for meta-black-box-optimization. In *Proceedings of the 13th International Conference on Learning Representations*, Singapore, 2025b.

Zeyuan Ma, Yue-Jiao Gong, Hongshu Guo, Wenjie Qiu, Sijie Ma, Hongqiao Lian, Jiajun Zhan, Kaixu Chen, Chen Wang, Zhiyang Huang, Zechuan Huang, Guojun Peng, Ran Cheng, and Yining Ma. MetaBox-v2: A unified benchmark platform for meta-black-box optimization. In *Advances in Neural Information Processing Systems 39*, 2025c.

Zeyuan Ma, Hongshu Guo, Yue-Jiao Gong, Jun Zhang, and Kay Chen Tan. Toward automated algorithm design: A survey and practical guide to meta-black-box-optimization. *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2025d.

Kazem Meidani, Parshin Shojaee, Chandan K. Reddy, and Amir Barati Farimani. SNIP: Bridging mathematical symbolic and numeric realms with unified pre-training. In *Proceedings of the 12th International Conference on Learning Representations*, Vienna, Austria, 2024.

Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pp. 829–836, Dublin, Ireland, 2011.

Ali Wagdy Mohamed, Anas A Hadi, Ali Khater Mohamed, Prachi Agrawal, Abhishek Kumar, and P. N. Suganthan. Problem definitions and evaluation criteria for the CEC 2021 on single objective bound constrained numerical optimization. In *Proceedings of the IEEE Congress of Evolutionary Computation*, 2021.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 25444–25470, Honolulu, HI, 2023.

Tung Nguyen, Sudhanshu Agrawal, and Aditya Grover. ExPT: Synthetic pretraining for few-shot experimental design. In *Advances in Neural Information Processing Systems 36*, pp. 45856–45869, New Orleans, LA, 2023.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 17473–17498, 2022.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pp. 4819–4825, 2020.

Raphael Patrick Prager and Heike Trautmann. Pflacco: Feature-based landscape analysis of continuous and constrained optimization problems in Python. *Evolutionary Computation*, pp. 1–25, 2023.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv:1511.05952*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Mhd Ali Shehadeh and Jakub Kudela. Benchmarking global optimization techniques for unmanned aerial vehicle path planning. *arxiv:2501.14503*, 2025.

Yunqi Shi, Ke Xue, Lei Song, and Chao Qian. Macro placement by wire-mask-guided black-box optimization. In *Advances in Neural Information Processing Systems 37*, pp. 6825–6843, New Orleans, LA, 2023.

Dong-Hee Shin, Young-Han Son, Hyun Jung Lee, Deok-Joong Lee, and Tae-Eui Kam. Offline model-based optimization for real-world molecular discovery. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 3485–3494, Vancouver, Canada, 2025.

Lei Song, Chenxiao Gao, Ke Xue, Chenyang Wu, Dong Li, Jianye Hao, Zongzhang Zhang, and Chao Qian. Reinforced in-context black-box optimization. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence*, pp. 8939–8947, Montreal, Canada, 2025.

Xingyou Song, Sagi Perel, Chansoo Lee, Greg Kochanski, and Daniel Golovin. Open source Vizier: Distributed infrastructure and API for reliable and flexible blackbox optimization. In *Proceedings of the 1st International Conference on Automated Machine Learning (AutoML)*, pp. 8/1–17, 2022.

Xingyou Song, Yingtao Tian, Robert Tjarko Lange, Chansoo Lee, Yujin Tang, and Yutian Chen. Position: Leverage foundational models for black-box optimization. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 46168–46180, Vienna, Austria, 2024.

Jianyong Sun, Xin Liu, Thomas Bäck, and Zongben Xu. Learning adaptive differential evolution algorithm from optimization experiences by policy gradient. *IEEE Transactions on Evolutionary Computation*, 25(4):666–680, 2021.

Sebastian Thrun and Lorien Y. Pratt. Learning to learn: Introduction and overview. In *Learning to Learn*, pp. 3–17. Springer, 1998.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 23–30, 2017.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-Bench: Benchmarks for data-driven offline model-based optimization. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 21658–21676, Baltimore, MD, 2022.

Tomer Tsaban, Julia K Varga, Orly Avraham, Ziv Ben-Aharon, Alisa Khramushin, and Ora Schueler-Furman. Harnessing protein folding neural networks for peptide–protein docking. *Nature Communications*, 13:176, 2022.

A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 31*, pp. 6000–6010, Long Beach, CA, 2017.

Diederick Vermetten, Furong Ye, Thomas Bäck, and Carola Doerr. MA-BBOB: A problem generator for black-box optimization using affine combinations and shifts. *ACM Transactions on Evolutionary Learning Optimization*, 5(1):1–19, 2025.

L. S. Vygotsky. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, MA, 1978.

Chen Wang, Yue-Jiao Gong, Zhiguang Cao, and Zeyuan Ma. Instance generation for meta-black-box optimization through latent space reverse engineering. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, Singapore, 2026.

Rose E. Wang, Jesse Mu, Dilip Arumugam, Natasha Jaques, and Noah Goodman. In the ZONE: Measuring difficulty and progression in curriculum generation. In *Deep Reinforcement Learning Workshop at NeurIPS 2022*, 2022.

Wenxuan Wang, Kai Wu, Yujian Betterest Li, Dan Wang, and Xiaoyu Zhang. Synthetic series-symbol data generation for time series foundation models. In *Advances in Neural Information Processing Systems 39*, 2025.

Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3751–3760, Sydney, Australia, 2017.

D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

Ke Xue, Jiacheng Xu, Lei Yuan, Miqing Li, Chao Qian, Zongzhang Zhang, and Yang Yu. Multi-agent dynamic algorithm configuration. In *Advances in Neural Information Processing Systems 36*, pp. 20147–20161, New Orleans, LA, 2022.

Ke Xue, Ruo-Tong Chen, Rong-Xi Tan, Xi Lin, Yunqi Shi, Siyuan Xu, Mingxuan Yuan, and Chao Qian. BBOPlace-Bench: Benchmarking black-box optimization for chip placement. *arXiv:2510.23472*, 2025.

A Additional Background

A.1 Further Developments in Automated Curriculum Learning

Several works further extend UED and prioritized replay to more complex settings. PLR$^{\perp}$ (Jiang et al., 2021a) extends PLR to stochastic environments by reducing noise from epistemic uncertainty. ACCEL (Parker-Holder et al., 2022) further uses evolutionary strategies to progressively mutate and generate more complex environments from the PLR buffer (Bhatt et al., 2022). More recently, these UED principles have also been applied to more complex domains, such as generating synthetic environments for LLM agents (Guo et al., 2025b).

B A Brief Introduction of the Employed Method Backbone

B.1 SYMBOL

SYMBOL (Chen et al., 2024) is a MetaBBO framework that learns optimizers by symbolically generating update rules. Instead of using a fixed hand-crafted algorithm, SYMBOL trains a symbolic equation generator, parameterized by RL policy, to output an explicit update expression built from a predefined set of basic operators and optimization-related operands. The generated symbolic expression is then executed as the optimizer's update rule to propose new candidate solutions during search, which yields an adaptive yet interpretable optimization procedure.

Practically, the equation generator is meta-trained over the CEC function class (Mohamed et al., 2021), which is augmented with random shift and rotation. The policy is trained via Proximal Policy Optimization (Schulman et al., 2017). The reward consists of two parts: A base reward indicating the optimizer's absolute performance and a guided reward forcing the optimize the mimic the behavior of a teacher algorithm, typically instantiated by MadDE (Biswas et al., 2021).

B.2 ExPT

ExPT (Nguyen et al., 2023) is a pretraining-adaptation framework for few-shot offline BBO, i.e., optimizing the black-box function f using only a few-shot dataset $\{\mathbf{x}_i, f(\mathbf{x}_i)\}_{i=1}^{N}$. ExPT pretrains a transformer model (Vaswani et al., 2017) with a large amount of synthetic data, so that it can adapt to a new objective using only a few labeled observed (x, y) pairs. ExPT uses *inverse modeling*: given a small set of context points $(\mathbf{x}_{1:m}, y_{1:m})$ and a set of desired target outputs $y_{m+1:N}$, the model approximates $p(\mathbf{x}_{m+1:N} \mid \mathbf{x}_{1:m}, y_{1:m}, y_{m+1:N})$, and samples candidate $\mathbf{x}$ during inference.

To obtain substantial pretraining data, ExPT samples synthetic functions from a Gaussian process prior with an RBF kernel (Rasmussen & Williams, 2006), $\tilde{f} \sim \mathcal{GP}(0, K)$ where $K(x, x') = \sigma^2 \exp\left(-\frac{\|x-x'\|^2}{2\ell^2}\right)$. Here σ (variance) controls the scale of function values and ℓ (length scale) controls smoothness. Specifically, ExPT randomizes σ and ℓ to increase function diversity. For each sampled function, it evaluates $\tilde{f}$ on a set of unlabeled inputs, then splits the resulting input-output pairs into a small context set and a target set to form training examples.

C Implementation Details

C.1 Detailed Setting and Hyperparameters for HATS

To quantify the utility performance gap in Equation (1), following SYMBOL to select MadDE (Biswas et al., 2021) as a learnable teacher, we use MadDE as the baseline algorithm to evaluate the hardness of the generated instances. Other hyperparameters for HATS are listed in Table 2.

Table 2: Hyperparameter setting for HATS.

Parameter Name	Values
Utility range $[u_{\min}, u_{\max}]$	[-0.5, 4.0]
Decay rate γ for bandit logits	0.9
Update rate η for bandit logits	0.1
Clipping bound L for bandit logits	1.5
Balancing coefficient α for staleness in replay buffer	0.1
Optimizer	Adam (Kingma & Ba, 2015)
Learning rate	1×10^{-3}

C.2 Implementation Details of Compared Baselines

For BBOB functions, we use the inference from MetaBox (Ma et al., 2025c) (https://github.com/MetaEvo/MetaBox/tree/v2.0.0/src/environment/problem/SOO/COCO_BBOB).

For CEC functions, we directly use the implementation provided by SYMBOL (Chen et al., 2024) (https://github.com/MetaEvo/Symbol/blob/main/dataset/cec_dataset.py).

For MA-BBOB (Vermetten et al., 2025), we use the implementation by IOHProfiler (Doerr et al., 2019) by randomly selecting a set of 24 function instances and train the agent using this set.

For Diverse-BBO, we directly use the `pickle` file provided by (Wang et al., 2026) (https://github.com/MetaEvo/Diverse-BBO/blob/main/save_gp_functions/all_256_programs.pickle). However, we find that the provided function instances are prone to outliers due to the symbolic function property, which may cause damage the training process. Therefore, we take a dynamic strategy that once we meet an outlier that cause numerical overflow, we filter out the current instance and resume the checkpoint from last episode to enable training for Diverse-BBO.

C.3 Implementation Details of HATS on ExPT

At each pretraining iteration, we sample GP hyperparameters θ to generate synthetic functions and train ExPT on the collected data from the functions. We define the utility of a sampled configuration θ by the training negative log-likelihood: $U_{\text{ExPT}}(\theta) = -\log p_\phi(\mathbf{x}_{m+1:N} \mid \mathbf{x}_{1:m}, y_{1:m}, y_{m+1:N})$ where p_ϕ denotes the distribution parameterized by the model, and use it as the feedback signal for utility-aware replay. This follows the HATS principle that U should reflect the training value of an instance for the current model.

In our implementation, we apply HATS by scheduling the two core GP hyperparameters that govern instance generation and difficulty: the lengthscale ℓ and the function scale σ. Let $\theta = (\ell, \sigma)$, $\ell \sim \mathcal{U}[1.0, 10.0]$, $\sigma \sim \mathcal{U}[5.0, 10.0]$. These prior ranges are set following the ExPT. We perform utility-aware replay over previously sampled θ for training.

Following ExPT, we evaluate the trained model on 4 tasks from Design-Bench (Trabucco et al., 2022), a prevalent benchmark suite for offline BBO. These tasks include 2 continuous tasks, Ant Morphology (Brockman et al., 2016) and D'Kitty Morphology (Ahn et al., 2020), and 2 discrete

tasks, TF-Bind-8 and TF-Bind-10 (Barrera et al., 2016). The objective of Design-Bench is to maximize the objective score, which is different to our main experiment in Table 1.

D Additional Experimental Results

D.1 Analysis of Robust MAB

We empirically find that deciding the function class solely based on a vanilla MAB can lead to probability collapse. During the training of SYMBOL (Chen et al., 2024) on 10 CEC function classes (Mohamed et al., 2021), we find that the agent struggles on the `bi_Rastrigin` function. As training progresses, the probability of selecting `bi_Rastrigin` converges to 0.55, the theoretical maximum probability of a function class, i.e., the MAB collapses to this arm. This results in poor performance for the agent. As shown in Figure 5, after 30000 episodes, the agent trained with vanilla MAB suffers from catastrophic forgetting.

In contrast, Robust MAB assigns a more balanced probability to the `bi_Rastrigin` arm, showing smoother and more consistent improvement on the test set throughout training.

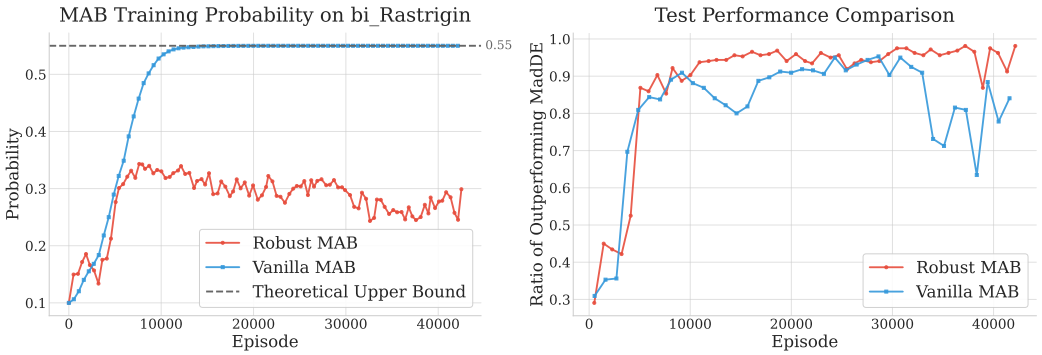


Figure 5: Learning dynamics of Robust MAB and Vanilla MAB. The left sub-figure plots MAB probability of choosing the `bi_Rastrigin` function, while the right one displays the test performance throughout training.

D.2 Ablation Study: Does Each Component of HATS Contribute to the Final Performance?

We ablate key components in HATS to isolate their contributions by: (i) removing the multi-armed bandit for function class selection, (ii) removing utility-aware replay for instance parameters. Experimental results in Figure 6 clearly demonstrates the effectiveness of each component in HATS.

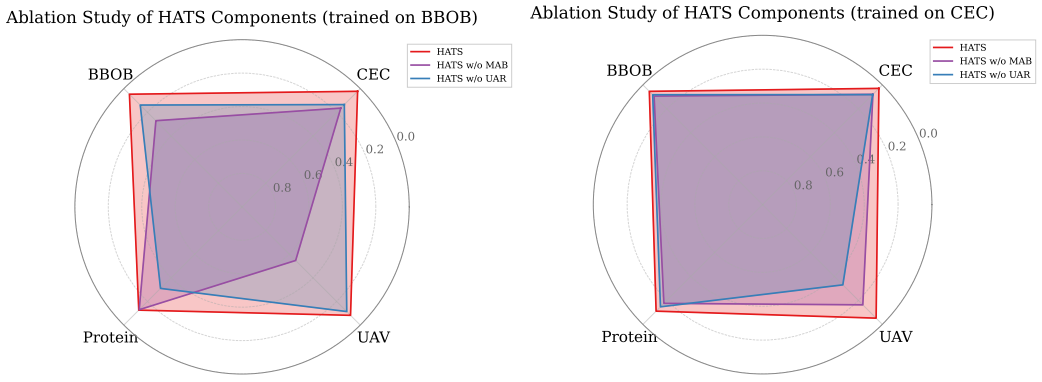


Figure 6: Ablation results on HATS components, MAB and UAR.

D.3 Generalization to Few-Shot Offline BBO

We additionally examine the effectiveness of HATS on the few-shot data-driven offline BBO scenario. We build upon ExPT (Nguyen et al., 2023), a framework that directly models the final solution conditioned on the few-shot data. ExPT is pretrained under Gaussian processes with varying hyperparameters. We apply HATS to schedule these hyperparameters. Details of ExPT and our HATS implementation can be found in Appendix C.3. As shown in Figure 7, combined with HATS, the performance of ExPT can be further enhanced, showing the versatility of HATS.

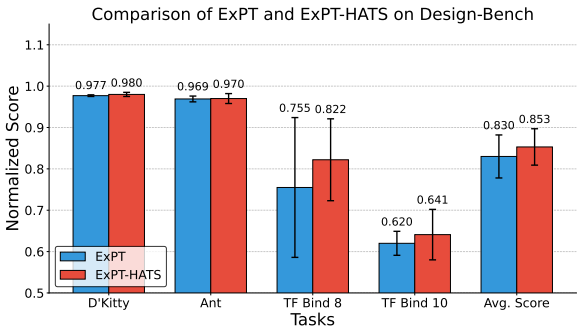


Figure 7: Normalized score (maximization) on Design-Bench (Trabucco et al., 2022), comparing ExPT and ExPT-HATS.

D.4 Case Study: How does SYMBOL Combined with HATS Generalize Better than DR?

Figure 8 compares the behaviors of SYMBOL-DR and SYMBOL-HATS on an unseen shifted-and-rotated 2D Rastrigin instance. In the left panel, SYMBOL-HATS reaches the true optimum within fewer iterations, and its trajectory is more clustered with fewer large cross-region jumps, indicating a more stable search behavior. In contrast, in the right panel, SYMBOL-DR repeatedly searches around $(0, 0)$ and makes frequent long-range jumps across the domain, failing to reach the true optimum. Consistent with these observations, the bottom curve shows that SYMBOL-HATS reduces the objective faster, whereas SYMBOL-DR decreases more slowly and levels off at a worse objective.

D.5 Optimization Curve

In Figures 9 to 11, we visualize the optimization curve of all compared methods on BBOB, CEC, Protein and UAV. Note that for BBOB and CEC, we augment the testing instances by randomly sampling unseen parameters for testing, we plot the detailed performance of each function by averaging objective value at each iteration. For protein docking and UAV, we visualize the average performance over all test problems. The objective value are normalized to [0,1] with respect to each instance. These results clearly show the superiority of our proposed method, HATS.

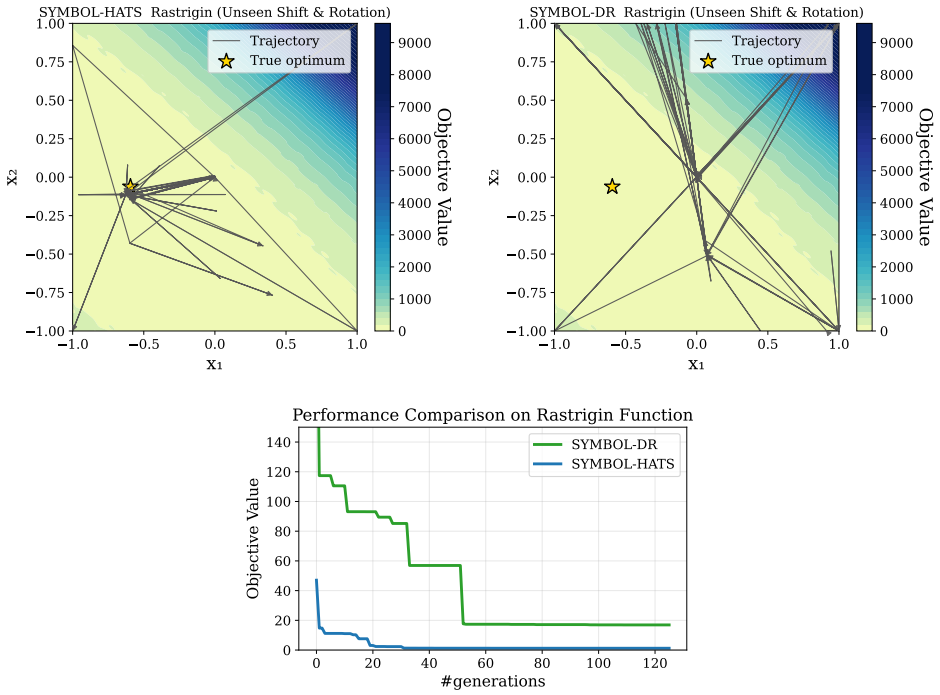


Figure 8: Performance comparison on Rastrigin: SYMBOL trained with HATS (termed as SYMBOL-HATS) versus SYMBOL-DR. The top panels visualize the 2D objective landscape of an unseen shifted-and-rotated Rastrigin function with optimization trajectories produced by SYMBOL-HATS (left) and SYMBOL-DR (right), and the bottom panel compares their objective values across generations. SYMBOL-HATS exhibits more stable search behavior.

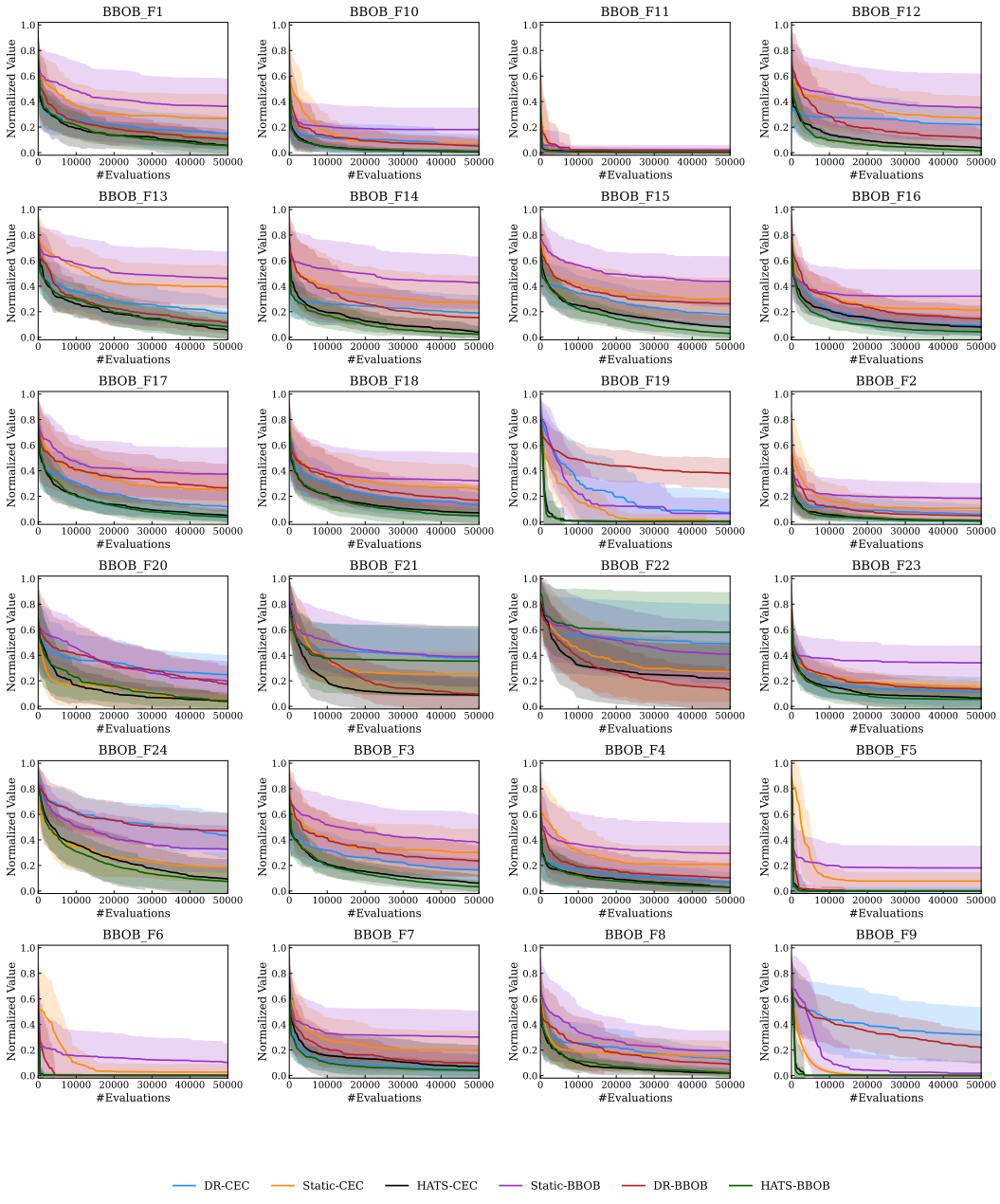


Figure 9: Optimization curve of different methods on BBOB.

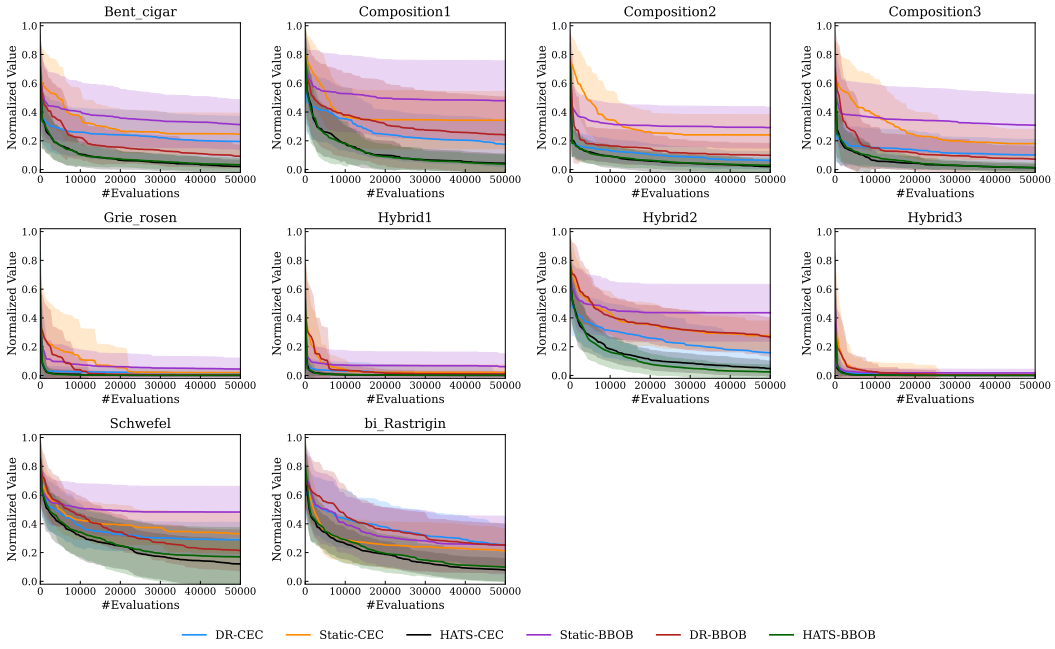


Figure 10: Optimization curve of different methods on CEC.

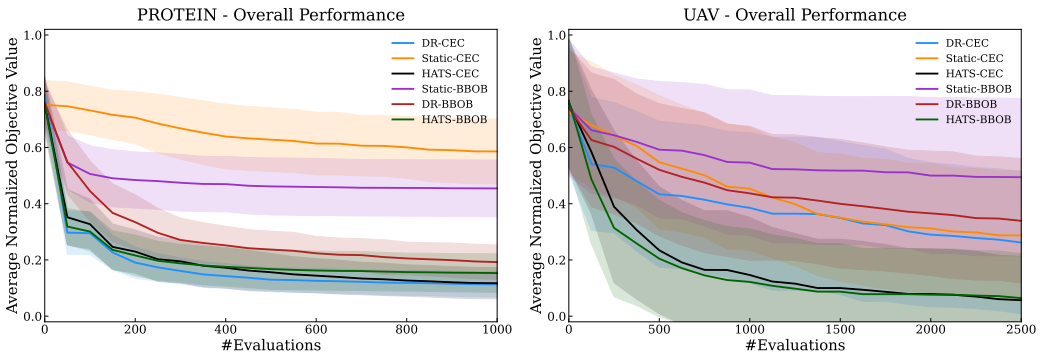


Figure 11: Optimization curve of different methods on Protein Docking and UAV.