Enhancing Zero-Shot Black-Box Optimization via Efficient Population Modeling, Interaction, and Stable Gradient Approximation

Muqi Han
Guangzhou Institute of Technology
Xidian University
Key Laboratory of Data and Intelligent System Security Ministry of Education
mqhan@stu.xidian.edu.cn

Xiaobin Li
School of Artificial Intelligence
Xidian Univeristy
22171214784@stu.xidian.edu.cn

Kai Wu*
School of Artificial Intelligence
Xidian Univeristy
kwu@xidian.edu.cn

Xiaoyu Zhang
School of Cyber Engineering
Xidian University
xiaoyuzhang@xidian.edu.cn

Handing Wang
School of Artificial Intelligence
Xidian Univeristy
hdwang@xidian.edu.cn

Abstract

Zero-shot optimization aims to achieve both generalization and performance gains on solving previously unseen black-box optimization problems over SOTA methods without task-specific tuning. Pre-trained optimization models (POMs) address this challenge by learning a general mapping from task features to optimization strategies, enabling direct deployment on new tasks.

In this paper, we identify three essential components that determine the effectiveness of POMs: (1) task feature modeling, which captures structural properties of optimization problems; (2) optimization strategy representation, which defines how new candidate solutions are generated; and (3) the feature-to-strategy mapping mechanism learned during pre-training. However, existing POMs often suffer from weak feature representations, rigid strategy modeling, and unstable training.

To address these limitations, we propose EPOM, an enhanced framework for pre-trained optimization. EPOM enriches task representations using a cross-attention-based tokenizer, improves strategy diversity through deformable attention, and stabilizes training by replacing non-differentiable operations with a differentiable crossover mechanism. Together, these enhancements yield better generalization, faster convergence, and more reliable performance in zero-shot black-box optimization.

1 Introduction

Black-box optimization (BBO) problems are ubiquitous in machine learning and engineering. The characteristic of BBO is that the algorithm can evaluate $f(\mathbf{x})$ for any solution $\mathbf{x}$; however, access to

*Corresponding author

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

additional information about f, such as Hessian matrices and gradients, is unavailable. BBO problems include hyperparameter optimization (HPO) [1], neuroevolution [2, 3, 4], neural architecture search (NAS) [5], and algorithm selection [6], etc.

Traditionally, solving BBO tasks requires expert-designed or heavily tuned optimization algorithms for each new problem, making the process inefficient and non-scalable. Thus, the paradigm of solving new optimization problems without any task-specific tuning is extremely valuable.

Meta-learning, or learning to learn [7], particularly in the context of meta-black-box optimization [8, 9] or learning to optimize [10], covers a wide range of scenarios. It improves the performance of the optimizer on the target task by pre-training it on similar tasks. However, they typically exhibit poor generalization to new and unseen tasks. Although some methods such as LES [11] and LGA [12] can generalize to novel settings, their performance still falls significantly short compared to state-of-the-art optimizers specifically designed for these settings.

To better highlight our contribution, we introduce the concept of zero-shot optimization, which aims to simultaneously achieve strong generalization and high performance on new optimization problems without requiring task-specific training. A particularly promising direction in this area is the development of pre-trained optimization models (POMs) [13]. These models are designed to learn general-purpose optimization behaviors in a wide variety of training tasks and effectively transfer that knowledge to previously unseen problems.

We view a POM as a function that maps task-specific features to optimization strategies. This process involves three key components: (1) **Task feature modeling**, which captures characteristics of the problem (e.g. normalized fitness values and centralized rankings); (2) **Optimization strategy representation**, which determines how candidate solutions are generated and refined; (3) **Mapping mechanism**, which learns the transformation from features to strategies during pre-training.

Although recent POMs have made progress, they remain limited in three areas:

- The expressiveness of task features is insufficient to generalize across diverse problem structures;
- The strategy generation process lacks flexibility and diversity;
- The training process suffers from gradient instability due to non-differentiable operations.

To overcome these limitations, we propose **EPOM (Enhanced Pretrained Optimization Model)**, which systematically enhances all three components:

- **Feature modeling** is improved by a cross-attention-based tokenizer that captures decision-variable-level information and encodes it into fixed-length task representations;
- **Strategy representation** is enhanced through deformable attention, enabling dynamic and diversity-aware interactions among population;
- **Training stability** is ensured by replacing sampling-based crossover operations with a differentiable, weighted-sum mechanism, leading to smoother gradient flow.

By jointly improving feature modeling, strategy generation, and training robustness, EPOM achieves superior generalization and performance in zero-shot black-box optimization. This work contributes to the growing body of meta-learning literature on learning-to-optimize methods, and offers a scalable pathway toward universal optimization agents.

2 Related Work

Manually Designed Population-Based BBO Algorithms. Traditional population-based black-box optimization (BBO) algorithms, such as genetic algorithms [14], evolution strategies [15], particle swarm optimization [16], and differential evolution [17], have been widely used to tackle optimization challenges. State-of-the-art methods like CMA-ES [18] and L-SHADE [19] rely heavily on expert knowledge and trial-and-error development, resulting in limited flexibility, high development costs, and suboptimal generalization [20].

Meta-Learned Population-Based BBO Algorithms. Meta-learned BBO algorithms improve generalizability through meta-learning, can be categorized by their training strategies. One category

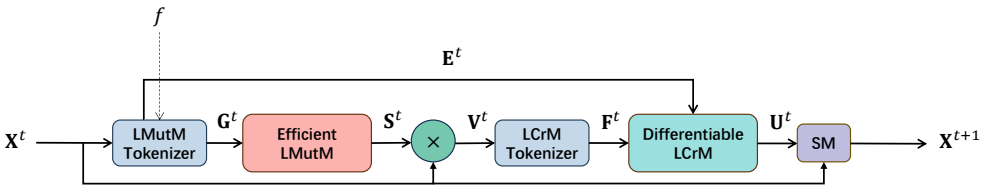


Figure 1: The overall framework of EPOM. The LMutM Tokenizer and LCrM Tokenizer are designed to generate input tokens for the Efficient LMutM and Differentiable LCrM, which are responsible for obtaining the mutant population $\mathbf{V}^t$ and the candidate population $\mathbf{U}^t$, respectively. Finally, the SM produces the offspring population based on $\mathbf{X}^t$ and $\mathbf{U}^t$.

employs bi-level optimization [21], as seen in [22], which adapts optimizers to specific task classes. LES [11] and LGA [12] use neuroevolution with self-attention mechanisms, but their reliance on external search algorithms such as OPENAI-ES [23] leads to slow training and limited scalability. Meta-MOGA [24] addresses multi-objective BBO problems by parameterizing mutation and crossover operators with multi-head self-attention and the selection operator with an MLP, whose parameters are optimized via an evolutionary algorithm. Another adopts reinforcement learning (RL) [25], for example, Shala et al. [26] use RL to meta-learn policies for adjusting CMA-ES parameters, however, suffers from training instability. Moreover, Q-Mamba [4] offline learns a meta-level policy for dynamic algorithm configuration over an optimization task distribution, while ConfigX [27] meta-learns a universal configuration policy over a modular evolutionary algorithm space. Alternatively, EvoTF [28] incorporates algorithm distillation, but requires large-scale datasets and struggles to surpass the performance of its source algorithms. Besides these, B2Opt [29] and POM [13] leverage self-supervised learning, however, B2Opt is restricted by iteration limits, and POM faces challenges in scaling with model size.

LLMs for Optimization. In the field of optimization, a series of optimization methods based on Large Language Models (LLMs) have emerged. They are widely applied in areas such as NP-hard problems [30, 31], algorithm evolution [32, 33, 34, 35, 36], reward design [37], and Neural Architecture Search (NAS) [38, 39]. LLMs play a crucial role in optimization, especially in sampling new solutions. However, as stated in [40], their optimization strategies rely on externally-introduced natural selection mechanisms, limiting their effectiveness in numerical optimization scenarios. For example, LLaMoCo [41] and EoH [42] use LLMs to generate code for optimization problems. The former depends on well-designed instructions and prompts, while the latter faces high evaluation costs, restricting practical applications. Moreover, TNPs [43], ExPT [44], and LICO [45] use transformer structures to address the BBO problem. Nevertheless, TNPs relies on the context information of the target problem, and neither ExPT nor LICO can be directly applied to tasks with different dimensions from the training ones. In summary, although LLM-based optimization methods have made progress, compared with pre-trained BBO methods, they have insufficient cross-task solution-generation capabilities and poor generalizability, which remain key issues to be overcome in the optimization field.

3 EPOM

A black-box optimization problem can be transformed as a minimization problem, and constraints may exist for corresponding solutions: $\min_{\mathbf{x}} f(\mathbf{x}), s.t.\ x_i \in [l_i, u_i]$, where $\mathbf{x} = (x_1, x_2, \cdots, x_d)$ represents the solution of optimization problem f, the lower and upper bounds $\mathbf{l} = (l_1, l_2, \cdots, l_d)$ and $\mathbf{u} = (u_1, u_2, \cdots, u_d)$, and d is the dimension of $\mathbf{x}$. A population consists of n individuals, denoted as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^{\mathrm{T}}$.

3.1 Review of POM

3.1.1 Main Parts

The POM primarily consists of three modules: Learned Mutation Module (**LMutM**), Learned Crossover Module (**LCrM**) [2], and Selection Module (**SM**).

1) LMutM. LMutM generates the mutant population $\mathbf{V}^t$ through the multi-head self-attention mechanism [46]:

$$\mathbf{S}^t \leftarrow \mathrm{LMutM}(\mathbf{H}^t), \quad \mathbf{V}^t = \mathbf{S}^t \times \mathbf{X}^t, \tag{1}$$

where $\mathbf{V}^t, \mathbf{X}^t \in \mathbb{R}^{n\times d}$, $\mathbf{H}^t \in \mathbb{R}^{n\times 2}$ and $\mathbf{S}^t \in \mathbb{R}^{n\times n}$. Here, n denotes the population size and d denotes the problem dimension. $\mathbf{X}^t$ represents the parent population, $\mathbf{H}^t = [\mathbf{h}_1^t, \mathbf{h}_2^t, \ldots, \mathbf{h}_n^t]$ is obtained by tokenizing $\mathbf{X}^t$ and serves as the input to LMutM, where each token $\mathbf{h}_i^t$ encodes the mutation features of the corresponding individual $\mathbf{x}_i^t$, including: 1) $\hat{f}_i^t$: the normalized fitness $f(\mathbf{x}_i^t)$ of $\mathbf{x}_i^t$; 2) $\hat{r}_i^t$: the centralized ranking of $\mathbf{x}_i^t$. $\mathbf{S}^t$ is the weight matrix computed by the attention mechanism.

2) LCrM. LCrM generates the crossover rate $\mathbf{cr}^t = [cr_1^t, cr_2^t, \cdots, cr_n^t]$ ($\mathbf{cr}^t \in \mathbb{R}^{1\times n}$, cr_i^t is the crossover rate of the i-th individual) by an FFN [46] and performs the crossover operation to obtain the candidate population $\mathbf{U}^t \in \mathbb{R}^{n\times d}$:

$$\mathbf{cr}^t \leftarrow \mathrm{LcrM}(\mathbf{Z}^t),\ \mathbf{U}^t = gumbel\text{-}softmax([\mathbf{X}^t, \mathbf{V}^t], \mathbf{cr}^t) \tag{2}$$

Where $\mathbf{Z}^t = [\mathbf{z}_1^t, \mathbf{z}_2^t, \cdots, \mathbf{z}_n^t] \in \mathbb{R}^{n\times 3}$ is the crossover feature matrix utilized by LCrM. Each token $\mathbf{z}_i^t$ encodes the crossover features of the i-th candidate individual $\mathbf{u}_i^t$, including: 1) $\hat{f}_i^t$: the normalized fitness $f(\mathbf{x}_i^t)$ of $\mathbf{x}_i^t$; 2) $\hat{r}_i^t$: the centralized ranking of $\mathbf{x}_i^t$; 3) sim_i^t: the cosine similarity between x_i^t and v_i^t. The $gumbel\text{-}softmax$ trick [47], which independently operates across dimensions by taking $[x_{i,j}^t, v_{i,j}^t]$ $(j = 1, 2, \ldots, d)$ as two categories and $[cr_i^t, 1 - cr_i^t]$ as their respective probabilities, provides a differentiable approximation of the crossover operation.

3) SM. Finally, SM [48], a *1-to-1* selection strategy is executed between $\mathbf{U}^t$ and $\mathbf{X}^t$ to produce the next-generation population $\mathbf{X}^{t+1}$, where $\mathbf{X}^{t+1} = \mathrm{SM}(\mathbf{X}^t, \mathbf{U}^t)$.

3.1.2 Issues

Issues with LMutM. 1) Insufficient and non-diverse strategies: The mutation strategies are characterized by the weight matrix $\mathbf{S}^t$ derived from the self-attention mechanism, where each individual interacts with the entire population. However, such global interaction can hinder the efficiency and diversity of the mutation process. 2) Training instability: The mask operation randomly masks parts of $\mathbf{S}^t$, which may obscure significant signals, thereby slowing down convergence and causing training instability. 3) Insufficient input features: The input $\mathbf{H}^t$ of LMutM only includes fitness and ranking information, overlooking the distribution characteristics of the population. This limits the POM's convergence and generalization abilities.

Issues with LCrM. LCrM uses the gumbel softmax trick to address the non-differentiability of crossover operation. However, the imprecise gradient approximation can lead to instability during model training.

3.2 Proposed EPOM

Our goal is to improve the strategic efficiency and trainability of POM. We have redesigned the LMutM and LCrM of POM and proposed the EPOM, as shown in Fig. 1.

3.2.1 Overall Architecture

First, the LMutM Tokenizer encodes $\mathbf{X}^t$ into fixed-dimensional tokens $\mathbf{E}^t$ using a cross-attention mechanism [46], similar in spirit to prior cross-attention-based tokenization methods [49]. These tokens are then concatenated with the tokens $\mathbf{H}^t$ from the original LMutM in POM to form $\mathbf{G}^t$.

[2] To avoid potential confusion between the abbreviations of *Learned Mutation Module* and *Learned Crossover Module* (formerly LMM and LCM) with *Large Multimodal Models* and *Large Context Models*, we have revised their abbreviations to *LMutM* and *LCrM*, respectively.

Subsequently, the Efficient LMutM takes $\mathbf{G}^t$ as input and generates the mutant population $\mathbf{V}^t$ through two self-attention mechanisms. The first self-attention mechanism identifies, for each parent individual, the subset of population members it should attend to. The second mechanism, a deformable self-attention module [50], computes the attention weight vector $\mathbf{s}_i^t$ for each individual i based on the keys corresponding to its selected individuals $\mathbf{X}_i^t$. Then, the mutant individual $\mathbf{v}_i^t$ is obtained by multiplying $\mathbf{s}_i^t$ with $\mathbf{X}_i^t$. Next, the LCrM Tokenizer processes $\mathbf{V}^t$ in the same manner as the LMutM Tokenizer to generate $\mathbf{F}^t$, and then feeds $[\mathbf{E}^t, \mathbf{F}^t]$ into the Differentiable LCrM. The Differentiable LCrM produces the candidate population $\mathbf{U}^t$ through two cross-attention mechanisms. The first mechanism determines, for each parent individual, the subset of mutant individuals from $\mathbf{V}^t$ it should attend to. The second mechanism, a deformable cross-attention module [50], computes the attention weights between each parent and the mutant individuals it attends to. The candidate individual is then obtained as a weighted combination of the parent and the attended mutants. Finally, $\mathbf{X}^t$ and $\mathbf{U}^t$ are passed to the Selection Module (SM) to generate the offspring population $\mathbf{X}^{t+1}$.

3.2.2 Tokenizer

1) LMutM Tokenizer. In POM, the dimensions of the inputs ($\mathbf{H}^t$ or $\mathbf{Z}^t$) to LMutM and LCrM are 2 and 3, respectively. However, such low-dimensional representations may fail to adequately capture the characteristics of the optimization landscape, limiting the information available to LMutM and LCrM. To address issue 3) in Section 3.1.2, we design a tokenizer that maps each individual in the population from the problem dimension d to a fixed dimension $\hat{d}_t$ using a cross-attention mechanism:

$$
\begin{aligned}
\hat{\mathbf{x}}_i^t &= \begin{cases} \mathbf{x}_i^t, & \text{if} \quad d \quad \text{is even} \\ append(\mathbf{x}_i^t, 0), & \text{otherwise} \end{cases} \\
\hat{\mathbf{K}}_T &= reshape(\hat{\mathbf{x}}_i^t, (-1, 2)) \\
\mathbf{K}_T &= \hat{\mathbf{K}}_T \times \mathbf{W}_{1K} + \mathbf{b}_{1K}, \quad \mathbf{V}_T = \hat{\mathbf{K}}_T \times \mathbf{W}_{1V} + \mathbf{b}_{1V} \\
\hat{\mathbf{e}}_i^t &= Tanh(\frac{\mathbf{Q}_T \times \mathbf{K}_T^{\mathrm{T}}}{\sqrt{d_{1V}}}) \times \mathbf{V}_T, \quad \mathbf{e}_i^t = reshape(\hat{\mathbf{e}}_i^t, (-1,))
\end{aligned}
\tag{3}
$$

where $\mathbf{x}_i^t \in \mathbb{R}^d$ is the i-th individual in the t-th generation, The $append$ and $reshape$ operations are functionally equivalent to their PyTorch counterparts. $\mathbf{W}_{1K} \in \mathbb{R}^{2\times d_{1QK}}$, $\mathbf{W}_{1V} \in \mathbb{R}^{2\times d_{1V}}$, $\mathbf{b}_{1K}$ and $\mathbf{b}_{1V}$ are bias vectors with dimensions d_{1QK} and d_{1V}, respectively. $\mathbf{Q}_T \in \mathbb{R}^{d_T \times d_{1QK}}$ represents the shared query across the population. All parameters, including $\mathbf{W}_{1K}$, $\mathbf{W}_{1V}$, $\mathbf{b}_{1K}$, $\mathbf{b}_{1V}$, and $\mathbf{Q}_T$, are learnable.

By using Eq.(3), we obtain a $d_T \times d_{1V}$-dimensional token for each individual $\mathbf{x}_i^t$, and subsequently construct the population token matrix $\mathbf{E}^t = [\mathbf{e}_1^t, \mathbf{e}_2^t, \cdots, \mathbf{e}_n^t]$. Then we concatenate $\mathbf{E}^t$ with the token matrix $\mathbf{H}^t$ from the LMutM of POM to form $\mathbf{G}^t = [\mathbf{H}^t | \mathbf{E}^t]$, where $\mathbf{G}^t \in \mathbb{R}^{n\times \hat{d}_t}$ and $\hat{d}_t = 2 + d_T \times d_{1V}$.

2) LCrM Tokenizer. Similarly to the LMutM Tokenizer, the LCrM Tokenizer transforms the mutant population $\mathbf{V}^t$ into tokens $\mathbf{F}^t$ with a fixed dimension d_t according to Eq. (3), where $d_t = d_T \times d_{1V}$.

3.2.3 Efficient LMutM

For issues 1) and 2) in Section 3.1.2, we introduce Efficient LMutM, which leverages an deformable attention mechanism to allow individuals to adaptively select a subset of individuals for information exchange. This approach avoids global interactions, improving the model's performance and diversity. Additionally, we incorporate a dropout layer that is active during both the training and testing phases within the EPOM, introducing more randomness to the model.

We adapt two self-attention mechanism in the Efficient LMutM. The first self-attention identifies the individuals within the parent population that each individual should atttend to according to the input

tokens $\mathbf{G}^t$. The attended individuals can be computed using Eq. (4).

$$\begin{aligned}
&\mathbf{Q}_{1M} = \mathbf{G}^t \times \mathbf{W}_{2Q} + \mathbf{b}_{2Q}, \quad \mathbf{K}_{1M} = \mathbf{G}^t \times \mathbf{W}_{2K} + \mathbf{b}_{2K} \\
&\mathbf{V}_{1M} = \mathbf{G}^t \times \mathbf{W}_{2V} + \mathbf{b}_{2V}, \quad \hat{\mathbf{N}}_t = Softmax(\frac{\mathbf{Q}_{1M} \times \mathbf{K}_{1M}^{\mathrm{T}}}{\sqrt{d_{2V}}}) \times \mathbf{V}_{1M}, \\
&\mathbf{N}_t = top\text{-}p\text{-}sampling(\hat{\mathbf{N}}_t|p)
\end{aligned} \tag{4}$$

where $\mathbf{W}_{2Q}, \mathbf{W}_{2K} \in \mathbb{R}^{\hat{d}_t \times d_{2QK}}$, $\mathbf{b}_{2Q}, \mathbf{b}_{2K} \in \mathbb{R}^{d_{2QK}}$, $\mathbf{W}_{2V} \in \mathbb{R}^{\hat{d}_t \times d_{out}}$, and $\mathbf{b}_{2V} \in \mathbb{R}^{d_{out}}$ denote learnable parameters, and d_{out} is the fixed output dimension. If the population size $n \leq d_{out}$, the first n dimensions are used to represent the selection probability of each individual. Otherwise, selection is restricted to the first d_{out} individuals. The $top\text{-}p\text{-}sampling$ function selects $top\text{-}m_i$ individuals via top-p sampling [51], ensuring $\sum_{i=1}^{m} prob(\mathbf{x}_i^t) = p$. Applying top-p sampling to $\hat{\mathbf{N}}_t$ yields the index matrix $\mathbf{N}_t \in \mathbb{R}^{n \times k}$, where $k = \max\{m_i\}_{i=1}^{n}$, representing the selected individuals for each member of the population.

The second mechanism, deformable self-attention [50], computes the attention weight vector $\mathbf{s}_i^t$ for each individual i. Then, we derive the mutant population $\mathbf{V}^t = [\mathbf{v}_1^t, \mathbf{v}_2^t, \cdots, \mathbf{v}_n^t]$, where $\mathbf{v}_i^t = \mathbf{s}_i^t \times \mathbf{X}_i^t$.

Table 1: Training Functions for EPOM and POM. $z_i = x_i - \omega_i$.

ID	Functions	Range
TF1	$\sum_i \lvert x_i - \omega_i \rvert$	$x \in [-10, 10], \omega \in [-10, 10]$
TF2	$\sum_i \lvert (x_i - \omega_i) + (x_{i+1} - \omega_{i+1}) \rvert + \sum_i \lvert x_i - \omega_i \rvert$	$x \in [-10, 10], \omega \in [-10, 10]$
TF3	$\sum_i z_i^2$	$x \in [-100, 100], \omega \in [-50, 50]$
TF4	$\max\{\lvert z_i \rvert, 1 \leq i \leq d\}$	$x \in [-100, 100], \omega \in [-50, 50]$
TF5	$\sum_{i=1}^{d-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$	$x \in [-100, 100], \omega \in [-50, 50]$

3.2.4 Differentiable LCrM

To resolve the issue in LCrM, we propose a differentiable LCrM, replacing the gumbel-softmax trick with a weighted summation approach. Specifically, the differentiable LCrM takes $[\mathbf{E}^t, \mathbf{F}^t]$ as inputs and generates the crossover weight $\mathbf{cw}^t \in \mathbb{R}^{n \times 2}$. The i-th candidate individual can be computed by Eq. (5).

$$\mathbf{u}_i^t = cw_{i,0}^t \cdot \mathbf{x}_i^t + cw_{i,1}^t \cdot mean(\mathbf{V}_i^t), \tag{5}$$

where $\mathbf{V}_i^t \in \mathbb{R}^{j \times d}$ denotes j mutant individuals selected by parent individual $\mathbf{x}_i^t$. The $mean$ function computes the average along each dimension of the input matrix. Through Eq. (5), we obtain the candidate population $\mathbf{U}^t = [\mathbf{u}_1^t, \mathbf{u}_2^t, \cdots, \mathbf{u}_n^t]$. The remaining task is handled by **SM**, which peforms a one-to-one selection strategy to generate the offspring population $\mathbf{X}^{t+1}$.

The training set, and training strategy of EPOM are consistent with those of POM, as detailed in the Appendix A. After EPOM has been trained, we use Algorithm 1 to solve new problems.

3.2.5 Loss Function

The loss function of POM for the i-th training function consists of two components: 1) the difference in the mean fitness of adjacent generations, normalized l_i^1; and 2) the mean standard deviation across the dimensions of the population l_i^2. Ideally, l_i^1 encourages POM to converge toward the optimal point, and l_i^2 encourages population diversity. However, in practical applications, we found that l_i^2 has a limited effect on population diversity, therefore, we replaced it with crowding distance [52]. The loss function of EPOM for the i-th training function can be discribed as Eq.(6).

Algorithm 1 Driving EPOM to Solve Problem

Input: Generations T, population size n, BBO problem f.
Output: The optimal $\mathbf{X}^T$ found.

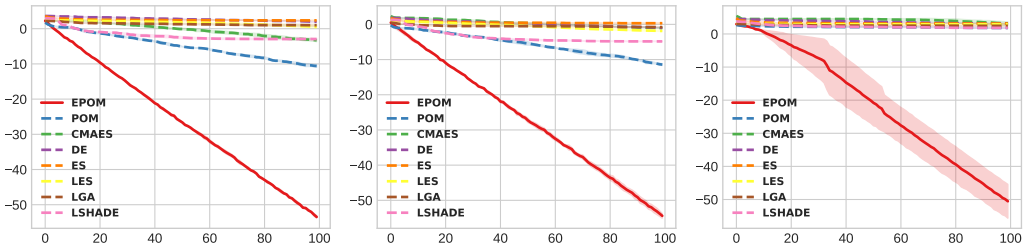
```
EPOM loads the trained parameter θ.
Randomly sample an initial population X^0 of size n.
for t = 0, 1, ..., T − 1 do
    Construct E^t based on X^t and f using LMutM Tokenizer.
    G^t ← [H^t|E^t].
    S^t ← Efficient LMutM(G^t).
    V^t ← S^t × X^t.
    Build F^t based on V^t using LCrM Tokenizer.
    U^t ←Differentiable LCrM(E^t, F^t).
    X^{t+1} ← SM(X^t, U^t).
end for
```

$$
\begin{aligned}
l_i^t &= l_i^1 + \alpha l_{cd} \\
&= \frac{\frac{1}{|\mathbf{X}^t|}\sum_{\mathbf{x}\in\mathbf{X}^t} f_i(\mathbf{x}|\omega^i) - \frac{1}{|\mathbf{X}^{t-1}|}\sum_{\mathbf{x}\in\mathbf{X}^{t-1}} f_i(\mathbf{x}|\omega^i)}{\left|\frac{1}{|\mathbf{X}^{t-1}|}\sum_{\mathbf{x}\in\mathbf{X}^{t-1}} f_i(\mathbf{x}|\omega^i)\right|} + \frac{\alpha}{|\mathbf{X}^t|}\sum_{i=2}^{n-1}\frac{f_i(\mathbf{x}_{i+1:n}^t|\omega^i) - f_i(\mathbf{x}_{i-1:n}^t|\omega^i)}{f_i(\mathbf{x}_{n:n}^t|\omega^i) - f_i(\mathbf{x}_{1:n}^t|\omega^i)}
\end{aligned} \tag{6}
$$

where $\mathbf{x}_{i:n}^t$ denotes the individual ranked i-th in ascending order (for a minimization problem) in the t-th generation. α is a hyperparameter, and we found that setting it to 0.1 better balances the convergence of the population toward the optimal value and its diversity.

4 Experiments

4.1 Experimental Setup



(a) F1 d=30 (b) F2 d=30 (c) F3 d=30

Figure 2: Part of the convergence curves of EPOM and other baselines. It shows the logarithmic convergence curve of these algorithms on functions in BBOB. See Appendix C for more details.

Baselines. Our core approach is a population-based pretraining BBO algorithm; therefore, we focus on comparisons with other population-based methods rather than non-population methods such as Bayesian optimization. Moreover, we do not compare with LLM-based approaches [30, 31, 32, 33, 34, 37, 38, 39], as these methods lack zero-shot optimization capabilities.

Heuristic Population-based BBO Algorithms. We compare against the following established population-based BBO algorithms: (a) **DE (DE/rand/1/bin)** [53]: A classic numerical optimization algorithm. (b) **ES (μ,λ-ES)**: A well-known variant of evolutionary strategies. (c) **CMA-ES** [18]: Often considered the state-of-the-art method for continuous domain optimization in challenging settings (e.g., ill-conditioned, non-convex, multimodal problems). (d) **L-SHADE** [19]: A state-of-the-art variant of DE.

Table 2: Results of BBOB. EPOM is trained on TF1-TF5 with d =10. The best results are indicated in bold, and the suboptimal results are underlined. The "+/=/-" at the bottom indicates that EPOM's performance is better/same/worse than the given baseline on the given dimension setting and function.

d	F	EPOM	POM	ES	DE	CMA-ES	LSHADE	LES	LGA
30	F1	**4.92E-54(5.26E-54)**	3.72E-11(3.95E-11)	2.30E+02(1.36E+01)	9.46E+01(1.17E+01)	7.79E-04(8.97E-04)	1.28E-03(7.36E-04)	4.93E+00(4.94E+00)	1.13E+01(7.12E+00)
	F2	**8.39E-55(8.27E-55)**	4.69E-12(3.85E-12)	2.18E+00(5.24E-01)	1.10E-01(4.35E-03)	8.45E-02(1.64E-02)	1.47E-05(2.12E-06)	1.45E-02(6.38E-03)	1.75E-01(5.88E-02)
	F3	**1.64E-44(2.84E-44)**	6.57E+01(1.49E+01)	1.41E+03(1.26E+02)	1.02E+03(5.47E+01)	2.47E+03(2.39E+03)	7.12E+01(9.31E+00)	8.10E+02(1.27E+02)	2.82E+02(2.03E+01)
	F4	**1.92E-40(3.31E-40)**	6.95E+01(4.28E+01)	3.35E+03(6.76E+02)	1.99E+03(5.61E+02)	2.21E+02(1.02E+00)	1.04E+02(4.24E+00)	6.11E+02(1.50E+02)	3.76E+02(3.85E+01)
	F5	3.28E+02(5.05E+01)	3.61E+01(3.13E+01)	5.91E+01(4.64E+00)	1.32E+00(2.70E-01)	0.00E+00(0.00E+00)	0.00E+00(0.00E+00)	1.99E+02(5.46E+01)	0.00E+00(0.00E+00)
	F6	**1.90E-47(2.90E-47)**	1.69E-09(1.62E-09)	3.97E+02(9.66E+00)	5.29E+02(1.74E+02)	8.99E-02(6.01E-03)	1.54E-01(8.94E-02)	1.11E+01(9.11E+00)	2.25E+01(6.53E+00)
	F7	**6.83E-37(7.89E-37)**	3.78E-13(3.38E-13)	1.61E+03(5.19E+01)	7.62E+03(9.02E+02)	3.44E+00(7.67E-01)	1.25E+01(6.46E+00)	1.20E+02(4.80E+01)	6.97E+01(2.45E+01)
	F8	**0.00E+00(0.00E+00)**	6.23E-06(4.98E-06)	4.21E+05(6.85E+04)	3.26E+05(4.66E+04)	3.15E+02(3.90E+02)	3.08E+01(3.53E+00)	3.01E+03(2.79E+03)	1.63E+03(3.60E+02)
	F9	1.86E+02(8.10E-01)	1.60E+02(1.94E+01)	4.44E+05(8.88E+04)	7.06E+05(9.64E+04)	**4.17E+01(1.20E+01)**	5.85E+01(5.42E+01)	2.37E+03(8.49E+02)	1.38E+03(5.39E+02)
	F10	**5.58E-48(9.24E-48)**	2.24E+03(2.11E+03)	3.56E+06(1.48E+06)	2.33E+07(6.30E+06)	3.39E+05(1.18E+05)	1.16E+04(5.72E+03)	7.58E+04(3.58E+04)	2.67E+05(6.29E+04)
	F11	**3.99E-48(4.42E-48)**	7.38E+00(8.33E-01)	1.59E+03(5.31E+02)	5.73E+03(8.62E+02)	5.55E+03(1.21E+03)	1.53E+02(1.13E+02)	2.36E+02(3.17E+01)	3.95E+02(1.72E+02)
	F12	**1.46E-46(1.19E-46)**	5.13E-04(4.11E-04)	4.18E+09(4.62E+08)	1.37E+10(8.87E+08)	2.91E+11(2.89E+10)	4.10E+05(4.53E+05)	1.04E+08(7.97E+07)	9.59E+07(3.51E+07)
	F13	**3.61E-25(2.49E-25)**	6.76E-05(3.90E-05)	1.57E+03(6.29E+01)	1.07E+03(8.97E+01)	9.66E+00(1.62E+00)	2.44E+00(1.41E+00)	8.61E+01(4.07E+01)	2.40E+02(4.05E+01)
	F14	**8.36E-29(7.43E-29)**	2.29E-04(6.42E-05)	9.04E+01(1.08E+01)	5.84E+02(8.93E+01)	1.92E+00(1.14E+00)	4.38E-02(2.39E-02)	6.01E+00(1.88E+00)	4.02E+00(9.65E-01)
	F15	**2.74E-03(4.75E-03)**	7.84E+01(2.37E+01)	1.62E+03(1.29E+02)	4.31E+03(6.26E+02)	4.27E+04(3.66E+04)	1.16E+02(1.08E+01)	8.73E+02(2.02E+02)	2.84E+02(2.33E+01)
	F16	3.54E+01(5.25E+00)	2.55E+01(1.42E+00)	4.62E+01(4.62E+00)	5.44E+01(5.74E+00)	3.18E+01(3.66E+00)	1.64E+01(5.59E+00)	**7.17E+00(1.16E+00)**	3.24E+01(1.07E+00)
	F17	**1.46E-27(5.82E-28)**	2.79E-05(1.29E-05)	2.47E+01(7.36E+00)	2.43E+01(4.93E+00)	3.78E-01(8.36E-02)	4.67E-01(9.78E-02)	9.74E+00(4.15E+00)	2.21E+00(3.62E-01)
	F18	**4.56E-27(3.83E-27)**	1.30E-01(1.86E-01)	9.84E+01(1.68E+01)	1.19E+02(4.66E+01)	2.26E+00(5.51E-01)	9.34E-01(3.55E-01)	3.43E+01(1.25E+01)	1.21E+01(3.22E+00)
	F19	**4.41E+00(6.06E-01)**	4.82E+00(2.55E-01)	5.43E+01(4.16E+00)	5.00E+01(1.17E+01)	5.94E+00(4.07E-01)	5.44E+00(4.67E-01)	1.61E+01(2.59E+00)	7.06E+00(2.56E-01)
	F20	-1.69E+00(4.86E+00)	-1.32E+01(3.32E+00)	1.25E+05(3.14E+04)	1.08E+05(2.55E+04)	3.27E+00(1.03E-01)	3.13E+00(9.10E-02)	**-2.72E+01(1.26E+01)**	9.09E+01(1.05E+02)
	F21	7.63E+01(2.03E+00)	3.36E+01(1.03E+01)	8.80E+01(6.16E-01)	8.56E+01(7.64E-01)	**2.89E+00(5.34E-02)**	1.44E+01(1.26E+01)	1.99E+01(9.86E+00)	9.98E+00(2.89E+00)
	F22	7.35E+01(1.47E+00)	1.57E+01(1.03E+01)	8.92E+01(1.82E+00)	8.57E+01(6.39E-01)	1.96E+00(5.02E-03)	**1.14E+00(7.17E-01)**	1.68E+01(4.44E+00)	9.91E+00(5.50E+00)
	F23	3.26E+00(1.70E-01)	3.68E+00(2.22E-01)	3.61E+00(5.29E-01)	3.49E+00(3.33E-01)	3.75E+00(6.23E-01)	3.38E+00(3.16E-01)	**3.01E+00(4.00E-01)**	4.38E+00(1.38E-01)
	F24	3.33E+02(1.22E+01)	2.81E+02(2.11E+01)	2.82E+02(2.17E+01)	3.20E+02(3.56E+01)	2.08E+02(2.20E+00)	**1.84E+02(3.16E+00)**	7.08E+02(7.80E+01)	3.69E+02(4.32E+01)
100	F1	**2.33E-53(2.03E-53)**	1.48E-11(8.37E-12)	1.60E+03(3.45E+01)	4.62E+03(5.31E+02)	4.34E+01(4.29E+00)	1.64E+01(8.78E-01)	2.20E+02(4.99E+01)	1.13E+02(2.07E+01)
	F2	**4.26E-55(5.23E-55)**	4.70E-12(2.55E-12)	4.08E+01(6.19E+00)	2.24E+01(3.00E+00)	4.17E+01(9.20E+00)	7.58E-02(4.38E-02)	4.56E+00(1.29E+00)	3.28E+00(5.75E-01)
	F3	3.36E-01(5.82E-01)	**1.07E-09(1.16E-09)**	1.06E+04(7.18E+02)	4.77E+04(2.87E+03)	3.24E+04(8.39E+03)	8.71E+02(9.08E+01)	2.72E+03(1.89E+02)	1.82E+03(5.85E+01)
	F4	5.02E+03(8.69E+03)	**1.39E-07(1.93E-07)**	6.28E+04(7.31E+03)	2.96E+05(2.45E+04)	3.77E+03(3.11E+02)	1.29E+03(1.69E+02)	5.15E+03(1.25E+03)	2.49E+03(1.51E+02)
	F5	1.82E+03(4.32E+01)	3.04E+02(4.14E+01)	2.03E+01(1.42E+01)	4.64E+00(2.13E+00)	1.63E+02(2.83E+02)	**3.98E+00(4.69E+00)**	1.30E+03(1.07E+02)	4.05E+00(4.50E+00)
	F6	**4.82E-46(8.34E-46)**	9.32E-10(5.17E-10)	2.46E+03(2.04E+02)	9.15E+03(2.22E+02)	2.65E+02(1.05E+02)	4.00E+01(5.60E+00)	4.37E+02(4.98E+01)	2.09E+02(1.11E+01)
	F7	**9.29E-30(1.61E-29)**	2.42E-13(1.55E-13)	1.11E+04(2.21E+03)	6.11E+04(4.18E+03)	2.79E+03(3.55E+02)	1.96E+02(7.71E+01)	1.43E+03(3.93E+02)	9.43E+02(3.33E+02)
	F8	**0.00E+00(0.00E+00)**	3.01E-08(1.57E-08)	2.09E+07(2.75E+05)	1.60E+08(2.16E+07)	6.06E+04(2.47E+04)	1.35E+04(7.05E+03)	2.40E+05(1.44E+04)	9.43E+04(6.50E+04)
	F9	6.43E+02(9.70E-02)	**6.41E+02(8.48E-01)**	1.97E+07(1.59E+06)	2.18E+08(2.65E+07)	1.12E+05(8.38E+04)	4.06E+03(9.38E+02)	3.71E+05(1.73E+04)	1.07E+05(3.15E+04)
	F10	**1.07E-20(1.85E-20)**	2.34E+01(4.04E+01)	5.73E+07(1.15E+07)	3.29E+08(1.13E+07)	7.27E+07(4.91E+07)	4.19E+05(3.99E+04)	2.82E+06(1.24E+06)	3.83E+06(6.94E+05)
	F11	**9.46E-03(1.64E-02)**	1.71E+01(1.48E+00)	4.63E+03(5.42E+02)	2.41E+04(2.95E+02)	3.25E+04(5.30E+03)	4.59E+02(8.92E+01)	7.82E+02(4.67E+01)	1.27E+03(2.04E+02)
	F12	**4.95E-47(1.95E-47)**	1.43E-04(1.06E-04)	4.15E+10(1.75E+09)	4.86E+11(8.89E+10)	1.91E+12(5.61E+11)	9.01E+08(4.73E+08)	3.83E+09(3.59E+08)	2.12E+09(1.32E+09)
	F13	**1.52E-24(2.32E-24)**	7.23E-05(7.37E-05)	4.18E+03(8.22E+01)	6.65E+03(4.61E+02)	6.35E+02(1.15E+02)	3.89E+02(6.17E+01)	1.53E+03(1.26E+02)	9.26E+02(7.82E+01)
	F14	**1.49E-27(1.98E-27)**	9.07E-05(6.57E-05)	4.51E+02(6.12E+01)	3.85E+03(5.14E+02)	4.15E+02(6.83E+01)	7.45E+00(3.08E+00)	3.57E+01(6.65E+00)	4.11E+01(5.20E+00)
	F15	**6.06E-47(1.05E-46)**	4.88E+02(4.17E+01)	9.88E+03(7.02E+02)	6.86E+04(8.80E+03)	3.37E+04(1.93E+04)	1.05E+03(9.66E+01)	3.61E+03(2.92E+02)	1.65E+03(1.35E+01)
	F16	4.92E+01(1.43E+00)	4.72E+01(4.72E-01)	8.41E+01(3.87E+00)	1.90E+02(1.40E+01)	5.36E+01(3.48E+00)	3.44E+01(3.21E+00)	**1.29E+01(6.39E-01)**	5.58E+01(1.67E+00)
	F17	**1.03E-26(1.74E-26)**	5.50E-07(2.32E-07)	1.26E+03(3.78E+02)	1.77E+04(8.83E+02)	5.71E+00(1.24E+00)	2.61E+00(9.12E-02)	2.10E+01(2.67E+00)	1.20E+01(1.33E+00)
	F18	**6.80E-26(1.10E-25)**	5.94E-06(4.45E-06)	1.73E+03(1.13E+02)	2.66E+04(3.33E+03)	2.65E+01(4.37E+00)	1.06E+01(1.25E+00)	5.66E+01(1.16E+01)	4.16E+01(5.48E+00)
	F19	**1.62E+00(1.29E+00)**	6.74E+00(4.70E-01)	5.37E+02(5.46E+01)	5.32E+03(2.05E+03)	1.08E+01(1.71E+00)	8.95E+00(2.98E-01)	2.75E+01(3.36E+00)	1.30E+01(1.38E+00)
	F20	-1.63E+00(6.36E+00)	**-5.08E+00(7.15E-01)**	1.56E+06(8.58E+04)	5.16E+06(5.28E+05)	3.98E+04(1.36E+04)	1.70E+03(3.56E+02)	4.66E+04(2.02E+04)	2.56E+04(7.96E+03)
	F21	7.75E+01(1.06E+00)	4.03E+01(7.57E+00)	2.10E+02(4.08E+01)	1.22E+03(2.28E+02)	6.56E+01(1.25E+01)	**1.37E+01(3.04E+00)**	7.62E+01(7.83E-01)	7.35E+01(9.27E-01)
	F22	8.28E+01(1.01E-01)	5.95E+01(1.27E+00)	2.33E+02(1.93E+01)	1.46E+03(4.77E+02)	7.02E+01(2.84E+00)	**3.12E+01(1.82E+01)**	7.62E+01(5.88E+00)	8.01E+01(7.78E+00)
	F23	**4.35E+00(9.52E-01)**	4.83E+00(2.54E-01)	5.14E+00(2.01E-01)	4.84E+00(5.12E-01)	5.23E+00(3.62E-01)	5.12E+00(3.16E-01)	5.21E+00(1.89E-01)	7.15E+00(1.68E-01)
	F24	1.37E+03(3.12E+01)	1.24E+03(1.23E+02)	1.25E+03(5.46E+01)	1.82E+03(1.75E+01)	1.61E+03(6.24E+01)	**9.20E+02(5.85E+01)**	3.52E+03(3.79E+02)	3.37E+03(2.95E+02)
+/=/-	-	-/-/-	32/0/16	44/0/4	45/0/3	38/0/10	36/0/12	38/0/10	40/0/8

These algorithms are implemented as follows: DE and ES using Geatpy [54], CMA-ES and IPOP-CMA-ES using cmaes[3], and L-SHADE using pyade[4].

Pretrained BBO Algorithms. For comparison with EPOM, we select three recent meta-learned BBO algorithms: (a) **POM** [13]: The Latest pretrained optimization model with the best performance. (b) **LES** [11]: A learnable evolutionary strategy that uses a data-driven approach to enhance generalization performance and search efficiency. (c) **LGA** [12]: A data-driven learnable genetic algorithm that adapts to unseen optimization problems, search dimensions, and evaluation budgets.

For all algorithms, we train EPOM and POM on the TS problem, with a maximum of 100 evolution generations, $n = 100$, and a problem dimension of 10.

Benchmarks BBOB [55, 56] is a well-established benchmark suite for evaluating optimization algorithms. It includes a diverse set of high-dimensional continuous functions, encompassing single-peak, multi-peak, rotated, and distorted functions, along with functions exhibiting characteristics such as Lipschitz continuity and second-order differentiability.

4.2 Results

We evaluate the generalization ability of EPOM on 24 BBOB benchmark functions with dimensions $d = 30$ and $d = 100$. Table 2 summarizes the performance comparison across all algorithms (see

[3]https://github.com/CyberAgentAILab

[4]https://github.com/xKuZz/pyade

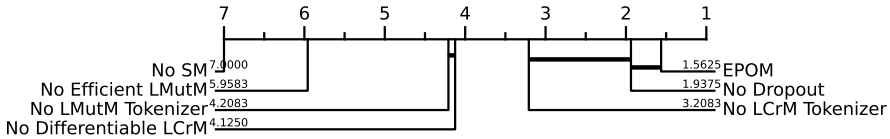


Figure 4: Results of ablation study on BBOB. The metric used to evaluate performance is the optimal value of the function found, with smaller values being better. Here, $d = 30$.

Appendix Figures 6 and 8 for detailed results). The fitness value of the best individual in the final generation for each function and dimension setting is reported.

Compared to POM, EPOM achieves superior results on 32 out of 48 functions across dimensions $d = 30$ and $d = 100$ (66.7%), indicating consistent performance gains across both low- and high-dimensional settings. This demonstrates EPOM's improved scalability and robustness. Moreover, EPOM outperforms all baseline methods in most cases, highlighting its strong generalization capability across diverse optimization landscapes.

EPOM exhibits significantly better convergence properties, as evidenced by its ability to achieve extremely small objective values. In contrast, POM often struggles to achieve comparable precision, particularly in high-dimensional problems. This highlights EPOM's enhanced ability to model population information and exploit fitness landscapes effectively.

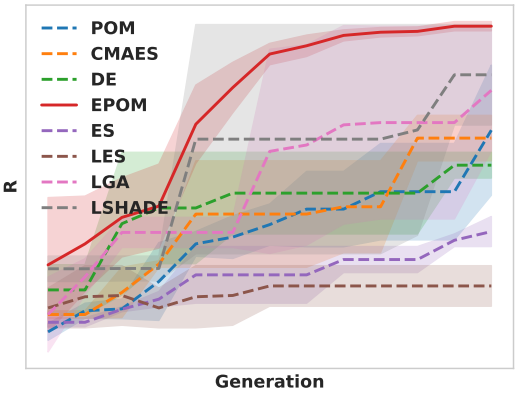


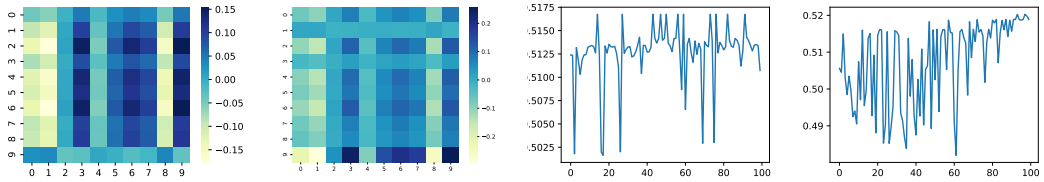
Figure 3: The results of robot control in the Bipedal Walker task.

Bipedal Walker [57]. The Bipedal Walker task aims to optimize a fully connected neural network with $d = 874$ parameters over $k = 800$ time steps to enhance robot locomotion control. In Fig. 3, EPOM achieves rapid and high-quality convergence, whereas LSHADE exhibits lower effectiveness, and CMA-ES, DE, and LES suffer from premature convergence.

4.3 Ablation Study

The results of ablation study for the designed modules are shown as Fig. 4. Configurations include the following items: (a) *No Efficient LMutM*: where the Efficient LMutM is excluded, and a simple *DE/rand/1/bin* mutation operator is employed; (b) *No LMutM Tokenizer*: the LMutM Tokenizer is excluded; (c) *No Diffrentiable LCrM*: indicating the absence of the learnable crossover operation, using only binomial crossover; (d) *No LCrM Tokenizer*: the LCrM Tokenizer is excluded; (e) *No Dropout*: the dropout layer for $\mathbf{S}^t$ is excluded; (f) *No SM*: the 1-to-1 selection module is excluded.

Upon observing the experimental results, we discovered that the performance of the model without SM deteriorated significantly. This is because the model loses its evolutionary direction. The impact of the modules can be roughly ranked as follows: *Efficient LMutM* $\succ$ *LMutM Tokenizer* $\succ$ *Differentiable LCrM* $\succ$ *LCrM Tokenizer* $\succ$ *Dropout*. This indicates that the Efficient LMutM introduces a more powerful representational ability to the model, addresses the problem of strategy inefficiency in POMs, and enhances the model's search capabilities. The LMutM Tokenizer can effectively extract the distribution characteristics of the population. The introduction of this characteristic significantly improves the model's decision-making ability, enabling it to better understand the fitness landscape. The Differentiable LCrM and LCrM Tokenizer can achieve information interaction between new and old individuals, strike an adaptive balance between exploration and exploitation, and enhance the generalization and convergence of the model.



(a) F1 step 1 (b) F1 step 50 (c) F1 No.1 individual (d) F1 No.50 individual

Figure 5: (a)(b) Visualization results of some information selection strategies of Efficient LMutM. The numbers 0-9 represent the population ranking, and the smaller the ranking, the higher the fitness. The population size here is 10 and parameter p for top-p-sampling is set to 1. (c)(d) Visualization results of crossover weight of some individuals of Differentiable LCrM. The x label represents the generation, and the y label denotes the crossover weight of $\mathbf{X}^t$. The population size is 100. (a), (b), (c) and (d) are all tested on BBOB $d = 30$

4.4 Visualization Analysis

Analysis of Efficient LMutM To conduct an in-depth analysis of the Efficient LMutM strategies, we visualize the evolution of $\mathbf{S}^t$ through heat maps, as shown in Fig. 5 (further details can be found in Appendix D). These heat maps reveal two key insights: 1) Similar to the original version, Efficient LMutM effectively balances exploration and exploitation by dynamically adjusting the weights of individuals based on their performance; and 2) Efficient LMutM adaptively generates diverse mutant strategies in response to varying task landscapes and individual performance.

Analysis of Differentiable LCrM We visualize the crossover strategies of Differentiable LCrM. As shown in Fig. 5 (refer to Appendix Figures 15-19 for additional details), Differentiable LCrM demonstrates its dynamic crossover strategies, generating variable crossover weights based on the overall performance of individuals and the characteristics of task landscapes.

5 Conclusion

We propose the Enhanced Pretrained Optimization Models, a novel framework designed to enhance the performance and robustness of POMs. The experimental results demonstrate that EPOM addresses the key limitations of POM, including insufficient task feature modeling, inefficient strategy generation, and unstable training process. By introducing the LMutM/LCrM Tokenizer, Efficient LMutM, and Differentiable LCrM, EPOM achieves state-of-the-art performance in zero-shot optimization, particularly in high-dimensional landscapes. These advancements make EPOM a highly promising candidate for real-world black-box optimization problems, offering superior scalability, robustness, and efficiency compared to existing methods.

EPOM's ability to outperform POM and other baselines across a wide range of functions, especially in high-difficulty scenarios, underscores its potential as a universal black-box optimizer. This work not only advances the state of the art in zero-shot optimization but also paves the way for broader adoption in machine learning and engineering applications.

However, the performance of EPOM in heterogeneous search spaces, involving optimization tasks with diverse data types (e.g., images, strings), remains underexplored. We identify this as an important avenue for future work.

Acknowledgement

This work was supported in part by Natural Science Basic Research Program of Shaanxi under Grant 2025JC-QYCX-060, in part by the National Natural Science Foundation of China under Grant 62206205, 62472345, and 62471371, in part by the Young Talent Fund of Association for Science and Technology in Shaanxi, China under Grant 20230129, in part by the Guangdong High-level Innovation Research Institution Project under Grant 2021B0909050008, and in part by the Guangzhou Key Research and Development Program under Grant 202206030003.

References

[1] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[2] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[3] 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 *The Twelfth International Conference on Learning Representations*, 2024.

[4] Zeyuan Ma, Zhiguang Cao, Zhou Jiang, Hongshu Guo, and Yue-Jiao Gong. Meta-black-box-optimization through offline q-function learning. *arXiv preprint arXiv:2505.02010*, 2025.

[5] Qing Ye, Yanan Sun, Jixin Zhang, and Jiancheng Lv. A distributed framework for ea-based nas. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1753–1764, 2021.

[6] 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*, pages 1–13, 2024.

[7] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[8] 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*, 2025.

[9] Xu Yang, Rui Wang, Kaiwen Li, and Hisao Ishibuchi. Meta-black-box optimization for evolutionary algorithms: Review and perspective. *Swarm and Evolutionary Computation*, 93:101838, 2025.

[10] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research*, 23(189):1–59, 2022.

[11] Robert 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 Companion Conference on Genetic and Evolutionary Computation*, pages 29–30, 2023.

[12] Robert Lange, Tom Schaul, Yutian Chen, Chris Lu, Tom Zahavy, Valentin Dalibard, and Sebastian Flennerhag. Discovering attention-based genetic algorithms via meta-black-box optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 929–937, 2023.

[13] Xiaobin Li, Kai Wu, Yujian Betterest Li, Xiaoyu Zhang, Handing Wang, and Jing Liu. Pretrained optimization model for zero-shot black box optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[14] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[15] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[16] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[17] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11:341–359, 1997.

[18] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[19] Ryoji Tanabe and Alex S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1658–1665, 2014.

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

[21] Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10045–10067, 2022.

[22] Hugo Siqueira Gomes, Benjamin Léger, and Christian Gagné. Meta learning black-box population-based optimizers. *arXiv preprint arXiv:2103.03526*, 2021.

[23] Xingwen Zhang, Jeff Clune, and Kenneth O Stanley. On the relationship between the openai evolution strategy and stochastic gradient descent. *arXiv preprint arXiv:1712.06564*, 2017.

[24] Tianyu Li, Kai Wu, Xiaobin Li, Xiangyi Teng, and Jing Liu. Meta-moga: Meta-learning multi-objective genetic algorithm. In *2025 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–4, 2025.

[25] 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 A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 10775–10795. Curran Associates, Inc., 2023.

[26] Gresa Shala, André Biedenkapp, Noor Awad, Steven Adriaensen, Marius Lindauer, and Frank Hutter. Learning step-size adaptation in cma-es. In *International Conference on Parallel Problem Solving from Nature*, pages 691–706. Springer, 2020.

[27] 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 AAAI Conference on Artificial Intelligence*, volume 39, pages 26982–26990, 2025.

[28] Robert Tjarko Lange, Yingtao Tian, and Yujin Tang. Evolution transformer: In-context evolutionary optimization. *arXiv e-prints*, pages arXiv–2403, 2024.

[29] Xiaobin Li, Kai Wu, Xiaoyu Zhang, Handing Wang, and Jing Liu. B2opt: Learning to optimize black-box optimization with little budget. *arXiv preprint arXiv:2304.11787*, 2023.

[30] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, pages 1–3, 2023.

[31] Elliot Meyerson, Mark J Nelson, Herbie Bradley, Arash Moradi, Amy K Hoover, and Joel Lehman. Language model crossover: Variation through few-shot prompting. *arXiv preprint arXiv:2302.12170*, 2023.

[32] Fei Liu, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Algorithm evolution using large language model. *arXiv preprint arXiv:2311.15249*, 2023.

[33] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[34] Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. Evolution through large models. In *Handbook of Evolutionary Machine Learning*, pages 331–366. Springer, 2023.

[35] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language mode, 2024.

[36] Niki van Stein and Thomas Bäck. Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 2024.

[37] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[38] Angelica Chen, David M Dohan, and David R So. Evoprompting: Language models for code-level neural architecture search. *arXiv preprint arXiv:2302.14838*, 2023.

[39] Muhammad U Nasir, Sam Earle, Julian Togelius, Steven James, and Christopher Cleghorn. Llmatic: Neural architecture search via large language models and quality-diversity optimization. *arXiv preprint arXiv:2306.01102*, 2023.

[40] Beichen Huang, Xingyu Wu, Yu Zhou, Jibin Wu, Liang Feng, Ran Cheng, and Kay Chen Tan. Exploring the true potential: Evaluating the black-box optimization capability of large language models. *arXiv preprint arXiv:2404.06290*, 2024.

[41] Zeyuan Ma, Hongshu Guo, Jiacheng Chen, Guojun Peng, Zhiguang Cao, Yining Ma, and Yue-Jiao Gong. Llamoco: Instruction tuning of large language models for optimization code generation, 2024.

[42] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model, 2024.

[43] Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling, 2023.

[44] Tung Nguyen, Sudhanshu Agrawal, and Aditya Grover. Expt: synthetic pretraining for few-shot experimental design. *Advances in Neural Information Processing Systems*, 36, 2024.

[45] Tung Nguyen and Aditya Grover. Lico: Large language models for in-context molecular optimization. *arXiv preprint arXiv:2406.18851*, 2024.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[47] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[48] Kai Wu, Penghui Liu, and Jing Liu. Decn: Automated evolutionary algorithms via evolution inspired deep convolution network, 2023.

[49] Zeyuan Ma, Jiacheng Chen, Hongshu Guo, and Yue-Jiao Gong. Neural exploratory landscape analysis for meta-black-box-optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[50] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4794–4803, 2022.

[51] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[52] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[53] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2010.

[54] Jazzbin. Geatpy: The genetic and evolutionary algorithm toolbox with high performance in python, 2020.

[55] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical report, Citeseer, 2010.

[56] 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.

[57] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[58] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Tasks, Loss Function & MetaGBT

Tasks. We form a training task set $TS = \{f_i(\mathbf{X}|\omega^j)\}$,where $i \in [1, 5]$ and $j \in [1, N]$, comprising $4N$ tasks derived from Table 1 in appendix, where ω_i denotes the task parameter influencing the function's landscape offset. Our selection of these functions for the training task is motivated by their diverse landscape features.

MetaGBT. The pseudocode for *MetaGBT* is presented in Algorithm 2. Initially, we sample the *EPOM* parameter θ from a standard normal distribution. The objective of *MetaGBT* is to iteratively update θ to bring it closer to the global optimum θ^*. In line 2, we sample a population for each task in *TS*. Lines 3, 4 and 5 involve the resampling of task parameters for all tasks in *TS*, thereby altering the task landscape, augmenting training complexity, and enhancing the learning of robust optimization strategies by EPOM. The final loss function (line 10) is determined by computing the average of the loss functions for all tasks. Subsequently, in line 12, we update θ using a gradient-based optimizer, such as Adam [58]. The trained *EPOM* is then ready for application in solving an unknown BBO problem, as depicted in Algorithm 1.

Algorithm 2 MetaGBT

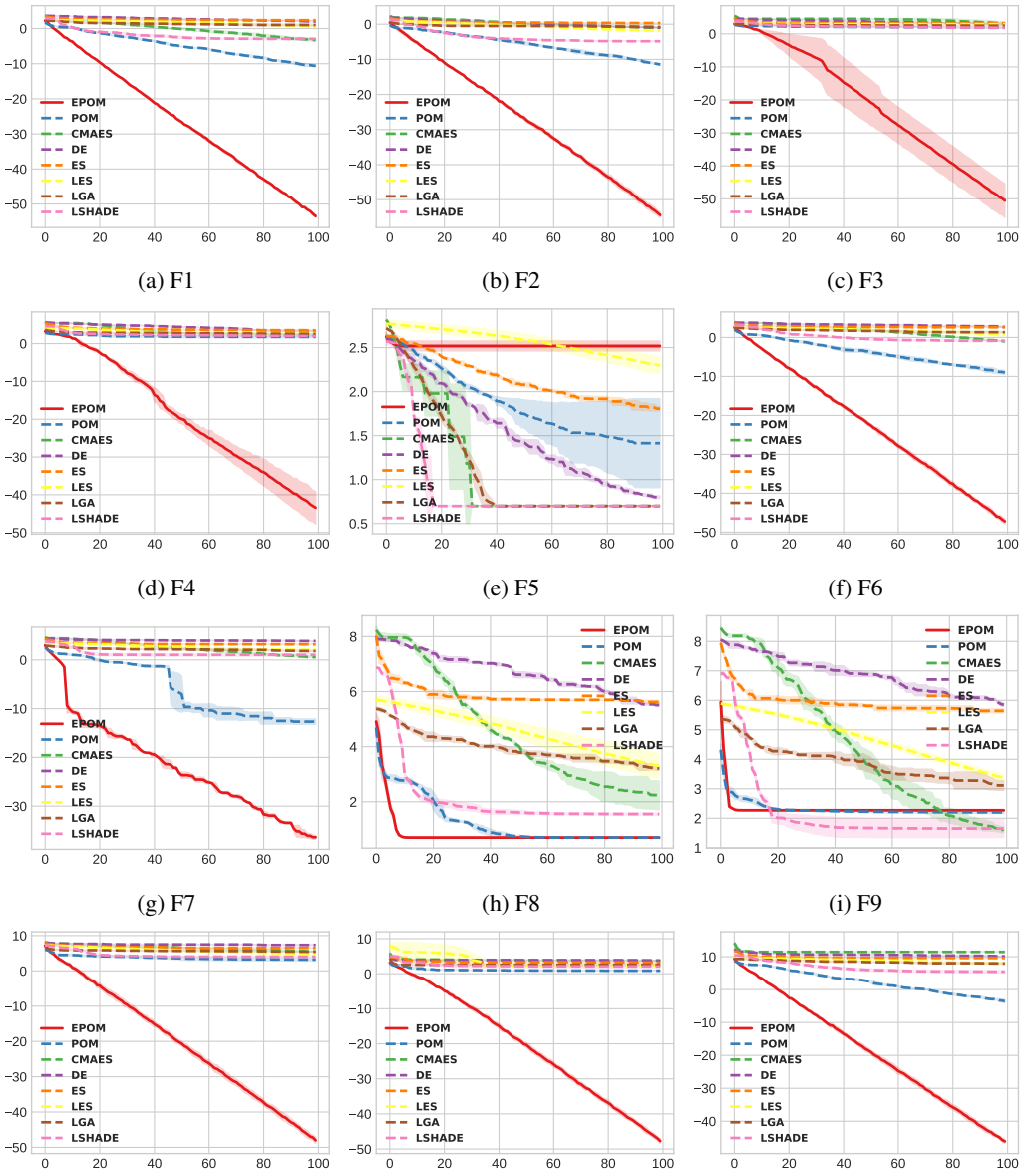
Input: T, n, training set TS.
Output: The optimal θ.
1: Randomly sample the parameter θ of *EPOM*.
2: **while** not done **do**
3: Sample $|TS|$ populations of size n to obtain the population set $pop \leftarrow [\mathbf{X}_1^0, \mathbf{X}_2^0, \cdots, \mathbf{X}_{|TS|}^0]$.
4: **for** $i = 1, 2, \ldots, |TS|$ **do**
5: Randomly sample ω^i for the f_i in TS.
6: **end for**
7: **for** $t = 1, 2, \ldots, T$ **do**
8: **for** $i = 1, 2, \ldots, |TS|$ **do**
9: $\mathbf{X}_i^t \leftarrow EPOM(\mathbf{X}_i^{t-1}, 1|\theta)$.
10: $loss_i^t \leftarrow l_i(\mathbf{X}_i^t, \mathbf{X}_i^{t-1}, f_i, \omega^i, \lambda)$.
11: **end for**
12: $\theta \leftarrow$ Update θ using Adam based on $\frac{1}{|TS|}\sum_i loss_i^t$.
13: **end for**
14: **end while**

B Parameters

The primary control parameters of CMA-ES and L-SHADE are automatically adjusted. For LGA and LES, we utilized the optimal parameters provided by the authors without modifications. Other hyperparameters were tuned using grid search to identify the optimal combinations, and multiple experiments were conducted accordingly. Detailed parameter settings are presented in Table 3. Each experiment reports the mean and standard deviation of the results from various sets of experiments, with a consistent population size of 100 across all trials. All experiments are performed on a device with GeForce RTX 3090 24G GPU, Intel Xeon Gold 6126 CPU and 64G RAM.

Table 3: Detailed parameter settings for all baselines.

Algorithm	**item**	**setting**
POM	$d_m = 1000$	Standard Settings for POM (M).
	$d_c = 4$	
CMA-ES	Initial $\sigma = \frac{\textbf{upper\_bounds+lower\_bounds}}{2} * \frac{2}{5}$	$2/5$ is a hyperparameter, and we determine this hyperparameter between $[0.1, 1]$ using a grid search, with a step of 0.1.
	Initial μ	$\mu =$ **lower_bounds** $+ (randn(d) *$ (**upper_bounds** $-$ **lower_bounds**)), where $randn(d)$ stands for sampling a d-dimensional vector from a standard normal distribution.
LSHADE	$memory\_size = 6$	We use a grid search to determine this parameter, the search interval is $[1, 10]$, and the search step is 1.
ES	$selFuc = urs$	We use a grid search to determine this parameter, the search interval is [*dup*, *ecs*, *etour*, *otos*, *rcs*, *rps*, *rws*, *sus*, *tour*, *urs*] [54].
	$Nsel = 0.5$	we determine this hyperparameter between $[0.1, 0.8]$ using a grid search, with a step of 0.1.
DE	$F = 0.5$	we determine this hyperparameter between $[0.1, 0.9]$ using a grid search, with a step of 0.1. [54].
	$XOVR = 0.5$	we determine this hyperparameter between $[0.1, 0.9]$ using a grid search, with a step of 0.1.
LGA	All parameters	We use the pre-trained optimal parameters provided by the authors.
LES		



(a) F1 (b) F2 (c) F3

(d) F4 (e) F5 (f) F6

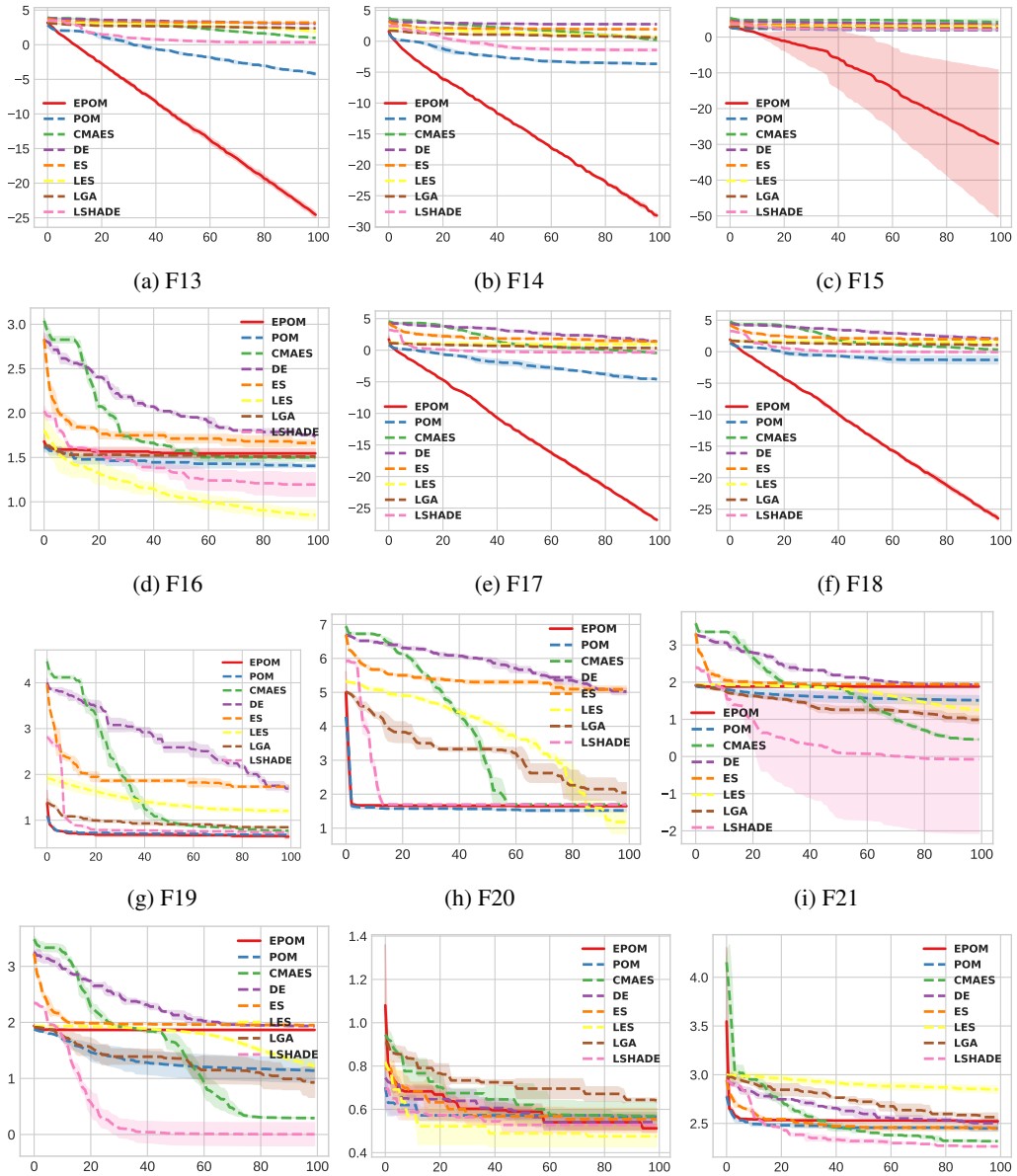
(g) F7 (h) F8 (i) F9

(j) F10 (k) F11 (l) F12

Figure 6: The log convergence curves of EPOM and other baselines on F1-F12. It shows the convergence curve of these algorithms on functions in BBOB with $d = 30$.

C Visualization Results of BBOB

The visualization results of all BBOB experiments are shown in Figure 6-9.



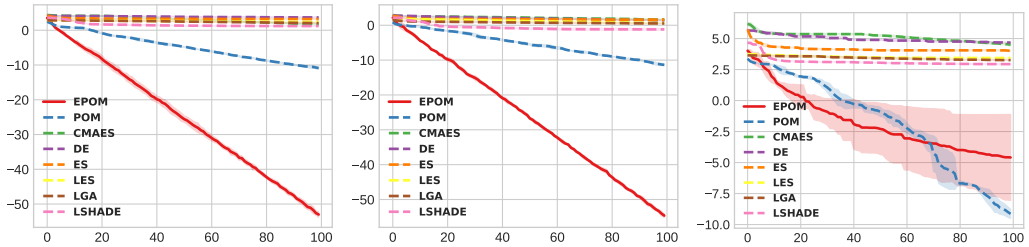
(a) F13 (b) F14 (c) F15

(d) F16 (e) F17 (f) F18

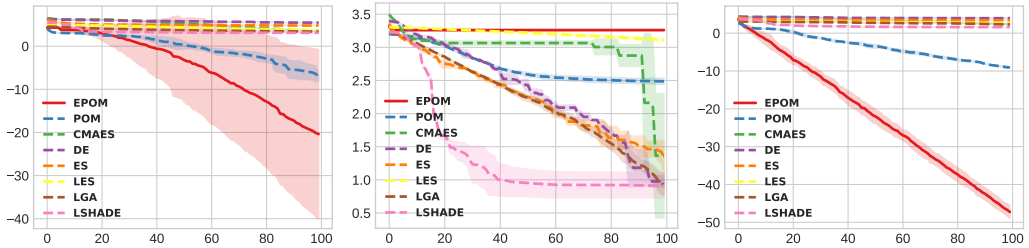
(g) F19 (h) F20 (i) F21

(j) F22 (k) F23 (l) F24

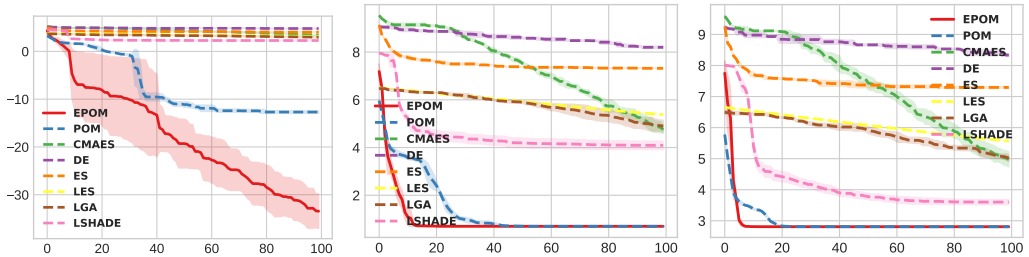
Figure 7: The log convergence curves of EPOM and other baselines on F13-F24. It shows the convergence curve of these algorithms on functions in BBOB with $d = 30$.



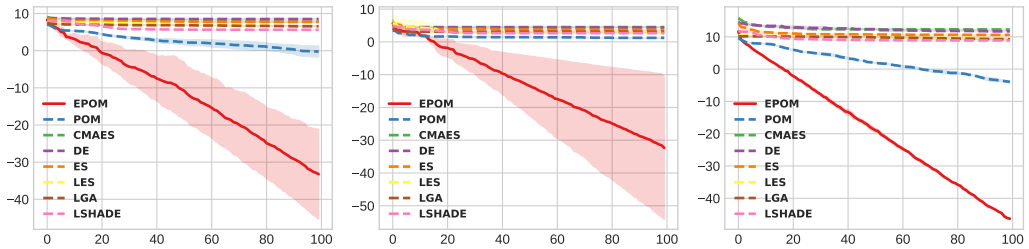
(a) F1 (b) F2 (c) F3



(d) F4 (e) F5 (f) F6

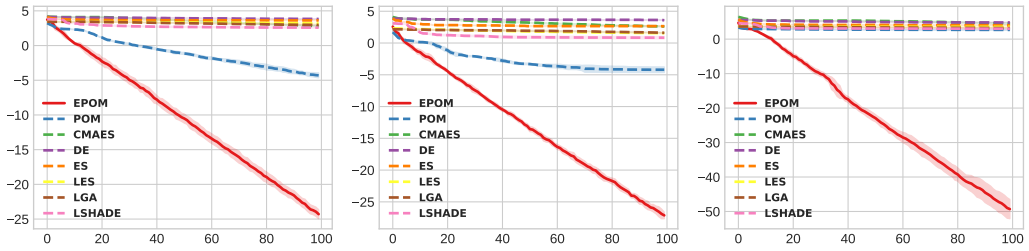


(g) F7 (h) F8 (i) F9

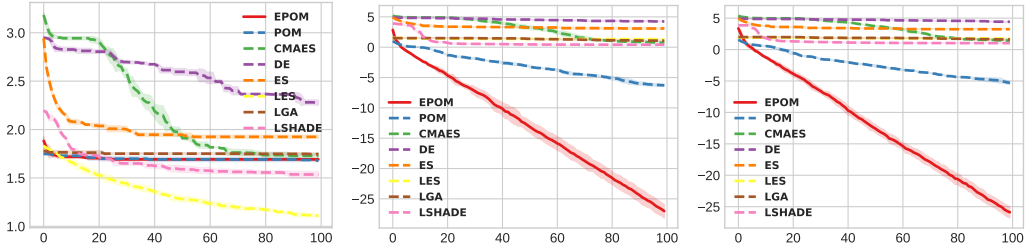


(j) F10 (k) F11 (l) F12

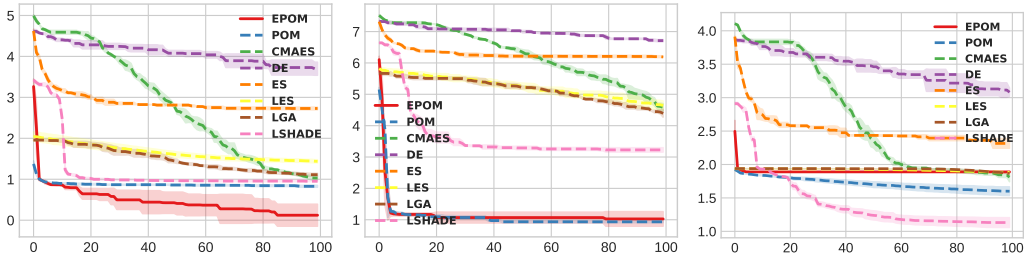
Figure 8: The log convergence curves of EPOM and other baselines on F1-F12. It shows the convergence curve of these algorithms on the functions in BBOB with $d = 100$.



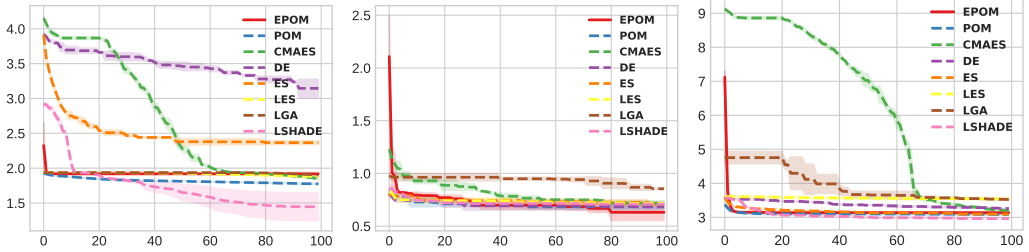
(a) F13 (b) F14 (c) F15



(d) F16 (e) F17 (f) F18



(g) F19 (h) F20 (i) F21

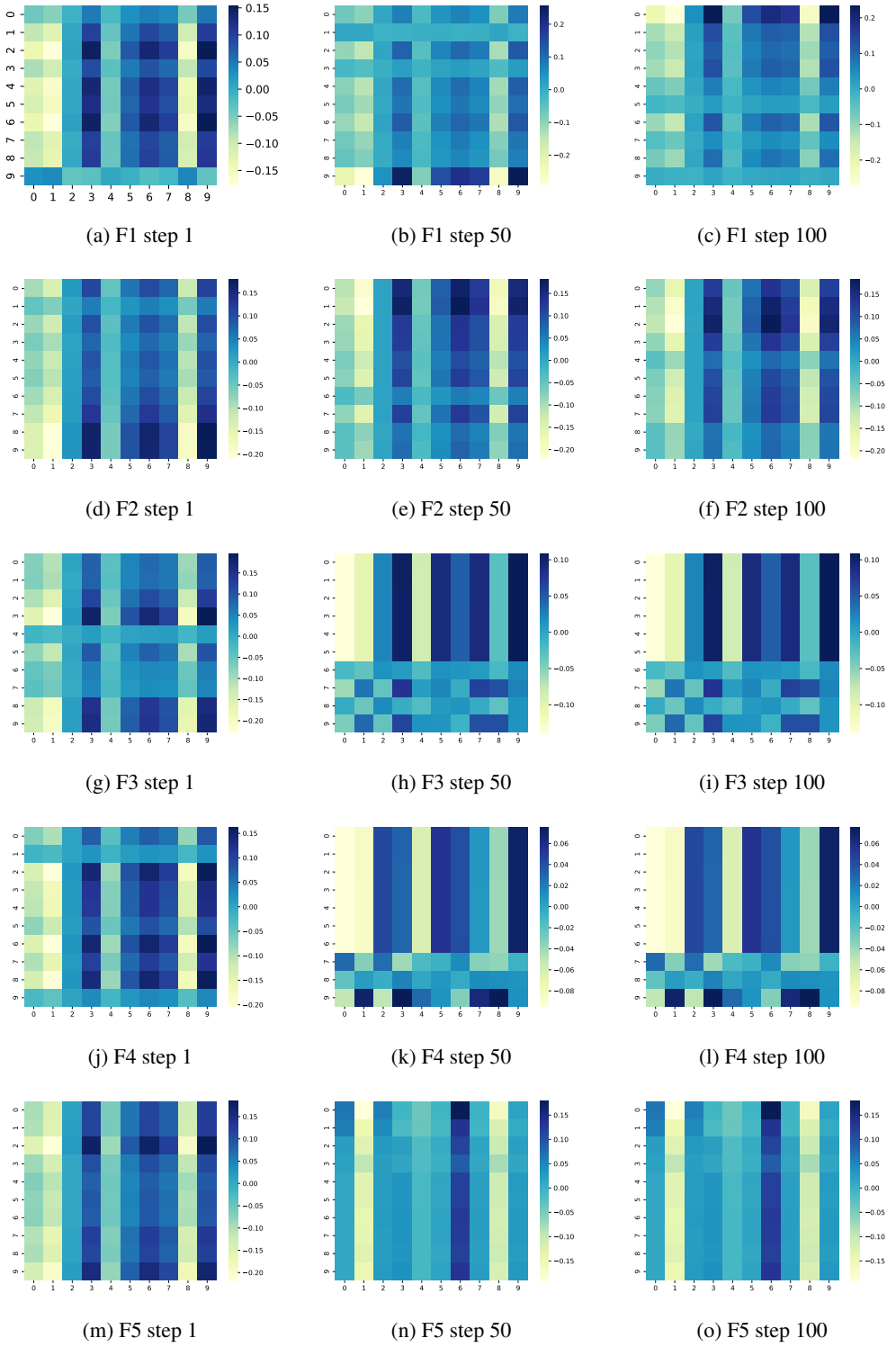


(j) F22 (k) F23 (l) F24

Figure 9: The log convergence curves of EPOM and other baselines on F13-F24. It shows the convergence curve of these algorithms on the functions in BBOB with $d = 100$.

D Visualization Results of Efficient LMutM

The visualization results of Efficient LMutM are shown in Figure 10-14.



(a) F1 step 1 (b) F1 step 50 (c) F1 step 100

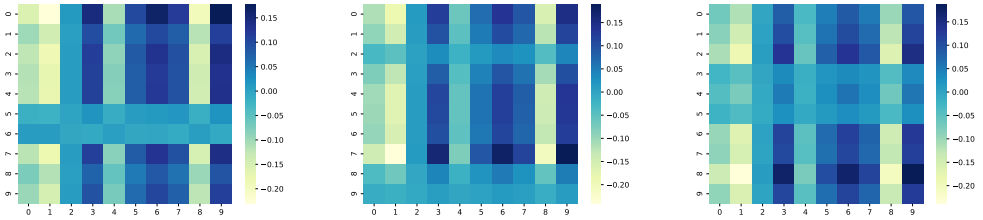
(d) F2 step 1 (e) F2 step 50 (f) F2 step 100

(g) F3 step 1 (h) F3 step 50 (i) F3 step 100

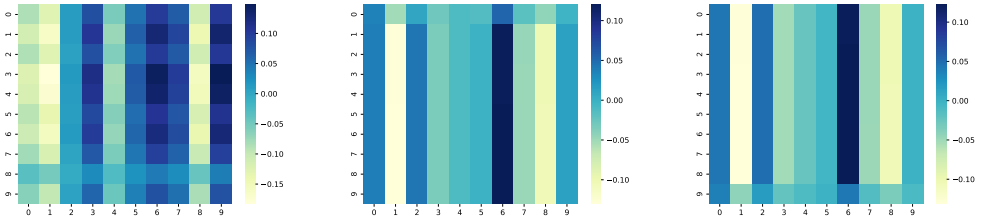
(j) F4 step 1 (k) F4 step 50 (l) F4 step 100

(m) F5 step 1 (n) F5 step 50 (o) F5 step 100

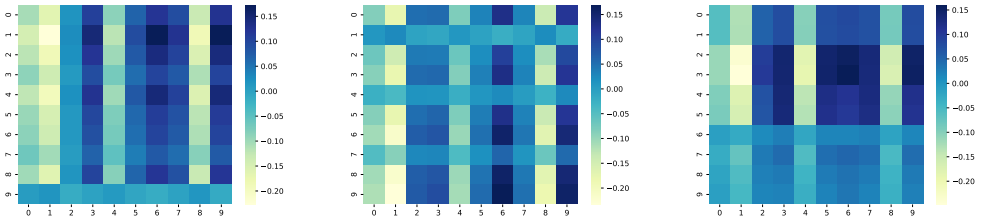
Figure 10: F1-F5 Visualization results of some information selection strategies of Efficient LMutM. 0->9 represents the population ranking, and the smaller the ranking, the higher the fitness.



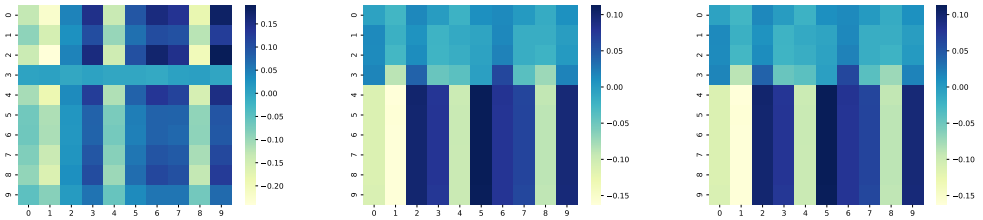
(a) F6 step 1 (b) F6 step 50 (c) F6 step 100



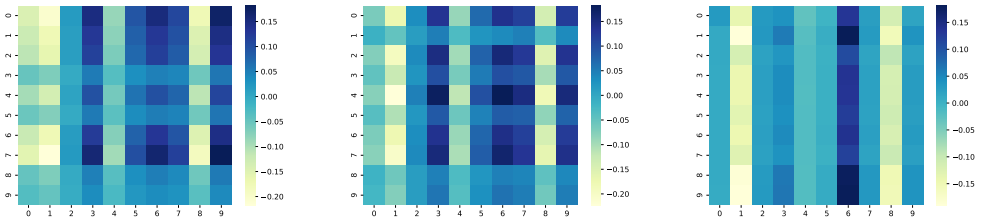
(d) F7 step 1 (e) F7 step 50 (f) F7 step 100



(g) F8 step 1 (h) F8 step 50 (i) F8 step 100

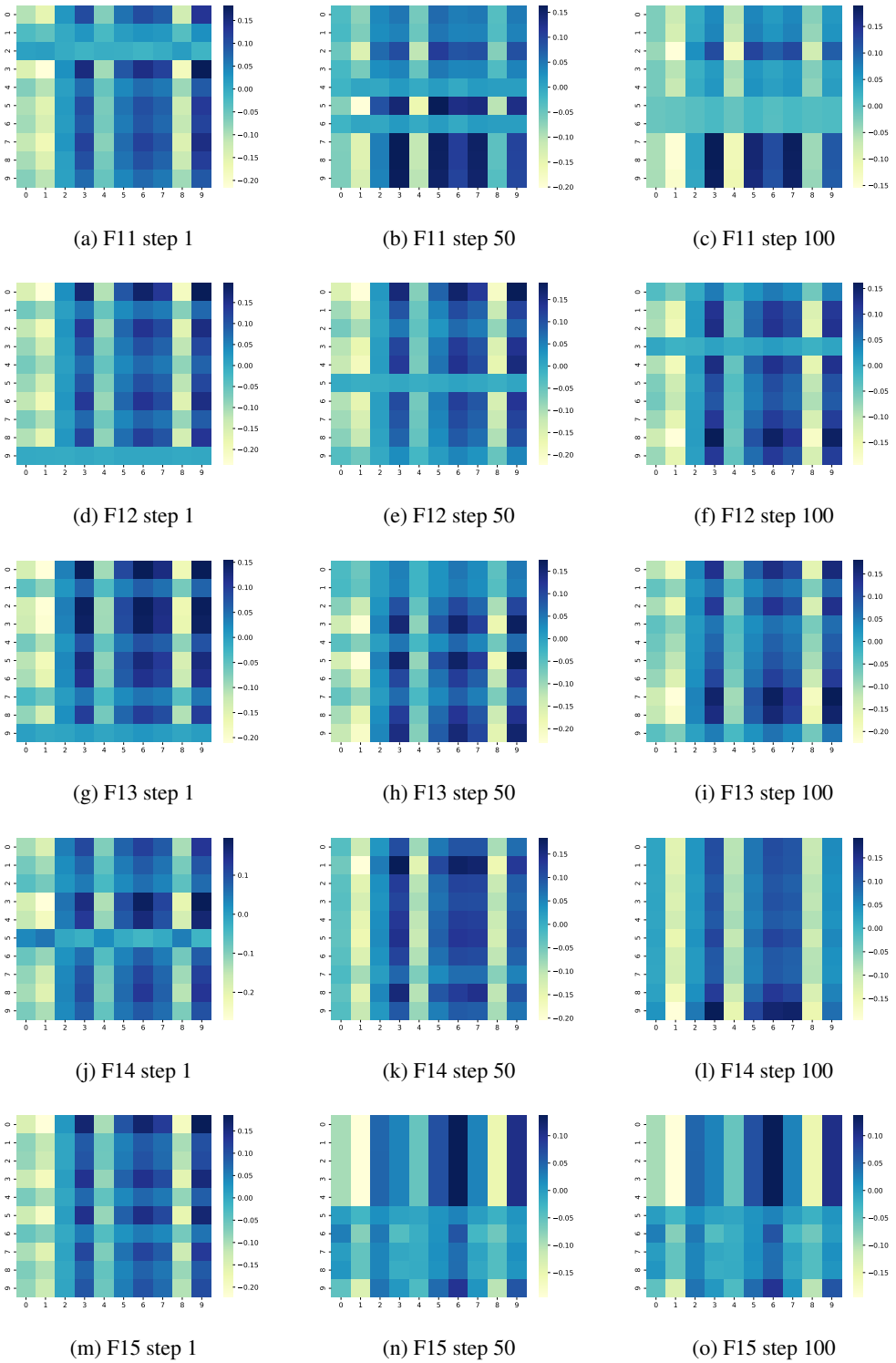


(j) F9 step 1 (k) F9 step 50 (l) F9 step 100



(m) F10 step 1 (n) F10 step 50 (o) F10 step 100

Figure 11: F6-F10 Visualization results of some information selection strategies of Efficient LMutM. 0->9 represents the population ranking, and the smaller the ranking, the higher the fitness.



(a) F11 step 1 (b) F11 step 50 (c) F11 step 100

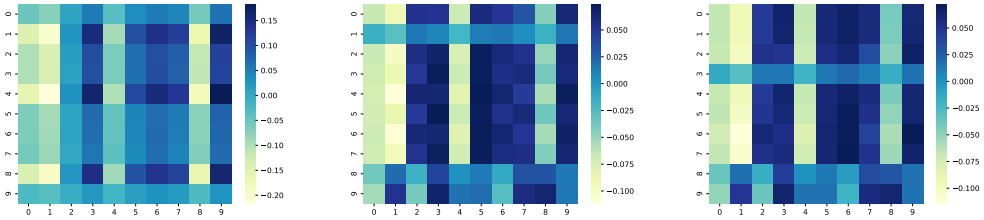
(d) F12 step 1 (e) F12 step 50 (f) F12 step 100

(g) F13 step 1 (h) F13 step 50 (i) F13 step 100

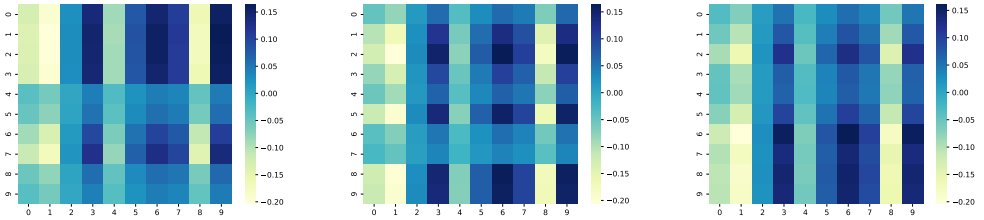
(j) F14 step 1 (k) F14 step 50 (l) F14 step 100

(m) F15 step 1 (n) F15 step 50 (o) F15 step 100

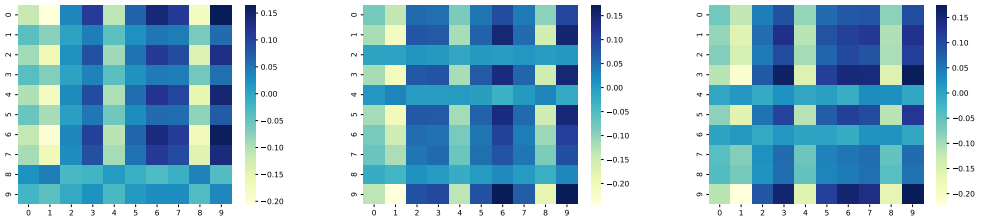
Figure 12: F11-F15 Visualization results of some information selection strategies of Efficient LMutM. 0->9 represents the population ranking, and the smaller the ranking, the higher the fitness.



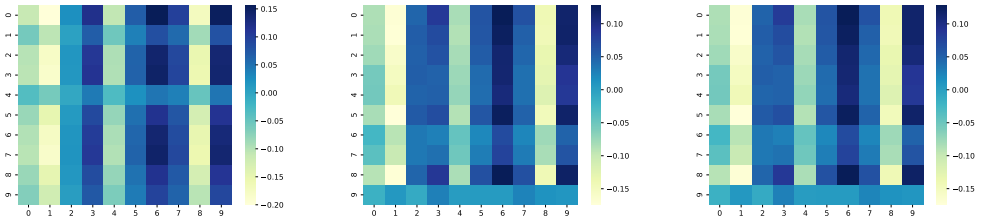
(a) F16 step 1 (b) F16 step 50 (c) F16 step 100



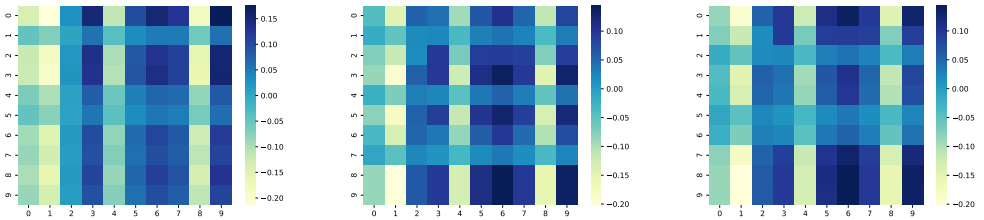
(d) F17 step 1 (e) F17 step 50 (f) F17 step 100



(g) F18 step 1 (h) F18 step 50 (i) F18 step 100

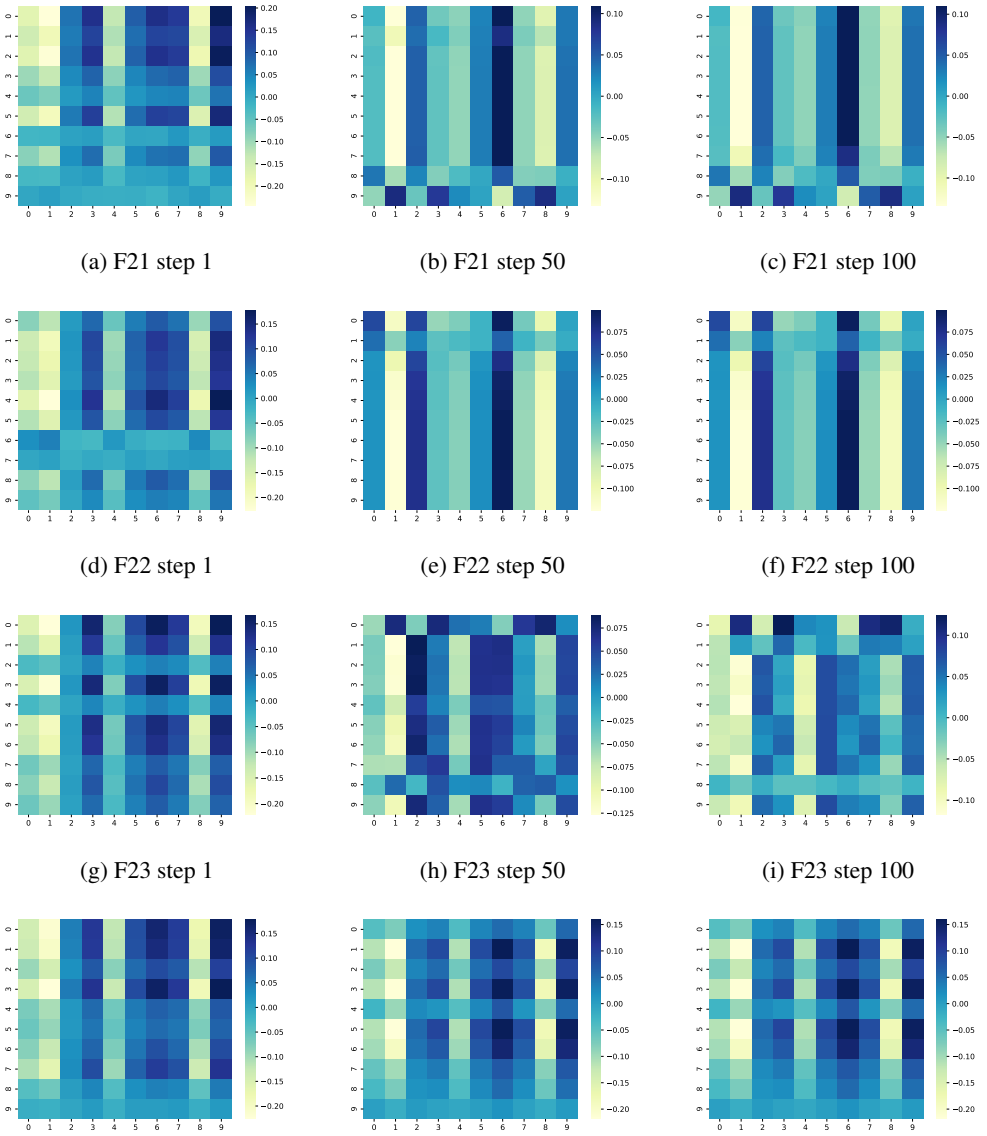


(j) F19 step 1 (k) F19 step 50 (l) F19 step 100



(m) F20 step 1 (n) F20 step 50 (o) F20 step 100

Figure 13: F16-F20 Visualization results of some information selection strategies of Efficient LMutM. 0->9 represents the population ranking, and the smaller the ranking, the higher the fitness.



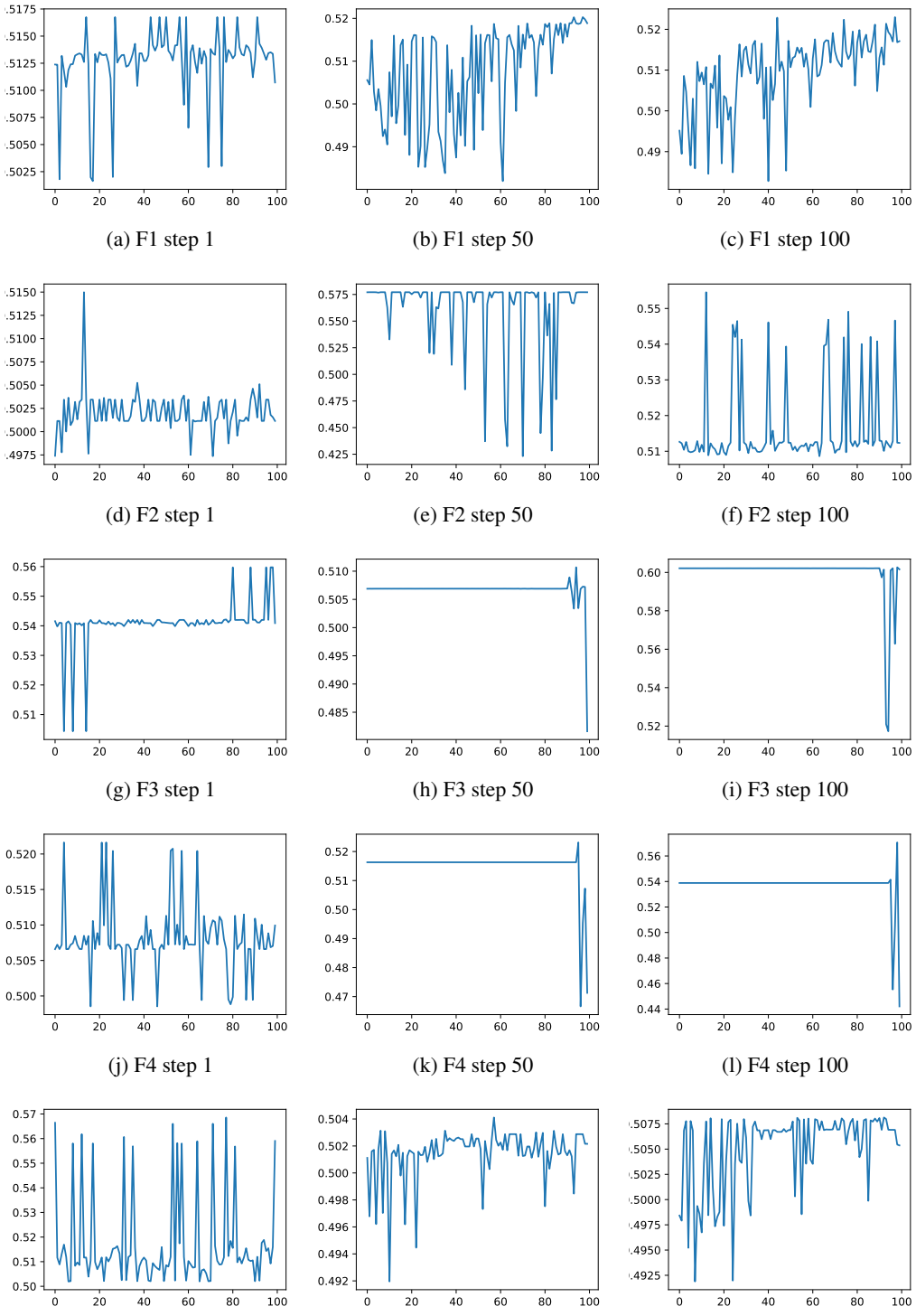
(a) F21 step 1 (b) F21 step 50 (c) F21 step 100

(d) F22 step 1 (e) F22 step 50 (f) F22 step 100

(g) F23 step 1 (h) F23 step 50 (i) F23 step 100

(j) F24 step 1 (k) F24 step 50 (l) F24 step 100

Figure 14: F21-F24 Visualization results of some information selection strategies of Efficient LMutM. 0->9 represents the population ranking, and the smaller the ranking, the higher the fitness.



(a) F1 step 1 (b) F1 step 50 (c) F1 step 100

(d) F2 step 1 (e) F2 step 50 (f) F2 step 100

(g) F3 step 1 (h) F3 step 50 (i) F3 step 100

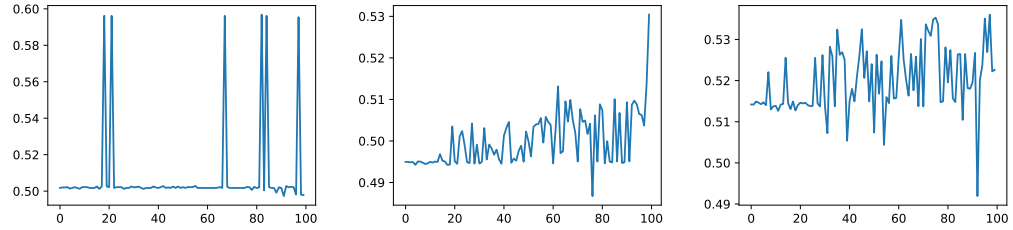
(j) F4 step 1 (k) F4 step 50 (l) F4 step 100

(m) F5 step 1 (n) F5 step 50 (o) F5 step 100

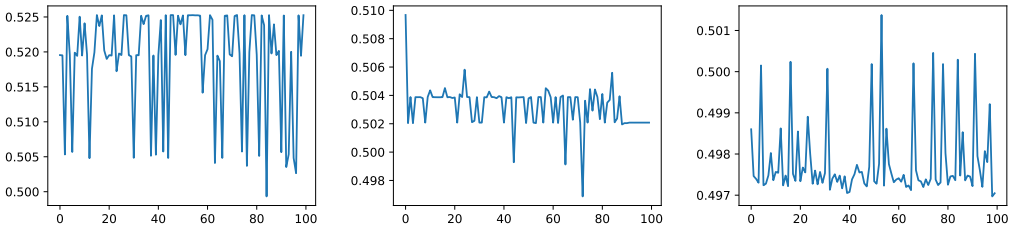
Figure 15: F1-F5 Visualization results of crossover weight of Differentiable LCrM.

E Visualization Results of Differentiable LCrM

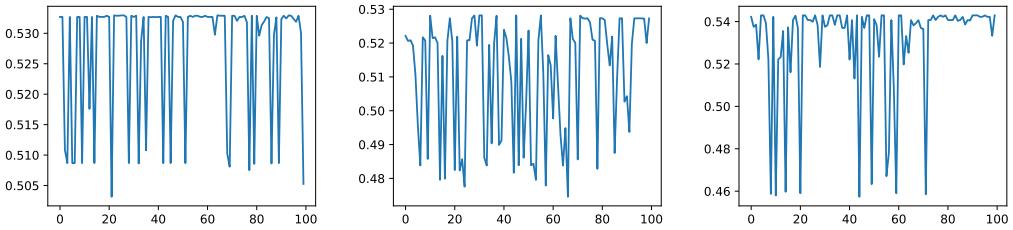
The visualization results of Differentiable LCrM are shown in Figure 15-19.



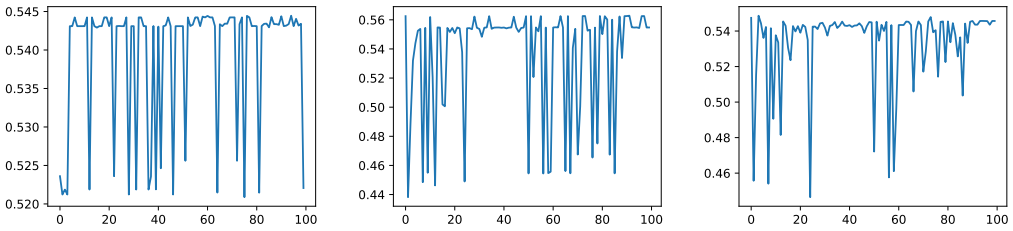
(a) F6 step 1 (b) F6 step 50 (c) F6 step 100



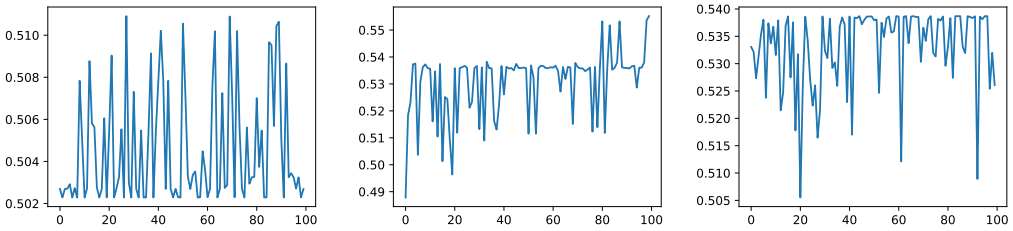
(d) F7 step 1 (e) F7 step 50 (f) F7 step 100



(g) F8 step 1 (h) F8 step 50 (i) F8 step 100

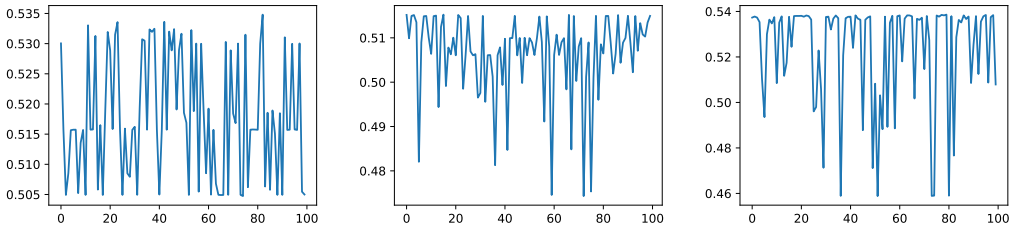


(j) F9 step 1 (k) F9 step 50 (l) F9 step 100

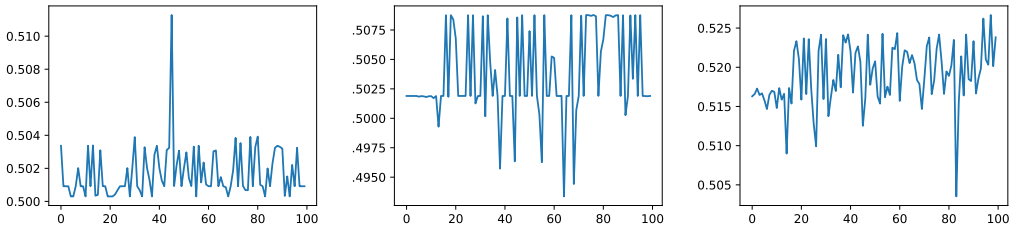


(m) F10 step 1 (n) F10 step 50 (o) F10 step 100

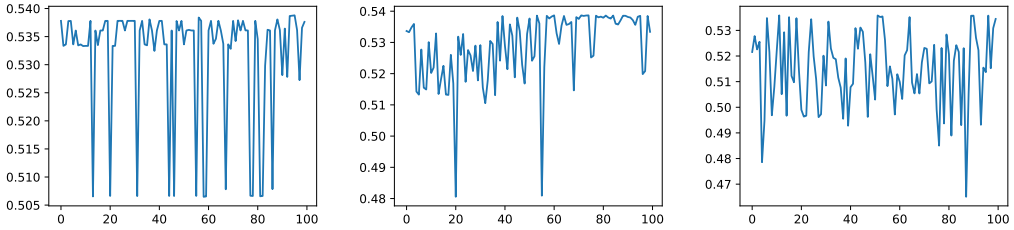
Figure 16: F6-F10 Visualization results of crossover weight of Differentiable LCrM.



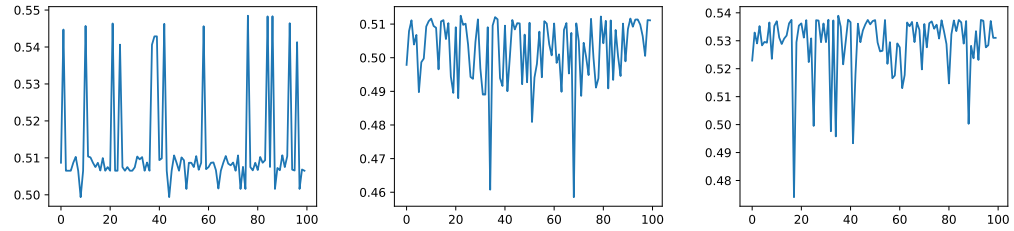
(a) F11 step 1 (b) F11 step 50 (c) F11 step 100



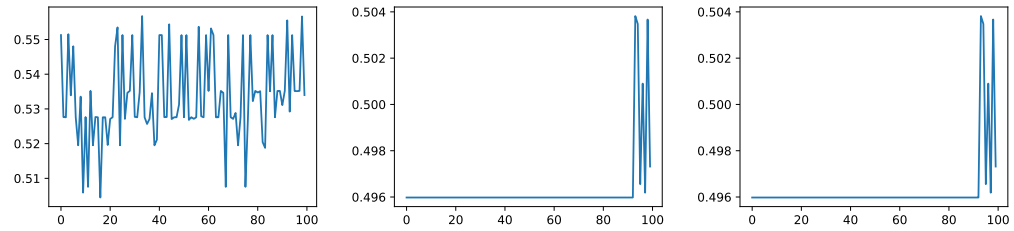
(d) F12 step 1 (e) F12 step 50 (f) F12 step 100



(g) F13 step 1 (h) F13 step 50 (i) F13 step 100

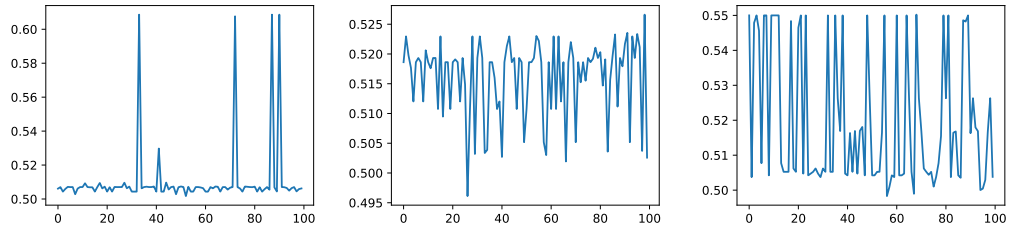


(j) F14 step 1 (k) F14 step 50 (l) F14 step 100

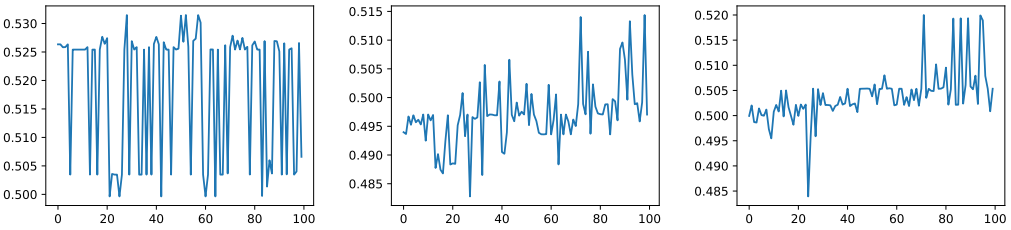


(m) F15 step 1 (n) F15 step 50 (o) F15 step 100

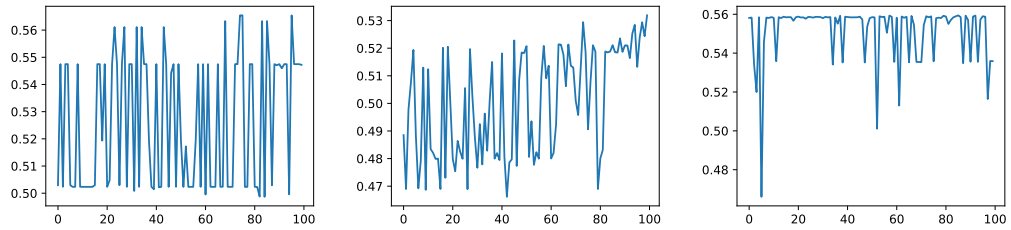
Figure 17: F11-F15 Visualization results of crossover weight of Differentiable LCrM.



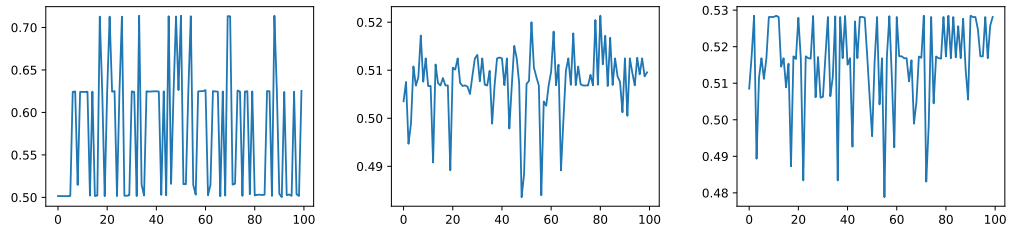
(a) F16 step 1 (b) F16 step 50 (c) F16 step 100



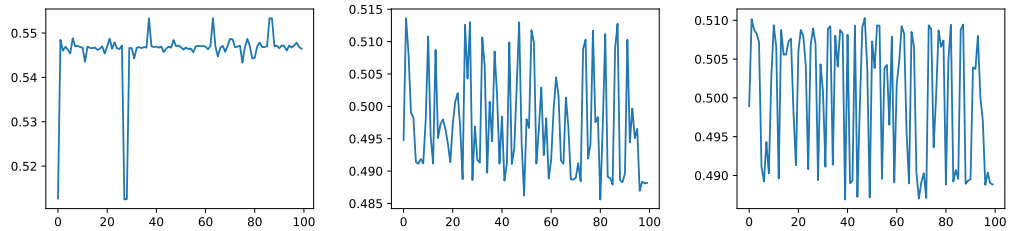
(d) F17 step 1 (e) F17 step 50 (f) F17 step 100



(g) F18 step 1 (h) F18 step 50 (i) F18 step 100

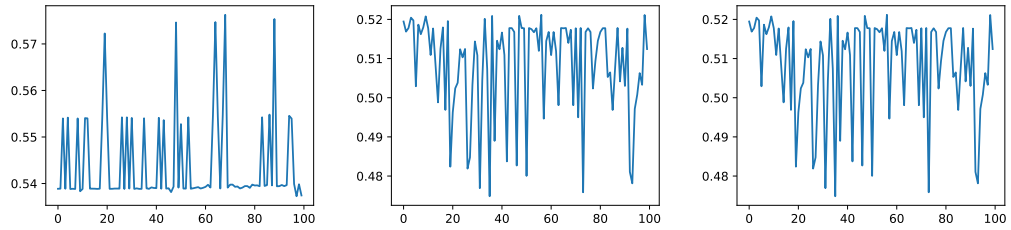


(j) F19 step 1 (k) F19 step 50 (l) F19 step 100

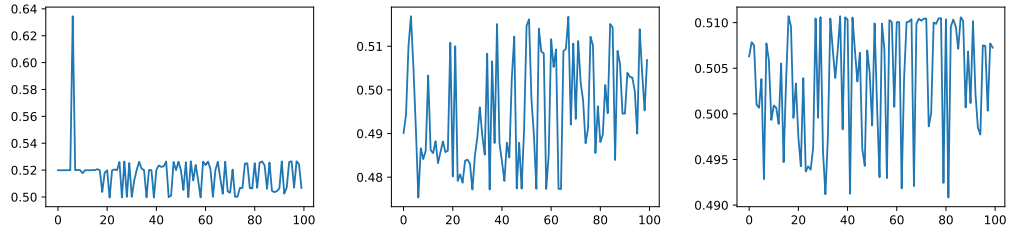


(m) F20 step 1 (n) F20 step 50 (o) F20 step 100

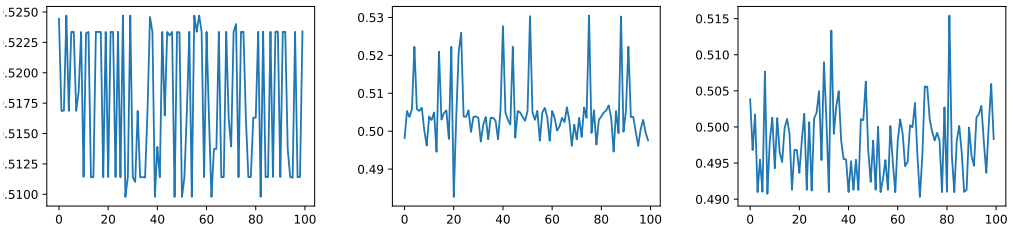
Figure 18: F16-F20 Visualization results of crossover weight of Differentiable LCrM.



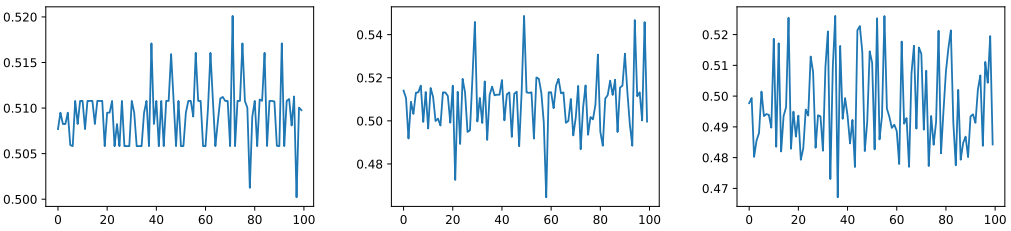
(a) F21 step 1 (b) F21 step 50 (c) F21 step 100



(d) F22 step 1 (e) F22 step 50 (f) F22 step 100



(g) F23 step 1 (h) F23 step 50 (i) F23 step 100



(j) F24 step 1 (k) F24 step 50 (l) F24 step 100

Figure 19: F21-F24 Visualization results of crossover weight of Differentiable LCrM.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Abstract.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Conclusion.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All experimental settings have been described. Once the article is accepted, the code will be released.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Once the article is accepted, the code will be released.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental settings have been described.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Conclusions.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: the paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: the paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: the paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.