ADAPTIVE EXPLORATORY LANDSCAPE ANALYSIS FOR LLM-BASED META-BLACK-BOX OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) have recently advanced Meta-Black-Box Optimization (MetaBBO) by enabling optimizer design with reduced expert intervention, while existing approaches typically rely on fixed optimization-state representations, such as predefined Exploratory Landscape Analysis (ELA) indicators. We argue that adaptive optimization-state perception is more effective than fixed optimization-state representations for improving MetaBBO performance. Motivated by this idea, we propose AdaELA, a plug-and-play adaptive ELA selection framework for LLM-based MetaBBO. AdaELA dynamically selects informative ELA indicators through a population-aware self-attention encoder and a cross-attention-based selection mechanism, and provides the selected landscape information to LLM-based meta-policies for optimizer control. We formulate the training of AdaELA as a multi-task optimization problem and optimize its parameters using a neuroevolution strategy. Extensive experiments demonstrate that AdaELA consistently improves different LLM-based MetaBBO methods, generalizes to unseen optimization tasks, and enables further performance improvements through fine-tuning. This work highlights adaptive optimization-state perception as a promising direction toward more generalizable and effective MetaBBO.

1 INTRODUCTION

Black-Box Optimization (BBO) focuses on finding high-quality solutions when the mathematical form, gradient, or other internal information of the objective function is unavailable. Since only function evaluations are required, BBO has been widely used to solve many complex optimization problems Larson et al. (2019). However, traditional black-box optimizers usually rely on manually designed search rules and expert knowledge, which may limit their ability to adapt to different problem landscapes and different stages of the search process Du et al. (2026a). To improve this adaptability, Meta-Black-Box Optimization (MetaBBO) introduces a learning-based meta-policy to adjust the behavior of an optimizer according to its current search state Ma et al. (2023; 2024). This approach reduces the reliance on expert knowledge and manual design, while improving the ability of the optimizer to generalize to unseen problems. Reinforcement Learning (RL) has become a common way to learn such a meta-policy, allowing the optimizer to select operators, tune parameters, or adjust search strategies through interaction with different optimization environments Guo et al. (2025); Du et al. (2026b). More recently, Large Language Models (LLMs) have also been introduced as meta-policies in MetaBBO, using optimization-state information to generate control decisions during the search process Huang et al. (2026); Hu et al. (2026).

RL-based MetaBBO generally follows a bi-level optimization paradigm, consisting of a meta-level and a low-level Ma et al. (2026). As illustrated on the left side of Fig. 1, the meta-level trains a meta-policy across multiple optimization environments to learn how to control the optimizer, while the low-level optimizer performs the actual search and provides feedback for updating the meta-policy. In contrast, as highlighted by the red box in Fig. 1, LLM-based MetaBBO methods can directly employ pre-trained LLMs as meta-policies due to their general reasoning capability, without further adapting the LLM parameters Guo et al. (2024b). Despite this difference, both types of MetaBBO rely on an effective perception of the low-level optimization status so that the meta-policy can understand the ongoing search process and make suitable control decisions. Such optimization status can be represented through population statistics, fitness changes, search history, or landscape features Guo et al. (2024a); Jin et al. (2026). Among these, Exploratory Landscape Analysis (ELA) provides

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

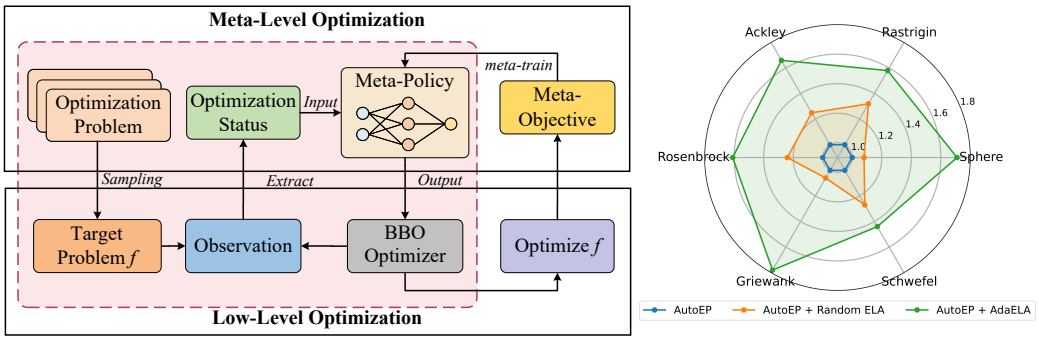

Figure 1: **Left**: Bi-level paradigm of MetaBBO. The meta-policy continuously monitors the low-level optimization progress and dynamically generates control decisions to guide the low-level BBO optimizer. **Right**: Relative performance of Original AutoEP (blue line), AutoEP with random ELA replacement (orange line), and AutoEP with AdaELA (green line) on different optimization problems (larger is better; see Eq. 5).

a structured way to characterize the current optimization landscape through a set of numerical indicators, making it a widely used approach for constructing optimization-state representations Renau et al. (2020); Mersmann et al. (2011). However, the usefulness of different ELA indicators may vary across optimization problems Bischl et al. (2012); Renau et al. (2019); Nikolikj et al. (2022), and whether they should be adaptively adjusted during different optimization stages remains underexplored. We argue that using a fixed optimization-state representation, such as a predefined set of ELA indicators, throughout the optimization process may provide less informative state perception, as the relevance of different state information may vary across optimization stages.

To examine the sensitivity of LLM-based MetaBBO to the provided ELA information, we conduct a simple motivating experiment using AutoEP Xu et al. (2026), an LLM-based MetaBBO method, as an example. We randomly replace part of the original ELA indicators used by AutoEP at every optimization iteration, such that the ELA combination varies dynamically throughout the search process. As shown on the right side of Fig. 1, this random strategy achieves better optimization performance than the original AutoEP on several representative problems. This observation suggests that a fixed set of ELA indicators may not always provide the most effective optimization-state representation for the meta-policy. **More importantly**, it motivates us to investigate whether dynamically adapting the provided landscape information can further improve optimizer control. We systematically examine this hypothesis in the subsequent experiments.

Based on this motivation, we propose AdaELA, a **plug-and-play** learnable framework that adaptively selects ELA indicators for LLM-based MetaBBO, enabling more effective optimizer control through informative landscape knowledge. As shown in Fig. 1, AdaELA achieves improved performance over fixed and random ELA strategies. We focus on LLM-based MetaBBO because RL-based MetaBBO usually relies on a fixed-dimensional state representation, making dynamically changing ELA combinations difficult without retraining the policy. In contrast, the generality of LLMs allows them to flexibly handle different ELA combinations without additional training. Designing AdaELA involves three main challenges: 1) AdaELA should adapt to diverse optimization problems, including problems with different **landscape characteristics** and **dimensions**. 2) AdaELA should be compatible with different LLM-based MetaBBO methods, allowing the same ELA selection mechanism to work with different optimizer control strategies. 3) AdaELA should generalize beyond the training setting and remain effective on previously unseen optimization problems and MetaBBO methods. We address these challenges as follows:

1. We implement AdaELA as a neural network, which combines a two-stage self-attention encoder with a cross-attention mechanism. Meanwhile, we construct an ELA indicator library $\mathcal{L}$ to provide a broad set of candidate landscape indicators. By modeling the relation between the current search state and different ELA indicators, AdaELA can dynamically adjust the information provided to the LLM to improve optimization performance, while maintaining scalability across different problem dimensions and population sizes.

2. To enable AdaELA to work with different LLM-based MetaBBO methods, we construct a multi-task operating space Ω that includes multiple MetaBBO methods evaluated on a shared set of optimization problems with different dimensions. AdaELA is optimized across these tasks, enabling the same ELA selection mechanism to adapt to different optimizer control strategies.
3. Our experiments show that AdaELA can transfer well to previously unseen LLM-based MetaBBO methods and optimization problems without additional training. Moreover, when further fine-tuned on an unseen MetaBBO method, AdaELA can achieve additional performance improvements.

Our main contribution is the finding that dynamically adapting the optimization information provided to LLM-based MetaBBO can effectively improve their performance. Based on this observation, we propose AdaELA to adaptively select suitable ELA indicators during the optimization process. This work opens a new direction for adaptive optimization-state perception in MetaBBO and further promotes the use of LLMs for optimizer control. **More broadly**, we argue that the effectiveness of optimization information varies across different problems and optimization stages, motivating adaptive perception mechanisms for LLM-based MetaBBO. AdaELA instantiates this idea through adaptive perception of ELA-based optimization states, rather than focusing solely on ELA selection. This principle can also be extended to MetaBBO methods that characterize optimization states without ELA-based representations (see Section 3.4), and our additional experiments on RL-based MetaBBO in Appendix C further suggest the broader applicability of adaptive optimization-state perception beyond LLM-based MetaBBO.

2 Related Work

LLM-Based Meta-Black-Box Optimization. MetaBBO typically formulates optimizer control as a meta-level optimization problem, where a meta-policy learns to guide the behavior of a low-level optimizer according to the observed optimization states, with the goal of improving its performance across a distribution of optimization tasks rather than on a single problem instance Ma et al. (2026); Yang et al. (2025). Existing LLM-based MetaBBO methods employ different mechanisms to control the optimization process. At a relatively coarse level, LLMs can select a suitable solver from a predefined candidate pool, allowing different optimizers to be employed during the search process Wu et al. (2024). A finer level of control is achieved through algorithm configuration, where LLMs dynamically adjust optimizer hyperparameters or operators to modify search behavior Zhong et al. (2024); Zhang & Yi (2025); Xu et al. (2026). At the solution level, LLMs can directly generate, perturb, or refine candidate solutions, thereby partially or even fully taking over the role of the underlying optimizer in proposing new solutions Yang et al. (2024); Wang et al. (2024). These approaches mainly differ in the granularity of control over an ongoing optimization process Ma et al. (2025b). However, existing studies mainly focus on how LLMs control the optimization process, while much less attention has been paid to what optimization information should be provided to the LLM and how this information should be represented. AdaELA addresses this gap by adaptively constructing the optimization-state representation provided to the LLM according to the current search status. In addition, another line of research focuses on automated algorithm design, where LLMs synthesize new heuristics or optimization workflows rather than controlling an ongoing search process Liu et al. (2024); Ye et al. (2024); Van Stein & Bäck (2024). Since this setting differs from the online optimizer-control scenario considered in AdaELA, automated algorithm design methods are beyond the scope of this work.

Landscape Analysis. Landscape analysis provides a principled way to characterize optimization problems from observable samples and has long served as an information interface for automated algorithm selection and configuration Rice (1976); Kerschke & Trautmann (2019). For continuous black-box optimization, ELA represents a problem using numerical descriptors computed from sampled solutions and their objective values Mersmann et al. (2011). Classical ELA relies on handcrafted features capturing complementary properties such as fitness-distance relationships, dispersion, ruggedness, and funnel structures Jones & Forrest (1995); Lunacek & Whitley (2006); Muñoz et al. (2014); Kerschke et al. (2015). While interpretable, these representations depend on manually designed descriptors and can be sensitive to the underlying sampling strategy Renau et al. (2020). Recent studies have also explored learning-based landscape representations, where neural models

directly encode sampled points or fitness-based representations without relying exclusively on predefined feature sets Prager et al. (2021); Seiler et al. (2022). DeepELA further learns transferable landscape representations through self-supervised Transformer pretraining, reducing the dependence on labeled task-specific training data Seiler et al. (2025).

Most landscape-analysis methods focus on static problem characterization Cenikj et al. (2026), whereas MetaBBO additionally requires representations that reflect the evolving optimization state. NeurELA extends neural landscape analysis to this setting by learning dynamic representations from observations generated during optimization, allowing landscape information to serve as state input for different MetaBBO tasks Ma et al. (2025a). This development reflects a broader transition from characterizing intrinsic problem properties toward representing the interaction between the landscape and the ongoing search process. For LLM-based MetaBBO, this naturally leads to the question of which landscape and search-state information should be provided to the LLM and how such information should be organized into an effective optimization-state representation. AdaELA addresses this issue through adaptive perception of ELA-based optimization states.

3 METHODOLOGY

3.1 FRAMEWORK

As illustrated in Fig. 1, LLM-based MetaBBO methods generate control decisions for low-level optimizers based on the optimization information provided to the LLM-based meta-policy. However, the effectiveness of optimization information varies across different problems and optimization stages. Therefore, we propose AdaELA, a learnable adaptive information perception framework that dynamically determines suitable ELA indicators for LLM-based MetaBBO according to the current optimization state. At each optimization iteration, AdaELA first encodes the current population and the ELA indicators from the library $\mathcal{L}$ into corresponding representations. The population representation is then processed by a two-stage population self-attention encoder to capture the current optimization state. A cross-attention mechanism subsequently models the relationship between the search state and the corresponding ELA indicator values, producing a selection probability for each indicator through a sigmoid function. Indicators with probabilities higher than a predefined threshold τ are selected and delivered to the LLM-based meta-policy, which generates control decisions for the low-level optimizer. Through this process, AdaELA dynamically selects more informative landscape information for the LLM during optimization.

To enable AdaELA to generalize across diverse optimization problems and MetaBBO methods, we formulate its training as a multi-task optimization problem over a task space consisting of different optimization problems and MetaBBO algorithms. The parameters of AdaELA are optimized by a neuroevolution strategy with the objective of maximizing the performance improvement brought to the integrated MetaBBO methods.

3.2 ARCHITECTURE OF ADAELA

Our architecture is designed to adapt to diverse optimization problems and various MetaBBO algorithms. At each optimization iteration t, as illustrated in Fig. 2, AdaELA separately encodes the current population $\{\boldsymbol{x}_i^t, y_i^t\}_{i=1}^n$ and the corresponding ELA indicator values $\{z_i^t\}_{i=1}^m$. For a d-dimensional optimization problem, the decision variables are normalized using min-max normalization according to their search ranges, while the objective values are normalized based on the running minimum and maximum objective values observed up to iteration t. This ensures unified representation and generalization by scaling all values to [0, 1]. The normalized population is then reorganized into a structured representation $\{\{x_{i,j}^t, y_i^t\}_{i=1}^n\}_{j=1}^d \in \mathbb{R}^{d\times n\times 2}$, where each element contains the j-th decision variable and the corresponding objective value of an individual. Subsequently, the structured population representation is linearly projected through an embedding matrix $W_{\text{pop}} \in \mathbb{R}^{2\times h}$ to obtain the population token representation $Q_{\text{pop}}^t \in \mathbb{R}^{d\times n\times h}$. For each ELA indicator, a shared linear embedding matrix $W_{\text{ELA}} \in \mathbb{R}^{1\times h}$ is used to obtain its embedded representation. To preserve the semantic identity of different ELA indicators, an indicator-specific learnable embedding is added to obtain ELA token representation $Q_{\text{ELA}}^t \in \mathbb{R}^{m\times h}$. Here, h denotes the hidden dimension.

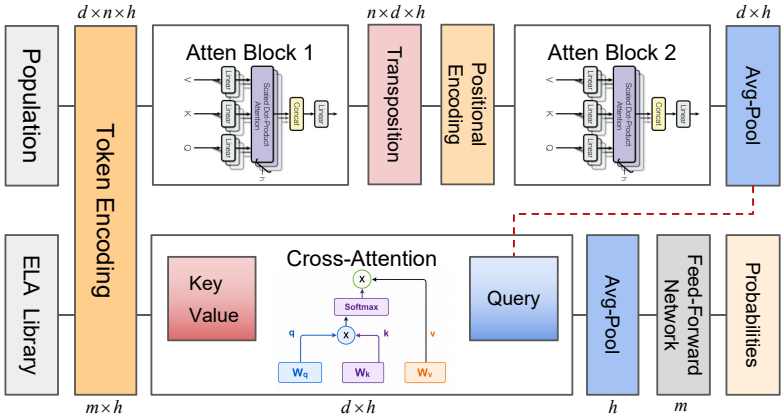

Figure 2: The architecture of AdaELA.

Two-Stage Population Self-Attention Encoder. The attention block in AdaELA follows the Transformer architecture Vaswani et al. (2017), where layer normalization Ba et al. (2016) is adopted instead of batch normalization Ioffe & Szegedy (2015) to improve the stability under different optimization environments. Given the population representation $Q^t_{\text{pop}} \in \mathbb{R}^{d\times n\times h}$, AdaELA employs a two-stage self-attention encoder to enhance information sharing at **individual** and **variable** levels:

1) In the first stage, an attention block is applied along the individual dimension. Specifically, for each decision variable, the representations of n individuals are treated as a sequence and processed by the attention block, enabling information exchange among different solutions. The resulting representation is denoted as $H^t_1 \in \mathbb{R}^{d\times n\times h}$.

2) To further model dependencies among decision variables, the first two dimensions of H^t_1 are transposed, obtaining ${H^t_1}^T \in \mathbb{R}^{n\times d\times h}$. Cosine/Sine positional encoding is then added to preserve the order information of decision variables. Subsequently, another attention block is applied along the variable dimension, producing $H^t_2 \in \mathbb{R}^{n\times d\times h}$. Finally, the output is aggregated along the individual dimension through pooling, resulting in the final search state representation $S_t \in \mathbb{R}^{d\times h}$, which is used for the subsequent ELA selection process.

Cross-Attention based ELA Selection. The cross-attention module in AdaELA is designed to model the relation between the current optimization state and different ELA indicators. Unlike self-attention, which focuses on information interaction within the same representation space, cross-attention enables the optimization state to interact with external landscape information. Specifically, the search state representation $S_t \in \mathbb{R}^{d\times h}$ obtained from the two-stage population self-attention encoder is used as the Query, while the ELA token representation $Q^t_{\text{ELA}} \in \mathbb{R}^{m\times h}$ serves as the Key and Value. Through this interaction, AdaELA evaluates the relevance between the current search state and different ELA indicators, thereby extracting landscape information that is more suitable for the current optimization process. The cross-attention operation and the subsequent ELA selection process are formulated as:

$$H_t = \text{Softmax}\left(\frac{(S_t W_Q)(Q^t_{\text{ELA}} W_K)^T}{\sqrt{h}}\right)(Q^t_{\text{ELA}} W_V) \in \mathbb{R}^{d\times h}, \tag{1}$$

$$p_t = \text{MLP}(H^{\text{pool}}_t), \tag{2}$$

where $H^{\text{pool}}_t \in \mathbb{R}^h$ is obtained by applying average pooling over the problem-dimension, MLP is implemented as a single-layer feed-forward network that maps H^{pool}_t to an m-dimensional output, followed by a sigmoid activation function to produce the selection probabilities of the m ELA indicators. At each optimization step t, the selected ELA set is determined according to:

$$\mathcal{Z}_t = \left\{ z^t_i \mid p_{t,i} > \tau,\ i = 1, \ldots, m \right\}. \tag{3}$$

The proposed attention-based design can naturally scale across different problem dimensions and population sizes of low-level optimizers. Meanwhile, the normalized encoding strategy reduces the

impact of different variable ranges and objective scales, enabling AdaELA to adapt to diverse optimization problems with different characteristics. Additional technical details about the architecture of AdaELA can be found in **Appendix A.1**.

3.3 Training Method

Multi-Task Operating Space. To enable AdaELA to work across different optimization problems and LLM-based MetaBBO methods, we construct a multi-task operating space Ω. We first define an algorithm set $\mathcal{A} = \{A_1, A_2, \ldots, A_K\}$, which contains different LLM-based MetaBBO methods, and a problem set $\mathcal{D} = \{D_1, D_2, \ldots, D_L\}$, which contains optimization problems with diverse landscape properties and dimensions. The multi-task operating space is then constructed as the Cartesian product of these two sets, i.e., $\Omega = \mathcal{A} \times \mathcal{D}$. Each task $\mathcal{T}_{k,l} \in \Omega$ represents the application of a MetaBBO method A_k to an optimization problem D_l.

Meta-Objective of AdaELA. The training goal of AdaELA is to obtain a single parameter set θ^* that performs well across the whole multi-task operating space, rather than specializing in a single optimization problem or MetaBBO method. Therefore, we define the meta-objective as maximizing the expected performance gain of AdaELA over all tasks in Ω. Mathematically, this is expressed as:

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{\mathcal{T}\sim\Omega} \left[\mathcal{J}(\theta \mid \mathcal{T})\right] = \arg\max_{\theta} \frac{1}{K \times L} \sum_{k=1}^{K} \sum_{l=1}^{L} \mathcal{J}\left(\theta \mid \mathcal{T}_{k,l}\right), \tag{4}$$

where $\mathcal{J}\left(\theta \mid \mathcal{T}_{k,l}\right)$ measures the performance gain brought by AdaELA on task $\mathcal{T}_{k,l}$. To measure $\mathcal{J}(\theta \mid \mathcal{T}_{k,l})$, we adopt the Z-score normalized performance gain following Ma et al. (2023; 2025a). Each task is independently executed Q times. Let $f^{\theta}_{k,l,q}$ denote the objective value obtained with AdaELA in the q-th run, while $\mu_{k,l}$ and $\sigma_{k,l}$ denote the mean and standard deviation of the original MetaBBO method over Q independent runs. For minimization problems, the performance gain is defined as:

$$\mathcal{J}(\theta \mid \mathcal{T}_{k,l}) = \frac{1}{Q} \sum_{q=1}^{Q} -\frac{f^{\theta}_{k,l,q} - \mu_{k,l}}{\sigma_{k,l} + \epsilon}. \tag{5}$$

Neuroevolution. Since the meta-objective of AdaELA is non-differentiable, we employ a neuroevolution strategy to optimize the parameters of AdaELA. Specifically, Fast Covariance Matrix Adaptation Evolution Strategy (FCMAES) Li et al. (2018) is adopted to optimize θ based on the meta-objective defined in Eq. 4. Accordingly, the fitness of a parameter set θ is defined as

$$\mathcal{F}(\theta) = \frac{1}{KLQ} \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{q=1}^{Q} -\frac{f^{\theta}_{k,l,q} - \mu_{k,l}}{\sigma_{k,l} + \epsilon}, \tag{6}$$

where a larger $\mathcal{F}(\theta)$ indicates a greater performance improvement over the original MetaBBO methods across the multi-task operating space Ω. During the optimization process, evaluating a single parameter set requires executing the corresponding LLM-based MetaBBO methods over all tasks in Ω with multiple independent runs, resulting in a large computational cost. Therefore, we employ the **Ray framework** Moritz et al. (2018) to parallelize the evaluation across different tasks and repeated runs, thereby accelerating the neuroevolution process.

3.4 Integration with LLM-based MetaBBO Methods

AdaELA is designed as a plug-and-play module that can be integrated into different LLM-based MetaBBO methods without modifying their original optimization frameworks. According to whether the original method incorporates ELA information, AdaELA adopts different integration strategies. For MetaBBO methods that already use ELA indicators, such as AutoEP, AdaELA replaces the original manually designed ELA information construction strategy with an adaptive selection mechanism, enabling the LLM-based meta-policy to receive more informative landscape information during optimization. For methods that do not utilize ELA information, such as OPRO, which directly provides population information as the optimization context, AdaELA introduces the selected ELA indicators as complementary landscape information while preserving the original

population-based input. **Notably**, AdaELA essentially realizes adaptive optimization-state perception by dynamically adjusting the optimization-state representation provided to LLM-based meta-policies, which aligns with our motivation that the effectiveness of different optimization-state information varies across optimization problems and stages.

Furthermore, to improve the interpretability of the provided landscape information, AdaELA supplements each selected ELA indicator with its semantic description when constructing the input for the LLM-based meta-policy. These descriptions provide the meaning of the selected indicators, allowing the LLM to better understand the current optimization landscape and make more effective decisions. In this work, we construct an ELA indicator library containing 25 indicators as the candidate landscape information pool for AdaELA. The detailed definitions and semantic descriptions of these indicators are provided in **Appendix A.2**.

4 Experimental Studies

In this section, we discuss the following research questions: **RQ1**: Can AdaELA generalize to unseen optimization problems? **RQ2**: Can AdaELA further generalize to unseen MetaBBO algorithms on unseen optimization problems? **RQ3**: Is the performance improvement of AdaELA attributed to adaptive ELA selection rather than simply providing richer landscape information? **RQ4**: Can AdaELA be further fine-tuned on unseen MetaBBO algorithms to achieve additional performance improvements? **RQ5**: What does AdaELA have learned? **RQ6**: How does model complexity affect the performance of AdaELA? Below, we first introduce the experimental settings and then address RQ1–RQ6. Additional experimental results and analyses are provided in **Appendix B**.

Experimental Setup. For the multi-task operating space Ω, we select $K = 3$ representative LLM-based MetaBBO methods, including AutoEP Xu et al. (2026), OPRO Yang et al. (2024), and LAOS Zhang & Yi (2025). We use the BBOB test suites in the COCO platform Hansen et al. (2021) as meta-training optimization problems, which contain diverse optimization landscapes with various characteristics. For meta-training, we employ FCMAES with a population size of 10 and 100 optimization iterations. Each BBOB function-instance is evaluated with $Q = 15$ independent runs across multiple problem dimensions to improve the generalization ability of AdaELA. The detailed dimension configuration is provided in **Appendix A.3**. For the AdaELA architecture, the hidden dimension is set to $h = 16$. Both self-attention and cross-attention modules adopt a single-head attention. The resulting model contains 5,225 learnable parameters in total. The selection threshold is set to $\tau = 0.5$ to determine whether an ELA indicator is provided to the LLM-based meta-policy. To improve computational efficiency, we parallelize the 150 meta-training pipelines using the Ray framework. For the low-level optimization process, the maximum number of function evaluations is set to 4,000. All LLM-based MetaBBO methods use the same LLM, Qwen3-30B.

Baselines. We compare AdaELA with three baseline variants to evaluate the effectiveness of adaptive ELA selection: 1) **Original**: The original MetaBBO methods with their default designs, without any modification of the provided optimization information. 2) **Random ELA**: At each optimization iteration t, this baseline randomly selects the same number of ELA indicators as AdaELA from $\mathcal{L}$, while keeping the selected indicators random. 3) **All ELA**: All available ELA indicators in the library $\mathcal{L}$ are directly provided to the meta-policy at each optimization iteration, without any selection process.

4.1 Generalization performance (RQ1-3)

To evaluate the generalization ability of AdaELA, we conduct experiments from three perspectives. For RQ1, we further evaluate AdaELA on two unseen optimization scenarios beyond the meta-training distribution, including the BBOB-NOISY test suite Hansen et al. (2021) and the Human-Powered Aircraft Design problem Namura (2025), to investigate whether AdaELA can generalize to different optimization landscapes and real-world optimization tasks. For RQ2, we directly integrate the trained AdaELA into three unseen LLM-based MetaBBO methods, including EvoLLM Lange et al. (2024), LEO Brahmachary et al. (2025), and PAIR Ali et al. (2025), without additional meta-training. These methods are evaluated on three test scenarios to examine whether AdaELA can generalize across different MetaBBO algorithms and optimization problems. Additionally, we compare AdaELA with three baselines to answer RQ3.

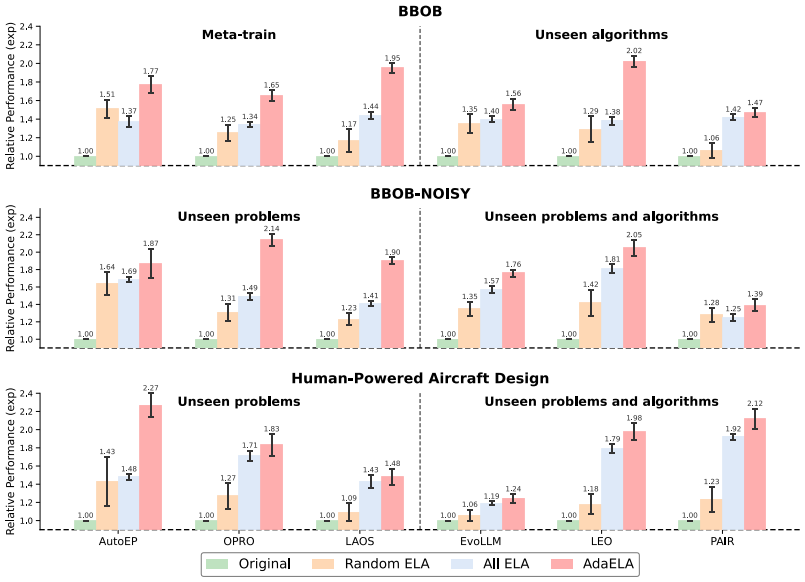

Figure 3: Generalization performance of AdaELA across unseen optimization problems and MetaBBO methods.

The relative performance shown in Fig. 3 is calculated according to Eq. 5, averaged over the corresponding test tasks, and then exponentiated for visualization, such that the original method corresponds to a reference value of 1 Ma et al. (2025a). AdaELA achieves clear performance gains on unseen optimization problems beyond the meta-training distribution, indicating that its learned information perception mechanism can adapt to diverse optimization landscapes. Moreover, it continues to improve performance when transferred to unseen MetaBBO methods and problems, demonstrating strong cross-task and cross-algorithm generalization. Additionally, the comparison with Random ELA demonstrates that AdaELA has learned an effective adaptive information perception mechanism. The comparison with All ELA further indicates that the improvement of AdaELA mainly comes from adaptively perceiving informative optimization states rather than simply providing additional landscape information. This may be attributed to the fact that excessive and redundant landscape information can introduce unnecessary context for the meta-policy, making it more difficult to identify the most relevant optimization states for generating effective optimizer control decisions.

4.2 Fine-tuning Performance (RQ4)

Although AdaELA demonstrates strong zero-shot generalization and can be directly integrated into unseen MetaBBO methods without additional meta-training, we further investigate whether task-specific adaptation can provide additional performance improvements. Specifically, we use the top-10 performing individuals obtained from the meta-training stage as the initial population of FCMAES and perform a small number of additional evolutionary iterations for adaptation. We select PAIR on BBOB and BBOB-NOISY, and EvoLLM on the Human-Powered Aircraft Design problem for fine-tuning. As shown in Fig. 4, the performance gain curves of the population mean with the corresponding standard deviation are presented over FCMAES generations. The results show that AdaELA can achieve further performance improvements on specific tasks with only a limited number of evolutionary adaptation iterations. Due to space limitations, we further analyze the generalization ability of the fine-tuned models on unseen problems in **Appendix B.2**.

4.3 What has AdaELA Learned? (RQ5)

To further analyze whether AdaELA learns stage-dependent landscape perception, we investigate the evolution of selected ELA indicators during the optimization process. Taking AutoEP as an example, we represent the selected ELA indicators at each optimization iteration as binary vectors and measure their differences from the initial optimization stage using the Jaccard distance. A larger

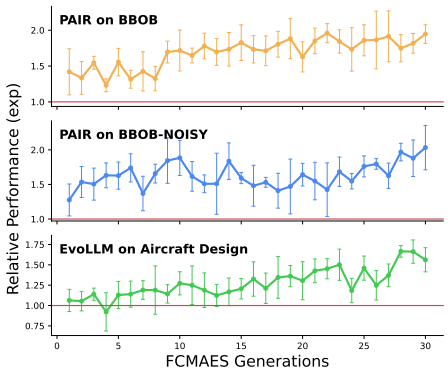

Figure 4: Performance gain curves of AdaELA fine-tuning.

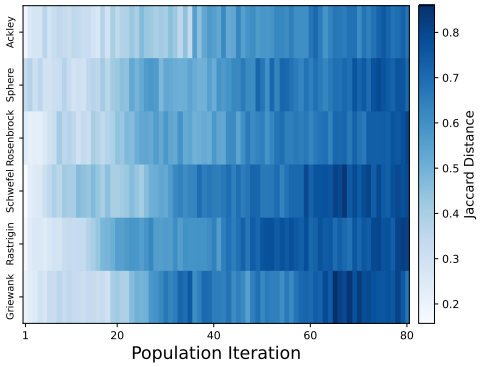

Figure 5: Stage-dependent optimization-state perception of AdaELA.

Jaccard distance indicates a greater change in the perceived landscape information compared with the early stage (see **Appendix B.3** for detailed calculation). As shown in Fig. 5, we visualize the evolution of ELA perception using heatmaps, where the horizontal axis represents the population iteration of the optimizer and the vertical axis denotes different optimization problems. The selected ELA combinations remain relatively stable during the initial optimization stage, while gradually exhibiting larger variations as the optimization proceeds. This indicates that AdaELA does not rely on a fixed ELA combination throughout the optimization process, but instead learns to adaptively adjust the perceived landscape information according to different optimization stages, enabling the LLM-based meta-policy to receive more suitable information for optimizer control.

4.4 Model Complexity Comparison (RQ6)

To further analyze the impact of model complexity in AdaELA's attention architecture, we compare 6 configurations with different numbers of attention heads $Atten \in \{1, 3, 5\}$ and hidden dimensions $h \in \{16, 64\}$. The results are visualized using a 2D scatter plot (see Fig. 6). We evaluate the influence of model complexity from two perspectives: relative performance on unseen optimization tasks to assess generalization ability, and relative performance on specific tasks. The experimental results reveal an interesting trade-off between model complexity and optimization performance. As model complexity increases, the relative performance on unseen tasks tends to decrease, while the relative performance on specific tasks improves. This indicates that larger models provide stronger fitting capability for task-specific optimization scenarios, but may reduce the transferability of learned optimization-state perception.

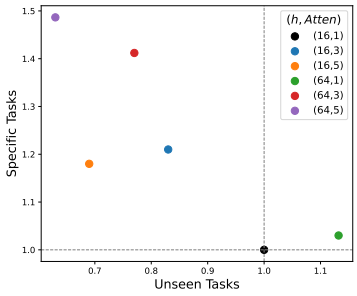

Figure 6: The influence of model complexity.

5 Conclusion

In this paper, we propose AdaELA, a plug-and-play adaptive optimization-state perception framework for LLM-based MetaBBO, which dynamically selects informative ELA indicators according to the current optimization state and provides effective landscape information for LLM-based optimizer control. Extensive experiments demonstrate that AdaELA consistently improves the performance of various LLM-based MetaBBO methods and generalizes well to unseen optimization problems and MetaBBO algorithms without additional meta-training. The results further support our viewpoint that adaptively perceiving informative optimization states is more effective than relying on fixed or excessive landscape information. While AdaELA still has certain limitations, such as the computational cost of meta-training, this work represents a significant step toward more adaptive and intelligent LLM-based optimization.

AI USE STATEMENT

In this work, we used generative AI tools only to assist with language polishing and improving the clarity of the manuscript. All AI-assisted text was carefully reviewed and revised by the authors. We take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Details of the AdaELA architecture, training procedure, experimental settings, ELA indicator library, and multi-task operating space are provided in Sections 3–4 and Appendix A. Additional experimental details are reported in Appendix B. Random seeds are fixed in all experiments, and the source code together with remaining implementation details will be released upon acceptance.

REFERENCES

Shady Ali, Mahmoud Ashraf, Seif Hegazy, Fatty Salem, Hoda Mokhtar, Mohamed Medhat Gaber, and Mohamed Taher Alrefaie. Pair: A novel large language model-guided selection strategy for evolutionary algorithms. *arXiv preprint arXiv:2503.03239*, 2025.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike Preuß. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pp. 313–320, 2012.

Shuvayan Brahmachary, Subodh M. Joshi, Aniruddha Panda, Kaushik Koneripalli, Arun Kumar Sagotra, Harshil Patel, Ankush Sharma, Ameya D. Jagtap, and Kaushic Kalyanaraman. Large language model-based evolutionary optimizer: Reasoning with elitism. *Neurocomputing*, 622: 129272, 2025.

Gjorgjina Cenikj, Ana Nikolikj, Gašper Petelin, Niki Van Stein, Carola Doerr, and Tome Eftimov. A survey of features used for representing black-box single-objective continuous optimization. *Swarm and Evolutionary Computation*, 101:102288, 2026.

Yukun Du, Haiyue Yu, Jiang Jiang, Shuaiwen Tang, Xiaotong Xie, Haobo Liu, Chongshuang Hu, and Shengkun Chang. Meta-black-box optimization can do search guidance for expensive constrained multi-objective optimization. In *Forty-third International Conference on Machine Learning*, 2026a.

Yukun Du, Haiyue Yu, Xiaotong Xie, Yan Zheng, Lixin Zhan, Yudong Du, Chongshuang Hu, Boxuan Wang, and Jiang Jiang. Meta-black-box optimization with bi-space landscape analysis and dual-control mechanism for saea. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 36891–36899, 2026b.

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

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, pp. 26982–26990, 2025.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 34133–34156, 2024b.

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.

Caie Hu, Haoyue Zhang, Tingyang Jiao, Xiaoliang Yu, and Shangbin Jiao. Large language model-driven evolutionary optimization: A hierarchical interaction framework. *Expert Systems with Applications*, pp. 134167, 2026.

Yuxiao Huang, Shenghao Wu, Wenjie Zhang, Jibin Wu, Liang Feng, and Kay Chen Tan. Autonomous multiobjective optimization using large language model. *IEEE Transactions on Evolutionary Computation*, 30(2):594–608, 2026.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456, 2015.

Xiao Jin, Yongxiong Wang, Haobo Liu, Yudong Du, and Yukun Du. Meta-black-box optimization with ensemble surrogate modeling for robustness–accuracy trade-off within saea. *Swarm and Evolutionary Computation*, 106:102425, 2026.

Terry Jones and Stephanie Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *ICGA*, volume 95, pp. 184–192, 1995.

Pascal Kerschke and Heike Trautmann. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary computation*, 27(1):99–127, 2019.

Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. Detecting funnel structures by means of exploratory landscape analysis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 265–272, 2015.

Robert Lange, Yingtao Tian, and Yujin Tang. Large language models as evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 579–582, 2024.

Jeffrey Larson, Matt Menickelly, and Stefan M Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, 2019.

Zhenhua Li, Qingfu Zhang, Xi Lin, and Hui-Ling Zhen. Fast covariance matrix adaptation for large-scale black-box optimization. *IEEE transactions on cybernetics*, 50(5):2073–2083, 2018.

Fei Liu, Tong Xialiang, 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. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223, 2024.

Monte Lunacek and Darrell Whitley. The dispersion metric and the cma evolution strategy. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 477–484, 2006.

Zeyuan Ma, Hongshu Guo, Jiacheng Chen, Zhenrui Li, Guojun Peng, Yue-Jiao Gong, Yining Ma, and Zhiguang Cao. Metabox: A benchmark platform for meta-black-box optimization with reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10775–10795, 2023.

Zeyuan Ma, Jiacheng Chen, Hongshu Guo, Yining Ma, and Yue-Jiao Gong. Auto-configuring exploration-exploitation tradeoff in evolutionary computation via deep reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1497–1505, 2024.

Zeyuan Ma, Jiacheng Chen, Hongshu Guo, and Yue-Jiao Gong. Neural exploratory landscape analysis for meta-black-box-optimization. In *International Conference on Learning Representations*, volume 2025, pp. 15373–15393, 2025a.

Zeyuan Ma, Yue-Jiao Gong, Hongshu Guo, Wenjie Qiu, Sijie Ma, Hongqiao Lian, Jiajun Zhan, Kaixu Chen, Chen Wang, Zhiyang Huang, Zechuan Huang, Guojun Peng, Ran Cheng, and Yining Ma. Metabox-v2: A unified benchmark platform for meta-black-box optimization. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 2025b. doi: 10.52202/085713-5017.

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*, 30(2):667–687, 2026.

Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 829–836, 2011.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 561–577, Carlsbad, CA, 2018.

Mario A Muñoz, Michael Kirley, and Saman K Halgamuge. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE transactions on evolutionary computation*, 19(1):74–87, 2014.

Nobuo Namura. Single and multi-objective optimization benchmark problems focusing on human-powered aircraft design. In *Evolutionary Multi-Criterion Optimization*, pp. 195–210, 2025.

Ana Nikolikj, Risto Trajanov, Gjorgjina Cenikj, Peter Korošec, and Tome Eftimov. Identifying minimal set of exploratory landscape analysis features for reliable algorithm performance prediction. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2022.

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke. Towards feature-free automated algorithm selection for single-objective continuous black-box optimization. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2021.

Quentin Renau, Johann Dreo, Carola Doerr, and Benjamin Doerr. Expressiveness and robustness of landscape features. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2048–2051, 2019.

Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. Exploratory landscape analysis is strongly sensitive to the sampling strategy. In *International Conference on Parallel Problem Solving from Nature*, pp. 139–153, 2020.

John R Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pp. 65–118. 1976.

Moritz Vinzent Seiler, Raphael Patrick Prager, Pascal Kerschke, and Heike Trautmann. A collection of deep learning-based feature-free approaches for characterizing single-objective continuous fitness landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 657–665, 2022.

Moritz Vinzent Seiler, Pascal Kerschke, and Heike Trautmann. Deep-ela: Deep exploratory landscape analysis with self-supervised pretrained transformers for single-and multiobjective continuous optimization problems. *Evolutionary Computation*, 33(4):513–540, 2025.

Mudita Sharma, Alexandros Komninos, Manuel López-Ibáñez, and Dimitar Kazakov. Deep reinforcement learning based parameter control in differential evolution. In *Proceedings of the genetic and evolutionary computation conference*, pp. 709–717, 2019.

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

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Niki Van Stein and Thomas Bäck. Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29 (2):331–345, 2024.

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.

Zeyi Wang, Songbai Liu, Jianyong Chen, and Kay Chen Tan. Large language model-aided evolutionary search for constrained multiobjective optimization. In *International Conference on Intelligent Computing*, pp. 218–230, 2024.

Xingyu Wu, Yan Zhong, Jibin Wu, Bingbing Jiang, and Kay Chen Tan. Large language model-enhanced algorithm selection: towards comprehensive algorithm representation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024.

Zhenxing Xu, Yizhe Zhang, Weidong Bao, Hao Wang, Ming Chen, Haoran Ye, Wenzheng Jiang, Hui Yan, and Ji Wang. Autoep: Llms-driven automation of hyperparameter evolution for metaheuristic algorithms. In *International Conference on Learning Representations*, volume 2026, pp. 139927–139945, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024.

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.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *Advances in neural information processing systems*, 37:43571–43608, 2024.

Yisong Zhang and Guoxing Yi. Laos: Large language model-driven adaptive operator selection for evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 517–526, 2025.

Rui Zhong, Shilong Zhang, Jun Yu, and Masaharu Munetomo. Geminide: A novel parameter adaptation scheme in differential evolution. In *2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS)*, pp. 33–38, 2024.

A TECHNICAL DETAILS

A.1 ARCHITECTURE DESIGN OF ADAELA

Token Encoding. To provide additional implementation details of AdaELA, we describe the detailed architecture of the population encoder and ELA selection module. At optimization iteration t, the current population is denoted as $\mathcal{P}^t = \{\boldsymbol{x}_i^t, y_i^t\}_{i=1}^n$, where $\boldsymbol{x}_i^t \in \mathbb{R}^d$ represents the decision vector of the i-th individual and y_i^t denotes its corresponding objective value. Meanwhile, the calculated ELA indicators are represented as $\{z_i^t\}_{i=1}^m$, where m is the number of candidate indicators. For convenience, we omit the superscript t in the following descriptions. Before encoding, the decision variables and objective values are normalized to obtain a unified representation. Specifically, the j-th decision variable and the objective value of the i-th individual are normalized as

$$\tilde{x}_{i,j} = \frac{x_{i,j} - l_j}{u_j - l_j}, \qquad \tilde{y}_i = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}}, \tag{7}$$

where l_j and u_j denote the lower and upper bounds of the j-th decision variable, respectively, and $y_{\min}$ and $y_{\max}$ denote the running minimum and maximum objective values observed up to the current optimization iteration t, respectively. Both normalized decision variables and objective values are mapped into $[0, 1]$, reducing the influence of different search ranges and objective scales across optimization problems. After normalization, the population is reorganized along the decision variable dimension to capture the interaction between each decision variable and the corresponding objective information. Specifically, the normalized population is transformed as $\{\{\tilde{x}_{i,j}, \tilde{y}_i\}_{i=1}^n\}_{j=1}^d \in \mathbb{R}^{d\times n\times 2}$. Subsequently, each variable-objective pair is projected into the hidden embedding space through a shared linear transformation:

$$Q_{\text{pop}} = \{[\tilde{x}_{i,j}, \tilde{y}_i]W_{\text{pop}}\}_{i=1,j=1}^{n,d} \in \mathbb{R}^{d\times n\times h}, \tag{8}$$

where $W_{\text{pop}} \in \mathbb{R}^{2\times h}$ is a learnable embedding matrix and h denotes the hidden dimension. The resulting Q_{pop} serves as the population token representation for the subsequent self-attention encoder. For the ELA indicators, each scalar indicator value is projected into the hidden space using a shared linear transformation. To preserve the semantic identity of different landscape indicators, a learnable indicator-specific embedding is further introduced. The ELA token representation is formulated as

$$Q_{\text{ELA}} = \{z_i W_{\text{ELA}} + \boldsymbol{e}_i\}_{i=1}^m \in \mathbb{R}^{m\times h}, \tag{9}$$

where $W_{\text{ELA}} \in \mathbb{R}^{1\times h}$ denotes the shared embedding matrix and $\boldsymbol{e}_i \in \mathbb{R}^h$ represents the learnable embedding associated with the i-th ELA indicator.

Two-Stage Population Self-Attention Encoder. We construct a two-stage attention block to aggregate optimization status information across candidate solutions and across each dimension of the decision variables. The attention block mainly follows designs of the original Transformer Vaswani et al. (2017), except that the layer normalization Ba et al. (2016) is used instead of batch normalization Ioffe & Szegedy (2015). For the j-th decision variable, the corresponding encoding vectors are denoted as $\boldsymbol{q}_j \in \mathbb{R}^{n\times h}$. The first attention block then performs the following computation:

$$\begin{aligned} \boldsymbol{g}_j &= \text{LN}\left(\boldsymbol{q}_j + \text{MHSA}(\boldsymbol{q}_j)\right), \\ \boldsymbol{v}_j &= \text{FF}^{(2)}\left(\text{ReLU}(\text{FF}^{(1)}(\boldsymbol{g}_j))\right), \\ \boldsymbol{h}_j &= \text{LN}(\boldsymbol{g}_j + \boldsymbol{v}_j). \end{aligned} \tag{10}$$

where $\text{MHSA}(\cdot)$ denotes the multi-head self-attention operation, $\text{FF}^{(1)}(\cdot)$ and $\text{FF}^{(2)}(\cdot)$ denote two fully connected layers, and $\text{LN}(\cdot)$ represents layer normalization. The output $\boldsymbol{h}_j$ holds identical shape with the input $\boldsymbol{q}_j$. After processing all encoding vectors through the first attention block, we obtain the first-stage representation $H_1 = \{\boldsymbol{h}_j\}_{j=1}^d \in \mathbb{R}^{d\times n\times h}$. Here, the first self-attention operation is performed along the candidate solution dimension, enabling information interaction among different solutions for the same decision variable. Each encoding vector can aggregate the information from other candidate solutions, thereby capturing the population-level optimization status. After obtaining the first-stage representation H_1, we transpose the first two dimensions and add Cosine/Sine positional encoding to preserve the ordering information of decision variables. The second attention block is then applied along the decision variable dimension. Finally, average pooling is performed along the candidate solution dimension to aggregate population-level information

and obtain the final optimization-state representation S. The overall computation process of the two-stage self-attention encoder is summarized as:

$$\begin{aligned} H_1 &= Attn^{(1)}(Q_{\text{pop}}) \in \mathbb{R}^{d\times n\times h}, \\ H_1^T &= \text{Transpose}(H_1) + \text{PE} \in \mathbb{R}^{n\times d\times h}, \\ H_2 &= Attn^{(2)}(H_1^T) \in \mathbb{R}^{n\times d\times h}, \\ S &= \text{AvgPool}_n(H_2) \in \mathbb{R}^{d\times h}. \end{aligned} \tag{11}$$

Cross-Attention Module. The cross-attention module is designed to model the relationship between the optimization-state representation and different ELA indicators. Specifically, the state representation S obtained from the two-stage self-attention encoder is used as the Query, while Q_{ELA} serves as both the key and value. The cross-attention operation follows Eq. (1) in the main text, producing the landscape-aware representation H. The output representation is further processed by average pooling and a feed-forward layer with sigmoid activation to obtain the selection probabilities of different ELA indicators. The overall computation after cross-attention can be summarized as:

$$\begin{aligned} H^{\text{pool}} &= \text{AvgPool}_d(H) \in \mathbb{R}^h, \\ \boldsymbol{p} &= \text{Sigmoid}(W_o H^{\text{pool}} + b_o) \in \mathbb{R}^m, \\ \mathcal{Z} &= \{z_i | p_i > \tau,\ i = 1, \ldots, m\}. \end{aligned} \tag{12}$$

where W_o and b_o denote the parameters of the output layer, and $\boldsymbol{p}$ represents the selection probability vector of all candidate ELA indicators. The selected indicators are subsequently provided to the LLM-based meta-policy for optimizer control.

A.2 Construction of the ELA Indicator Library

To support adaptive optimization-state perception, we construct an ELA indicator library $\mathcal{L}$ comprising 25 complementary descriptors of the sampled landscape and search dynamics. These indicators characterize fitness distributions, fitness–distance relationships, population geometry, local fitness variation, and optimization progress. At each optimization iteration, AdaELA selects a subset of indicators from $\mathcal{L}$ conditioned on the current optimization state. Each selected indicator is incorporated into the prompt of the LLM-based meta-policy together with its name, numerical value, and a semantic description. This contextualization conveys the optimization meaning of the numerical measurements and informs the meta-policy's optimizer-control decisions. Table 1 summarizes the complete indicator library, including the descriptions and computation formulas.

Table 1: Landscape indicators used in the Adaptive ELA framework.

ELA	Description	Formula
`fitness_mean`	Average objective value of the current population, summarizing overall search quality.	$\bar{f} = \frac{1}{n}\sum_i f_i$
`fitness_std`	Standard deviation of population fitness values, indicating fitness heterogeneity.	$\sqrt{\frac{1}{n}\sum_i (f_i - \bar{f})^2}$
`fitness_skewness`	Asymmetry of the fitness distribution, indicating an imbalance between good and poor solutions.	$\frac{1}{n}\sum_i \left(\frac{f_i - \bar{f}}{s}\right)^3$
`fitness_kurtosis`	Excess kurtosis of the fitness distribution, highlighting extreme fitness outliers.	$\frac{1}{n}\sum_i \left(\frac{f_i - \bar{f}}{s}\right)^4 - 3$
`fdc`	Correlation between the distance to the current best solution and fitness difference, measuring global search guidance.	$\text{corr}\left(\lVert x_i - x_{i^*}\rVert, \lvert f_i - f_{i^*}\rvert\right)$
`best_dispersion`	Average coordinate-wise variance of the best 10% solutions; a low value indicates a concentrated promising basin.	$\frac{1}{d}\sum_{j=1}^{d} \text{Var}(\{x_{i,j} : i \in \mathcal{B}_{10\%}\})$
`mean_nn_fitness_difference`	Average fitness difference between each solution and its nearest neighbor, measuring local landscape variation.	$\frac{1}{n}\sum_i \lvert f_i - f_{nn(i)}\rvert$
`mean_abs_variable_correlation`	Mean absolute Pearson correlation among decision-variable pairs, indicating linear variable coupling.	$\frac{2}{d(d-1)}\sum_{a<b} \lvert\text{corr}(X_{:,a}, X_{:,b})\rvert$
`mean_mutual_information`	Mean pairwise mutual information among decision variables, indicating nonlinear variable interactions.	$\frac{2}{d(d-1)}\sum_{a<b} \frac{\text{MI}(X_a;X_b) + \text{MI}(X_b;X_a)}{2}$

ELA	Description	Formula
pca_first_component_ratio	Fraction of population variance explained by the first principal component, indicating a dominant low-dimensional search direction.	$\frac{\lambda_1}{\sum_r \lambda_r}$
local_fitness_ruggedness	Average absolute fitness change per decision-space distance over sampled solution pairs; high values indicate a rugged landscape.	$\frac{1}{\lvert\mathcal{P}\rvert}\sum_{(i,j)\in\mathcal{P}}\frac{\lvert f_i-f_j\rvert}{\lVert x_i-x_j\rVert+\epsilon}$
fitness_gradient_variance	Variance of nearest-neighbor finite-difference fitness slopes; high values indicate uneven local geometry.	$\mathrm{Var}_i\left(\frac{f_i-f_{nn(i)}}{\lVert x_i-x_{nn(i)}\rVert+\epsilon}\right)$
population_diversity	Average pairwise Euclidean distance within the population, measuring the current degree of exploration.	$\frac{2}{n(n-1)}\sum_{i<j}\lVert x_i-x_j\rVert$
fitness_improvement_rate	Relative improvement of the best fitness compared with that observed k generations earlier; a positive value indicates progress.	$\frac{b_{t-k}-b_t}{\lvert b_{t-k}\rvert+\epsilon}$
stagnation_score	Fraction of recent generations without meaningful relative improvement; a value close to one indicates stagnation.	$\frac{1}{k}\sum_{\tau}\mathbb{I}\left[\frac{b_{\tau-1}-b_\tau}{\lvert b_{\tau-1}\rvert+\epsilon}\le\delta\right]$
nbc_mean_distance	Mean distance from each non-best solution to its nearest strictly better solution; a large value may indicate separated attraction basins.	$\frac{1}{\lvert\mathcal{I}\rvert}\sum_{i\in\mathcal{I}}\min_{j:f_j<f_i}\lVert x_i-x_j\rVert$
mean_better_solution_ratio	Mean fraction of strictly better solutions among the k nearest neighbors, indicating the clarity of local improvement directions.	$\frac{1}{n}\sum_i\frac{1}{k}\sum_{j\in\mathcal{N}_k(i)}\mathbb{I}[f_j<f_i]$
fitness_autocorrelation	Lag-one fitness autocorrelation along a nearest-neighbor path; a high value indicates a smoother landscape.	$\mathrm{corr}([f_{p_1},\dots,f_{p_{L-1}}],[f_{p_2},\dots,f_{p_L}])$
mean_distance_correlation	Mean pairwise distance correlation among decision variables, detecting general nonlinear dependence.	$\frac{2}{d(d-1)}\sum_{a<b}\mathrm{dCor}(X_{:,a},X_{:,b})$
condition_number	Condition number of the population covariance matrix; a high value indicates elongated or ill-conditioned search geometry.	$\frac{\lambda_{\max}(\Sigma)}{\lambda_{\min}(\Sigma)+\epsilon}$
pca_variance_entropy	Shannon entropy of PCA explained-variance ratios; low entropy indicates concentrated low-dimensional structure.	$-\sum_r p_r\log p_r,\quad p_r=\frac{\lambda_r}{\sum_j\lambda_j}$
elite_fitness_variance	Fitness variance among the best 20% solutions; a low value indicates a stable promising region.	$\mathrm{Var}(\{f_i : i\in\mathcal{B}_{20\%}\})$
local_smoothness_index	Mean absolute nearest-neighbor fitness slope, measuring local smoothness versus ruggedness.	$\frac{1}{n}\sum_i\frac{\lvert f_i-f_{nn(i)}\rvert}{\lVert x_i-x_{nn(i)}\rVert+\epsilon}$
fitness_convergence_ratio	Decay of the current best-fitness change relative to the first change in a recent window; a high value suggests convergence.	$\mathrm{clip}\left(1-\frac{\lvert b_t-b_{t-1}\rvert}{\lvert b_{t-k+1}-b_{t-k}\rvert+\epsilon},0,1\right)$
population_contraction_rate	Relative reduction in population diversity compared with k generations earlier; a high value indicates rapid contraction.	$1-\frac{\mathrm{Div}(X_t)}{\mathrm{Div}(X_{t-k})+\epsilon}$

A.3 Multi-Task Operating Space

To improve the generalization ability of AdaELA across different optimization scenarios, we construct a multi-task operating space consisting of diverse LLM-based MetaBBO methods and optimization problems. Specifically, the task space is defined as $\Omega = \mathcal{A} \times \mathcal{D}$, where $\mathcal{A}$ denotes the set of LLM-based MetaBBO algorithms and $\mathcal{D}$ represents the set of optimization problems. Each task corresponds to applying a MetaBBO method to a specific optimization problem.

To improve the generalization across different problem dimensions, we introduce dimension diversity during meta-training. Specifically, each BBOB function-instance is evaluated with $Q = 15$ independent runs, where the problem dimensions are set to 5, 10, and 15, with each dimension repeated five times. By incorporating multiple dimensions within the same optimization problem, AdaELA is encouraged to learn dimension-independent optimization-state perception patterns rather than relying on fixed-dimensional representations. Combined with the diversity of MetaBBO methods and optimization problems in Ω, this design improves the adaptability of AdaELA to unseen optimization scenarios with different dimensions and landscape characteristics.

B Additional Discussion

B.1 Additional Analysis of FCMAES Meta-Training

During FCMAES meta-training, we further analyze the evolutionary process of the population by recording the mean relative performance of all individuals at each generation. As illustrated in

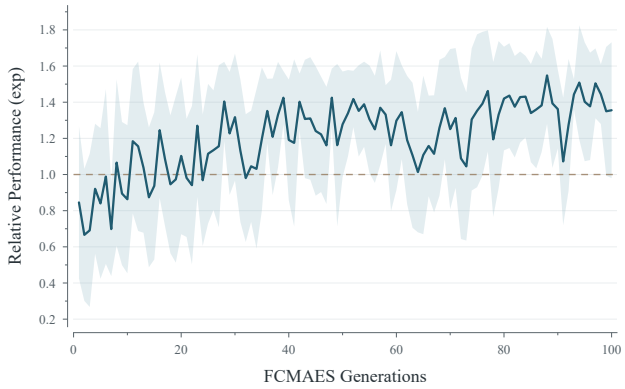

Figure 7: Evolution of relative population performance during FCMAES meta-training.

Fig. 7, the mean relative performance of the population gradually increases during the early stage of meta-training, indicating that FCMAES effectively explores the parameter space and progressively identifies more promising AdaELA configurations. As the evolution proceeds, the performance improvement gradually slows down and becomes relatively stable. **In particular**, the population performance has already reached a stable stage after approximately 50 iterations, while we use 100 iterations in our experiments to ensure sufficient optimization of AdaELA parameters. In practical applications, the number of meta-training iterations can be flexibly adjusted according to the desired balance between optimization effectiveness and computational cost. After meta-training, we select the AdaELA parameter configuration with the best relative performance among the evolved population as the final model for subsequent evaluations.

B.2 Generalization Analysis of Fine-tuned AdaELA

Although task-specific fine-tuning can further improve the performance of AdaELA on specific optimization scenarios, it is important to investigate whether such adaptation affects its generalization ability on unseen optimization environments. Therefore, we further evaluate the transferability of fine-tuned AdaELA across different benchmark suites. Specifically, we consider the three fine-tuning settings introduced in Section 4.2, including PAIR on BBOB, PAIR on BBOB-NOISY, and EvoLLM on the Human-Powered Aircraft Design problem. For each setting, the fine-tuned AdaELA is directly transferred to previously unseen optimization problems without additional fine-tuning. As shown in Fig. 8, the fine-tuned AdaELA achieves further improvements. This result indicates that task-specific fine-tuning does not merely improve performance on the target scenario, but can also further improve the performance of AdaELA on unseen problems.

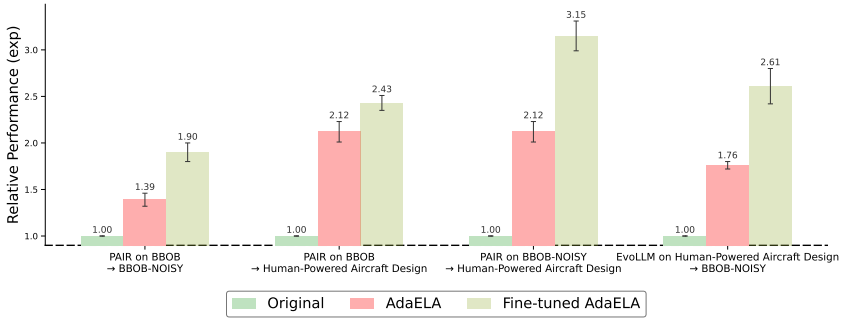

Figure 8: Generalization Analysis of Fine-tuned AdaELA.

B.3 Further analysis for RQ5

Additional Experimental Details. To quantify the stage-dependent variation of landscape information perceived by AdaELA, we represent the ELA indicators selected at optimization iteration t as a binary vector $\boldsymbol{b}^t \in \{0,1\}^m$, where $b_i^t = 1$ indicates that the i-th ELA indicator is selected and $b_i^t = 0$ otherwise. The Jaccard distance between the selected ELA sets at two iterations t and s is defined as

$$D_{\mathrm{J}}(t,s) = 1 - \frac{|\mathcal{Z}^t \cap \mathcal{Z}^s|}{|\mathcal{Z}^t \cup \mathcal{Z}^s|}, \tag{13}$$

where $\mathcal{Z}^t$ and $\mathcal{Z}^s$ denote the ELA indicator sets selected at iterations t and s, respectively. We define the first 10 optimization iterations as the early stage and measure the deviation of the perceived landscape information at iteration t by averaging its Jaccard distance to these early-stage selections:

$$\bar{D}_{\mathrm{J}}^t = \frac{1}{10}\sum_{s=1}^{10} D_{\mathrm{J}}(t,s). \tag{14}$$

A larger $\bar{D}_{\mathrm{J}}^t$ indicates a greater change in the ELA information perceived by AdaELA compared with the early optimization stage, while a smaller value indicates that the selected landscape information remains relatively similar to that in the early stage.

Effect of Stage-Dependent ELA Adaptation. To further investigate whether AdaELA indeed learns stage-dependent optimization-state perception, we conduct an additional ablation study by controlling the update interval K of the selected ELA indicators. We consider $K \in \{1, 10, \infty\}$, where $K = 1$ corresponds to the original AdaELA that updates the selected ELA set at every optimization iteration, $K = 10$ updates the selection every ten iterations, and $K = \infty$ selects ELA indicators only at the initial iteration and keeps the selected set fixed throughout the remaining optimization process. We evaluate these variants on AutoEP, a seen MetaBBO method involved in meta-training, and PAIR, an unseen MetaBBO method, while taking $K = 1$ as the reference baseline. As shown in Fig. 9 and Fig. 10, the optimization performance gradually deteriorates as K increases for both MetaBBO methods. In particular, the frozen variant with $K = \infty$ consistently performs worse than the original AdaELA. These results demonstrate that the effectiveness of AdaELA does not merely come from identifying a generally useful subset of ELA indicators. Instead, AdaELA learns to dynamically adjust the perceived optimization information according to the evolving search process, confirming the importance of stage-dependent adaptive optimization-state perception.

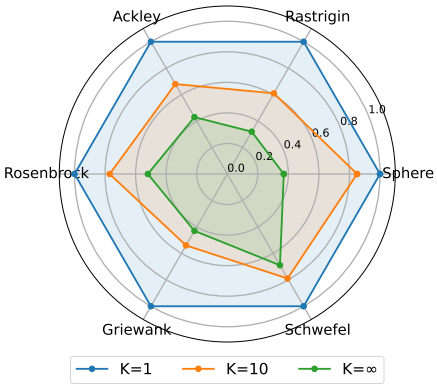

Figure 9: Effect of ELA Selection Update Interval for AutoEP.

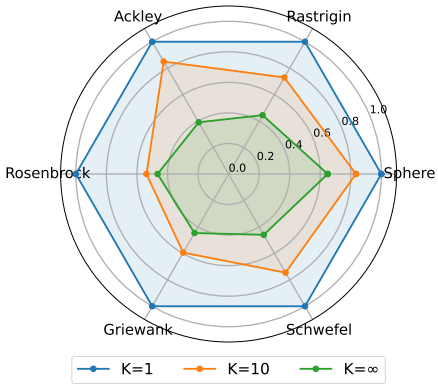

Figure 10: Effect of ELA Selection Update Interval for PAIR.

B.4 Computational Efficiency Analysis

To evaluate the computational efficiency of AdaELA, we measure the runtime under different population sizes and problem dimensions. Specifically, we consider multiple combinations of population size N and problem dimension D, and repeat each configuration 20 times to report the mean runtime and standard deviation. As shown in Table 2, we separately report the computational cost of calculating all indicators in the ELA library, the two-stage self-attention module, the cross-attention module,

and the complete AdaELA pipeline. The results show that the main computational overhead comes from calculating the ELA indicators, while the two-stage self-attention and cross-attention modules introduce only a small additional cost. Although the runtime increases with the population size and problem dimension, the overall computation remains lightweight, with the full pipeline requiring less than 0.45 seconds even under the largest evaluated setting. Therefore, AdaELA introduces only limited additional computational overhead during the optimization process. **Notably**, in our current implementation, the ELA indicators are computed sequentially. Since most indicators can be calculated independently, their computation can potentially be parallelized in practical implementations, which may further reduce the overall runtime.

Table 2: Computational efficiency of AdaELA under different population sizes and problem dimensions.

N	D	ELA library computation (ms)	Two-stage attention (ms)	Cross-attention (ms)	Full pipeline (ms)
50	5	28.178 ± 1.052	1.020 ± 0.202	0.121 ± 0.002	30.761 ± 0.982
50	10	116.314 ± 2.178	1.164 ± 0.209	0.121 ± 0.002	132.102 ± 12.353
50	15	281.529 ± 3.994	1.844 ± 0.214	0.130 ± 0.002	283.671 ± 3.254
100	5	35.486 ± 2.872	1.195 ± 0.050	0.121 ± 0.004	40.019 ± 5.323
100	10	139.764 ± 1.964	1.962 ± 0.171	0.121 ± 0.002	141.705 ± 2.101
100	15	321.481 ± 5.167	2.127 ± 0.176	0.128 ± 0.008	322.419 ± 6.406
200	5	52.995 ± 0.878	2.065 ± 0.155	0.140 ± 0.017	56.767 ± 0.724
200	10	186.922 ± 2.215	2.562 ± 0.196	0.126 ± 0.014	190.300 ± 2.485
200	15	413.536 ± 3.227	3.344 ± 0.268	0.132 ± 0.005	420.115 ± 4.194

B.5 Analysis of LLM Capability Influence

The experiments for RQ3 demonstrate that, under Qwen3-30B, the performance improvement of LLM-based MetaBBO mainly comes from adaptive ELA selection rather than simply providing richer landscape information. However, this observation may potentially be affected by the capability of the underlying LLM. Therefore, we further conduct experiments using GLM-5.3 and DeepSeek-V4-Pro through API calls to examine whether the conclusion remains consistent under different LLM configurations.

As shown in Fig. 11, we compare AdaELA with the All ELA variant on BBOB-NOISY and Human-Powered Aircraft Design using different LLMs. For each LLM configuration, the original design of each LLM-based MetaBBO method is used as the reference baseline, and the relative performance of AdaELA and All ELA is evaluated accordingly. The results show that AdaELA consistently maintains its advantage over All ELA across different LLM configurations, indicating that the conclusion of RQ3 is not dependent on a specific LLM capability. These results further support that the performance gain mainly originates from adaptive perception of informative landscape information rather than simply increasing the amount of ELA information provided to the LLM.

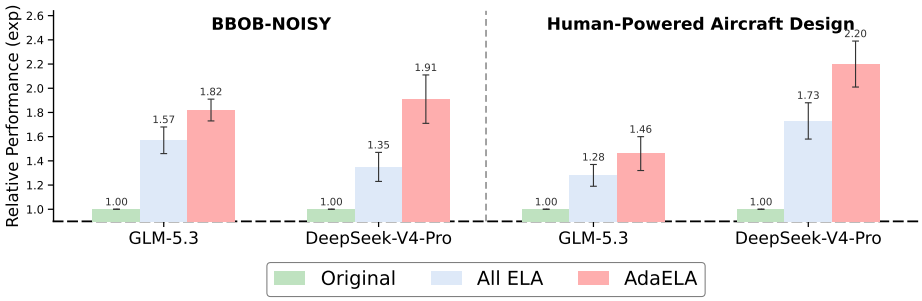

Figure 11: Comparison of AdaELA and All ELA under different LLM configurations.

B.6 Why Neural Landscape Representation Methods are Not Included as Baselines?

Neural landscape representation methods, such as NeurELA Ma et al. (2025a) and DeepELA Seiler et al. (2025), are closely related to landscape-based optimization-state representation. However, we do not include them as direct baselines for AdaELA because they differ from our method in both research objective and representation form. To clarify this choice, we summarize the main reasons below.

1. Different research problems. NeurELA and DeepELA mainly focus on learning new landscape representations, whereas AdaELA studies a different question: given existing ELA information, what information should be provided to the LLM at different optimization stages? Therefore, the two directions do not address exactly the same problem.

2. Different interpretability of information. NeurELA and DeepELA output latent representations that are difficult to interpret, whereas AdaELA provides explicit ELA indicators with clear numerical and semantic meanings. Since LLMs rely on understandable textual semantics for reasoning, they cannot directly interpret such latent vectors, making them difficult to use directly in the current LLM-based MetaBBO framework.

3. Complementary rather than conflicting. AdaELA and neural landscape representation methods are not conflicting directions. In the future, it may be possible to combine neural landscape representations with LLM-based MetaBBO and further introduce adaptive information perception. We view this as an interesting direction for future work.

C Prospects of Adaptive ELA for RL-based MetaBBO

Our central viewpoint is that the effectiveness of optimization information varies across different problems and optimization stages. Accordingly, adaptive optimization-state perception should not be limited to LLM-based MetaBBO, but may also benefit other types of meta-policies that rely on optimization-state representations. To further examine whether this principle extends beyond LLM-based meta-policies, we conduct an additional proof-of-concept experiment on RL-based MetaBBO using three representative methods, including LDE Sun et al. (2021), DEDDQN Sharma et al. (2019), and RL-DAS Guo et al. (2024a). Specifically, BBOB is used as the meta-training problem set, while three representative single-objective optimization problems outside the BBOB suite, including Ackley, Griewank, and Levy, are selected as unseen test problems.

At each test-time optimization step t, AdaELA selects an ELA indicator set $\mathcal{Z}^t \subseteq \mathcal{L}$ according to the current optimization state. Since $\mathcal{Z}^t$ may vary across different test steps, each distinct ELA combination corresponds to an RL meta-policy $\pi_{\mathcal{Z}^t}$, which is trained on BBOB using $\mathcal{Z}^t$ as the optimization-state representation and subsequently applied to the unseen test problem. We consider three variants for comparison: the original RL-based MetaBBO method, All ELA, where all ELA indicators are used as a fixed optimization-state representation and only one RL meta-policy needs to be trained, and AdaELA, where different ELA combinations selected by AdaELA require the corresponding RL meta-policies to be trained. Notably, AdaELA does not require training policies for all 2^{25} possible ELA combinations. A policy is trained only when the corresponding ELA combination is actually selected during optimization, while combinations that are never encountered require no additional training.

As shown in Fig. 12–14, we report the relative performance of different variants over population iterations on the three unseen test problems. For each RL-based MetaBBO method, its original design is used as the baseline with a relative performance of 1.0. The horizontal axis denotes the population iteration, while the vertical axis represents the relative performance. AdaELA achieves the best overall performance on the unseen problems, further supporting our viewpoint that adaptive optimization-state perception can also benefit RL-based MetaBBO. Although this implementation is effective, it is not computationally efficient because different ELA combinations require the training of multiple RL meta-policies. Nevertheless, these results provide useful insights for future RL-based MetaBBO research, where more flexible frameworks, such as hierarchical reinforcement learning Sutton et al. (1999); Bacon et al. (2017); Pateria et al. (2021), may be considered to realize adaptive optimization-state perception more efficiently.

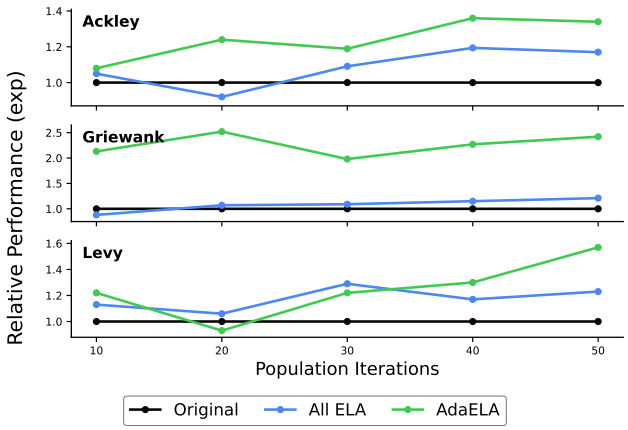

Figure 12: Relative performance of different optimization-state perception strategies for **LDE**.

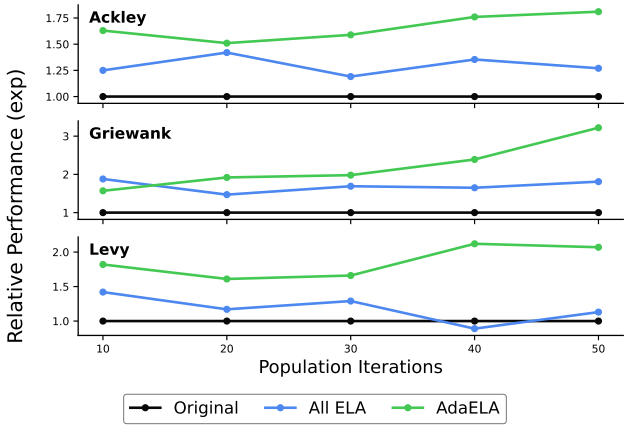

Figure 13: Relative performance of different optimization-state perception strategies for **DEDDQN**.

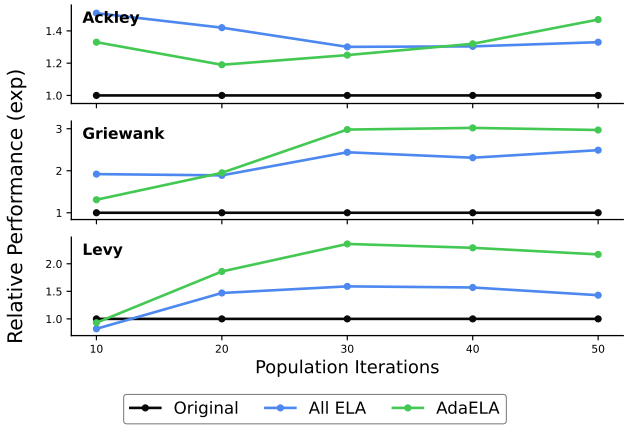

Figure 14: Relative performance of different optimization-state perception strategies for **RL-DAS**.