B2Opt: Learning to Optimize Black-box Optimization with Little Budget

Xiaobin Li[1], **Kai Wu**[1*], **Xiaoyu Zhang**[2], **Handing Wang**[1]

[1]School of Artificial Intelligence, Xidian University
[2]School of Cyber Engineering, Xidian University
22171214784@stu.xidian.edu.cn, kwu@xidian.edu.cn, xiaoyuzhang@xidian.edu.cn, hdwang@xidian.edu.cn

Abstract

The core challenge of high-dimensional and expensive black-box optimization (BBO) is how to obtain better performance faster with little function evaluation cost. The essence of the problem is how to design an efficient optimization strategy tailored to the target task. This paper designs a powerful optimization framework to automatically learn the optimization strategies from the target or cheap surrogate task without human intervention. However, current methods are weak for this due to poor representation of optimization strategy. To achieve this, 1) drawing on the mechanism of genetic algorithm, we propose a deep neural network framework called B2Opt, which has a stronger representation of optimization strategies based on survival of the fittest; 2) B2Opt can utilize the cheap surrogate functions of the target task to guide the design of the efficient optimization strategies. Compared to the state-of-the-art BBO baselines, B2Opt can achieve multiple orders of magnitude performance improvement with less function evaluation cost.

1 Introduction

Many tasks, such as neural architecture search (Elsken, Metzen, and Hutter 2019) and hyperparameter optimization (Hutter, Kotthoff, and Vanschoren 2019; Golovin et al. 2017), can be abstracted as black-box optimization (BBO), which means that although we can evaluate $f(x)$ for any $x \in X$, we have no access to any other information about f, such as the Hessian and gradients. Moreover, it is expensive to evaluate $f(x)$ in most cases. Various hand-designed algorithms, such as genetic algorithms (GAs) (Khadka and Tumer 2018), Bayesian optimization(Mutny and Krause 2018; Eriksson et al. 2019), and evolution strategy (ES) (Wierstra et al. 2014; Salimans et al. 2017), have been designed to solve BBO. Their purpose is to design optimization strategies that allow them to continuously sample better solutions.

Recently, the learning to optimize (L2O) framework (Andrychowicz et al. 2016; Chen et al. 2022) gives a new insight into optimization. They leverage the recurrent neural network (Chen et al. 2020, 2017; Li and Malik 2016; Wichrowska et al. 2017; Bello et al. 2017), multilayer perceptron (Metz et al. 2019), or Transformer (Gärtner et al. 2023) as the optimizer to develop optimization methods, aiming at reducing the laborious iterations of hand engineering (Vicol, Metz, and Sohl-Dickstein 2021; Flennerhag et al. 2022; Li and Malik 2016). The core of L2O is constructing a solid mapping from the initial solutions to the optimal solution. Moreover, several efforts (Cao et al. 2019; Chen et al. 2017; Lange et al. 2022) have coped with the black-box problems, their effectiveness may be hindered by the limited representation capabilities of optimization strategy and a large number of evaluations on expensive black-box functions. As such, they often deal with low-dimensional problems.

The GA uses a static sequence of operators for black-box optimization, which can result in inefficient sampling. Drawing from a recent study (Zhang et al. 2021) that compares the transformer architecture to EAs, noting parallels between genetic operators and transformer components, we've adapted the transformer's core module. This insight led to the creation of B2Opt, a parameterized optimization strategy representation.

We design three modules for mapping random solutions to optimal solutions. The Self-Attention (SA)-based Crossover module (SAC) generates potential individuals. Its output feeds into the Feed-Forward Network (FFN)-based Mutation module (FM) to avoid local optima. The Residual and Selection module (RSSM) selects the fittest individuals. These modules form a B2Opt Block (OB). By stacking OBs, we simulate GA iteration rules to represent robust optimization strategies.

Adapting B2Opt strategy for target tasks involves updating its parameters using task info. To minimize costly function evaluations, we create a cheap surrogate function set for B2Opt training. This set, resembling target BBO problems, includes initial populations and designed surrogates. Training uses gradient-based methods like SGD and Adam, eliminating the need for expensive function queries.

B2Opt undergoes rigorous testing on six standard functions, BBOB challenges (Finck et al. 2010) in high dimensions, neural network training, and the planar mechanical arm problem (Wang et al. 2022). The experimental results position B2Opt at the forefront, showcasing its superior performance compared to five leading EA baselines,

*Corresponding author

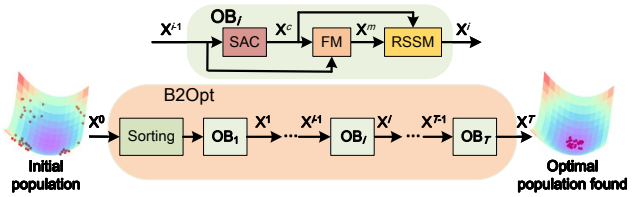


Figure 1: Overall architecture of B2Opt and OB. B2Opt comprises a stack of T OBs. These units can be configured to share or not share weights among themselves. In operation, the initial random population $\mathbf{X}^0$ is fed into B2Opt, which then generates the optimal population $\mathbf{X}^T$.

three state-of-the-art (SOTA) Bayesian optimization baselines, and three top-performing L2O methods. The highlights of this paper are shown as follows:

- Compared with the SOTA BBO methods, B2Opt achieves multiple orders of magnitude performance improvement with fewer function evaluations even if the training dataset is a low-fidelity set of the target black-box function.
- We design the B2Opt framework to realize the automated GA, which can better represent optimization strategies. It is easier to map random solutions to optimal solutions with fewer evaluation times during optimization.
- We design a new training strategy to reduce the evaluation cost of expensive objective functions during training. It uses cheap surrogate functions for expensive target tasks instead of directly evaluating the target black-box function.

2 Related Work

2.1 Population-based BBO Algorithm

Our work is closely related to the population-based BBO algorithm (Ros and Hansen 2008; Sampson 1976; Kennedy and Eberhart 1995; Gong et al. 2015; Storn and Price 1997; Stanovov, Akhmedova, and Semenkin 2022). However, with limited knowledge of the target problem, the algorithms mentioned above rely heavily on carefully hand-crafted designs, which is time-consuming and unreliable. Ultimately, they are unable to automatically modify their optimization strategies according to the target task, resulting in low-performance degradation and extreme inflexibility.

2.2 Meta-Learn BBO Algorithm

Meta-learn (or learn-to-learn) is also an important research direction of AutoML (Hutter, Kotthoff, and Vanschoren 2019). Here, we mainly focus on a learnable optimization strategy for BBO (Ma et al. 2024), divided into two parts.

Meta-Learn Whole BBO Algorithm Our proposal belongs to this type. These methods parametrize the optimization algorithm by an neural network processing the raw solution vectors and their associated fitness (Chen et al. 2017; TV et al. 2020; Cao et al. 2019; Gomes, Léger, and Gagné 2021; Wu et al. 2023; Li et al. 2024).

Meta-Learn Part BBO Algorithm This type only learns some parts of the algorithm, not the overall algorithm (Müller et al. 2023; Shala et al. 2020; Lange et al. 2022, 2023; Salimans et al. 2017; Guo et al. 2024).

The limitations of these meta-learn methods are shown as follows: 1) to train the meta-optimizer, a large number of expensive black-box functions need to be requested, which is very unrealistic; 2) the established loss function for training the meta-optimizer is challenging to optimize, resulting in a poor representation of the optimization strategy. Thus, we propose B2Opt overcome the above limitations.

2.3 LLM for Optimization

Various optimization approaches leveraging Large Language Models (LLMs) have emerged to address diverse BBO problem domains (Romera-Paredes et al. 2023; Meyerson et al. 2023; Liu et al. 2023; Yang et al. 2023; Lehman et al. 2023; Ma et al. 2023; Chen, Dohan, and So 2023; Nasir et al. 2023; Mo et al. 2024). These methods lack the universal applicability as pretrained BBO models due to a deficiency in generating capabilities across tasks. A more detailed introduction to these related works is presented in the Appendix A.

3 Framework of B2Opt

We introduce the background knowledge related to genetic algorithms (GA) and transformers in Appendix B.

3.1 Problem Definition

A black-box optimization problem can be transformed as a minimization problem, as shown in Eq. (1), and constraints may exist for corresponding solutions:

$$\min_{\mathbf{x}} \ f(\mathbf{x}), s.t.\ x_i \in [l_i, u_i] \tag{1}$$

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 $\mathbf{x}$. Suppose n individuals of one population be $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^T$, then B2Opt are required to find the population near the optimal solution. Note that we only have a very small number of function evaluations to achieve.

3.2 Self-Attention Crossover Module

Inspired by the crossover operator in GAs, we propose an SA-based module to generate potential solutions by maximizing information interaction among individuals in a population. Suppose a population $\mathbf{X}$ is arranged in a non-descending order of fitness, and $\mathbf{F} \in \mathbb{R}^{n\times 1}$ be the fitness matrix of $\mathbf{X}$. Then, this module can be represented as follows:

$$\mathbf{X}^c = SAC(\mathbf{X}, \mathbf{F}) \tag{2}$$

where $\mathbf{X}^c = [\mathbf{x}_1^c, \mathbf{x}_2^c, \cdots, \mathbf{x}_n^c]^T$ is the output population of the SAC module.

Since the order of individuals does not affect the population distribution, SA does not require position coding. Standard SA projects the input sequence $\mathbf{X}$ into a d-dimensional

space via the queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$). We remove these three mappings for enhanced transferability and $\mathbf{X}^c = \mathbf{AX}$. $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a self-attention matrix that can be learned to maximize inter-individual information interaction based on individual ranking information. Since $\mathbf{A}$ uses individual ranking information instead of fitness value information, $\mathbf{A}$ is invariant to the problems, which is beneficial to improve the generalization performance of the model. This is why the population needs to be sorted in non-descending order. We further introduce fitness information to assist in designing the SAC module:

$$\mathbf{A}^F = SA(\mathbf{F}) = Softmax\left(\mathbf{F}\mathbf{W}^Q(\mathbf{F}\mathbf{W}^K)^T/\sqrt{d_k}\right) \tag{3}$$

Thus, $\mathbf{X}^c = \mathbf{AX} + \mathbf{A}^F\mathbf{X}$. To better balance the roles of $\mathbf{A}$ and $\mathbf{A}^F$, we introduce two learnable weights $\mathbf{W}_1^c \in \mathbb{R}^{n\times 1}$ and $\mathbf{W}_2^c \in \mathbb{R}^{n\times 1}$. Therefore, the final *SAC* module is shown as follows:

$$\mathbf{X}^c = tile(\mathbf{W}_1^c) \odot (\mathbf{AX}) + tile(\mathbf{W}_2^c) \odot (\mathbf{A}^F\mathbf{X}) \tag{4}$$

where $\odot$ represents Hadamard product and the *tile* copy function extends the vector to a matrix.

3.3 FFN-based Mutation Module

In Transformer, each patch embedding carries on directional feature transformation through the FFN module. We take one linear layer as an example: $\mathbf{X} = \mathbf{X}\mathbf{W}^F$, where $\mathbf{W}^F$ is the weight of the linear layer, and it is applied to each embedding separately and identically. The similar formula format of mutation operator and FFN inspires us to design a learnable mutation module FM based on FFN with $ReLU$ activation function:

$$\mathbf{X}^m = FM(\mathbf{X}^c) = (ReLU(\mathbf{X}^c\mathbf{W}_1^F + \mathbf{b}_1))\mathbf{W}_2^F + \mathbf{b}_2 \tag{5}$$

where $\mathbf{X}^m$ is the population after the mutation of $\mathbf{X}^c$. $\mathbf{W}_2^F$ and $\mathbf{W}_1^F$ represent the weight of the second layer of FFN and the weight of the first layer of FFN, respectively. $\mathbf{b}_2$ and $\mathbf{b}_1$ represent the bias of the second layer and the first layer of FFN, respectively. FM is designed to escape the local optimum.

3.4 Selection Module

The residual connection in the transformer can be analogized to the selection operation in GA (Zhang et al. 2021). The RSSM generates the offspring population according to the following equation:

$$\begin{aligned}\hat{\mathbf{X}} &= RSSM(\mathbf{X}, \mathbf{X}^c, \mathbf{X}^m) \\ &= Sort(SM(\mathbf{X}, tile(\mathbf{W}_1^s) \odot \mathbf{X} \\ &+ tile(\mathbf{W}_2^s) \odot \mathbf{X}^c + tile(\mathbf{W}_3^s) \odot \mathbf{X}^m))\end{aligned} \tag{6}$$

where $\hat{\mathbf{X}}$ is the fittest population for the next generation; the learnable weights $\mathbf{W}_1^s \in \mathbb{R}^{n\times 1}$, $\mathbf{W}_2^s \in \mathbb{R}^{n\times 1}$, and $\mathbf{W}_3^s \in \mathbb{R}^{n\times 1}$ are the weights for $\mathbf{X}$, $\mathbf{X}^c$, and $\mathbf{X}^m$, respectively. $Sort(\mathbf{X})$ represents that $\mathbf{X}$ is sorted in non-descending order of fitness. We use quicksort to sort the population based on function fitness. These three learnable weight matrices realize the weighted summation of residual connections, thereby simulating a learnable selection strategy. Meanwhile, introducing residual structure enhances the model's representation ability, enabling B2Opt to form a deep architecture.

ID	Functions	Range
TF_1	$\sum_i \lvert w_i sin(z_i)\rvert$	$x_i \in [-10, 10]$ $b_i \in [-10, 10]$
TF_2	$\sum_i \lvert z_i\rvert$	$x_i \in [-10, 10]$ $b_i \in [-10, 10]$
TF_3	$\sum_i \lvert(z_i) + (z_{i+1})\rvert$ $+\sum_i \lvert z_i\rvert$	$x_i \in [-10, 10]$ $b_i \in [-10, 10]$
TF_4 (Sphere)	$\sum_i z_i^2$	$x_i \in [-100, 100]$ $b_i \in [-50, 50]$
TF_5	$\{\lvert z_i\rvert, 1 \le i \le D\}$	$x_i \in [-100, 100]$ $b_i \in [-50, 50]$
TF_6 (Rosenbrock)	$\sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 +$ $(z_i - 1)^2)$	$x_i \in [-100, 100]$ $b_i \in [-50, 50]$
TF_7 (Rastrigin)	$\sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i))$ $+10D$	$x_i \in [-5, 5]$ $b_i \in [-2.5, 2.5]$
TF_8 (Griewank)	$\sum_{i=1}^{D}\frac{z_i^2}{4000} -$ $\prod_{i=1}^{D}\cos(\frac{z_i}{\sqrt{i}}) + 1$	$x_i \in [-600, 600]$ $b_i \in [-300, 300]$
TF_9 (Ackley)	$-20\exp(-\frac{\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}}{5})$ $-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i))$ $+20 + \exp(1)$	$x_i \in [-32, 32]$ $b_i \in [-16, 16]$

Table 1: Training and testing functions. TF_1-TF_3 are training functions and TF_4-TF_9 are testing functions. Here, $d = \{10, 100\}$. $z_i = x_i - b_i$.

SM updates individuals based on a pairwise comparison between the offspring and input population regarding their fitness. Suppose that $\mathbf{X}$ and $\mathbf{X}^{'}$ are the input populations of SM. We compare the quality of individuals from $\mathbf{X}$ and $\mathbf{X}^{'}$ pairwise based on fitness. A binary mask matrix indicating the selected individual can be obtained based on the indicator function $l_{x>0}(x)$, where $l_{x>0}(x) = 1$ if $x > 0$ and $l_{x>0}(x) = 0$ if $x < 0$. SM forms a new population $\hat{\mathbf{X}}$ by employing Eq. (7).

$$\begin{aligned}\hat{\mathbf{X}} &= tile(l_{x>0}(\mathbf{M}_{F'} - \mathbf{M}_F)) \odot \mathbf{X} \\ &+ tile(1 - l_{x>0}(\mathbf{M}_{F'} - \mathbf{M}_F)) \odot \mathbf{X}^{'}\end{aligned} \tag{7}$$

where the *tile* copy function extends the indication vector to a matrix, $\mathbf{M}_F(\mathbf{M}_{F'})$ denotes the fitness matrix of $\mathbf{X}(\mathbf{X}^{'})$.

3.5 Structure of B2Opt

B2Opt comprises basic t B2Opt blocks (OBs), and parameters can be shared among these t OBs or not. The over-

all architecture of B2Opt and OB is shown in Figure 1. Each OB consists of SAC, FM, and RSSM in sequential order. $\mathbf{X}^0 \in \mathbb{R}^{n\times d}$ represents the initial population input into B2Opt, which must be sorted in non-descending fitness order. In Eq. (8), $\mathbf{X}^{i-1}$ is fed into OB_t to get $\mathbf{X}^i$, where $i \in [1, t]$. B2Opt realizes the mapping from the random initial population to the target population by stacking t OBs.

$$\mathbf{X}^i = OB(\mathbf{X}^{i-1}); \quad \mathbf{X}^c = SAC(\mathbf{X}^{i-1}, \mathbf{F}); \qquad (8)$$
$$\mathbf{X}^m = FM(\mathbf{X}^c); \quad \mathbf{X}^i = RSSM(\mathbf{X}^{i-1}, \mathbf{X}^c, \mathbf{X}^m)$$

3.6 Training of B2Opt

Goal We introduce the parameters θ of B2Opt, which need to be optimized. Here, we set $\theta = \{\mathbf{W}_1^c, \mathbf{W}_2^c, \mathbf{A}, \mathbf{W}_1^F, \mathbf{W}_2^F, \mathbf{b}_1, \mathbf{b}_2, \mathbf{W}_1^s, \mathbf{W}_2^s, \mathbf{W}_3^s\}$.

Surrogate Function Given a target problem $f(\mathbf{x} \mid \omega)$, we define its surrogate function as $\hat{f}(\mathbf{x} \mid \hat{\omega})$. $\mathbf{x}$ is the input of function, ω and $\hat{\omega}$ are the parameters of the target problem and the surrogate function respectively. Given tn target problems $\{f_i(\mathbf{x} \mid \omega_i) \mid i \in [1, tn]\}$, we can construct a corresponding set of surrogate task set composed of functions $\{\hat{f}_i(\mathbf{x} \mid \hat{\omega}_{i,j}) \mid i \in [1, sn], j \in [1, sn_\omega^i]\}$, where sn represents the number of surrogate functions, sn_ω^i represents the number of different arguments samples of surrogate function $\hat{f}_i$. In this paper, we build all surrogate functions manually. We do not limit the way to obtain the surrogate function, it can be manually designed or learned.

Training Dataset Training data is a crucial factor beyond the objective functions. Unless otherwise specified, we choose $TF1-TF3$ as the training function, see Table. 1 for details. This paper establishes the training set by constructing a set of surrogate functions related to the optimization objective. This training dataset only contains $(\mathbf{X}^0, \hat{f}_i(\mathbf{x} \mid \hat{\omega}))$, the initial population, and the surrogate objective function, respectively. The variance of $\hat{\omega}$ causes the shift in landscapes. The training dataset is designed as follows: 1) Randomly initialize the input population $\mathbf{X}^0$; 2) Randomly sample a surrogate objective function $\hat{f}_i(\mathbf{x} \mid \hat{\omega}_{i,j})$ by adjusting the parameter $\hat{\omega}$; 3) Evaluate $\mathbf{X}^0$ by $\hat{f}_i(\mathbf{x} \mid \hat{\omega}_{i,j})$; 4) Repeat Steps 1)-3) to generate the corresponding dataset. We show the training and testing function set designed as follows:

$$F^{train} = \{\hat{f}_i(\mathbf{x} \mid \hat{\omega}_{i,j}) \mid i \in [1, sn], j \in [1, sn_\omega^i]\} \qquad (9)$$
$$F^{test} = \{f_i(\mathbf{x} \mid \omega_i) \mid i \in [1, tn]\} \qquad (10)$$

Loss Function

$$l_i = \frac{\frac{1}{|\mathbf{X}^0|}\sum_{\mathbf{x}\in\mathbf{X}^0} f_i(\mathbf{x}|\omega) - \frac{1}{|E_\theta(\mathbf{X}^0)|}\sum_{\mathbf{x}\in E_\theta(\mathbf{X}^0)} f_i(\mathbf{x}|\omega)}{\left|\frac{1}{|\mathbf{X}^0|}\sum_{\mathbf{x}\in\mathbf{X}^0} f_i(\mathbf{x}|\omega)\right|} \qquad (11)$$

B2Opt attempts to search for individuals with high quality based on the available information. The loss function tells how to obtain the parameters of B2Opt to generate individuals closer to the optimal solution by maximizing the difference between the initial population and the output population of B2Opt. The following loss function $l_i(\mathbf{X}^0, f(\mathbf{x} \mid \omega), \theta)$ is employed (see Eq. 11), where θ denotes parameters of B2Opt (E). Equation (11) calculates the average fitness difference between input and output, further normalized within $[0, 1]$. To encourage B2Opt to explore the fitness landscape, for example, the constructed Bayesian posterior distribution over the global optimum can be added to Eq. (11). We have three ways to train B2Opt:

1. When we can construct a differentiable surrogate function of the objective function, we use the gradient information of Eq. (11) to train B2Opt.
2. When it is difficult to construct differentiable surrogate functions for the objective function, we can use REINFORCE (Williams 1992) to estimate these derivatives.
3. We can use ES (Vicol, Metz, and Sohl-Dickstein 2021) to train B2Opt with black-box training functions. Here, we focus on introducing B2Opt training through the first method.

Training B2Opt We then train B2Opt in a supervised mode. Since the gradient is unnecessary during the test process, B2Opt can solve BBO problems. To prepare B2Opt to learn a balanced performance upon different optimization problems, we design a loss function formulated as follows:

$$\arg\min_\theta l_\Omega(\Omega, \theta, f_i(\mathbf{x}|\omega)) = \frac{\sum_{\mathbf{X}^0\in\Omega} l_i(\mathbf{X}^0, f_i(\mathbf{x}|\omega_i^{train}), \theta)}{-K} \qquad (12)$$

We employ Adam method with a minibatch Ω to train B2Opt upon training dataset.

Detailed Training Process The goal of the training algorithm is to search for parameters θ^* of B2Opt. Before training starts, B2Opt is randomly initialized to obtain the initial parameters θ. The algorithm will then perform the following three steps in a loop until the training termination condition is satisfied:

1. **Step 1**, randomly initialize a minibatch Ω comprised of K populations $\mathbf{X}^0$;
2. **Step 2**, for each $f_i \in F^{train}$, given training data $(\mathbf{X}^0, f_i)$, update θ by minimizing the l_Ω (see lines 9 to 12 of Algorithm 1);
3. **Step 3**, given $\mathbf{X}^0$, update θ by minimizing $-1/m\sum_i l_\Omega$, where m is the number of functions in F^{train} (see lines 14 to 23 of Algorithm 1).

After completing the training process, the algorithm will output θ^*. The pseudocode of the training phase is described in Appendix D Algorithm 1.

3.7 Driving B2Opt to Solve New Problems

The trained B2Opt does not require the gradient information of the target problem and can be directly used to solve the BBO problem. In the testing phase, the goal of B2Opt is to map a random population $\mathbf{X}^0$ to a population $\mathbf{X}^T$, which is closer to the global optimum. A detailed introduction is given in Appendix D Algorithm 2.

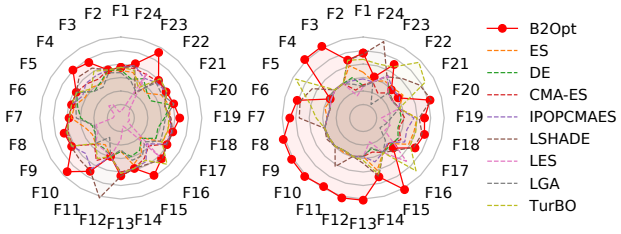


Figure 2: B2Opt's performance on BBOB. The problem dimensions of (left) and (right) are 10 and 100, respectively. This metric is the optimal value found on the function. To present the results more clearly, we normalize the experimental results. The higher the value here, the better the performance.

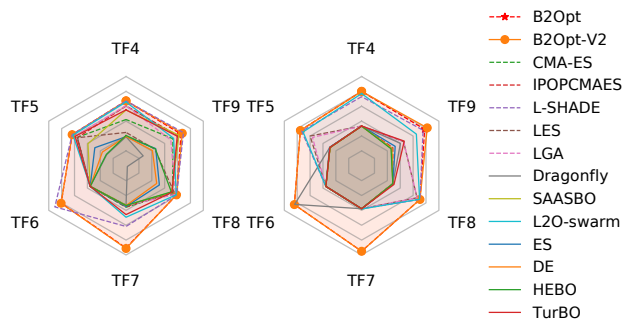


Figure 3: The compared results of all methods on six functions. The structure of B2Opt is *30 OBs with WS*. The left and right schematic diagrams are the test results when the dimensions are 10 and 100, respectively. The metric used for evaluation is the optimal function value found by the different methods. To plot this, we normalize the results. A larger value indicates better performance.

4 Experiments

4.1 Experimental Setup

Datasets This paper first employs 6 commonly used functions (TF_4-TF_9) to show the effectiveness of the proposed B2Opt. Then, we evaluate the performance of the proposed scheme on the planner mechanic arm problem (Cully et al. 2015; Mouret and Maguire 2020), a robotic control problem that has been widely used to evaluate the performance of BBO algorithms. We further analyze the performance of B2Opt in the field of neuroevolution. We evaluate the performance of training a convolutional neural network (Howard et al. 2017) using B2Opt on the MNIST classification task. Please see Appendix F for the detailed introduction of the tasks.

Baselines We design three B2Opt models, including *3 OBs with WS*, *5 OBs without WS*, and *30 OBs with WS*. *3 OBs with WS* represents that B2Opt has 3 OB modules, and these OBs share the weights of each other. Each OB consists of 1 SAC, 1 FM, and 1 RSSM. In general, B2Opt represents 30 OBs with WS. B2Opt is compared with the state-of-the-art BBO methods, including non-L2O and L2O-based. The reasons for choosing all these algorithms and the parameter settings are detailed in Appendix G.

1) Non-L2O: **EA baselines**. DE(DE/rand/1/bin) (Das and Suganthan 2010), ES((μ,λ)-ES), IPOP-CMA-ES (Auger and Hansen 2005), L-SHADE (Tanabe and Fukunaga 2014), and CMA-ES (Hansen 2016). **Bayesian optimization**. Dragonfly (Kandasamy et al. 2020), SAASBO (Eriksson and Jankowiak 2021) HEBO (Cowen-Rivers et al. 2022) and TurBO (Santoni et al. 2023) are employed as a reference.

2) L2O-based methods L2O-swarm (Cao et al. 2019), LES (Lange et al. 2022), LGA (Lange et al. 2023)

4.2 Results

Synthetic Functions Here, the structure of B2Opt is *30 OBs with WS*. The results on synthetic functions are provided in Fig. 3. We also plot the convergence curves of all the methods on TF_4-TF_9 with $d = \{10, 100\}$, as shown in Appendix E.1. B2Opt converges quickly and can obtain better solutions with little budget.

Discussion We delve into the underlying reasons behind B2Opt's remarkable efficacy (See Appendix I.1 for details).

From the perspective of landscape features, TF_1-TF_3 include the following features: unimodal, multimodal, separable, and non-separable. The landscape features included in TF_4-TF_9 are as follows: 1) TF_4: Unimodal, Separable; 2) TF_5: Unimodal, Separable; 3) TF_6: Multimodal, Non-separable, Having a very narrow valley from local optimum to global optimum, Ill-conditioned; 4) TF_7: Multimodal, Separable, Asymmetrical, Local optima's number is huge; 5) TF_8: Multi-modal, Non-separable, Rotated; 6) TF_9: Multi-modal, Non-separable, Asymmetrical

The landscape features of TF_4 and TF_5 can be found in TF_1-TF_3. TF_6-TF_9 all have new features. The interference strength of different characteristics with the landscape is arranged as follows: **Having a very narrow valley from local optimum to global optimum>Asymmetrical, Local optima's number is huge>Asymmetrical>Rotated**. Therefore, B2Opt has the best generalization performance on TF_4, TF_5, TF_8, the second-best generalization performance on TF_7 and TF_9, and the worst on TF_6.

Results on BBOB We explore the optimization capabilities and generalization performance of B2Opt on BBOB (Finck et al. 2010). The experimental results of B2Opt on BBOB are shown in Fig. 2. The convergence curves are shown in the Appendix E.3. We train B2Opt on the training function shown in Table 1. The B2Opt architecture we adopt contains 30 OBs and shares weights. We compare B2Opt with population-based baselines renowned for their performance in synthetic TF_4-TF_9 functions. Notably, B2Opt, trained on just three simple functions, attains superior performance in BBOB. Across 10 and 100 dimensions, B2Opt leads in 20 and 19 out of 24 functions, respectively. B2Opt achieves great results at little cost and demonstrates generalization capabilities far beyond its training distribution.

	r	DE	ES	CMA-ES	L2O	LSHE	IPOP	LGA	LES	TurBO	B2Opt	*Untrained*
SC	100	1.60(0.1)	10.7(0.4)	1.25(0.1)	40.4(3.89)	0.66(0.1)	1.17(0.1)	178(1.7)	0.48(0.1)	0.42(0.1)	**0.19(0.1)**	0.8(0.1)
	300	2.65(0.1)	35(2.4)	1.84(0.1)	69.5(3.8)	0.54(0.1)	1.21(0.1)	219(1.6)	0.59(0.1)	0.56(0.1)	**0.28(0.1)**	4.26(0.3)
	1000	155(1.5)	153(2.2)	154(2.3)	176(7.2)	**0.4(0.1)**	36.8(1.0)	446(1.4)	37.8(1.9)	96.8(0.2)	15.1(0.3)	126(0.9)
CC	100	1.17(0.1)	9.22(0.5)	0.78(0.1)	31.9(1.8)	0.25(0.1)	0.72(0.1)	71.5(0.3)	0.29(0.1)	0.17(0.1)	**0.15(0.0)**	0.94(0.1)
	300	13.6(0.5)	38.3(0.6)	5.22(0.5)	89.1(2.0)	54.5(5.3)	0.92(0.0)	129(0.4)	0.35(0.1)	0.45(0.0)	**0.17(0.0)**	18.6(0.4)
	1000	278(2.0)	235(1.4)	239(0.8)	262(3.0)	218(0.9)	167(1.2)	394(1.1)	168(3.3)	229(0.5)	**17.9(0.1)**	214(0.9)
w/t/l		0/0/6	0/0/6	0/0/6	0/0/6	1/0/5	0/0/6	0/0/6	0/0/6	0/1/5	-	0/0/6

Table 2: The results of all baselines on the planar mechanical arm. The structure of B2Opt is *30 OBs with WS*. Simple case (SC): searching for different angles with fixed lengths. Complex case (CC): searching for different angles and lengths. w/t/l represents win/tie/loss to B2Opt. L2O, LSHE and IPOP stands for L2O-SWARM, LSHADE and I-POP-CMA-ES, respectively.

Datasize	B2Opt	ES	DE	CMA-ES	I-POP-CMA-ES	L-SHADE	LGA	LES	TurBO
0.25	**0.69(0.01)**	0.21(0.02)	0.21(0.01)	0.57(0.04)	0.33(0.02)	0.31(0.03)	0.53(0.02)	0.14(0.02)	0.30(0.07)
0.5	**0.80(0.01)**	0.21(0.01)	0.24(0.01)	0.61(0.03)	0.37(0.04)	0.29(0.01)	0.51(0.01)	0.25(0.03)	0.33(0.03)
0.75	**0.85(0.00)**	0.21(0.01)	0.22(0.02)	0.61(0.03)	0.45(0.02)	0.30(0.07)	0.52(0.05)	0.23(0.07)	0.44(0.02)
1	**0.86(0.00)**	0.25(0.02)	0.23(0.02)	0.67(0.01)	0.42(0.02)	0.30(0.01)	0.54(0.04)	0.28(0.01)	0.38(0.05)

Table 3: The classification accuracy of all methods on the MNIST dataset. Datasize represents the proportion of data sets involved in training.

f	*Untrained*	*5 OBs without WS*	*30 OBs with WS*	*3 OBs with WS*
TF_4	0.28(0.09)	0.08(0.03)	**1.2e-4(5e-5)**	8.16(3.44)
TF_5	0.37(0.05)	0.15(0.03)	**0.008(0.002)**	1.47(0.40)
TF_6	45.8(16.9)	15.43(2.34)	**8.93(0.03)**	1891(1396)
TF_7	1.08(0.72)	4.43(1.82)	**0.01(0.03)**	35.72(8.52)
TF_8	0.69(0.09)	0.06(0.03)	**1e-5(2e-5)**	0.82(0.10)
TF_9	0.85(0.20)	0.29(0.07)	**0.01(0.003)**	3.28(1.00)

Table 4: The performance of different B2Opt structures. The metric used to evaluate is the minimum function value found in the output population. The lower values in the table indicate superior algorithm performance.

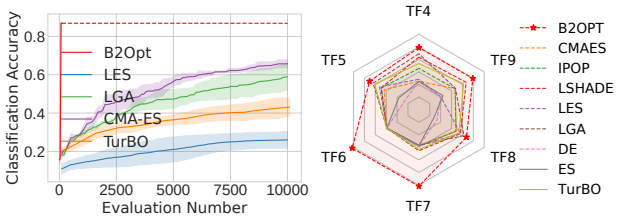
Planner Mechanic Arm The detailed experimental results are in Tables 2. Please see the Appendix E.2 for the convergence curve. Note that the parentheses in the table show the standard deviation if not otherwise specified. The structure of B2Opt is *30 OBs with WS*. *Untrained* represents the untrained B2Opt. We randomly selected 600 target points within the range of $r \leq 1000$ to form a set S, where r represents the distance from the target point to the origin of the mechanic arm. During the training process of B2Opt, a sample point set s is re-extracted from S for training every T training cycle. In the testing process, we extracted 128 target points (S^{test}) in the range of $r \leq 100$, $r \leq 300$, and $r \leq 1000$, respectively, for testing. The purpose of testing in three different regions is to further explore the generalization performance of B2Opt. B2Opt wins all SOTA methods and achieves the best results in comparison with other algorithms. B2Opt loses once to L-SHADE in SC with $r = 1000$. However, in the other 5 cases, B2Opt outperforms L-SHADE. In particular, on the complex problem (CC), B2Opt is more dominant and stable than L-SHADE. The optimization objective of this problem does not exist in the training sets of LGA and LES. A sharp performance degradation of LGA and LES can be observed. We also find the surprising phenomenon that *Untrained* outperforms most of the baselines, suggesting that the randomly initialized B2Opt also has some ability to produce and select potential solutions.

Neural Network Training The detailed results are shown in Table 3. The evaluation metric is the accuracy of the test set. When training B2Opt, the optimization objective of B2Opt is to minimize the cross-entropy loss, which is a surrogate function for metric accuracy. However, in the testing stage, the optimization goal of B2Opt and other baselines is to maximize the accuracy of the training set. We select 25%, 50%, 75%, and 100% data from the training set for training, respectively, which constitute surrogate problems with different levels of fidelity. Fig. 4(a) shows the convergence curve of B2Opt, LES, LGA, and CMA-ES in this task. B2Opt can achieve the best solution with the least number of evaluations.

4.3 Parameter Analysis

We analyze the effect of the deep structure, learning rate, and weight sharing between OBs on B2Opt.

B2Opt Architectures We consider the performance of different B2Opt architectures. The experimental results are shown in Table 4. They were sorted from good to worst based on their performance, and the result is *30 OBs with WS>5 OBs without WS>3 OBs with WS*. Deep architectures have better representation capabilities and also lead to better performance. *Untrained* represents that the parameters of *5 OBs without WS* are randomly initialized. *5 OBs without*



(a) (b)

Figure 4: (a). The convergence curves of B2Opt and baselines on the task of neural network training. The X-axis represents the number of evaluations, and the Y-axis represents the test accuracy. B2Opt stops iterating when the number of evaluations is 144, and the red dotted line is for convenient comparison. (b). Experimental results when $n = 25$, $d = 10$. The metric used for evaluation is the optimal function value found by the different methods. To plot this, we normalize the results. A larger value indicates better performance. IPOP stands for I-POP-CMA-ES.

WS outperforms *Untrained*, which demonstrates the effectiveness of the designed training process.

We also find an interesting phenomenon: *5 OBs without WS* outperforms *3 OBs with WS* in all cases. Our untrained deep architecture, *30 OBs with WS*, can achieve good results on simple planner mechanic arm problems, which shows that B2Opt retains the advantages of Transformer architecture and has strong generalization ability. We used untrained *5 OBs without WS* to test the complex planner mechanic arm problem, which performs poorly.

B2Opt yields improved outcomes with deeper architectures, but training deep B2Opt is challenging. ES for optimizing deep models, as explored in (Vicol, Metz, and Sohl-Dickstein 2021), represents a key future research direction.

Discussion. B2Opt comprises solution generation (crossover and mutation) and selection modules, enabling solution search even without training. In Table 2, the *Untrained* variant is outperformed by the trained B2Opt across all cases, although it surpasses several baselines. This is attributed to B2Opt's transformer-like structure, showcasing robust representation and search capabilities even without training. Table 4 demonstrates that *Untrained*, comprising 5 non-parameter-sharing OBs, outperforms *3 OBs with WS* but lags behind *5 OBs without WS* of the same depth. Deeper architectures, like *30 OBs with WS*, accentuate the performance gap of the untrained variant.

We further analyzed the impact of learning rate and population size on B2Opt performance, see Appendix J for details.

4.4 Ablation Study

This section considers the impact of different parts on B2Opt. We take B2Opt with 3 OBs and weight sharing as an example, which is trained on TF_1-TF_3 and tested on TF_4-TF_9. We remove SAC, FM, RSSM, and RC in B2Opt, respectively, and denote them as *Not SAC*, *Not FM*, *Not RSSM*, and *Not RC*. The experimental results are shown in Table 5. When their results are sorted from good to worst, the rank is B2Opt $>$ *Not FM* $>$ *Not RC* $\approx$ *Not SAC* $\approx$ *Not RSSM*. The role of FM is slightly weaker than that of the other three modules. Taken as a whole, the parts of SAC, RSSM, and RC are of equal importance. The absence of these core components can seriously affect the performance of B2Opt. At the same time, it also shows the effectiveness of the four proposed modules. The removal of any one of the modules in the crossover, mutation, and selection of EAs will degrade the performance of EAs. This shows that B2Opt implements a learnable EA framework that does not require human-designed parameters.

f	*Not SAC*	*Not FM*	*Not RC*	*Not RSSM*	B2Opt
TF_4	52.7(15.0)	23.3(6.68)	50.5(6.81)	271(346)	3.03(5.82)
TF_5	5.08(0.98)	2.94(0.41)	3.75(0.09)	3.51(1.49)	1.02(0.20)
TF_6	1e5(9e4)	1e4(1e4)	5e4(1e4)	7e6(1e8)	4e3(7e4)
TF_7	47.2(5.11)	19.7(4.27)	63.4(8.60)	40.6(8.10)	44.8(8.34)
TF_8	1.18(0.04)	1.19(0.07)	0.87(0.07)	3.28(1.14)	0.60(0.17)
TF_9	4.49(1.09)	3.42(0.78)	8.36(0.43)	3.56(0.72)	3.03(0.70)

Table 5: The results of ablation study. d = 10. The lower values in the table indicate superior algorithm performance.

We conducted additional visualization experiments to investigate B2Opt's internal optimization strategy (see Appendix K). B2Opt adjusts its strategy dynamically and adaptively for a balance between exploration and exploitation, aligning with our motivation.

5 Conclusions

The better performance than that of EA baselines, Bayesian optimization, and the L2O-based method demonstrates the effectiveness of B2Opt. Moreover, B2Opt can be well adapted to unseen BBO. Meanwhile, we experimentally demonstrate that the three proposed modules have positive effects. However, B2Opt still has room for improvement.

1) In the loss function, we do not effectively consider the diversity of the population, and the population can be regularized in the future;

2) The training set seriously affects the performance of B2Opt. If the similarity between the training set and the optimization objective is low, it will cause the performance of B2Opt to drastically degrade. Building the dataset as relevant to the target as possible is essential.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62206205 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

Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; Shillingford, B.; and De Freitas, N. 2016. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 3981–3989.

Auger, A.; and Hansen, N. 2005. A restart CMA evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, 1769–1776 Vol. 2.

Bello, I.; Zoph, B.; Vasudevan, V.; and Le, Q. V. 2017. Neural optimizer search with reinforcement learning. In *ICML*, 459–468. PMLR.

Cao, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2019. Learning to Optimize in Swarms. *NeurIPS*, 32: 15044–15054.

Chen, A.; Dohan, D. M.; and So, D. R. 2023. EvoPrompting: Language Models for Code-Level Neural Architecture Search. *arXiv preprint arXiv:2302.14838*.

Chen, T.; Chen, X.; Chen, W.; Heaton, H.; Liu, J.; Wang, Z.; and Yin, W. 2022. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research*, 23: 1–59.

Chen, T.; Zhang, W.; Jingyang, Z.; Chang, S.; Liu, S.; Amini, L.; and Wang, Z. 2020. Training Stronger Baselines for Learning to Optimize. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *NeurIPS*, volume 33, 7332–7343. Curran Associates, Inc.

Chen, Y.; Hoffman, M. W.; Colmenarejo, S. G.; Denil, M.; Lillicrap, T. P.; Botvinick, M.; and Freitas, N. 2017. Learning to learn without gradient descent by gradient descent. In *ICML*, 748–756. PMLR.

Cowen-Rivers, A. I.; Lyu, W.; Tutunov, R.; Wang, Z.; Grosnit, A.; Griffiths, R. R.; Maraval, A. M.; Jianye, H.; Wang, J.; Peters, J.; et al. 2022. Hebo: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74: 1269–1349.

Cully, A.; Clune, J.; Tarapore, D.; and Mouret, J.-B. 2015. Robots that can adapt like animals. *Nature*, 521: 503–507.

Das, S.; and Suganthan, P. N. 2010. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on Evolutionary Computation*, 15(1): 4–31.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1): 1997–2017.

Eriksson, D.; and Jankowiak, M. 2021. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In de Campos, C.; and Maathuis, M. H., eds., *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, 493–503. PMLR.

Eriksson, D.; Pearce, M.; Gardner, J.; Turner, R. D.; and Poloczek, M. 2019. Scalable global optimization via local Bayesian optimization. *NeurIPS*, 32.

Finck, S.; Hansen, N.; Ros, R.; and Auger, A. 2010. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical report, Citeseer.

Flennerhag, S.; Schroecker, Y.; Zahavy, T.; van Hasselt, H.; Silver, D.; and Singh, S. 2022. Bootstrapped Meta-Learning. In *ICLR*.

Gärtner, E.; Metz, L.; Andriluka, M.; Freeman, C. D.; and Sminchisescu, C. 2023. Transformer-Based Learned Optimization. In *CVPR*, 11970–11979.

Golovin, D.; Solnik, B.; Moitra, S.; Kochanski, G.; Karro, J.; and Sculley, D. 2017. Google vizier: A service for black-box optimization. In *KDD*, 1487–1495.

Gomes, H. S.; Léger, B.; and Gagné, C. 2021. Meta learning black-box population-based optimizers. *arXiv preprint arXiv:2103.03526*.

Gong, Y.-J.; Li, J.-J.; Zhou, Y.; Li, Y.; Chung, H. S.-H.; Shi, Y.-H.; and Zhang, J. 2015. Genetic learning particle swarm optimization. *IEEE Transactions on Cybernetics*, 46(10): 2277–2290.

Guo, H.; Ma, Z.; Chen, J.; Ma, Y.; Cao, Z.; Zhang, X.; and Gong, Y.-J. 2024. ConfigX: Modular Configuration for Evolutionary Algorithms via Multitask Reinforcement Learning. *arXiv preprint arXiv:2412.07507*.

Hansen, N. 2016. The CMA Evolution Strategy: A Tutorial. *ArXiv*, abs/1604.00772.

Howard, A.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv: Computer Vision and Pattern Recognition*.

Hutter, F.; Kotthoff, L.; and Vanschoren, J. 2019. *Automated machine learning: methods, systems, challenges*. Springer Nature.

Kandasamy, K.; Vysyaraju, K. R.; Neiswanger, W.; Paria, B.; Collins, C. R.; Schneider, J.; Poczos, B.; and Xing, E. P. 2020. Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly. *Journal of Machine Learning Research*, 21(81): 1–27.

Kennedy, J.; and Eberhart, R. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, 1942–1948. IEEE.

Khadka, S.; and Tumer, K. 2018. Evolution-Guided Policy Gradient in Reinforcement Learning. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *NeurIPS*, volume 31. Curran Associates, Inc.

Lange, R.; Schaul, T.; Chen, Y.; Lu, C.; Zahavy, T.; Dalibard, V.; and Flennerhag, S. 2023. Discovering Attention-Based Genetic Algorithms via Meta-Black-Box Optimization. In *GECCO*, GECCO '23, 929–937. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701191.

Lange, R. T.; Schaul, T.; Chen, Y.; Zahavy, T.; Dallibard, V.; Lu, C.; Singh, S.; and Flennerhag, S. 2022. Discovering Evolution Strategies via Meta-Black-Box Optimization. *arXiv preprint arXiv:2211.11260*.

Lehman, J.; Gordon, J.; Jain, S.; Ndousse, K.; Yeh, C.; and Stanley, K. O. 2023. Evolution through large models. In *Handbook of Evolutionary Machine Learning*, 331–366. Springer.

Li, K.; and Malik, J. 2016. Learning to optimize. *arXiv preprint arXiv:1606.01885*.

Li, X.; Wu, K.; Li, Y. B.; Zhang, X.; Wang, H.; and Liu, J. 2024. Pretrained Optimization Model for Zero-Shot Black Box Optimization. In *NeurIPS*.

Liu, F.; Tong, X.; Yuan, M.; and Zhang, Q. 2023. Algorithm Evolution Using Large Language Model. *arXiv preprint arXiv:2311.15249*.

Ma, Y. J.; Liang, W.; Wang, G.; Huang, D.-A.; Bastani, O.; Jayaraman, D.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*.

Ma, Z.; Guo, H.; Gong, Y.-J.; Zhang, J.; and Tan, K. C. 2024. Toward Automated Algorithm Design: A Survey and Practical Guide to Meta-Black-Box-Optimization. *arXiv preprint arXiv:2411.00625*.

Metz, L.; Maheswaranathan, N.; Nixon, J.; Freeman, D.; and Sohl-Dickstein, J. 2019. Understanding and correcting pathologies in the training of learned optimizers. In *ICML*, 4556–4565. PMLR.

Meyerson, E.; Nelson, M. J.; Bradley, H.; Moradi, A.; Hoover, A. K.; and Lehman, J. 2023. Language Model Crossover: Variation through Few-Shot Prompting. *arXiv preprint arXiv:2302.12170*.

Mo, S.; Wu, K.; Gao, Q.; Teng, X.; and Liu, J. 2024. AutoSGNN: Automatic Propagation Mechanism Discovery for Spectral Graph Neural Networks. In *The 39th AAAI*.

Mouret, J.-B.; and Maguire, G. 2020. Quality diversity for multi-task optimization. *GECCO*.

Müller, S.; Feurer, M.; Hollmann, N.; and Hutter, F. 2023. PFNs4BO: In-Context Learning for Bayesian Optimization. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, 25444–25470.

Mutny, M.; and Krause, A. 2018. Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *NeurIPS*, volume 31. Curran Associates, Inc.

Nasir, M. U.; Earle, S.; Togelius, J.; James, S.; and Cleghorn, C. 2023. LLMatic: Neural Architecture Search via Large Language Models and Quality-Diversity Optimization. *arXiv preprint arXiv:2306.01102*.

Romera-Paredes, B.; Barekatain, M.; Novikov, A.; Balog, M.; Kumar, M. P.; Dupont, E.; Ruiz, F. J.; Ellenberg, J. S.; Wang, P.; Fawzi, O.; et al. 2023. Mathematical discoveries from program search with large language models. *Nature*, 1–3.

Ros, R.; and Hansen, N. 2008. A simple modification in CMA-ES achieving linear time and space complexity. In *International Conference on Parallel Problem Solving from Nature*, 296–305. Springer.

Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; and Sutskever, I. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Sampson, J. R. 1976. Adaptation in natural and artificial systems (John H. Holland).

Santoni, M. L.; Raponi, E.; De Leone, R.; and Doerr, C. 2023. Comparison of High-Dimensional Bayesian Optimization Algorithms on BBOB. *arXiv preprint arXiv:2303.00890*.

Shala, G.; Biedenkapp, A.; Awad, N.; Adriaensen, S.; Lindauer, M.; and Hutter, F. 2020. Learning step-size adaptation in CMA-ES. In *International Conference on Parallel Problem Solving from Nature*, 691–706. Springer.

Stanovov, V.; Akhmedova, S.; and Semenkin, E. 2022. NL-SHADE-LBC algorithm with linear parameter adaptation bias change for CEC 2022 Numerical Optimization. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, 01–08. IEEE.

Storn, R.; and Price, K. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11: 341–359.

Tanabe, R.; and Fukunaga, A. S. 2014. Improving the search performance of SHADE using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, 1658–1665.

TV, V.; Malhotra, P.; Narwariya, J.; Vig, L.; and Shroff, G. 2020. Meta-learning for black-box optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 366–381. Springer.

Vicol, P.; Metz, L.; and Sohl-Dickstein, J. 2021. Unbiased gradient estimation in unrolled computation graphs with persistent evolution strategies. In *ICML*, 10553–10563. PMLR.

Wang, C.; Liu, J.; Wu, K.; and Wu, Z. 2022. Solving Multitask Optimization Problems With Adaptive Knowledge Transfer via Anomaly Detection. *IEEE Transactions on Evolutionary Computation*, 26(2): 304–318.

Wichrowska, O.; Maheswaranathan, N.; Hoffman, M. W.; Colmenarejo, S. G.; Denil, M.; Freitas, N.; and Sohl-Dickstein, J. 2017. Learned optimizers that scale and generalize. In *ICML*, 3751–3760. PMLR.

Wierstra, D.; Schaul, T.; Glasmachers, T.; Sun, Y.; Peters, J.; and Schmidhuber, J. 2014. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1): 949–980.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3): 229–256.

Wu, K.; Li, X.; Liu, P.; and Liu, J. 2023. DECN: Evolution inspired deep convolution network for black-box optimization. *arXiv preprint arXiv:2304.09599*.

Yang, C.; Wang, X.; Lu, Y.; Liu, H.; Le, Q. V.; Zhou, D.; and Chen, X. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Zhang, J.; Xu, C.; Li, J.; Chen, W.; Wang, Y.; Tai, Y.; Chen, S.; Wang, C.; Huang, F.; and Liu, Y. 2021. Analogous to Evolutionary Algorithm: Designing a Unified Sequence Model. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *NeurIPS*, volume 34, 26674–26688. Curran Associates, Inc.