arXiv:2304.03995v1 [cs.NE] 8 Apr 2023

通过元黑盒优化发现基于注意力的遗传算法

Robert Tjarko Lange*
柏林工业大学

Tom Schaul
DeepMind

Yutian Chen
DeepMind

Chris Lu*
牛津大学

Tom Zahavy
DeepMind

Valentin Dalibard
DeepMind

Sebastian Flennerhag
DeepMind

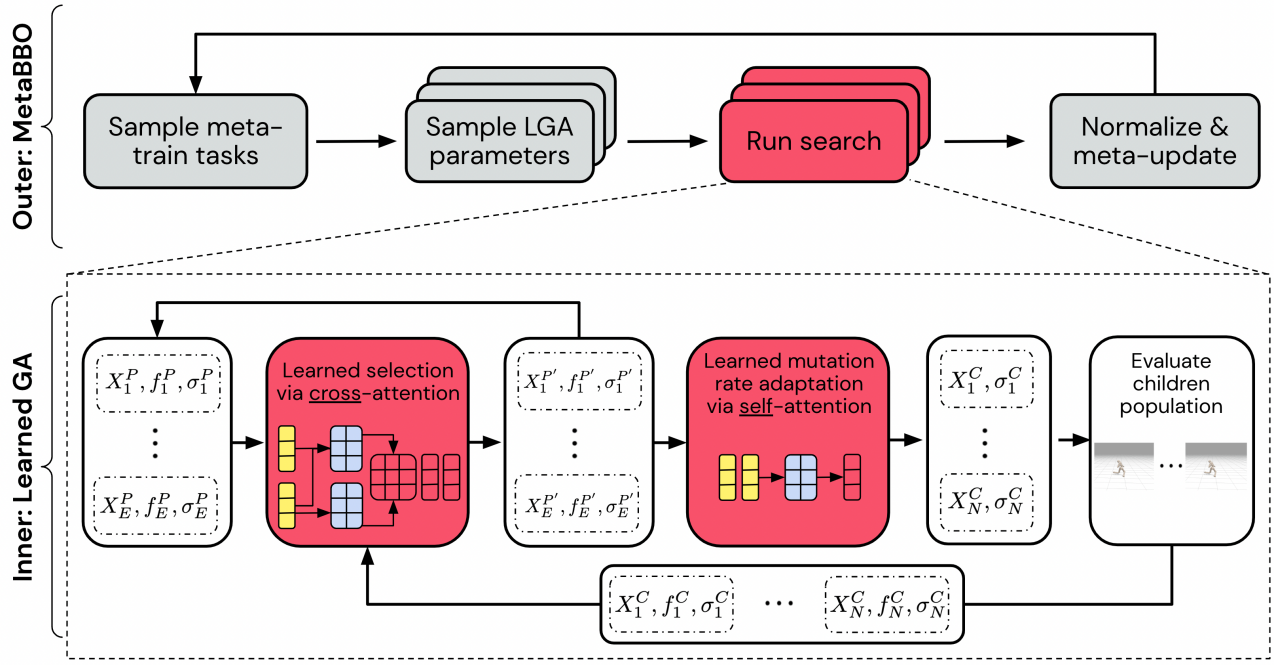


图 1：通过MetaBBO发现基于注意力的学习型遗传算法（LGA）。在每次元迭代中，从元进化优化器（EO）中采样一组内循环任务和一组候选解 LGA参数。随后，运行一次内循环搜索，并利用计算资源计算跨任务的归一化元适应度分数。我们更新元 EO 并进行迭代。

摘要

遗传算法构成了一类黑箱优化算法，其灵感来源于生物进化的原理。虽然它们提供了通用的优化工具，但其具体实现可能具有启发性，且往往仅受到宽泛的生物学直觉启发。在本研究中，我们探索了一种根本不同的方法：给定遗传算子的充分灵活的参数化形式，我们以数据驱动的方式发现全新的遗传算法。更具体地说，我们将选择和变异率自适应参数化为交叉注意力和自注意力模块，并使用元黑箱优化 在一组多样化的优化任务上演化它们的参数。最终得到的学习型遗传算法 优于最先进的自适应基线遗传算法，并且其泛化能力远超其元训练设置。学习到的算法可以应用于此前未见过的优化问题、搜索维度和评估预算。我们对所发现的算子进行了广泛分析，并开展了消融实验，突出了灵活的模块参数化所带来的优势，以及将学习到的算子迁移（“即插即用”）到传统遗传算法中的能力。

CCS 概念

• **计算方法** → **遗传算法**;

关键词

遗传算法，机器学习，元学习

* 实习期间在 DeepMind 完成。联系方式：robert.t.lange@tu-berlin.de。

1 引言

研究动机。遗传算法（GAs）提供了一系列受进化启发的优化算法，可灵活应用于黑箱优化（BBO）问题。它们通常依赖人工设计的算子，而这些算子带来了一套严格受限且最重要的是主观的人工先验。这会带来领域过拟合和泛化能力有限的风险。基于近期关于发现基于注意力的进化策略的研究结果 [25], , 我们提出，可以通过从数据中元学习有效的遗传算法算子来克服这些局限。这样一来，遗传算法本身的归纳偏置便会通过其参数化间接编码，并可通过优化驱动的方式加以发现，即提升其在相关任务分布上的元性能。**研究方法**。受 Set Transformer[27] 架构近期成功的启发，我们引入基于神经网络的架构来替代核心遗传算子：选择和变异率自适应被构造为点积注意力模块，可灵活应用于维度和种群规模各异的问题。由此产生的遗传算法系列能够根据具体的模块权重实现不同的操作。我们使用元黑箱优化（*MetaBBO*，[25]）在一组具有代表性的优化问题上对这些权重进行元进化。

结果。我们评估了元训练的学习型遗传算法（*LGA*）的性能和泛化能力。LGA能够远超其元训练分布进行泛化，并在多个黑箱优化基准测试（BBOB）[2, 13] 和神经进化问题上优于已有的基线遗传算法。这包括不同的优化函数、搜索维度数量和种群规模。此外，我们分析了所发现的神经网络遗传算法算子：选择算子学习到了一种自适应的截断形式，在父代种群中保持多样性和冗余。另一方面，学习到的变异率自适应算子会根据任务自动调整探索程度。我们研究了元进化任务分布的重要性：虽然仅使用五个 BBOB 函数也可以元进化出有效的学习型遗传算法，但我们展示了学习型遗传算法可能会对其元训练分布过拟合，因此必须确保充分的元正则化，以实现广泛的泛化。经过训练的学习型遗传算法对超参数的选择以及元训练过程的细节都具有鲁棒性。最后，我们展示了单个学习到的算子可以替代白箱遗传算法算子，并产生积极的迁移效应。

贡献。我们的贡献总结如下：

(1)我们提出了一种基于点积注意力的参数化方法，用于选择与变异率自适应遗传算法算子。(2) 我们基于其在元训练黑箱优化任务上的表现，通过元进化其参数来发现新的遗传算法。(3)所得到的LGA能够远远超出其元训练设置进行泛化，并在各种基准任务和评估预算下优于多种遗传算法基线。(4)我们对 LGA进行了多项消融实验，以评估学习各个算子模块的贡- 献。两个学习到的算子都对整体性能有所贡献。(5)我们强调了训练后的 LGA 对其超-参数的鲁棒性、所学习组件的可迁移性，以及 MetaBBO 过程的稳定性。

2 相关工作

通过元学习算法进行发现。近期的研究尝试通过在具有代表性的任务分布上对参数化组件进行元学习，以端到端优化的归纳偏置取代人工设计的算法。例如，这包括通过元梯度发现强化学习目标函数 [28, 33, 44], 调度算法超参数 [9, 34, 45, 47], , 以及对完整学习算法进行元学习 [18, 19, 42]。发现过程可以借助合适的神经网络架构来实现。我们提出的 LGA 架构利用注意力层，推导出一类基于神经网络的遗传算法。

元学习梯度优化。我们的工作与元学习基于梯度下降的学习规则这一目标密切相关 [1, 3, 31]。这些方法依赖于通过反向传播算法高效计算梯度，因此不适用于黑箱优化问题。一个小型神经网络处理梯度和标准优化器统计量（动量等），并输出权重变化量。随后，优化器网络的权重在任务分布上通过元学习得到 [30]。Metz 等人[29] 表明，这能够为深度学习任务带来竞争力极强的优化器。另一方面，我们通过 MetaBBO 发现的 LGA 提供了一种通用黑箱优化方法，不要求目标函数可微。

元学习种群优化。Shala 等人 [38]对 CMA-ES[15]中标量变异率的控制器进行元学习。Chen 等人 [5], 、Gomes 等人 [12], 和 TV 等人 [41] 此前曾针对低维黑箱优化问题，优化由神经网络参数化的完整算法。他们都使用循环网络来处理原始候选解及其相应的适应度分数。这些方法通常难以泛化到新的优化领域，并且往往受限于固定的种群规模和/或搜索维度。Lange 等人 [25] 最近利用点积自注意力对于输入排序的等变性质 [20, 27, 39] ，为进化策略学习自适应重组权重。所提出的 LGA 扩展了这一基于注意力的黑箱优化视角，以刻画遗传算法算子。成功完成元训练后，所学习的遗传算法能够泛化到未见过的种群和大型搜索空间。据我们所知，我们首次证明了元学习遗传算法能够泛化到具有挑战性的神经进化任务。最后，MetaBBO 方法不要求获取元任务最优解 [41]或教师算法 [38] 的相关知识。

基线遗传算法。在全文中，我们将其与以下四种具有竞争力的基线遗传算法进行比较：

- 高斯遗传算法 [35]：一种使用高斯扰动和固定变异强度，并采用截断选择的简单遗传算法。
- MR-1/5遗传算法 [35]：如果所有扰动中有 1/5 是有益的，则将变异率加倍。否则，该变异率减半。
- SAMR-GA [7]：基于简单的协同进化启发式和元变异率自适应逐父代变异率。
- GESMR-GA [21]：通过采用群组精英选择准则和变异率共享，避免变异率消失。

此外，我们还将 Sep-CMA-ES [36]作为神经进化任务的可扩展进化策略基线。每种基线遗传算法都通过小规模网格搜索遍历进行调优（见附录 B）。除此之外，我们采用了作者提供的设置。

3 背景

黑箱优化。在本文中，我们关注高效的连续黑箱优化：给定一个函数 $f(x)$：$\mathbb{R}^D \to \mathbb{R}$，其函数形式未知，也就是说我们无法计算其导数，我们希望找到它的全局最优解：

$$\min_{\mathbf{x}} f(\mathbf{x}), \text{s.t. } \mathbf{x}_d \in [l_d, u_d] \subset [-\infty, \infty], \forall d = 1, ..., D,$$

遗传算法。遗传算法提供了一类黑箱优化算法，它们会迭代评估一个由 N候选解 $X^C = [\mathbf{x}_1^C, ..., \mathbf{x}_N^C]^T \in \mathbb{R}^{N\times D}$（“子代”）组成的种群，其适应度为 $\mathbf{f}^C \in \mathbb{R}^N$。给定一组 E“父代”解 $X^P = [\mathbf{x}_1^P, ..., \mathbf{x}_E^P]^T \in \mathbb{R}^{E\times D}$ 及其对应的适应度 $\mathbf{f}^P \in \mathbb{R}^E$，算法会根据一种基于适应度的启发式选择准则，用子代替换父代。

大多数遗传算法都采用截断选择，其中所有子代和父代 $[\mathbf{x}_1^C, ..., \mathbf{x}_N^C, \mathbf{x}_1^P, ..., \mathbf{x}_E^P]^T$ 会根据其适应度进行联合排序。表现最佳的前 $-E$ 个解将替代父代档案：

$$X^{P'}, \mathbf{f}^{P'} = \texttt{Selection}(X^P, \mathbf{f}^P, X^C, \mathbf{f}^C).$$

通常，父代的数量设置得很小 $E \ll N$，甚至常常为 $E = 1$，从而实现一种爬山法。N子代候选解从父代中均匀地有放回采样：

$$\tilde{X}^P, \tilde{\mathbf{f}}^P = \texttt{Sample}(X^P, \mathbf{f}^P) \in \mathbb{R}^{N\times D}.$$

之后，使用变异率（MR） $\sigma \in \mathbb{R}_+$对其进行扰动，该变异率控制高斯噪声 $\epsilon_j \sim \mathrm{N}(\mathbf{0}_D, \mathbf{I}_D)$ 的强度：

$$X_j^C = \texttt{Mutation}(\tilde{X}_j^P, \sigma) = \tilde{X}_j^P + \sigma\epsilon_j \in \mathbb{R}^D, \ \forall j = 1, ..., N.$$

许多竞争性的遗传算法会保留一个父代特定的 [7] 变异率 $\boldsymbol{\sigma}^P \in \mathbb{R}^E$ 向量，并额外执行变异率自适应（MRA）步骤，以利用适应度评估过程中收集的信息来改进变异率：

$$\boldsymbol{\sigma}^P = \texttt{Adaptation}(\mathbf{f}^P, \boldsymbol{\sigma}^P, \mathbf{f}^C, \boldsymbol{\sigma}^C) \in \mathbb{R}^E.$$

在这种情况下，子代会根据其采样得到的个体特定变异率（$\boldsymbol{\sigma}^C \leftarrow \tilde{\boldsymbol{\sigma}}^P \in \mathbb{R}^N$）受到扰动，选择也会应用于子代的变异率。或者，GESMR-GA [21] 构建父代子群，并根据观察到的适应度提升在线协同进化群体级变异率。

通过点积自注意力进行集合运算。缩放点积注意力（SDPA）尤其适合描述执行集合运算的算法，因为它能够自然地强制实现置换不变函数。考虑 SDPA 的标准形式，它将一组 N输入词元 $X \in \mathbb{R}^{N\times D}$嵌入到$D_K$维的潜在查询 Q、键 K和值 V表示中：

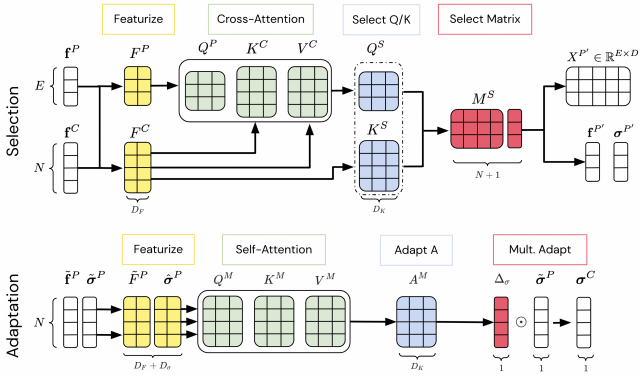
$$Q = XW_Q \in \mathbb{R}^{N\times D_K},$$
$$K = XW_K \in \mathbb{R}^{N\times D_K},$$
$$V = XW_V \in \mathbb{R}^{N\times D_K}.$$

输出 Y 是值的线性组合计算资源得到的：

$$Y = \text{softmax}\left(QK^T/\sqrt{D_K}\right)V \in \mathbb{R}^{N\times D_K}.$$

可以证明，这一变换对于 X中标记的排序是等变的，即对 X 的行进行置换，将对 Y [20, 39] 的行应用相同的置换。我们将利用这一合适的归纳偏置来刻画遗传算法，因为遗传算法本质上处理的是候选解及其适应度分数的集合。

图2：学习到的遗传算子。顶部：父代与子代适应度特征之间的交叉注意力选择。底部：通过对父代变异与适应度特征进行自注意力处理实现的 MRA。



4 基于注意力的遗传算子

现在，我们介绍遗传选择与适应算子的基于注意力的参数化方法（图 2）。接着，我们将在一组具有代表性的优化任务上对这些算子进行元优化，以提取有用的黑箱优化机制。我们先回答一个自然的问题：算子应处理哪些输入，才能在不同适应度尺度和解尺度之间实现泛化？**基于适应度分数与变异强度的注意力特征**。为了计算父代与子代之间的注意力分数，我们需要构造充分的特征，以调节有效选择和变异率自适应。此外，我们希望经元优化的操作能够泛化到不同测试优化尺度。因此，我们考虑尺度不变归一化方法，例如 z 分数和中心化排名（在 $[-0.5, 0.5]$ 中）。我们将原始适应度分数（D_F 维）和父代变异率（D_σ 维）都进行转换，构造一组由注意力层处理的特征：

- $F \in \mathbb{R}^{(N+E)\times D_F}$：父代与子代的联合适应度变换（z 分数和中心化排名）。
- $F^C = F_{1:N} \in \mathbb{R}^{N\times D_F}$：从联合变换中提取的子代适应度变换（z 分数和中心化排名）。
- $F^P = F_{N:} \ (N+E) \in \mathbb{R}^{E\times D_F}$：从联合变换中提取的父代适应度变换（z 分数和中心化排名）。
- $F^{P,} \in \mathbb{R}^{N\times D_F}$：（分开）选择操作后采样父代的适应度变换（z 分数和中心化排名）。
- $\hat{\boldsymbol{\sigma}}^P \in \mathbb{R}^{N\times D_\sigma}$：采样父代的变异强度变换（z 分数和 [-1, 1] 归一化）。
- $F^M = [\tilde{F}^P, \hat{\boldsymbol{\sigma}}^P] \in \mathbb{R}^{N\times(D_F+D_\sigma)}$：采样父代的适应度与变异率特征拼接。

适应度特征还包括一个布尔值，用于指示某个个体的表现是否优于迄今为止观察到的最佳适应度。 **通过交叉注意力进行选择**。我们用交叉SDPA层替代常见的基于排序的选择机制，该层比较子代和父代。它首先将父代和子代的适应度变换嵌入到查询、键和值中：

$$Q^P = F^P W_{Q^P} \in \mathbb{R}^{E\times D_K},$$
$$K^C = F^C W_{K^C}, \ V^C = F^C W_{V^C} \in \mathbb{R}^{N\times D_K}.$$

随后，我们计算归一化点积交叉注意力特征 A^S并构建一个选择矩阵 $M^S \in \mathbb{R}^{E\times(N+1)}$:

$$A^S = \text{softmax}\left(\frac{Q^P(K^C)^T}{\sqrt{D_K}}\right)V^C \in \mathbb{R}^{E\times D_K}$$
$$Q^S = A^S W_{Q^S} \in \mathbb{R}^{E\times D_k},\ K^S = F_C W_{K^S} \in \mathbb{R}^{N\times D_K}$$
$$M^S = \text{softmax}\left(\left[\frac{Q^S(K^S)^T}{\sqrt{D_K}}, \mathbf{1}_E\right]\right) \in \mathbb{R}^{E\times(N+1)}$$

在最终一行中，我们将一个 E 维全 1 向量拼接到 Q^S 和 K^S 的外积上。直观地说，这一列表示一个固定偏移量，用于指示父代是否复制了任何子代，或者是否未被替换。 M^S 的各行则指定每个子代替换父代的概率:

$$M^S = \text{softmax}\left(\begin{bmatrix} m_{11} & m_{12} & \dots & m_{1N} & 1 \\ m_{21} & m_{22} & \dots & m_{2N} & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 \\ m_{E1} & m_{E2} & \dots & m_{EN} & 1 \end{bmatrix}\right)$$

M^S 为行随机，即各行之和为1。例如， M^S_{11} 表示用子代1 替换父代1的概率，而 M^S_{1N+1} 则对应于不替换第一个父代。我们从类别分布中按行采样，以确定子代是否替换特定的父代。随后，我们使用选择矩阵更新父代档案（$X^{P'}$）和适应度档案（$\mathbf{f}^{P'}$）：

$$S \sim \text{Categorical}(M^S) \text{ with } S \in \mathbb{R}^{E\times(N+1)},$$
$$X^{P'} = X^C \cdot S_{:,1:N} + X^P \cdot \text{diag}(S_{:,N:N+1}),$$

同样，我们通过掩码加法得到$\mathbf{f}^P$ 和$\boldsymbol{\sigma}^P$ 。该选择算子可以通过用同一个子代替换多个父代槽位，灵活调节截断选择的程度。

基于自注意力的变异率自适应（MRA）。除选择外，我们还通过元学习实现 MRA。将采样父代的适应度和变异率特征拼接后输入缩放点积注意力层，该层输出一个子代特定特征矩阵 $A^M \in \mathbb{R}^{N\times D_K}$:

$$K^M = F^M W_{K^M},\ Q^M = F^M W_{Q^M},\ V^M = F^M W_{V^M} \in \mathbb{R}^{N\times D_K},$$
$$A^M = \text{softmax}\left(\frac{Q^M(K^M)^T}{\sqrt{D_K}}\right)V^M \in \mathbb{R}^{N\times D_K}.$$

随后，通过投影并重新参数化注意力输出，构造对变异率的乘法自适应:

$$\Delta_\sigma = \exp(0.5 \times A^M W_\sigma) \in \mathbb{R}^N$$

子代的变异率 $\boldsymbol{\sigma}^C$ 通过逐元素乘法得到:

$$\boldsymbol{\sigma}^C = \Delta_\sigma \odot \boldsymbol{\sigma}^{P'} \in \mathbb{R}^N$$

$\theta = \{W_{Q^P}, W_{K^C}, W_{V^C}, W_{Q^S}, W_{K^S}, W_{Q^M}, W_{K^M}, W_{V^M}, W_\sigma\}$表示联合注意力权重，用于刻画 LGA 的一个具体实例。我们使用较小的特征维度 $D_K = 16$，由此得到<1500 个可训练的元参数。总而言之，我们引入了两个基于点积注意力的算子，用于替代标准选择和变异率自适应操作。在全文中，我们重点讨论学习式选择和变异率自适应操作，但在附录 A 中，我们概述了如何利用自注意力进一步构建采样和交叉算子。

5 元训练、目标函数与任务

我们对遗传算法注意力模块的权重进行元优化，以在一组具有代表性的任务上执行黑箱优化。更具体地说，我们采用此前介绍的 MetaBBO 过程 [25]，并演化 LGA 参数，使其在由 10 个 BBOB [13] 函数构成的任务分布上的性能最大化。这些函数具有不同的性质，即可分性、条件性和多模态性（见表 1）。在每个元世代中（见图 1），我们首先从元进化优化算法中均匀采样一组黑箱优化任务和 LGA 参数。我们将刻画 LGA 的参数集合表示为 θ_i，其中包含 $i = 1, ..., M$ 个元种群个体。随后，通过运行内循环搜索，在所有任务上评估每个 LGA。我们计算聚合的元性能分数，以更新元进化优化算法。*MetaBBO-*

算法 1 学习型遗传算法的 MetaBBO 训练

输入： 元种群 M、元任务大小 J、内循环种群数-
ulation size N, 代数 T, MetaEO（例如 OpenAI-ES、 [37]）。
初始化元搜索分布 μ， $\Sigma = \sigma_{meta} I$。
当未完成 **执行**
包含 J黑箱优化任务，其中有 $\xi_l,\ \forall l = 1, ..., J.$
采样学习型遗传算法候选解： $\theta_i \sim \mathrm{N}(\mu,\ \Sigma)\ \forall i = 1,\ \dots\dots M$。
    在相同的 K 任务上评估所有 M 学习型遗传算法候选解:
    对于 $l = 1$、……、J，**执行**
        对于 $i = 1$、……、M，**执行**
            初始化搜索档案和适应度档案 X^P $, \boldsymbol{\sigma}^P, \mathbf{f}^P.$
            对于 $t = 0, ..., T-1$，**执行**
                采样： $\tilde{X}^P,\ \tilde{\mathbf{f}}^P,\ \tilde{\boldsymbol{\sigma}}^P \leftarrow$采样$(X^P,\ \mathbf{f}^P,\ \boldsymbol{\sigma}^P)$。
                执行 MRA： $\boldsymbol{\sigma}^C \leftarrow$ 适应$\theta_i($ $\tilde{\mathbf{f}}^P, \tilde{\boldsymbol{\sigma}}^P).$
                变异： $X^C \leftarrow$变异$(\tilde{X}^P,\ \boldsymbol{\sigma}^C)$。
                评估所有子代： $f(X^C_j|\xi_l)\ \forall j = 1,\ \dots\dots,\ N$。
                更新父代档案:
                $X^P,\ \mathbf{f}^P,\ \sigma^P \leftarrow$选择$\theta_i(\mathbf{f}^C,\ \mathbf{f}^P)$。
            结束循环
        结束循环
    结束循环
    收集适应度分数 $\left[\left[\left[\{f(x^C_{j,\ t}|\xi_l)\}_{j=1}^N\right]_{t=1}^T\right]_{l=1}^J\ |\theta_i\right]_{i=1}^M.$
    计算归一化聚合元适应度 $\{\bar{f}(\theta_i)\}_{i=1}^M$。
    更新元搜索 $\mu, \Sigma, \leftarrow \text{MetaEO}(\{\theta_i, \bar{f}\}_{i=1}^M|\mu, \Sigma).$
结束循环

目标函数基于每个 LGA 实例所收集的内循环适应度分数进行计算，其中 ξ_l 表示 $l = 1,..., J$ 任务的特定参数。对于每个候选解 θ_i，我们将最佳种群成员的最终性能最小化。随后，我们对元种群成员中的任务特定结果进行z-评分，并计算跨任务的中位数:

$$\left[[f(\theta_i|\xi_k)]_{i=1}^M\right]_{k=1}^K = \left[\left[\min_j\{f(x_{j,T}|\xi_l)\}_{j=1}^N|\theta_i\right]_{i=1}^M\right]_{l=1}^J$$
$$\{\bar{f}(\theta_i)\}_{i=1}^M = \texttt{Median}\left[\texttt{Z-Score}\left([f(\theta_i|\xi_k)]_{i=1}^M\right)\right]_{k=1}^K.$$

外循环使用 OpenAI-ES [37] 优化 θ，进行 750 代元训练，元种群规模为 $M = 512$。我们采样

图3：LGA MetaBBO。跨元代的元评估。在两个10维BBOB任务（顶部）和神经进化任务（底部）上对LGA进行评估。3次独立MetaBBO运行的均值和1.96倍标准误区间。

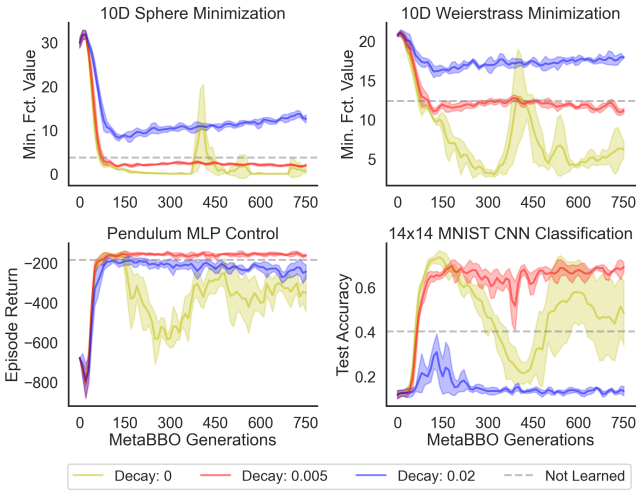
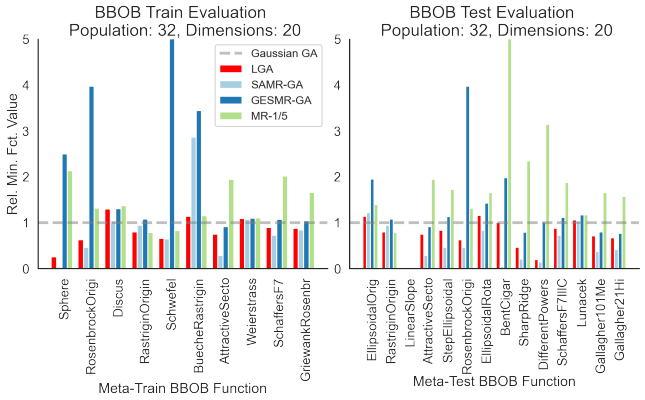


图 4：BBOB任务上的元评估。左：训练函数。右：留出函数。分数以高斯遗传算法基线方法的表现进行归一化。越低越好。取 50 次独立评估运行的平均值。



$J = 256$ 个任务，并在每个任务上独立评估采样得到的 LGA。每个 LGA 展开运行 $T = 50$ 代，每次迭代包含 $N = 16$ 个种群个体。整个过程中，我们假设 $E = N$，即父代数量等于子代数量。这并非限制，因为选择算子能够用同一个子代替换多个父代（见 7.1 节）。借助 JAX 库 [4, 23] 提供的自动向量化和设备并行能力，我们能够在多个加速器上大规模并行地对所有 θ_i 在所有 ξ_j 个任务上进行元评估。关于元训练设置和鲁棒性的更多细节与实验，请参见附录 B.1 和 C。

6 实验

现在，我们将对MetaBBO优化流程以及所发现的 LGA 进行全面的实验评估。为此，我们旨在回答以下问题：

(1)是否可以通过 MetaBBO，利用有限的一组元训练黑箱优化任务来元进化出具有竞争力的学习型遗传算法（第 6.1 节）？ (2) 所得到的学习型遗传算法在未见过的黑盒优化问题和不同搜索预算下，是否优于遗传算法基线（第 6.2 节）？ (3) 在有限的一组任务上发现的学习型遗传算法，在其元训练设置之外（例如超参数优化和神经进化；第 6.3 和 6.4 节）能够实现多大程度的泛化?

6.1 在 BBOB 函数上进行元训练

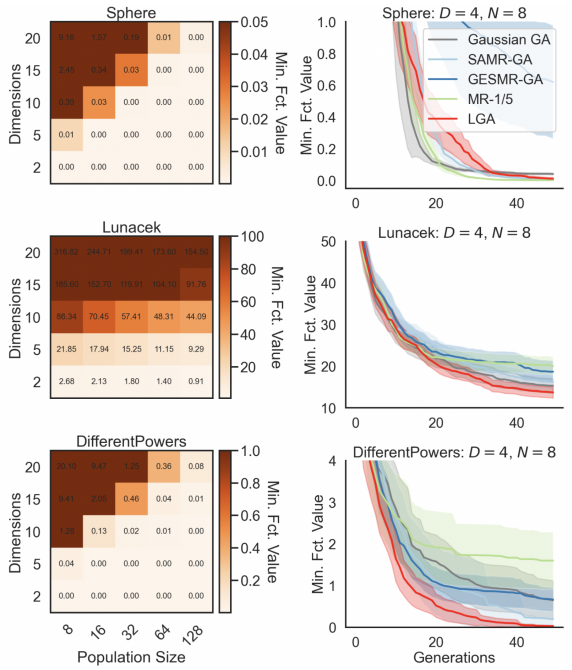
我们首先在一个任务分布上进行元进化，以优化LGA参数；该任务分布由10 个具有不同随机最优点偏移、评估噪声和所考虑问题维度($D \leq 10$)的 BBOB 函数组成。在整个元训练过程中，我们在多个不同的下游任务上评估优化后LGA 的性能。这些任务包括元训练期间见过的 BBOB函数、留出的元测试 BBOB函数以及小型神经进化任务。在图3中，我们绘制了元训练期间的详细评估曲线。经过 MetaBBO 训练的 LGA 很快学会了如何执行优化在低维 BBOB 元训练函数上。有趣的是，我们发现，MetaBBO 流程可能会使 LGA 对其元训练所用的 BBOB 任务过拟合。对于使用多层感知机策略的未见过的 Pendulum 控制任务，以及使用卷积神经网络的缩小版 14×14 MNIST 分类任务，LGA 的下游性能都会下降并变得不稳定。因此，我们研究了元正则化能否提升其在此类未见过的神经进化任务上的泛化能力。更具体地说，我们比较了三种不同的元均值正则化系数 $\lambda \in \{0,\ 0。005,\ 0。02\}$，它们会将元均值指数衰减至零，$\mu\lambda = (1 - \lambda)\mu$。我们观察到，采用恰当的 0。005 衰减，可以提升并稳定其在神经进化任务上的泛化能力。附录 C 进一步探讨了元任务分布、元目标和 LGA 注意力规模的影响。MetaBBO 流程对这些设置的选择总体上具有较强的鲁棒性。较小的注意力层便足以稳定地发现性能优异的学习型遗传算法。此外，这还能降低运行 LGA 所需的浮点运算次数和内存。只需一台单核 CPU 设备，即可轻松评估经过元训练的 LGA。

6.2 在 BBOB 函数上进行元测试

接下来，我们将在全部 BBOB 基准函数上全面评估 LGA 的性能，其中包括元训练期间未见过的测试函数。我们将其与 4 种有竞争力的遗传算法基线进行比较：高斯遗传算法、MR-1/5 遗传算法、SAMR-GA 和 GESMR-GA。我们比较了种群规模为 $N = 32$、搜索维度为 $D = 20$ 、运行 $T = 50$ 代时在所有 BBOB 函数上的性能。各代中的最佳函数值按高斯遗传算法的性能进行归一化。在图 4 中，我们发现，对于元训练期间见过的 BBOB 函数（左）和未见过的 BBOB 函数（右）中的大多数，LGA 的表现都优于所有基线方法。对于特征差异很大的函数（单峰/多峰、高/低条件性、可分离/不可分离）、不同搜索维度和种群规模，这一结论同样成立（附录图 17 和 18）。这进一步证明，LGA 实际上并未

所有基线方法均通过对父代档案大小和初始变异率尺度进行网格搜索来调优。附录 B.2 中报告了所有黑箱优化评估与调优设置。

图 5：LGA 泛化。左：不同预算和空间下的 BBOB 评估。右：各代的性能。50 次运行的均值和 1.96 倍标准误差区间。

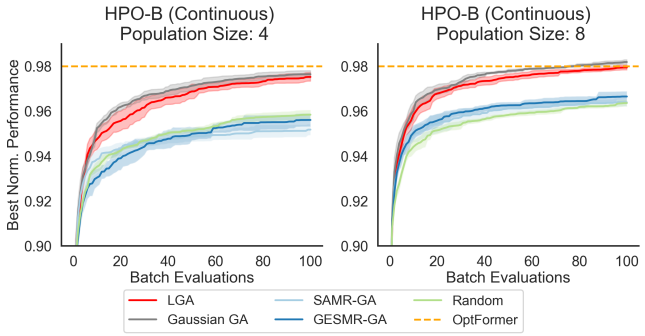


过拟合于元训练期间见过的黑箱优化函数，而是发现了一种通用遗传算法。在图 5 中，我们进一步展示了 LGA 可泛化到不同的种群规模和问题维度。元学习得到的遗传算法能够在更少的代数内达到更低的函数值（右图），并且在不同的问题设置下表现良好（左图）。随着维度增加、问题变得更加困难，LGA 可以通过增大种群规模进行补偿。

6.3 在连续HPO-B上的元测试

接下来，我们在 HPO-B 基准 [2]上测试 LGA 的性能。该基准涵盖大量超参数优化任务，包括 16 种不同的模型类型（支持向量机、XGBoost 等）及其各自的搜索空间（$D \in \{2、3、\ldots、16\}$）。每种模型都在 2 到 7 个不同的数据集上进行评估，因此总计有 86 个超参数搜索任务。我们考察连续版 HPO-B，该版本使用了一个先前拟合的代理模型。请注意，LGA 并未在此类超参数优化任务上接受过训练。图6比较了 LGA 与遗传算法基线和随机搜索基线方法的性能。此外，我们还报告了近期提出的 OptFormer 模型在总计评估 105 次后所达到的参考性能 [6] 。我们发现，LGA 的表现优于所考察的大多数遗传算法基线。对于所考察的两种种群规模，这一结果同样成立。LGA 还可以达到与 OptFormer 相近的性能，而 OptFormer 是在更多样化的任务分布上训练的。这凸显了 LGA 的迁移能力，以及其应用于元训练期间未见过的新优化领域的潜力。

图6：LGA 在 HPO-B [2] 上的评估。左：小种群规模（$N = 4$）。右：大种群规模（$N = 8$）。5 次运行的均值和 1.96 倍标准误差区间。



6.4 神经进化任务上的元测试

到目前为止，我们评估的都是发现的 LGA 在中等偏小搜索空间（即 $D \leq 20$）上的性能。但能否将 LGA 应用于神经进化场景，在具有数千个搜索维度、适应度景观特征也显然迥异的情况下使用？同样请注意，LGA 从未专门接受过进化此类高维基因组的训练，而这要求其学到的遗传算法算子具备很强的迁移能力。

所考虑的强化学习（RL）任务包括 4 个机器人控制任务（Pendulum-v1 [24] 和 5 个 Brax 任务 [10]），这些任务使用多层感知机策略；此外还包括 3 个 MinAtar 视觉控制任务（《太空入侵者》、《打砖块》和《Asterix》 [46]），这些任务使用基于卷积神经网络的策略。多层感知机基因组包含少于 1000 个权重，而 MinAtar 卷积神经网络包含约 50,000 个权重。因此，其搜索空间比 LGA 经过元训练的搜索空间高出许多个数量级（$D \leq 10$）。图 7 的前三行显示，在具有不同搜索空间、适应度景观和评估预算的 9 个强化学习任务上，LGA 能够与所有经过调优的基线遗传算法竞争。有趣的是，在卷积神经网络策略上的 LGA 与所考虑基线方法之间的性能差距最大，并且这一差距往往会随着搜索维度数量的增加而扩大。最后，在图 7 的最后一行中，我们展示了 LGA 也能够成功应用于 3 个图像分类任务，包括 MNIST、Fashion-MNIST 和 K-MNIST 分类（28×28 灰度图像）。这些任务使用一个小型卷积神经网络（2 个卷积层、ReLU 激活函数和线性读出层），其中包含 11274 个可进化权重。LGA 的泛化能力远远超出了元训练搜索范围（$T = 50$，而不是 4000），并且没有发生元过拟合 [26]。

7 对发现的 LGA 进行分析

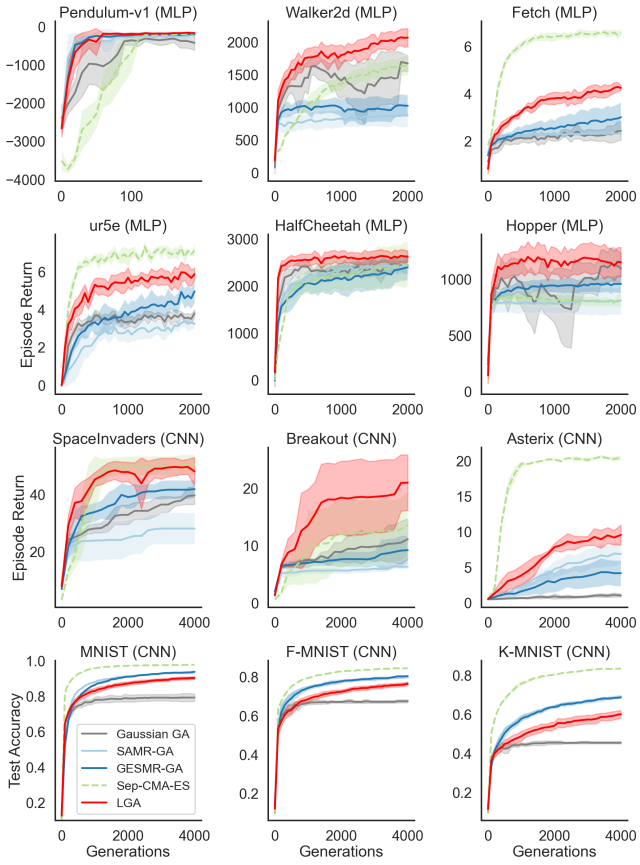
在确认经过元训练的 LGA 能够在未见过的优化问题上超越一组遗传算法基线后，我们现在将探究元学习遗传算法算子背后的机制、迁移能力和鲁棒性。

7.1 学习到的遗传算法算子的可视化

LGA 发现了黑箱遗传算子背后的哪些机制？它只是重新发现了基于适应度的截断选择，还是发现了更复杂的父代替换流程？在图 8 中，我们考察一个二维 Sphere 问题，并可视化用于更新父代档案 X^P 的选择掩码 $S_{:,\,1:N}$。我们观察到，选择算子会根据子代解相对于已见最优解的改进情况，用子代解替换父代。

此外，一个表现良好的子代通常会替换不止一个父代。这间接表明，选择算子已经通过元学习，能够动态调整其精英存档的大小，从而也调整有效的采样分布。在下一代中，替换了多个父代的子代会被更频繁地采样（允许重复采样）。此外，这也意味着存在一种鲁棒性机制：由于子代可以在父代档案中存储多次，因此它们不太可能在随机选择中被“遗忘”。另一方面，为了探索全局最优解附近的区域，变异率会随着代数增加而降低。此外，我们观察到 MR 会根据父代的表现呈现分组。父代表现不佳的子代往往具有较高的变异率。

图 7：LGA在神经进化任务上的评估，包括连续控制任务（顶部）、视觉控制（中部）和计算机视觉（底部）任务。5 次独立运行的均值和 1.96 标准误差区间。

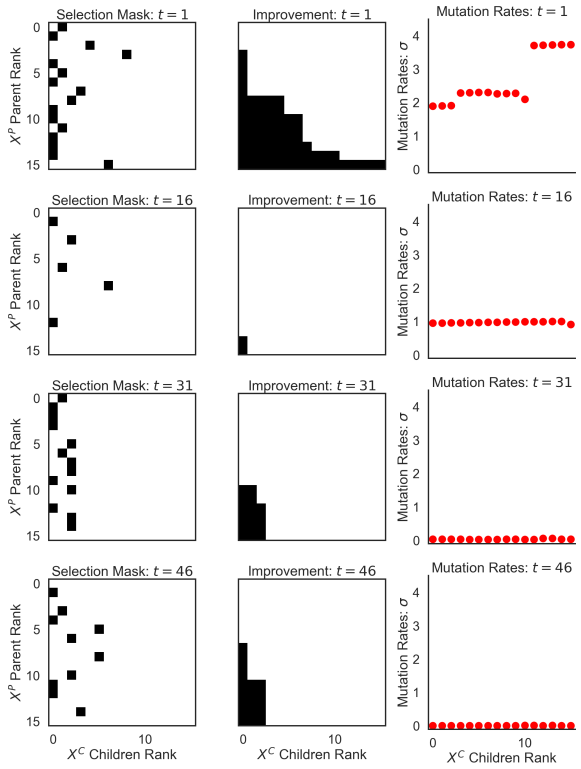


7.2 遗传算子的消融与迁移

不同的学习组件对 LGA 的整体性能有多大贡献？学习模块能否作为其他遗传算法的即插即用替代方案？为了回答这个问题，我们考虑了两类比较研究：

(1) **MetaBBO发现之前的算子消融**：我们使用数量可变的学习型遗传算子对 LGA 进行元训练。例如，我们将选择算子固定为白盒截断选择，并且仅元学习变异率适- 应。这使我们能够量化不同学习要素之间的联合贡献和协同效应。

图8：LGA在2维Sphere任务上的选择算子。左：采样得到的选择矩阵 $S_{:,1:N} \in \mathbb{R}^{E\times N}$ 中：表示子代适应度分数是否超过父代的矩阵。右：不同子代的变异率。行表示4个不同的代 $t \in \{1, 16, 31, 46\}$。



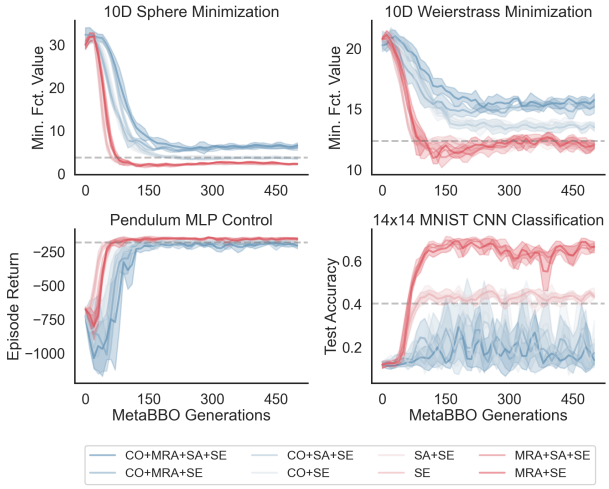
(2) **MetaBBO 发现后的算子迁移**：元训练完成后，我们探讨能否将学习到的算子替换到其他遗传算法中？这使我们能够进一步评估，特定的学习到的算子是否对下游遗传算法计算过拟合，还是能够作为遗传计算的可迁移归纳偏置。

在第一项研究中，我们比较了注意力参数化选择（SE）、变异率自适应（MRA）、交叉（CO）和采样（SA）的不同元训练组合。对于每种组合，我们在图 9 中绘制了其跨元世代的评估性能。我们观察到，MRA 对于在神经进化任务上取得良好表现至关重要。直观地说，这可以用神经网络权重所对应的解参数尺度较小来解释。遗传算法能够灵活地降低其扰动强度，因此受益匪浅。另一方面，交叉不利于 LGA 泛化到 MNIST卷积神经网络

附录 A 中还介绍了 CO 和 SA 参数化方法。在正文中，我们主要关注学习到的变异率自适应和选择。

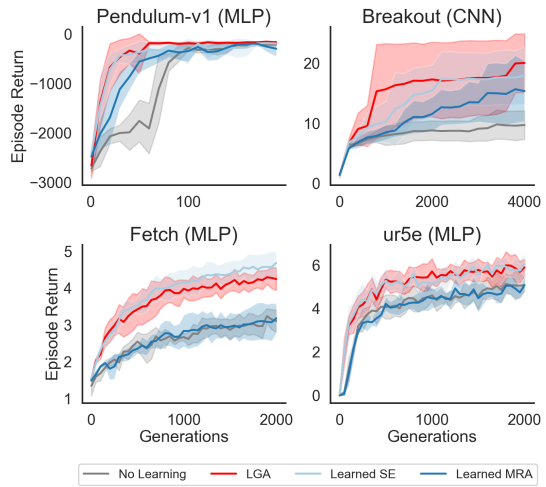
神经进化任务。这种现象可能归因于难以为不同的神经网络基因组找到有益的交叉配对。我们还观察到，学习得到的采样并未显著提升 LGA 的性能。我们推测，这是因为选择机制对子代采样产生了间接影响。最后，整体表现最佳的配置仅对选择和 MRA 进行元学习。

图 9：MetaBBO 期间 LGA 算子消融的可视化。对两个 10 维 BBOB（顶部）和神经进化任务（底部）进行 LGA 评估。我们报告了 3 次独立 MetaBBO 运行的均值和 1.96 倍标准误区间。



接下来，我们考虑用学习到的选择算子和 MRA 算子替代高斯遗传算法基线中的截断选择和固定变异率。图 10 展示了这一方法可以在四个神经进化任务上成功实现。替换选择算子或加入学习式 MRA 算子，都能提升高斯遗传算法的性能。这些学习到的算子可以作为即插即用替代项，并且具有可迁移归纳偏置。

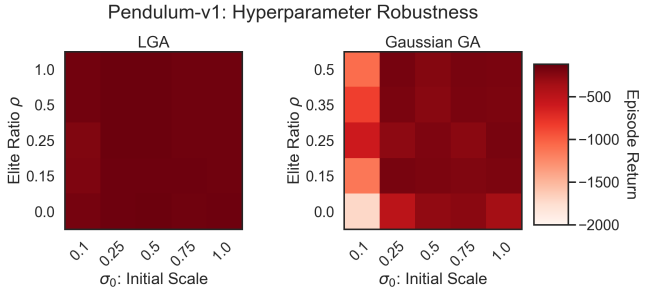
图 10：将学习到的算子迁移至高斯遗传算法。5 次运行的均值和 1.96 倍标准误区间。



7.3 LGA 的超参数鲁棒性

最后，我们评估 LGA对其剩余超参数选择的敏感性。更具体地说，我们比较 LGA与基线遗传算法在不同初始变异率尺度 σ_0 和父代档案大小 $E = \lceil \rho \times N \rceil$下的性能，其中 ρ表示构成父代数量的种群个体比例。请注意，尽管LGA针对 $\rho = 1$（即 $E = N$）进行了元训练，但我们发现它能够泛化到许多不同的档案大小，并且对初始尺度参数具有鲁棒性（Pendulum 控制任务；见图 11）。在D.2节中，我们为所有神经进化任务提供了相同的分析。与所考虑的基线遗传算法相比，LGA对超参数的敏感性要低得多。这凸显了 MetaBBO 过程所诱导的 LGA 的鲁棒性。

图 11：LGA 的超参数鲁棒性。Pendulum- v1 在不同精英比例和初始变异率下的性能。$\rho = 0$ 使用单个父代 $E = 1$。结果取 5 次独立评估运行的平均值。



8 结论

总结。在本研究中，我们利用进化优化，通过元学习发现新颖的遗传算法。我们借鉴了遗传算法算子执行集合运算这一洞见，并使用新颖的注意力模块对其进行参数化，从而引入置换等变性的归纳偏置。我们在 BBOB、HPO-B 和神经进化任务上的基准测试结果，凸显了将灵活的遗传算法参数化与数据驱动的元进化相结合的潜力。

局限性。我们使用强大的神经网络层来表征遗传算法算子。尽管这种方法灵活且可解释（第 7.1 节），其底层机制仍有一部分难以理解。未来还需进一步研究，以全面逆向工程发现的算子。在附录 E 中，我们提供了初步见解，揭示注意力输入与其输出之间的简单线性关系。我们认为，这些发现有助于设计新的“白箱”遗传算法，并以发现的“灰箱”学习型遗传算法为参考。此外，我们的分析凸显了元正则化的重要性，以及自动设计元训练课程的潜力。

未来工作。我们有兴趣明确地对学习型遗传算法进行正则化，以维持其父代档案中的多样性。这或许能为元学习质量多样性方法 [8]搭建桥梁。此外，还可以对一种灵活的进化优化算法进行参数化，使其能够在遗传算法对多个候选解进行探索与传统进化策略的单一搜索分布模式之间进行插值。最终，我们相信，通过同时协同进化元任务分布和学习型遗传算法，可以发现更优的元学习遗传算法。

参考文献

[1] Marcin Andrychowicz、Misha Denil、Sergio Gomez、MatthewW Hoffman、David Pfau、Tom Schaul、BrendanShillingford 和 NandoDe Freitas。2016 年。通过梯度下降学习如何通过梯度下降进行学习。《神经信息处理系统进展》29（2016）。[2] Sebastian Pineda Arango、Hadi S Jomaa、MartinWistuba和 Josif Grabocka。2021 年。HPO-B：基于OpenML 的大规模可复现黑箱超参数优化基准。arXiv预印本arXiv:2106.06257（2021）。[3] Samy Bengio、Yoshua Bengio、JocelynCloutier和Jan Gescei。1992 年。论突触学习规则的优化。载于《生物与人工网络中的最优性》？ Routledge，281–303。[4] James Bradbury、Roy Frostig、Peter Hawkins、Matthew James Johnson、Chris Leary、Dougal Maclaurin、George Necula、Adam Paszke、Jake VanderPlas、Skye Wanderman-Milne和Qiao Zhang。2018 年。JAX：Python+NumPy 程序的可组合变换。（2018）。http://github.com/google/jax[5] Yutian Chen、Matthew W. Hoffman、Sergio Gómez Colmenarejo、Misha Denil、Timothy P. Lillicrap、Matt Botvinick 和Nando de Freitas。2017 年。不通过梯度下降而通过梯度下降学习如何学习。载于第 34 届国际机器学习会议论文集（机器学习研究论文集），DoinaPrecup 和YeeWhye Teh（编），第 70 卷。PMLR，748–756。https: //proceedings.mlr.press/v70/chen17e.html[6] Yutian Chen、Xingyou Song、Chansoo Lee、Zi Wang、Qiuyi Zhang、David Dohan、Kazuya Kawakami、Greg Kochanski、Arnaud Doucet、Marc' aurelioRanzato 等。2022年。借助变换器实现通用超参数优化器的学习。arXiv 预印本 arXiv:2205.13320（2022）。[7] Jeff Clune、Dusan Misevic、Charles Ofria、Richard E Lenski、SantiagoF Elena和Rafael Sanjuán。2008 年。自然选择无法在崎岖适应度景观上优化长期适应所需的变异率。《公共科学图书馆·计算生物学》4，9（2008），e1000187。[8] Antoine Cully 和YiannisDemiris。2017年。质量与多样性优化：统一的模块化框架。《IEEE进化计算汇刊》22，2 2017 245 259（），–。[9] Sebastian Flennerhag、Yannick Schroecker、Tom Zahavy、Hado van Hasselt、DavidSilver 和 SatinderSingh。2021 年。自举元学习。arXiv预印本 arXiv:2109.04504（2021）。[10] C DanielFreeman、Erik Frey、Anton Raichuk、Sertan Girgin、Igor Mordatch 和 OlivierBachem。2021年。Brax——用于大规模刚体仿真的可微物理引擎。arXiv预印本 arXiv:2106.13281（2021）。[11] C DanielFreeman、Erik Frey、Anton Raichuk、Sertan Girgin、Igor Mordatch 和 OlivierBachem。2021年。Brax——用于大规模刚体仿真的可微物理引擎。arXiv 预印本 arXiv:2106.13281（2021）。[12] Hugo Siqueira Gomes、BenjaminLéger和Christian Gagné。2021年。元学习黑箱基于种群的优化器。arXiv 预印本 arXiv:2103.03526（2021）。[13] Nikolaus Hansen、Anne Auger、SteffenFinck 和 Raymond Ros。2010 年。实参数黑箱优化基准测试 2010：实验设置。博士学位论文。INRIA。[14] Nikolaus Hansen、Steffen Finck、RaymondRos和 AnneAuger。2009 年。实参数黑箱优化基准测试 2009：含噪函数定义。博士学位论文。INRIA。[15] NikolausHansen 和 AndreasOstermeier。2001年。进化策略中的完全去随机化自适应。《进化计算》9，2（2001），159–195。[16] Charles R Harris、K Jarrod Millman、Stéfan J Van Der Walt、Ralf Gommers、Pauli Virtanen、David Cournapeau、Eric Wieser、Julian Taylor、Sebastian Berg、Nathaniel J Smith 等。2020585，7825 2020 357 362 。使用 NumPy 进行数组编程。《自然》（），–。[17] JohnD Hunter。2007年。Matplotlib：一个二维图形环境。《科学与工程计算》9，03（2007），90–95。[18] Louis Kirsch、Sebastian Flennerhag、Hado van Hasselt、Abram Friesen、JunhyukOh 和 YutianChen。2022年。将对称性引入黑箱元强化学习。载于AAAI人工智能大会论文集，第 36 卷。7202–7210。[19] LouisKirsch和 JürgenSchmidhuber。2021年。元学习反向传播并对其进行改进。《神经信息处理系统进展》34（2021），14122–14134。[20] Jannik Kossen、Neil Band、Clare Lyle、Aidan N Gomez、ThomasRainforth和YarinGal。2021年。数据点之间的自注意力：超越深度学习中的单个输入输出对。《神经信息处理系统进展》34（2021），28742–28756。[21] Akarsh Kumar、Bo Liu、RistoMiikkulainen 和PeterStone。2022年。通过群体精英选择实现有效的变异率自适应。arXiv 预印本 arXiv:2204.04817（2022）。[22] Robert TjarkoLange。2021 年。MLE-Infrastructure：用于分布式机器学习实验的一组轻量级工具。（2021）。http://github.com/mle-infrastructure[23] Robert Tjarko Lange。2022年。evosax：基于 JAX 的进化策略。（2022）。http://github.com/RobertTLange/evosax

[24] Robert Tjarko Lange。2022年。gymnax：基于 JAX 的强化学习环境库。（2022年）。http://github.com/RobertTLange/gymnax[25] Robert Tjarko Lange、Tom Schaul、Yutian Chen、Tom Zahavy、Valenti Dallibard、Chris Lu、Satinder Singh 和 Sebastian Flennerhag。2022年。通过元黑箱优化发现进化策略。arXiv 预印本 arXiv:2211.11260（2022年）。[26] Robert Tjarko Lange 和 Henning Sprekeler。2022年。学会不学习：计算机模拟中的先天与后天。载于 AAAI人工智能大会论文集，第36卷。7290–7299。[27] Juho Lee、Yoonho Lee、Jungtaek Kim、Adam Kosiorek、Seungjin Choi 和 Yee Whye Teh。2019年。Set Transformer：一种用于基于注意力的置换不变神经网络的框架。载于国际机器学习会议。PMLR，3744–3753。[28] Chris Lu、Jakub Grudzien Kuba、Alistair Letcher、Luke Metz、Christian Schroeder de Witt 和 Jakob Nicolaus Foerster。2022年。发现式策略优化。载于 ICML 2022 强化学习中的决策感知研讨会。[29] Luke Metz、James Harrison、C Daniel Freeman、Amil Merchant、Lucas Beyer、James Bradbury、Naman Agrawal、Ben Poole、Igor Mordatch、Adam Roberts 等。2022年。VeLO：通过扩大规模训练通用学习型优化器。arXiv 预印本 arXiv:2211.09760（2022年）。[30] Luke Metz、Niru Maheswaranathan、C Daniel Freeman、Ben Poole 和 Jascha Sohl-Dickstein。2020年。任务、稳定性、架构与计算资源：训练更有效的学习型优化器，并用它们训练自身。arXiv 预印本 arXiv:2009.11243（2020年）。[31] Luke Metz、Niru Maheswaranathan、Jeremy Nixon、Daniel Freeman 和 Jascha Sohl-Dickstein。2019年。理解并纠正学习型优化器训练中的病态问题。载于国际机器学习会议。PMLR，4556–4565。[32] Andrew Y Ng 和 Michael I Jordan。2000年。PEGASUS：一种适用于大型马尔可夫决策过程和部分可观测马尔可夫决策过程的策略搜索方法。载于第十六届人工智能不确定性会议（UAI）论文集。406–415。[33] Junhyuk Oh、Matteo Hessel、Wojciech M Czarnecki、Zhongwen Xu、Hado P van Hasselt、Satinder Singh 和 David Silver。2020年。发现强化学习算法。神经信息处理系统进展 33（2020），1060–1070。[34] Jack Parker-Holder、Raghu Rajan、Xingyou Song、André Biedenkapp、Yingjie Miao、Theresa Eimer、Baohe Zhang、Vu Nguyen、Roberto Calandra、Aleksandra Faust 等。2022年。自动化强化学习（AutoRL）：综述与开放问题。人工智能研究期刊 74（2022），517–568。[35] Ingo Rechenberg。1973年。进化策略：基于生物进化原理的技术系统优化（1973年）。[36] Raymond Ros 和 Nikolaus Hansen。2008年。对协方差矩阵自适应进化策略的一项简单修改，实现线性时间和空间复杂度。载于自然启发式并行问题求解国际会议。施普林格，296–305。[37] Tim Salimans、Jonathan Ho、Xi Chen、Szymon Sidor 和 Ilya Sutskever。2017年。进化策略：强化学习的一种可扩展替代方法。arXiv 预印本 arXiv:1703.03864（2017年）。[38] Gresa Shala、André Biedenkapp、Noor Awad、Steven Adriaensen、Marius Lindauer 和 Frank Hutter。2020年。在协方差矩阵自适应进化策略中学习步长自适应。载于自然启发式并行问题求解国际会议。施普林格，691–706。[39] Yujin Tang 和 David Ha。2021年。作为变换器的感觉神经元：用于强化学习的置换不变神经网络。神经信息处理系统进展 34（2021），22574–22587。[40] Yujin Tang、Yingtao Tian 和 David Ha。2022年。EvoJAX：硬件加速的神经进化。arXiv 预印本 arXiv:2202.05008（2022年）。[41] Vishnu TV、Pankaj Malhotra、Jyoti Narwariya、Lovekesh Vig 和 Gautam Shroff。2019年。用于黑箱优化的元学习。载于欧洲机器学习与数据库知识发现联合会议。施普林格，366–381。[42] Jane X Wang、Zeb Kurth-Nelson、Dhruva Tirumala、Hubert Soyer、Joel Z Leibo、Remi Munos、Charles Blundell、Dharshan Kumaran 和 Matt Botvinick。2016年。学习如何进行强化学习。arXiv 预印本 arXiv:1611.05763（2016年）。[43] Michael L Waskom。2021年。Seaborn：统计数据可视化。开源软件期刊 6，60（2021），3021。[44] Zhongwen Xu、Hado P van Hasselt、Matteo Hessel、Junhyuk Oh、Satinder Singh 和 David Silver。2020年。基于在线发现的目标函数进行元梯度强化学习。神经信息处理系统进展 33（2020），15254–15264。[45] Zhongwen Xu、Hado P van Hasselt 和 David Silver。2018年。元梯度强化学习。神经信息处理系统进展 31（2018）。[46] Kenny Young 和 Tian Tian。2019年。MinAtar：一个受 Atari 启发、用于开展全面且可复现的强化学习实验的测试平台。arXiv 预印本 arXiv:1903.03176（2019年）。[47] Tom Zahavy、Zhongwen Xu、Vivek Veeriah、Matteo Hessel、Junhyuk Oh、Hado P van Hasselt、David Silver 和 Satinder Singh。2020年。一种自调节演员-评论家算法。神经信息处理系统进展 33（2020），20913–20924。

致谢
本研究由 DeepMind 资助。我们感谢 Nemanja Rakićević 对本稿件提出的宝贵意见。

A 基于注意力的采样& 交叉算子

除了学习式选择和MRA外，我们还在第 7.2节中对一个基于学习的注意力式采样与交叉算子进行了实验。这里我们给出它们的形式化定义。

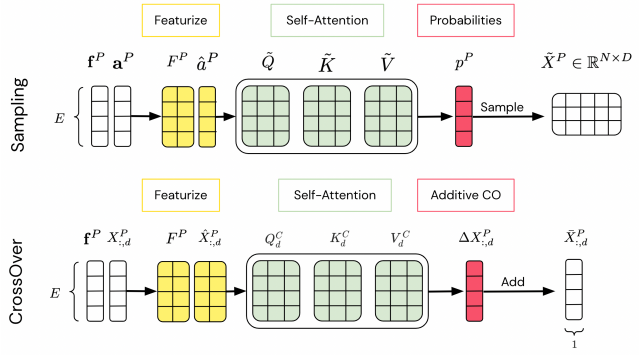
通过自注意力进行子代采样的分布。我们使用父代适应度 $\mathbf{f}^P$ 及其变换 F^P 以及年龄计数器 a^P及其双曲正切函数变换$\hat{}a^P$来计算查询、键和值。之后，输出被投影并归一化为概率分布：

$$\tilde{Q} = [F^P, \hat{a}^P]W_{\tilde{Q}}, K = [F^P, \hat{a}^P]W_{\tilde{K}} \in \mathbb{R}^{E\times D_K}, \tilde{V} = [F^P, \hat{a}^P]W_{\tilde{V}} \in \mathbb{R}^{E\times 1}$$

$$p^P = \text{softmax}\left(\frac{\tilde{Q}\tilde{K}^T}{\sqrt{D_K}}\right)\tilde{V} \in \mathbb{R}^{E\times 1}$$

$$\tilde{X}^P, \tilde{\mathbf{f}}^P, \tilde{\boldsymbol{\sigma}}^P = \texttt{Sample}(X^P, \mathbf{f}^P, \boldsymbol{\sigma}^P | p^P).$$

图 12：额外的学习式遗传算子。上：基于自注意力的父代采样概率。下：自注意力加法交叉算子。



通过自注意力进行交叉。对于父代矩阵 $X^{P:d} \in \mathbb{R}^E$的每个维度 d，我们首先构建一组归一化特征，用于衡量所有父代之间的多样性（例如，z分数标准化特征、归一化距离等），从而得到 $\hat{}X^{P:d} \in \mathbb{R}^{E\times D_E}$。然后，我们将 $[F^P, \hat{}X^{P:d}] \in \mathbb{R}^{E\times (D_F+D_E)}$拼接起来，通过嵌入得到查询、键和值，并计算缩放点积自注意力层的输出：

$$Q_d^C = [F^P, \hat{X}_{:,d}^P]W_{Q^C}, K_d^C = [F^P, \hat{X}_{:,d}^P]W_{K^C},$$

$$V_d^C = [F^P, \hat{X}_{:,d}^P]W_{V^C}, Z_d = \text{softmax}\left(\frac{Q_d^C K_d^{C^T}}{\sqrt{D_K}}\right)V_d^C \in \mathbb{R}^{E\times D_K}.$$

参数特定输出 Z_d 经过线性投影，以获得对父代的加法变化。随后，将交叉输出与原始父代档案中的维度向量相加，计算父代之间的维度交叉 X^P：

$$\Delta X_{:,d}^P = Z_d W_{\Delta X} \text{ and } \tilde{X}_{:,d}^P = X_{:,d}^P + \Delta X_{:,d}^P \in \mathbb{R}^{E\times 1}, \ \forall d = 1, ..., D.$$

该操作可以沿维度 D 使用例如 JAX 提供的自动向量化工具进行高效并行化。

B 超参数设置 B.1 LGADiscovery 的 MetaBBO 设置

函数	参考文献	性质	任务
球函数	Hansen 等人 [页 5, 13]	可分离（独立）	小
Rosenbrock 函数	Hansen 等人 [第 40, 13] 页	中等条件	中等
圆盘函数	Hansen 等人 [页 55, 13]	高条件数	中等
Rastrigin函数	Hansen 等人 [第 75, 13] 页	多峰（局部）	中等
Schwefel函数	Hansen 等人 [第 100, 13]页	多峰（全局）	中等
Bueche-Rastrigin函数	Hansen 等人 [第 20, 13] 页	可分离（独立）	大型
吸引扇区	Hansen 等人 [第 30, 13] 页	中等条件	大型
魏尔斯特拉斯	Hansen 等人 [第 80, 13] 页	多峰（全局）	大型
SchaffersF7	Hansen 等人 [第 85, 13] 页	多峰（全局）	大型
Griewank-Rosen	Hansen 等人 [第 95, 13] 页	多峰（全局）	大型

表 1：基于 BBOB 的元黑箱优化任务族。

我们为所有元成员 θ_i 共享任务参数、初始化和固定随机性。固定随机性和基于任务的归一化增强了元优化的稳定性 [“Pegasus技巧”, 32]。

参数	值	参数	值
MetaEO	OpenAI-ES [37]	M：元种群	512
α_0，衰减，最终	{0.01,0.999,0.001}	J：元任务	256
σ_0，衰减，最终	{0。1，0。999，0。001}	中心化排名	✓
元目标	minN-finalT	D_K，注意力头	{16，2}
内循环 D	~ [2, 10]	内循环 T	50
内循环 σ_0	~ [0.01, 0.5]	内循环 N	16
偏移量 $x^\star - c$	~ [−5, 5]	内循环噪声	Hansen 等人 [14]

表2：元BBO超参数（图3）。

B.2 BBOB评估设置

参数	值	参数	值
种群 N	32	维度 D	20
代 T	50	X^P 初始化	~ [−5, 5]

表 3：BBOB超参数（图 4、5）。

我们通过网格搜索调整了所有基线方法和 LGA 的精英比例 $\rho \in \{0.0, 0.15, 0.25, 0.35, 0.5, 1.0\}$ 和初始 $\sigma_0 \in \{0.1, 0.25, 0.5, 0.75, 1.0\}$。

B.3 HPO-B（连续）评估设置

图6针对两种不同的种群规模 $N \in \{4, 8\}$ 以及 $T = 100$生成。遗传算法档案初始化为 0.5，我们遵循基准测试代码库中概述的评估协议。我们通过网格搜索调整了所有基线方法和 LGA 的精英比例 $\rho \in \{0.0, 0.15, 0.25, 0.35, 0.5, 1.0\}$ 以及初始 $\sigma_0 \in \{0.1, 0.25, 0.5, 0.75, 1.0\}$。

B.4 神经进化评估设置

参数	值	参数	值
MNIST N	128	MNIST T	4000
MNIST卷积神经网络 L	8 [5, 5] & 16 [3,3]	MNIST 批大小	1024
Brax N	256	Brax T	2000
观测归一化	✓	X^P 初始化	0
回合步数	500	蒙特卡洛评估	8
Brax 多层感知机 L	0 - 仅读出层	Brax 激活函数	Tanh
MinAtar N	256	MinAtar T	4000
MinAtar 卷积神经网络 L	16 [3, 3] 滤波器	MinAtar 多层感知机 L	1 修正线性单元 32 个单元
回合步数 500 蒙特卡洛评估			16

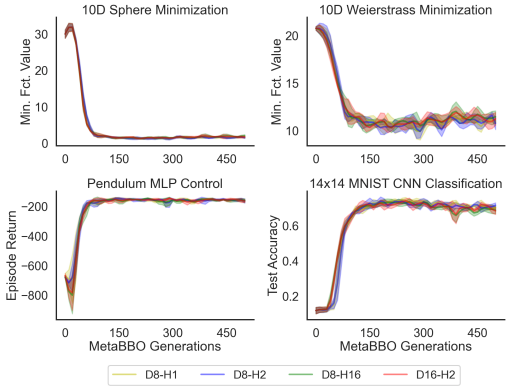
表 4：Neurevolution 超参数（图 7、10、11）。

MNIST 扫描：$\rho \in \{0.0, 0.25, 0.5, 1.0\}$ 和 $\sigma_0 \in \{0.01, 0.025, 0.05\}$。
Brax 扫描：$\rho \in \{0.0, 0.25, 0.5, 1.0\}$ 和 $\sigma_0 \in \{0.025, 0.05, 0.1\}$。
MinAtar 扫描：$\rho \in \{0.0, 0.25, 0.5, 1.0\}$ 和 $\sigma_0 \in \{0.05, 0.075, 0.1\}$。

C 额外的 MetaBBO 结果

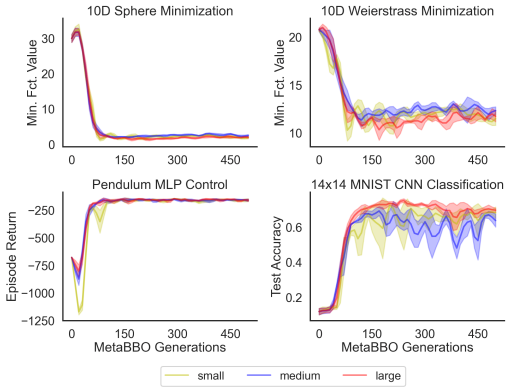
C.1 注意力特征维度&注意力头

图 13：LGA MetaBBO——不同模型尺寸。我们报告 3 次独立运行的均值/1.96ste 区间。



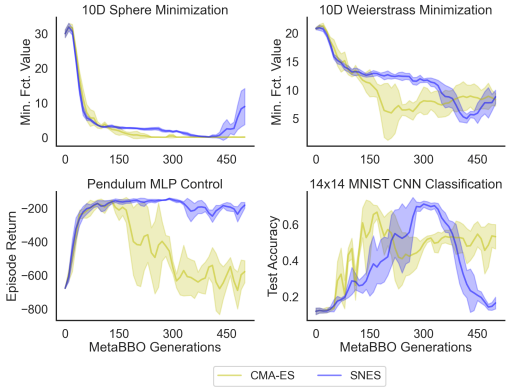
C.2 元训练任务对比

图 14：LGA 黑箱优化——不同的元任务分布。我们报告 3 次独立运行的均值/1.96 倍标准误区间。



C.3 元优化器比较

图 15：LGA MetaBBO——不同的元进化优化方法。我们报告 3 次独立运行的均值/1.96 倍标准误区间。



C.4 元目标函数比较

我们还比较了不同的元目标函数，它们以不同的方式构建元适应度：

minN-minT：在一代内的所有种群评估以及所有代之间取最小值。

minN-finalT：在最后一代中评估的所有种群评估结果中取最小值。

均值N-最小T：计算每一代中各成员的平均性能，并在各代之间最小化这一分数。**均值N-最终T**：计算最后一代中所有成员的平均性能。

$$\left[\left[f(\theta_i|\xi_k)\right]_{i=1}^{M}\right]_{k=1}^{K} = \left[\left[\min_t \left[\min_j \{f(x_{j,t}|\xi_l)\}_{j=1}^{N}\right]_{t=1}^{T} |\theta_i\right]_{i=1}^{M}\right]_{l=1}^{J} \quad \text{(minN-minT)}$$

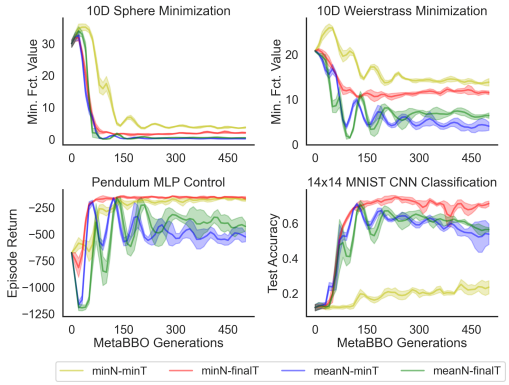
$$\left[\left[f(\theta_i|\xi_k)\right]_{i=1}^{M}\right]_{k=1}^{K} = \left[\left[\min_j \{f(x_{j,T}|\xi_l)\}_{j=1}^{N}|\theta_i\right]_{i=1}^{M}\right]_{l=1}^{J} \quad \text{(minN-finalT)}$$

$$\left[\left[f(\theta_i|\xi_k)\right]_{i=1}^{M}\right]_{k=1}^{K} = \left[\left[\min_t \left[\frac{1}{N}\sum_{j=1}^{N} f(x_{j,T}|\xi_l)|\theta_i\right]_{t=1}^{T} |\theta_i\right]_{i=1}^{M}\right]_{l=1}^{J} \quad \text{(meanN-minT)}$$

$$\left[\left[f(\theta_i|\xi_k)\right]_{i=1}^{M}\right]_{k=1}^{K} = \left[\left[\frac{1}{N}\sum_{j=1}^{N} f(x_{j,T}|\xi_l)|\theta_i\right]_{i=1}^{M}\right]_{l=1}^{J} \quad \text{(meanN-finalT)}$$

对 MetaBBO 比较研究的解读。 LGA 的发现过程对于所考虑的 MetaBBO 设定总体上具有较强的稳健性。较小的注意力模块参数化（单头和 $D_k = 8$）就足以学习强大的遗传算法算子。此外，较小的任务分布（单个带随机偏移/噪声的 Sphere 函数）便能在 BBOB任务上取得优异表现。对于 MNIST分类，则需要更丰富的函数多样性。元优化器和目标函数的选择更为重要。OpenAI-ES（图3）和均值N-最终T表现最佳。

图16：LGA MetaBBO——不同的元目标。我们报告了 3 次独立运行的均值/标准差区间。



D 附加评估结果 D.1 BBOB 详细结果

图 17：BBOB 训练函数评估详情。

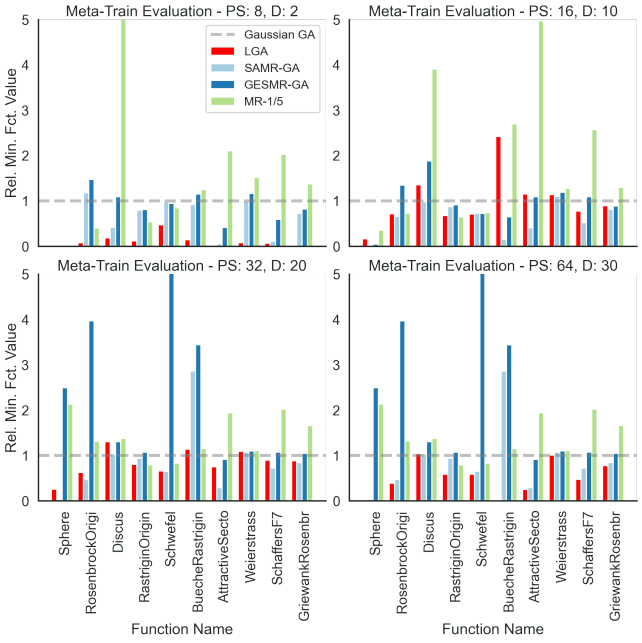
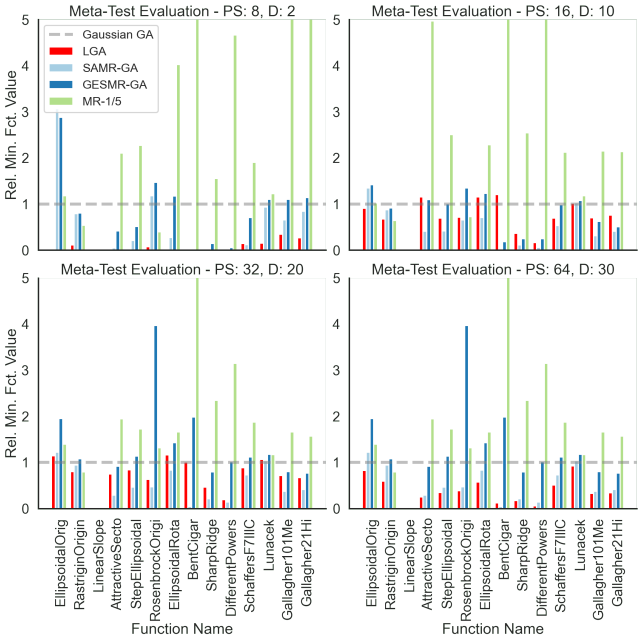


图 18：BBOB 测试函数评估详情。



D.2 神经进化参数鲁棒性

图 19：超参数鲁棒性 - MNIST 任务。

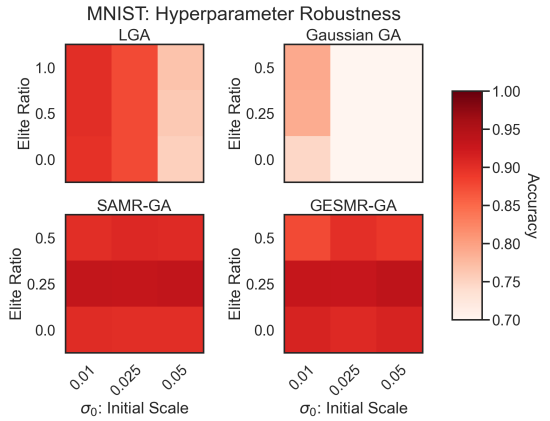


图 20：超参数鲁棒性 - F-MNIST 任务。

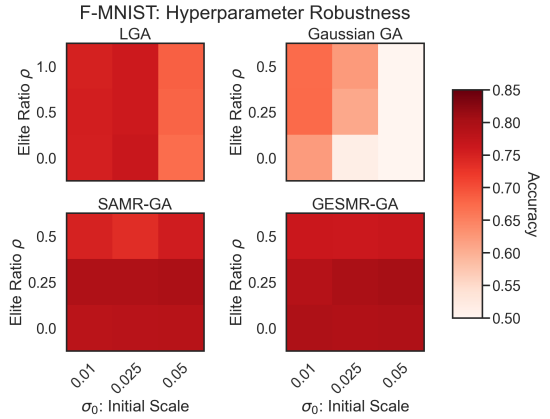


图 21：超参数鲁棒性 - K-MNIST 任务。

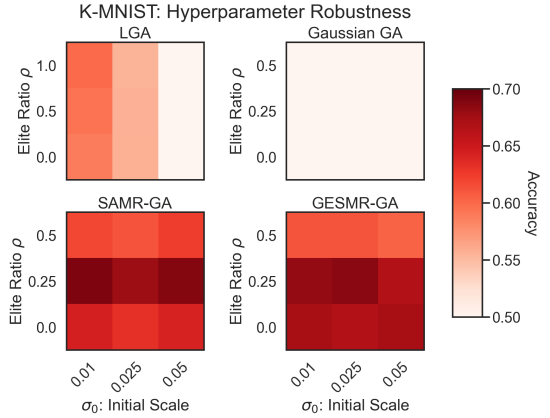


图 22：超参数鲁棒性 - Breakout任务。

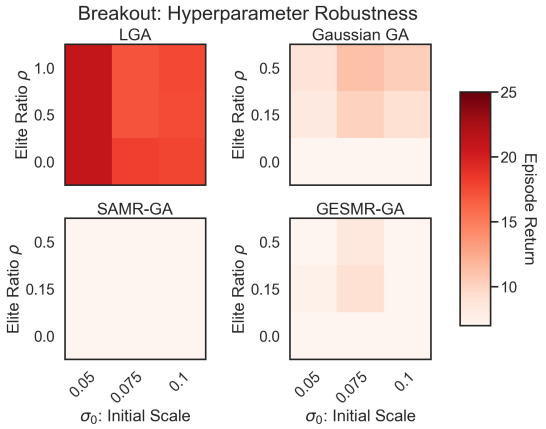


图 23：超参数鲁棒性 - Asterix任务。

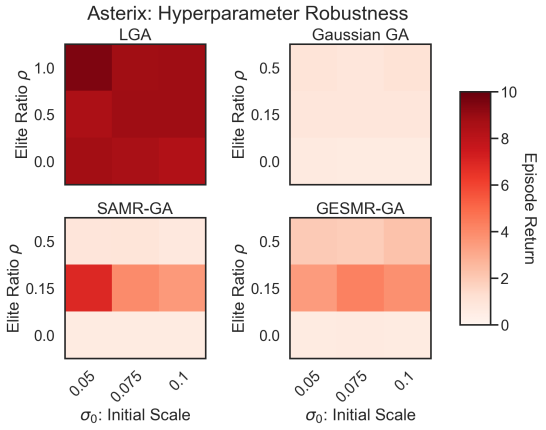


图 24：超参数鲁棒性 - SpaceInvaders任务。

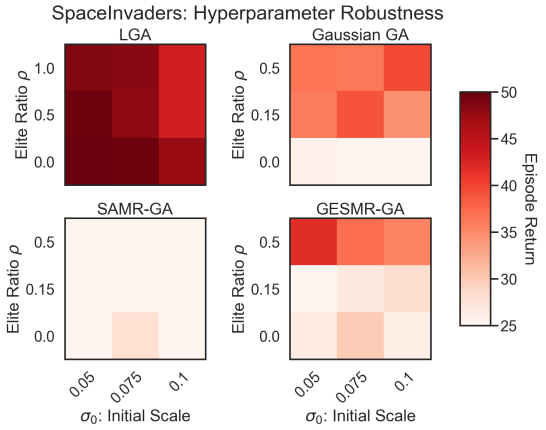


图 25：超参数鲁棒性 - Hopper任务。

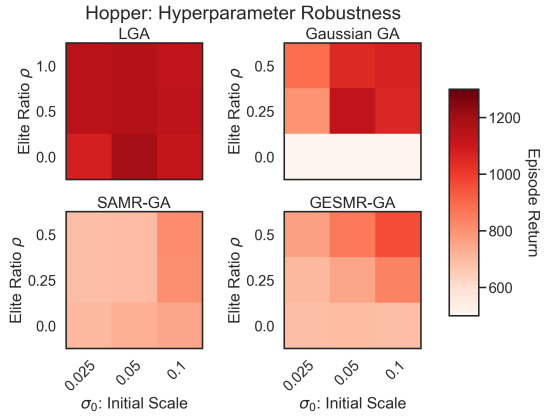


图 26：超参数鲁棒性 - HalfCheetah任务。

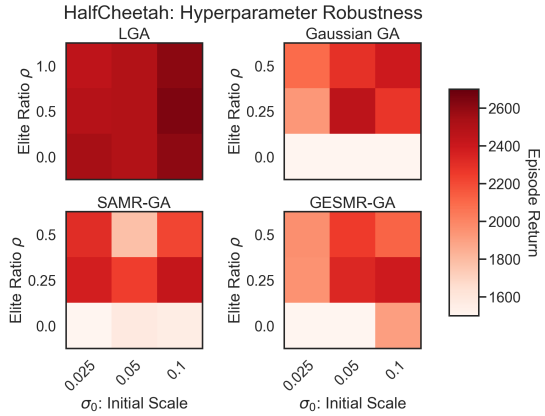
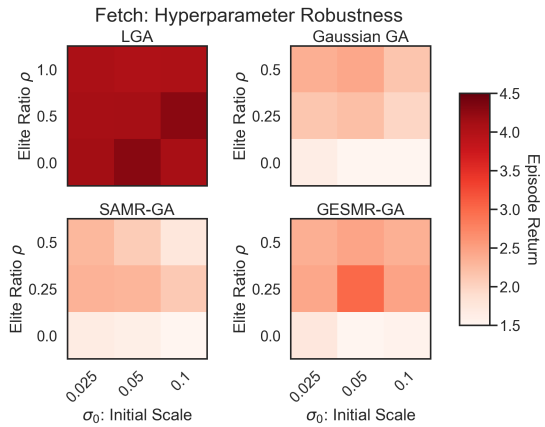


图 27：超参数鲁棒性 - Fetch任务。



E 逆向工程学习到的遗传算法

我们进一步研究了注意力输入特征与学习到的算子之间是否存在简单的关系

图 28：超参数鲁棒性——Walker2d 任务。

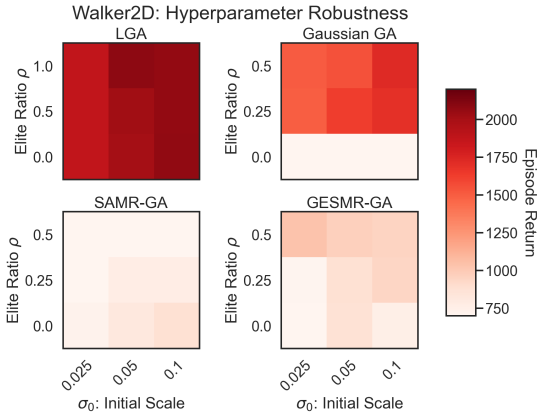


图 29：超参数鲁棒性 - ur5e 任务。

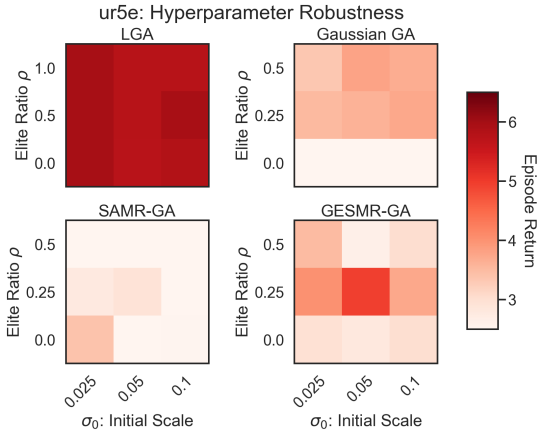
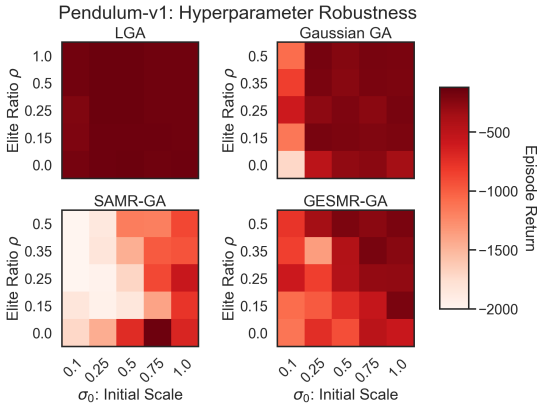


图30：超参数鲁棒性 - Pendulum-v1 任务。



输出。更具体地说，我们探讨了变异率自适应乘数 Δ_σ 和选择逻辑值 m_{ij} 是否可以由子代的归一化适应度特征（z 分数和中心化排名）解释。在图31中，我们绘制了所有特征、逻辑值和变异乘数。

在一次LGA评估运行中完成 2 维 Sphere任务。我们可以观察到，子代的性能与其选择概率之间存在明显的正相关。此外，对于表现良好的解，变异率会降低。这或许为在无需使用可能较为不透明的神经网络模块的情况下，对新发现的遗传算法进行逆向工程提供了可能性。

图31：上：适应度特征与 MRA 之间的线性关系。下：适应度特征与选择逻辑值之间的线性关系。



F 软件、计算资源要求

本项目得益于免费提供的开源软件。包括以下内容：

- NumPy：Harris等人 [16]
- Matplotlib：Hunter [17]
- [43] Seaborn：Waskom JAX：Bradbury等人 [4]
- Evosax: Lange [23]
- Gymnax: Lange [24]
- [46] MinAtar：Young 和 Tian；EvoJAX：Tang [40] 等。
- Brax：Freeman 等。 [11]
- MLE-Infrastructure：Lange [22]

网络检查点及其配套的架构代码已在 evosax中公开。所有实验（包括元训练和评估）均使用 JAX 库实现，以对适应度滚动评估进行并行化。每次 MetaBBO 元训练均在 4个 RTX2080Ti NvidiaGPU 上运行，大约需要 2.5小时。LGA下游BBOB、HPO-B 和gym 任务评估在 CPU集群上运行，使用 2 个CPU核心。每次运行持续 2 至 5 分钟。最后，神经进化任务在独立的 NVIDIA V100S 和 A100 GPU上运行。Brax评估根据控制任务的不同，需要 30 分钟至 1.5 小时。计算机视觉评估实验约需 10分钟，而在 V100S GPU 上，MinAtar 实验持续约 1 小时。