基于共享状态表示和个体策略表示的高效进化强化学习

谢旭东

2023/12/10

摘要

深度强化学习 (Deep Reinforcement Learning, Deep RL) 和进化算法 (Evolutionary Algorithms, EA) 是策略优化的两种主要范式，它们具有不同的学习原理，即基于梯度和无梯度。深度强化学习具有稀疏回报，缺乏有效探索，和对超参数敏感的特性，而 EA 不依赖于梯度信息进行策略优化。将深度强化学习与 EA 相结合，融合其互补优势，设计新的方法是一个很有吸引力的研究方向。然而，现有的结合深度强化学习和 EA 的工作有两个共同的缺点:1)RL 代理和 EA 代理分别学习它们的策略，忽略了有用的公共知识的有效共享;2) 参数级策略优化不能保证 EA 端的语义级行为演化。在本文中，提出了带有双尺度状态表示和策略表示的进化强化学习 (ERL-Re2)，这是一种针对上述两个缺点的新解决方案。ERL-Re2 的关键思想是双尺度表示: 所有 EA 和 RL 策略共享相同的非线性状态表示，同时保持单独的线性策略表示。状态表示传达了所有智能体集体学习的环境的表达性共同特征; 线性策略表示为有效的策略优化提供了有利的空间，可以执行新的行为级交叉和突变操作。此外，线性策略表示允许在策略扩展值函数逼近器 (PeVFA) 的帮助下方便地泛化策略适应度，进一步提高适应度估计的样本效率。

关键词: 强化学习；进化算法；进化强化学习；表征学习；

1 引言

强化学习 (RL) 在机器人控制、游戏 AI，供应链等领域取得了不俗的成就。利用像深度神经网络这样的函数逼近，可以通过可靠的梯度更新的试错来有效地学习策略。然而，众所周知，RL 是不稳定的，勘探能力差，并且在梯度信号有噪声和信息量不足的情况下挣扎。相比之下，进化算法 (EA) 是一类黑盒优化方法，被证明可以与强化学习竞争。EA 通过维持个体种群并通过迭代寻找有利的解决方案来模拟自然进化过程。在每次迭代中，高适应度的个体通过遗传和变异选择产生后代，低适应度的个体被淘汰。与 RL 不同的是，EA 是无梯度的，并且具有以下几个优势: 强大的勘探能力、鲁棒性和稳定的收敛性。尽管有这些优点，但 EA 的一个主要瓶颈是由于总体的迭代评估而导致的低样本效率。当策略空间较大时，这个问题变得更加严峻。由于 EA 和 RL 具有不同且互补的优势，因此很自然的想法是将这两种异构策略优化方法结合起来，设计更好的策略优化算法。近年来，在这个方向上做出了许多努力 [8] [10] [20]。一个代表性的作品是 ERL [8] ，它结合了遗传算法 (GA) [12] 和 DDPG [13]。

ERL 同时维持一个进化种群和一个 RL 代理。然而，现在观察到大多数现有方法很少突破其 EA 或 RL 组件的性能上限 (例如，MuJoCo 上的 Swimmer 和 Humanoid 分别由 EA 和 RL 主导)。这表明 EA 和 RL 的优势没有充分混合。可以将此归因于两个主要缺点：首先，EA 和 RL 的每个代理单独学习其策略。个体学习到的状态表示不可避免地会冗余而专业化 [4]，从而减慢了学习速度，限制了收敛性能。其次，典型的进化变异发生在参数层面 (如网络权重)。它不能保证语义层面的进化，并可能导致策略崩溃 [1]。本文的课程论文复现工作主要有以下几个创新点：1 基于双尺度表示的概念，提出了一种集成 EA 和 RL 的新方法.2 设计了具有明确遗传语义的行为级交叉和突变.

2 相关工作

2.1 进化强化学习

最近，一个新兴的研究领域正在从不同角度整合 EA 和 RL 的优势，设计新的方法。[18] 例如，结合 EA 和 RL 进行有效的策略优化， [9] [2]。利用 EA 逼近连续动作空间中的贪心动作选择 [3]。基于种群的 RL 超参数调谐。 [11] [15]。以及可解释的 RL 策略的遗传编程 [7]。在这项工作中，作者重点将 EA 和 RL 结合起来进行有效的策略优化，ERL [9] 首先提出了一种混合框架，其中 DDPG 代理与遗传群体一起训练。RL 代理不断从种群提供的经验中获益，而种群定期包含 RL 代理的副本。与此同时 CEM-RL [14] 结合了 CEM 和 TD3 的方法，这种方法特别的是，TD3 的 critic 函数用于为 CEM 种群中一半的个体提供更新梯度。后来，ERL 成为了一个流行的框架，在此基础上进行了许多改进。CERL [14] 将单个 RL 代理扩展为具有不同超参数设置的多个 RL 代理，以更好利用 RL 的那一方 [16]。 [5] 通过策略蒸馏和策略梯度算法分别设计了基于梯度的交叉和变异。 [2] 设计了 q 过滤蒸馏交叉和 Proximal 突变，以减轻常规遗传算子在参数水平上造成的策略崩溃。所有这些工作都没有在代理之间共享，并且 EA 和 RL 的每个代理都学习其自己的状态表示，这是低效的。相比之下，作者的方法 ERL-Re2 利用了一个表达丰富的状态表示函数，该函数由所有代理共享和学习。这些工作的另一个共同点是，演化变异是在参数级别（即策略网络）施加的。尽管存在 GPO 和 PDERL，由于策略参数的非线性特性，对策略行为的遗传算法操作的语义不能保证。在 ERL-Re2 中，作者提出了具有线性策略表示的行为级别交叉和变异算子，这些算子具有明确的语义。

3 本文方法

3.1 基于表征的进化强化学习

之前将进化算法（EA）和强化学习（RL）结合用于策略优化的算法，主要遵循图1所示，左侧所示的 ERL 交互架构，其中种群的策略提供各种经验供 RL 训练使用，而 RL 方面将其策略注入种群以参与迭代演化。然而，存在两个重要问题：1）每个代理保持独立的非线性策略并在参数空间中进行搜索，这是低效的，因为每个代理必须独立且反复地学习共同且有用的知识。2）参数级别的扰动可能导致灾难性失败，使参数级别的演化异常不稳定。为了解决前述的问题，作者提出了基于双尺度表示的策略构建方法，该方法用于维护和优化 EA 种群和 RL 代理。策略构建方法如图1右侧所示。具体而言，EA 和 RL 代理的策略都由共享的非

线性状态表示，$z_t = Z_\phi(s_t) \in \mathbb{R}^d$(给定状态 st) 和一个单独的线性策略表示 $W \in \mathbb{R}^{(d+1)\times|\mathcal{A}|}$ 组成。agent i 结合共享状态表示和策略表示进行决策:

$$\pi_i(s_t) = \text{act}(Z_\phi(s_t)^\top W_{i,[1:d]} + W_{i,[d+1]}) \in \mathbb{R}^{|\mathcal{A}|}$$

其中 $W_{[m(:n)]}$ 代表矩阵 W 的切片，该切片由行 m 到 n 组成，act(·) 则是代表某个激活函数，如 tanh 等。同时我们还用 $\mathbb{P} = \{W_1, W_2, ..., W_n\}$ 来表示 EA 种群，用 W_{rl} 来表示 RL 代理。直观地，作者期望共享状态表示 Z_ϕ 对学习过程中遇到的所有可能的策略都有用。它应该包含环境中与决策相关的一般特征，比如普遍的特征，而不是仅仅特定于任何单一的策略。通过共享状态表示 Z_ϕ，它不需要每个代理学习如何独立地表示状态。因此，通过与 EA 种群和 RL 代理以集体方式学习，可以实现更高的效率和更具表现力的状态表示。由于 Z_ϕ 负责表达能力，每个个体策略表示可以具有直观的线性形式，这种形式易于优化和评估 (使用 PeVFA)。作者的这项工作实际上是将端到端的强化学习转化成了增加了一个状态表征的强化学习，状态表征是由一个三层的线性网络加一个激活函数组成，且种群的每个个体以及 RL 代理的行为决策都要接受这个状态表征网络的输出来进行决策。

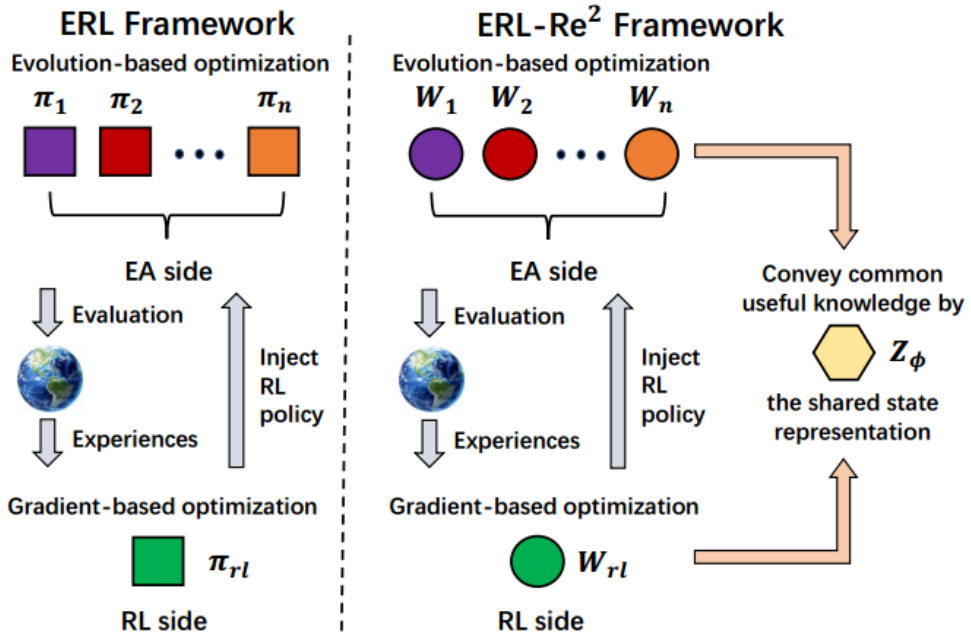

图 1. ERL 和 ERL-RE^2

3.2 新的交叉和变异算子

在遗传进化过程中，涉及到两个遗传算子: 交叉和突变。典型的 k 点交叉和高斯突变直接作用于参数水平，很容易导致决策的大幅波动，甚至导致策略崩溃。这是因为策略参数对行为是高度非线性的，这样的参数级运算符很难保证其自然语义。回想一下矩阵形式的线性策略表示 W。每一行 W[i] 决定了行动的第 i 个维度，因此更改 W[i] 不会影响其他维度的行为。基于更清晰的语义粒度，作者提出了新的行为级交叉和突变:

$$\begin{aligned}(W_{c_1}, W_{c_2}) &= (W_{p_1} \otimes_{d_1} W_{p_2}, W_{p_2} \otimes_{d_2} W_{p_1}) = b\text{-Crossover}(W_{p_1}, W_{p_2}),\\ W_{m_1} &= W_{p_1} \otimes_{\hat{d}_1} P_1 = b\text{-Mutation}(W_{p_1}),\end{aligned} \quad (1)$$

其中 $d_1, d_2, \hat{d}_1$ 是所有维度指标的随机抽样子集，P_1 是随机生成的扰动矩阵，$A \otimes_d B$ 意为 A 在 d 的对应行被 B 取代。通过对特定行为维度施加变化，同时不会对其他行为维度产生干扰，这两个算子 (即 b-Crossover 和 b-Mutation) 可以更有效和稳定，并且在遗传语义意义上提供更多信息。

3.3 PevFA 和 H 步自举

传统的价值函数是在特定策略上定义的。最近，提出了一种新的扩展称为策略扩展值函数逼近器 (PeVFA) [6] 来保留多个策略的值。具体地说，给定策略 π 的某种 χ_π 表示，用 参数化的 PeVFA 额外取 χ_π 作为输入，即 $\mathbb{Q}_\theta(s, a, \chi_\pi)$。通过显式的策略表示 χ_π，PeVFA 的一个吸引人的特点是策略 (或策略空间) 之间的价值泛化。PeVFA 最初是为了利用策略改进路径上的局部值泛化来改进在线强化学习而提出的。在作者的工作中，作者采用 PeVFA 对 EA 种群进行值估计，这自然很适合 PeVFA 的能力。后来作者提出，EA 群体 P 的进化主要包括:(1) 相互作用与选择;(2) 遗传进化。大多数先前估算适应度的方法是对种群中每个个体 rollout 几个 episode 并计算蒙特卡洛的适应度。作者提出了一个新的基于 PeVFA 的代理适应度。对于种群中每个个体，都有 p 的概率以下式来计算适应度:

$$\hat{f}(W_i) = \sum_{t=0}^{H-1} \gamma^t r_t + \gamma^H \mathbb{Q}_\theta(s_H, \pi_i(s_H), W_i). \tag{2}$$

由于 PeVFA，可以方便地估计代理适应度并有效地降低样本成本。此外，一个关键点是 f(Wi) 不存在并行工作中提出的代理中的 off-policy 偏差，代理中的偏离策略偏差来自于特定 EA 代理与 RL 代理之间的值函数不匹配，以及初始状态分布与重放缓冲之间的不匹配。相比之下，作者的方法通过从 PeVFA 引导并使用部分 rollout 来消除这种不匹配。根据 MC 或者作者使用的 H 步自举，具有高适应度的个体更有可能被选为父母。

3.4 算法框架

先在其中优化 EA 和 RL 代理的单个表示，其中 RL criticQ_ψ 和 PeVFA$\mathbb{Q}_\theta$ 的更新参照:

$$\begin{aligned} \mathcal{L}_{\mathbb{Q}}(\theta) &= \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}, W_i\sim\mathbb{P}} \left[\left(r + \gamma \mathbb{Q}_{\theta'}\left(s', \pi_i(s'), W_i\right) - \mathbb{Q}_\theta\left(s, a, W_i\right) \right)^2 \right], \\ \mathcal{L}_Q(\psi) &= \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[\left(r + \gamma Q_{\psi'}\left(s', \pi'_{rl}(s')\right) - Q_\psi\left(s, a\right) \right)^2 \right], \end{aligned} \tag{3}$$

旨在减少当前 Q 值和未来 Q 值的差距。RL agent 从重放缓冲区 $\mathcal{D}$ 获得的经验学习 RL,agent 的 loss 函数是这样定义的:

$$\mathcal{L}_{\mathrm{RL}}(W_{rl}) = -\mathbb{E}_{s\sim\mathcal{D}} \Big[Q_\psi\left(s, \pi_{rl}(s)\right) \Big] \tag{4}$$

, 设定参照了 DDPG 算法，旨在更新策略使得取得的 Q 值更大。 [19]

在个体表示的尺度上进行优化后，共享状态表示将朝着统一的方向进行优化，这源于所有 EA 和 RL 代理的值函数最大化，通过这种方式，将共享状态表示朝着更优的策略空间方向进行优化，从而实现后续的策略优化:

$$\mathcal{L}_Z(\phi) = -\mathbb{E}_{s\sim\mathcal{D}, \{W_j\}_{j=1}^K\sim\mathbb{P}} \Big[Q_\psi\left(s, \pi_{rl}\left(s\right)\right) + \sum_{j=1}^{K} \mathbb{Q}_\theta\left(s, \pi_j\left(s\right), W_j\right) \Big] \tag{5}$$

整个训练的流程如图2所示. 算法中遗传算法的流程如算法2。

Algorithm 1 ERL-RE^2

Input: EA 总体大小 n，使用 MC 估计的概率 p，部分 rollout 长度 H
Initialize: 重放缓冲区 $\mathcal{D}$，共享状态表示函数 Z_ϕ，RL 代理 W_{rl}，EA 种群 $\mathbb{P} = \{W_1, \cdots, W_n\}$，RL critic$Q_\psi$ 和 PeVFA$\mathbb{Q}_\theta$。(此处省略目标网络)
if Random Number > p **then**
    对种群 $\mathbb{P}$ 中的每个个体 rollout 并使用 H 步自举 ▷ 2
else
    对种群 $\mathbb{P}$ 中的每个个体 rollout 并采用 MC 来估计适应度
end if
在一个 episode 中对 RL 代理 rollout
将 $\mathbb{P}$ 和 W_{rl} 产生的经验存储到重放缓冲区 $\mathcal{D}$ 中
用 $\mathcal{D}$ 的数据来训练 RL critic Q_ψ 和 PeVFA $\mathbb{Q}_\theta$ ▷ 3
在行为水平上对 $\mathbb{P} = \{W_1, \cdots, W_n\}$ 进行遗传操作 (即选择、交叉和突变) ▷ 1
更新 RL 代理 W_{rl} 的参数 RL criticQ_ψ ▷ 4
更新状态共享表征，朝 $\mathbb{Q}_\theta$ 和 Q_ψ 的和最大的方向更新 Z_ϕ ▷ 5

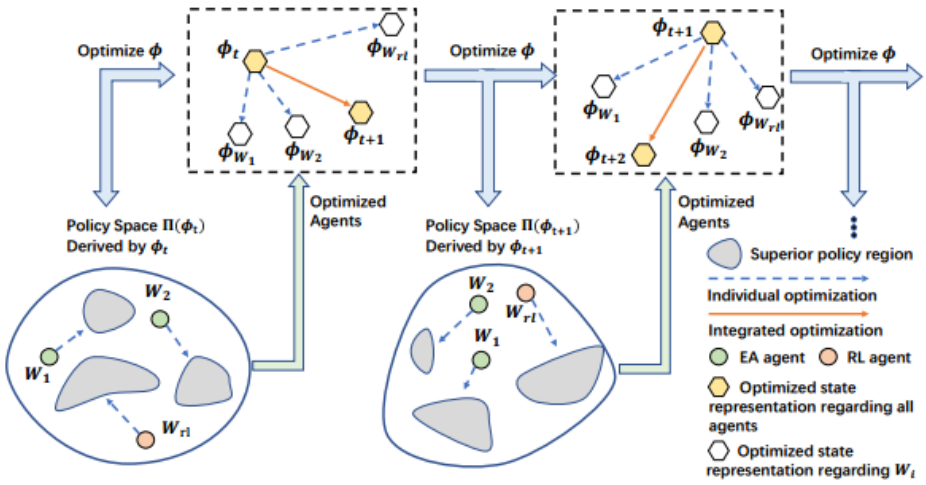

图 2. 训练流程

Algorithm 2 ERL-Re2 中的遗传算法

Input: EA 种群大小 n，概率 α 和 β

Output: 共享状态表示函数 $Z\phi$，EA 种群 $P = \{W_1, \ldots, W_n\}$

初始化：共享状态表示函数 $Z\phi$，EA 种群 $P = \{W_1, \ldots, W_n\}$
repeat
    # 获取适应度
    使用 $Z\phi$ 进行 EA 策略的 rollout，得到适应度
    # 执行选择算子
    根据适应度对种群进行排名，并选择最好的一些（在 ERL-Re2 中为 1 个）策略作为精英 E
    随机选择 3 个不是精英的策略，并保留最好的策略作为获胜者，重复此过程以收集一组获胜者 Win
    将未被选为精英和获胜者的策略收集为抛弃者 Dis
    # 执行交叉运算符 b-Crossover(W_{w1}, W_{w2})
    while Dis 不为空 **do**
        从 Dis 中选择两个抛弃者 W_{di}, W_{dj}
        从 E 中随机选择 W_i 并克隆 W_i 作为一个父母 W_{p1}
        从 Win 中随机选择 W_j 并克隆 W_j 作为另一个父母 W_{p2}
        for 每个动作维度的索引 ai **do**
            if 随机数 < 0.5 **then**
                $W_{p1} = W_{p1} \otimes aiW_{p2}$
            else
                $W_{p2} = W_{p2} \otimes aiW_{p1}$
            end if
        end for
        使用 W_{p1} 和 W_{p2} 替换 W_{di} 和 W_{dj}，并从 Dis 中删除 W_{di} 和 W_{dj}
    end while
    # 执行变异算子 b-Mutation(W_{pi})
    for 所有非精英代理 W **do**
        if 随机数 < 0.9 **then**
            for 每个动作维度的索引 ai **do**
                if 随机数 $< \alpha$ **then**
                    对 $W[ai]$ 中随机选择的 β 个参数添加高斯扰动
                end if
            end for
        end if
    end for
until 达到最大训练步数

4 复现细节

4.1 主要工作

本篇论文是开源代码的，源代码可以在 https://github.com/yeshenpy/ERL-Re2 上找到。本次复现的工作主要如下：1. 复现了原论文在 MUJOCO 上 Humanoid-v2 和 Ant-v2 的环境的结果。2. 修改了论文里面一些小 bug，如在评估进化算子对种群作用的影响（输出父代和子代的 fitness）这一段,evaluate 函数缺少了参数 state_embedding。3. 将 PDERL 算法 [2] 中的 q-蒸馏进化算子用于此论文的进化框架，并对比测试了最后的 reward 以及算子应用前后种群适应度的改变。

Listing 1: 算子评估

```
def crossover_inplace(self, gene1: GeneticAgent, gene2: GeneticAgent):
    # 评估父代
    trials = 5
    if self.args.opstat and self.stats.should_log():
        test_score_p1 = 0
        for eval in range(trials):
            episode = self.evaluate(gene1, self.state_embedding,
            is_render=False, is_action_noise=False, store_transition=False)
            test_score_p1 += episode['reward']
        test_score_p1 /= trials

        test_score_p2 = 0
        for eval in range(trials):
            episode = self.evaluate(gene2, self.state_embedding,
            is_render=False, is_action_noise=False, store_transition=False)
            test_score_p2 += episode['reward']
        test_score_p2 /= trials
```

4.2 实验环境搭建

本次复现主要在 MUJOCO 的连续控制环境上进行复现。MUJOCO 是一个用于物理仿真的物理引擎和运动动力学库，提供了一个高效的物理引擎，用于模拟多关节动力学系统的运动。这些系统可以包括机器人、生物学模型以及其他复杂的多体动力学系统。本次复现用的是 MUJOCO210 版本，环境仿真的效果如图3。此外，安装了 mujoco-py 和 gym 来提供对 MUJOCO 的 python 接口，版本的不兼容会导致各种 error，经过多次对比发现 MUJOCO210 和 mujoco-py==2.1.2.14 和 gym==0.21.0 可兼容的，并且可以运行大部分的强化学习程序。此外，本次复现中深度学习的环境是用的 pytorch，绘图的工具用的 matplotlib 工具。

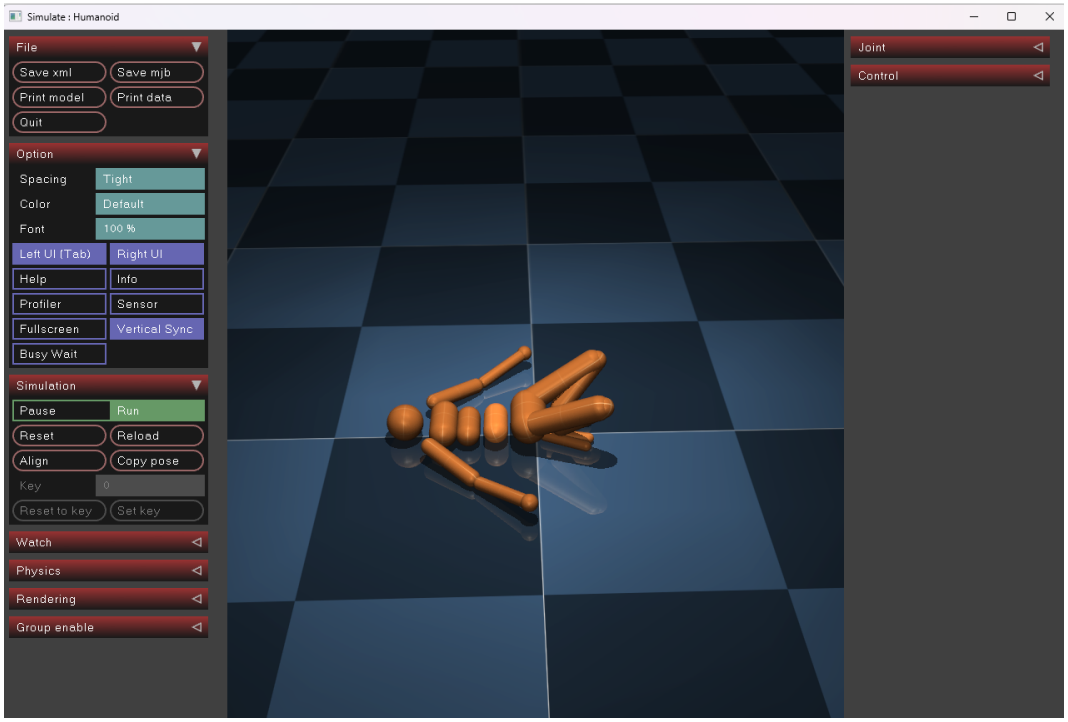

图 3. MUJOCO210

5 实验结果分析

5.1 论文结果复现

本复现工作首先在 HalfCheetah-v2 环境中对 seed=2，seed=3，seed=7 三个随机种子进行了测试，测试的超参数设置与原论文一致，H 步自举概率 p：0.3 H：200 种群大小 n：5 突变率 ： 1.0。实验结果如图4

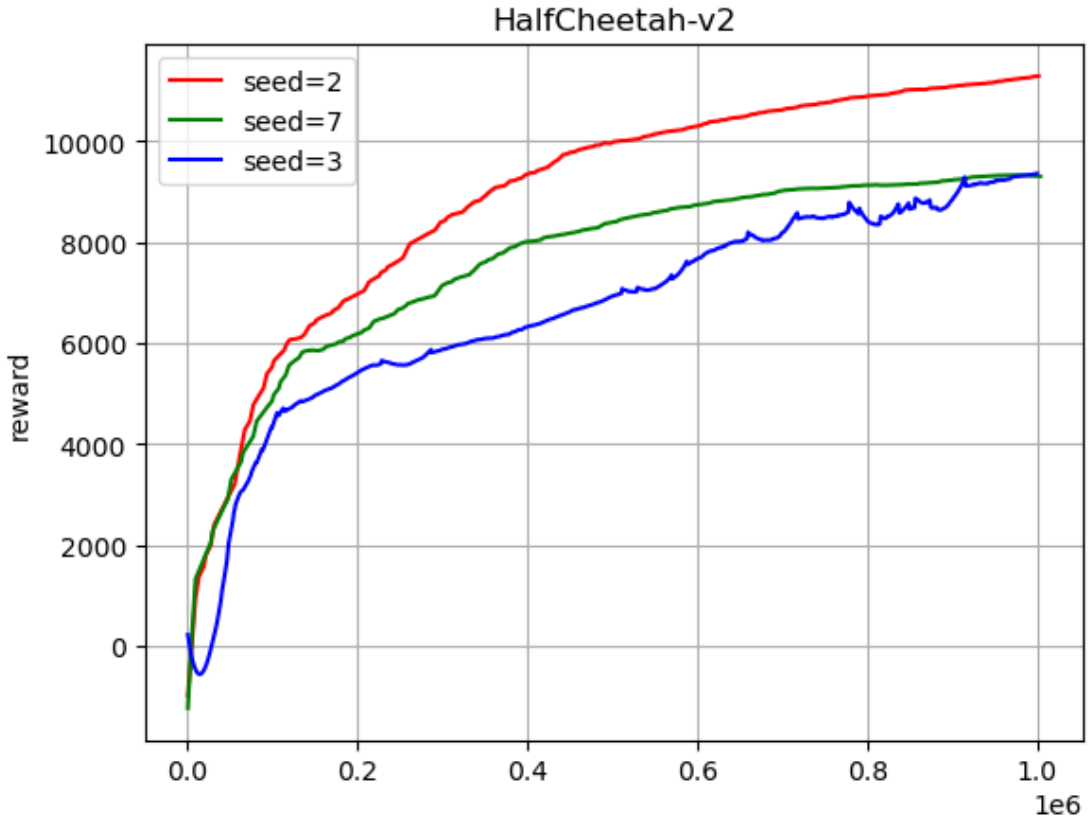

图 4. HalfCheetah-v2 中实验结果示意

随后用差值函数法对三个随机种子的结果取了平均值：5

随后在 Ant-v2 环境中用同样的随机种子进行了测试，测试的超参数设置与原论文一致，H 步自举概率 p：0.5 H：200 种群大小 n：5 突变率 ：0.7。实验结果如图6 随后用差值函数法对三个随机种子的结果取了平均值：7

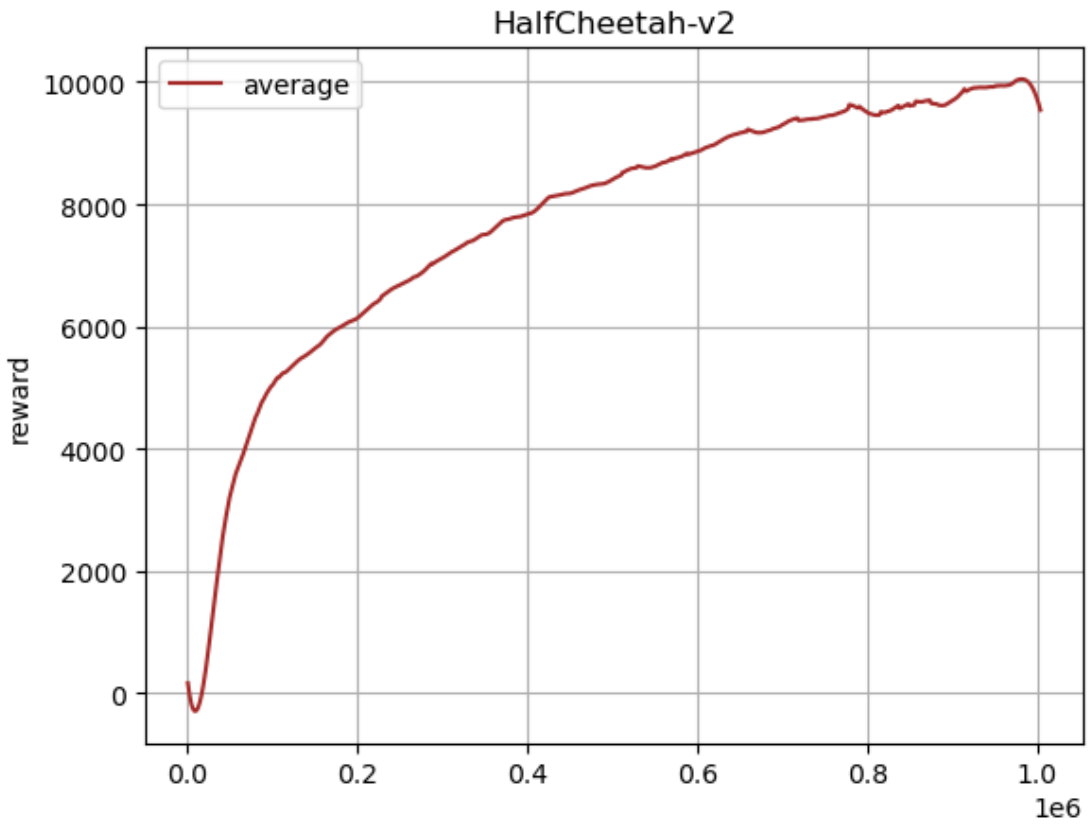

图 5. average HalfCheetah-v2

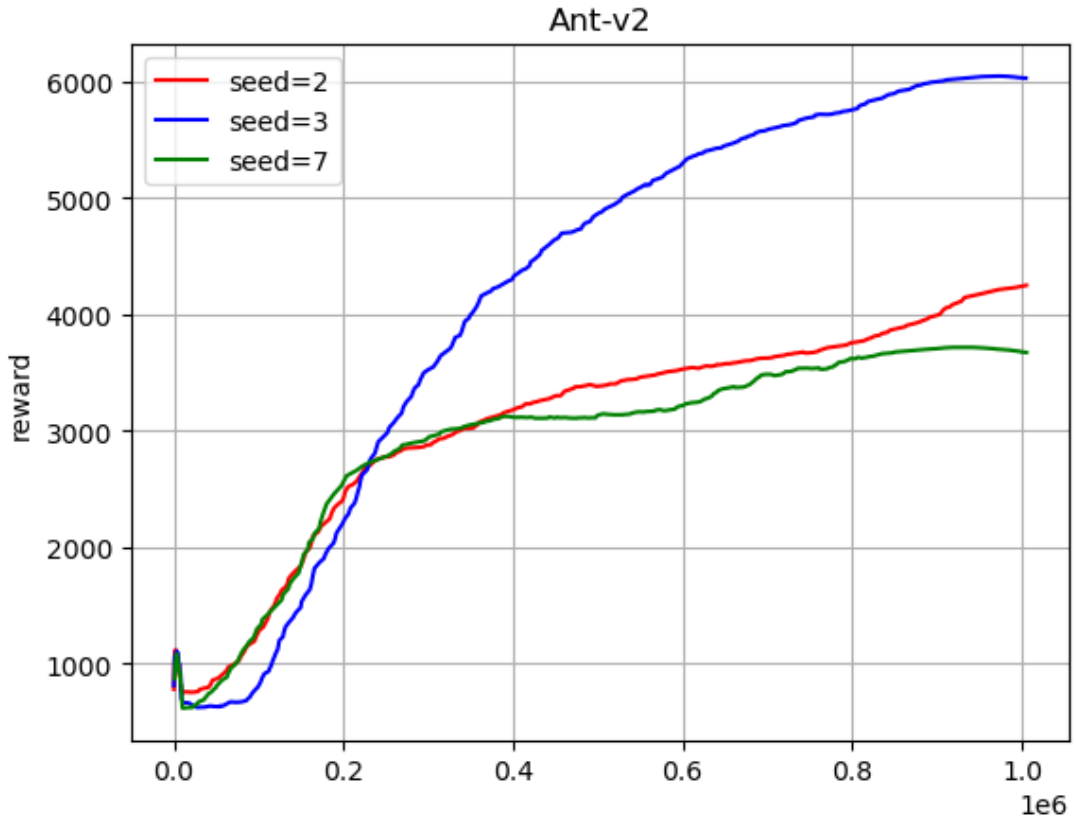

图 6. Ant-v2 中实验结果示意

article algorithm algpseudocode

可以发现最优秀的随机种子是可以达到原文的表现的，如 HalfCheetah-v2 环境中 seed 设为 2 和 Ant-v2 环境中 seed 设为 3。但在 HalfCheetah-v2 环境中 seed=7 和 seed=3 的效果以及 Ant-v2 环境中 seed=2 和 seed=7 的表现并不是很好。综上所述，这个算法稳定性有待提升，且对随机种子的敏感度很高。

5.2 q-蒸馏交叉算子

这个部分主要对 q-蒸馏交叉算子和本论文框架结合，并与原论文效果进行比较，随后对比了 q-蒸馏交叉算子和本论文的交叉算子对亲本适应度的改变。q-蒸馏交叉算子是在 PDERL [2] 中提出的安全交叉算子如图8，通过蒸馏得到的策略的适应度一般至少与亲本的适应度一样好，在复现的过程中，我猜测算法的效果不稳定可能是交叉算子引起的对亲本行为的灾难性遗忘，尝试使用 q-蒸馏交叉算子来尽可能保留亲本的行为。如图9，我对 seed=2，3 和 7 三个种子进行了用 distil 算子代替后的 ERL-RE2 框架的实验，其他超参数设置和原论文相同，并在使用了插值函数后取了平均值然后与原论文比较，结果如图9。可以看到和原论文的效果差不多，据此得出结论，原论文的效果不稳定并不仅仅是遗传算子的原因。 我还对两种算子对亲本的影响进行了比较，在 HalfCheetah-v2 环境中随机选择的八对双亲绘制了这个度量。每组柱状图给出了双亲的适应度和两种交叉得到的策略。所有这些值都由第一个亲本的适应度归一化。如图10所示，spring1 和 spring2 为文中基于行为的交叉得到的两个后代，distil 为蒸馏交叉得到的后代。我们可以知道：1. 蒸馏交叉得到的后代适应度一般不会比任意双亲都差，是很好

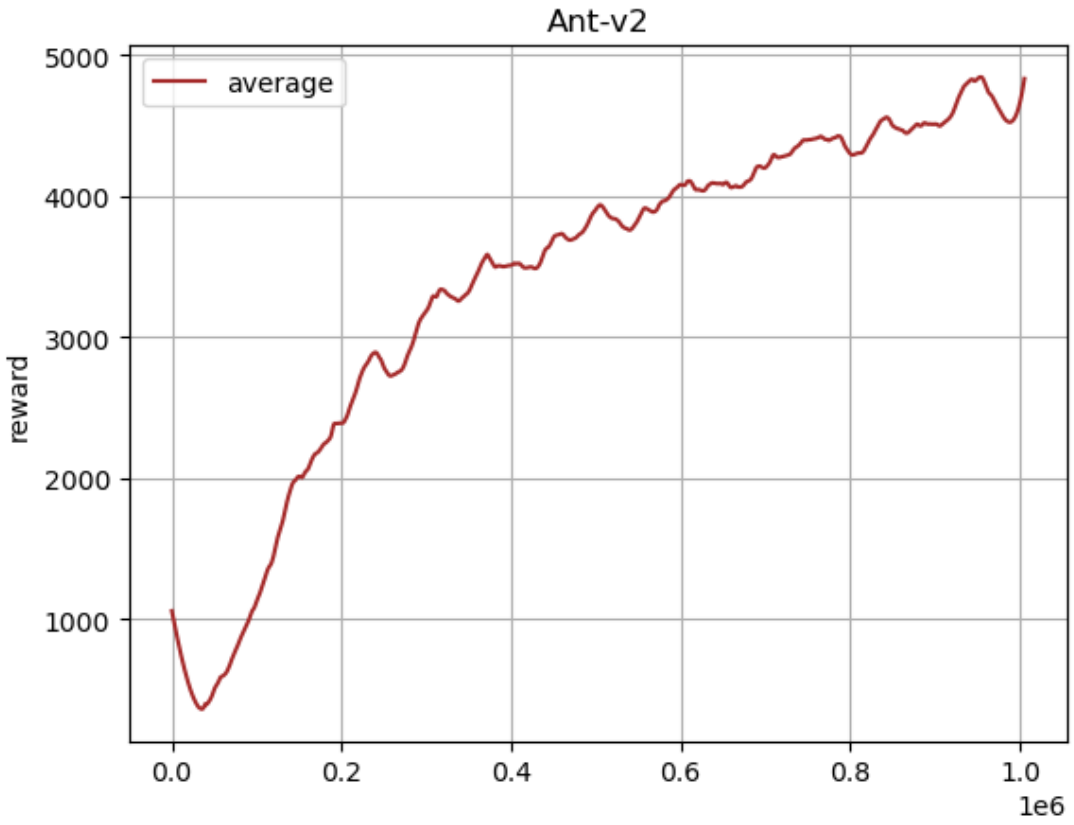

图 7. average Ant-v2

Algorithm 1: Distillation Crossover

Input : Parent policies μ_x, μ_y with memory R_x, R_y
Output: Child policy μ_z with memory empty R_z

Add latest $\frac{\kappa}{2}$ transitions from R_x to R_z
Add latest $\frac{\kappa}{2}$ transitions from R_y to R_z
Shuffle the transitions in R_z
Randomly initialise the weights $\boldsymbol{\theta}_z$ of μ_z with the weights of one of the parents
for $e \leftarrow 1$ **to** *epochs* **do**
    for $i \leftarrow 1$ **to** κ / N_C **do**
        Sample state batch of size N_C from R_z
        Optimise $\boldsymbol{\theta}_z$ to minimise $L(C)$ using SGD
    end
end

图 8. q-distil

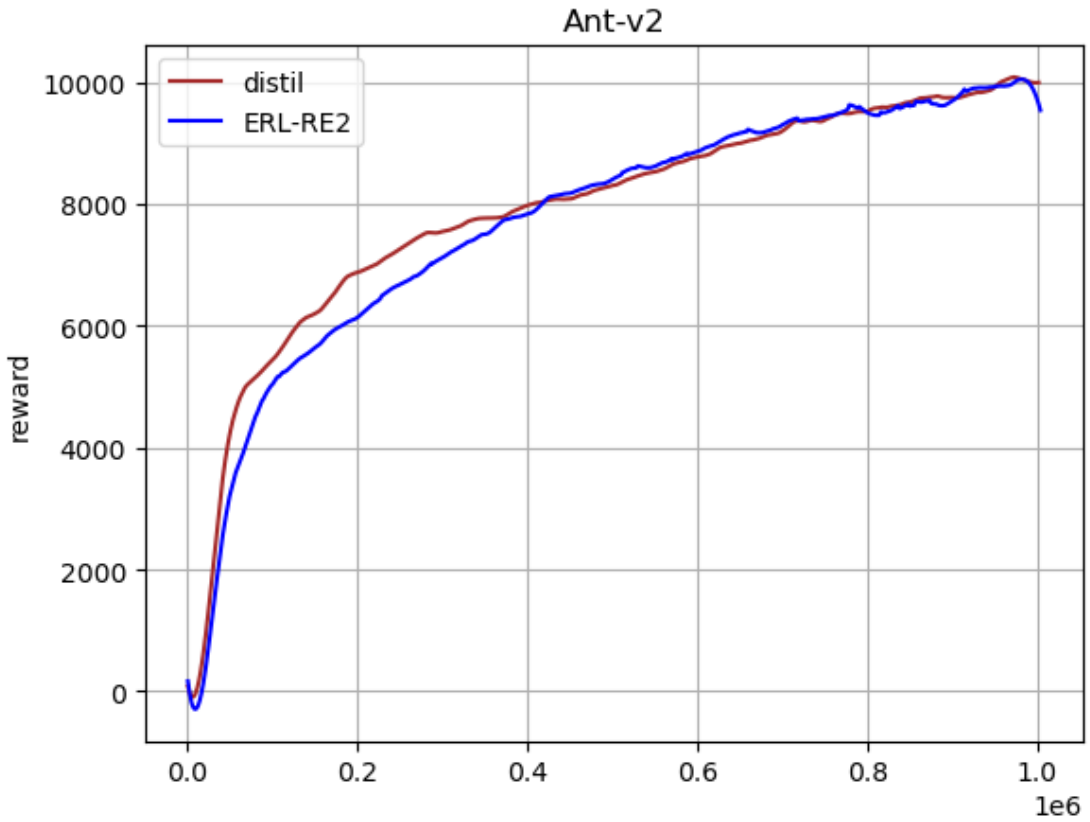
Ant-v2
distil
ERL-RE2
reward
10000
8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6

图 9. 性能比较

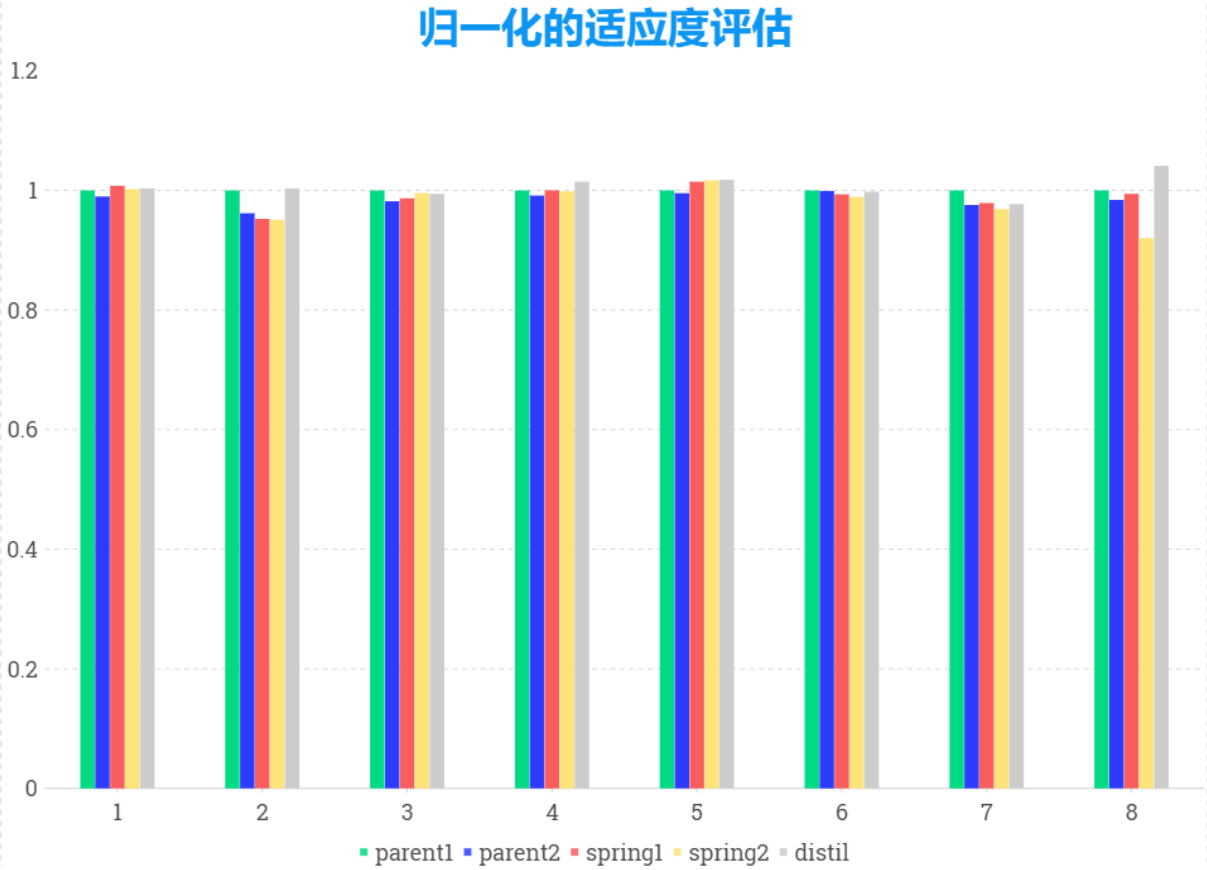

图 10. 蒸馏算子评估

且稳定的交叉算子，但比起原文采用的算子，并没有显示出优越性。2. 交叉算子的影响对亲本来说不大，不是该算法的主要影响因素。

6 可能的不稳定原因

1. 作者并没有对 PeVFA 的使用进行消融实验，PeVFA 的准确度和实用性有待考证。

2. 对于状态表征的奖励更新方式如式3，使得每个个体和 RL 代理的 Q 值和最大，这种更新方式没有和 RL 的梯度训练解耦，这意味着对于 RL 代理来说每轮训练都要改变两次策略，这可能是使得不稳定的原因之一。在近来高效的结合表征学习的强化学习算法中如 TD7 [17]，表征的训练是和 RL 代理解耦的。如下式

$$\mathcal{L}(f,g) := (g(f(s),a) - |f(s')|_x)^2 = \left(z^{sa} - |z^{s'}|_x\right)^2$$

仅仅改变了输入的特征，使得输入的状态和动作具有更高维的表示，而没有影响到 RL 代理的参数。

7 总结与展望

本次复现的论文将表征学习应用于进化强化学习领域，提出了两个新的遗传算子，采用了 PeVFA 对种群适应度进行估计。论文复现主要对两个 MOJOCO 的连续环境进行了测试，并着重研究了交叉算子对算法的影响。复现过程中发现论文的性能很不稳定，并尝试替换了

算子后发现算子对算法性能影响甚微。不是论文性能的主要影响因素。未来的工作中，我将把更优秀的 [17] 的表征学习的方法运用到本算法中，尝试提高算法的样本效率和稳定性。

参考文献

[1] B. Day Bodnar and P. Lió. Proximal distilled evolutionary reinforcement learning. *Proc. AAAI Conf. on Artificial Intelligence*, 2020.

[2] B. Day C. Bodnar. Proximal distilled evolutionary reinforcement learning. *Proc. AAAI Conf. on Artificial Intelligence*, 2020.

[3] B. Day C. Bodnar. Evolutionary action selection for gradient-based policy learning. *arXiv*, 2022.

[4] M. Rowland R. Dadashi J. Quan M. G. Bellemare Dabney, A. Barreto and D. Silver. The value-improvement path: Towards better representations for reinforcement learning. *Proc. AAAI Conf. on Artificial Intelligence*, 2021.

[5] T. Gangwani and J. Peng. Policy optimization by genetic distillation. *Proc. Int. Conf. on Learning Representations*, 2018.

[6] J. Hao C. Chen D. Graves D. Li C. Yu H. Mao W. Liu Y. Yang W. Tao H. Tang, Z. Meng and L. Wang. What about inputting policy in value function: Policy representation and policy-extended value function approximator. *Proc. AAAI Conf. on Artificial Intelligence*, 2022.

[7] D. Hein. Interpretable reinforcement learning policies by evolutionary computation. *PhD thesis*, 2019.

[8] Khadka, K. Tumer Khadka, and K. Tumer. Evolution-guided policy gradient in reinforcement learning. *Proc. Conf. on Neural Information Processing Systems*, 2018.

[9] S. Khadka and K. Tumer. Evolution-guided policy gradient in reinforcement learning. *Proc. Conf. on Neural Information Processing Systems*, 2018.

[10] T. Nassar Z. Dwiel E. Tumer S. Miret Y. Liu Khadka, S. Majumdar and K. Tumer. Collaborative evolutionary reinforcement learning. *ICML*, 2019.

[11] S. Osindero W. M. Czarnecki J. Donahue A. Razavi O. Vinyals T. Green I. Dunning K. Simonyan C. Fernando M. Jaderberg, V. Dalibard and K. Kavukcuoglu. Population based training of neural networks. *arXiv*, 2017.

[12] Mitchell. An introduction to genetic algorithms. *MIT press*, 1988.

[13] A. Pritzel N. Heess T. Erez Y. Tassa D. Silver P. Lillicrap, J. J. Hunt and D. Wierstra. Continuous control with deep reinforcement learning. *Proc. Int. Conf. on Learning Representations*, 2016.

[14] A. Pourchot and O. Sigaud. Cem-rl: combining evolutionary and gradient-based methods for policy search. *Proc. Int. Conf. on Learning Representations*, 2019.

[15] K. W. Pretorius and N. Pillay. Population based reinforcement learning. 2021.

[16] T. Nassar Z. Dwiel E. Tumer S. Miret Y. Liu S. Khadka, S. Majumdar and K. Tumer. Collaborative evolutionary reinforcement learning. in icml, volume 97 of proceedings of machine learning. 2019.

[17] Edward J. Smith Shixiang Shane Gu Doina Precup David Meger Scott Fujimoto, Wei-Di Chang. For sale: State-action representation learning for deep reinforcement learning. *Proc. Conf. on Neural Information Processing Systems*, 2023.

[18] O Sigaud. Combining evolution and deep reinforcement learning for policy search: a survey. *Acm digital library*, 2022.

[19] A. Pritzel N. Heess T. Erez Y. Tassa D. Silver T. P. Lillicrap, J. J. Hunt and D. Wierstra. Continuous control with deep reinforcement learning. *Proc. Int. Conf. on Learning Representations*, 2016.

[20] Y. Chang B. Liang X. Wang Wang, T. Zhang and B. Yuan. A surrogate-assisted controller for expensive evolutionary reinforcement learning. *Information science*, 2022.