Evolutionary Intelligence (2024) 17:1185–1195
https://doi.org/10.1007/s12065-022-00801-3

RESEARCH PAPER

Qualitative differences between evolutionary strategies and reinforcement learning methods for control of autonomous agents

Nicola Milano[1] · Stefano Nolfi[1]

Received: 21 September 2022 / Revised: 1 November 2022 / Accepted: 17 November 2022 / Published online: 7 December 2022

Abstract
In this paper we analyze the qualitative differences between evolutionary strategies and reinforcement learning algorithms by focusing on two popular state-of-the-art algorithms: the OpenAI-ES evolutionary strategy and the Proximal Policy Optimization (PPO) reinforcement learning algorithm – the most similar methods of the two families. We analyze how the methods differ with respect to: (i) general efficacy, (ii) ability to cope with rewards which are sparse in time, (iii) propensity/capacity to discover minimal solutions, (iv) dependency on reward shaping, and (v) ability to cope with variations of the environmental conditions. The analysis of the performance and of the behavioral strategies displayed by the agents trained with the two methods on benchmark problems enable us to demonstrate qualitative differences which were not identified in previous studies, to identify the relative weakness of the two methods, and to propose ways to ameliorate some of those weaknesses. We show that the characteristics of the reward function has a strong impact which vary qualitatively not only for the OpenAI-ES evolutionary algorithm and the PPO reinforcement learning algorithm but also for other reinforcement learning algorithms, thus demonstrating the importance of optimizing the characteristic of the reward function to the algorithm used.

Keywords Evolutionary strategies · Reinforcement learning · Embodied agents · Agent-based simulation

1 Introduction

Evolutionary algorithms (EA) and reinforcement learning algorithms (RLA) represent two well-established techniques for training embodied and situated agents. Both methods permit training agents from scratch on the basis of a fitness or reward function which rates how well the agent is behaving.

In this article we analyze the qualitative difference between the two methods. We briefly discuss the general efficacy, and the ability to cope with rewards which are sparse in time by briefly reviewing the evidence available in the literature. We propose three additional qualitative differences, which were not previously discussed in the literature: the propensity/capacity to discover minimal solutions, the dependency on reward shaping, and the ability to cope with variations of the environmental conditions. For these qualitative differences we provide original experimental evidence.

Clarifying the qualitative differences permits choosing the most suitable algorithm and hyperparameters in a more informed way. Moreover, it permits identifying ways to mitigate the weakness of the algorithm chosen.

Clearly, carrying out a systematic comparison of all existing algorithms on a large set of benchmark problems is out of the scope of a single paper. This also in consideration of the fact that identifying qualitative differences requires in depth studies which go behind the mere comparison of performance measures. Consequently, our analysis does not aim to be exhaustive. The objective of this paper is to illustrate qualitative differences which have not been analyzed in previous studies and to start collecting experimental data which can be used to characterize some of the existing algorithms with respect to those differences.

✉ Nicola Milano
nicola.milano@istc.cnr.it

Stefano Nolfi
stefano.nolfi@istc.cnr.it

[1] Laboratory of Autonomous Robots and Artificial Life, Institute of Cognitive Science and Technologies, National Research Council, Rome, Italy

For our analysis we focus primarily on two popular state-of-the-art algorithms: the OpenAI-ES evolutionary strategy [1] and the Proximal policy optimization reinforcement learning algorithm (PPO) [2]. We start from these algorithms also in consideration of the fact that they are the most similar among the evolutionary and reinforcement learning families. Indeed, they both operate on-policy and they both optimize a gradient on the basis of the Adam stochastic optimizer (other evolutionary algorithms operate without estimating a gradient and without using a stochastic optimizer). However, for some of the analysis we also consider two off-policy reinforcement learning algorithms: the Twin Delayed DDPG (T3D) [3] and the Soft Actor-Critic (SAC) [4]. As we will see, the analysis of those algorithms also reveals interesting qualitative differences among reinforcement learning algorithms.

The OpenAI-ES algorithm [1] belongs to the class of natural evolutionary strategies (NES) [5–8]. Let F denote the objective function acting on parameters θ, the algorithm generates a population, i.e. a set of solutions vector also called genotypes, with a Gaussian distribution over θ parametrized by ψ and tries to maximize the mean objective value $E_\theta \sim p_\psi$ F(θ) over the population by using the Adam stochastic optimizer. During each iteration it takes a gradient step on ψ by using the following estimator:

$$\nabla_\psi E_{\theta\sim p_\psi} F(\theta) = E_{\theta\sim p_\psi}\left[F(\theta)\nabla_\psi \log p_\psi(\theta)\right] \tag{1}$$

The Proximal Policy Optimization Algorithm (PPO, [2]) is an actor-critic on-policy reinforcement learning method [9–12] which uses a parametrized stochastic policy. It operates on the basis of a surrogate gradient that penalizes excessive divergence from the previous policy by clipping the gradient in a proximal trusted zone of the search space.

It updates the policy by using the Adam stochastic optimizer via:

$$\theta_{k+1} = \arg\underset{\theta}{max} \underset{s,a\sim\pi_{\theta_k}}{\mathrm{E}} \left[L\left(s,a,\theta_k,\theta\right)\right] \tag{2}$$

where s are the observation states, a the actions taken, θ are the parameters of the network policy and L is the loss defined as:

$$L\left(s,a,\theta_k,\theta\right) = min\left(\frac{\pi_\theta(a\mid s)}{\pi_{\theta_k}(a\mid s)}A^{\pi_{\theta_k}}(s,a), \mathrm{clip}\left(\frac{\pi_\theta(a\mid s)}{\pi_{\theta_k}(a\mid s)}, 1-\epsilon, +\epsilon\right)A^{\pi_{\theta_k}}(s,a)\right) \tag{3}$$

where $\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$ is the probability ratio of performing those actions with the current and the previous version of the policy, $A^{\pi_{\theta_k}}(s,a)$ is the advantage, and ϵ is the threshold used to clip the gradient.

The Twin Delayed DDPG (TD3, [3]) is an off-policy reinforcement learning method which operates with a deterministic policy combined with two Q-functions. The usage of the smaller Q-value computed by the two Q-functions permit to reduce the problems caused by the overestimation of the Q-values. The algorithm learns the control policy π_θ and two Q-functions Q_{ϕ_1}, Q_{ϕ_2} in an alternate way by mean square error Bellman minimization:

$$y\left(r,s',d\right) = r + \gamma(1-d)\underset{i=1,2}{min} Q_{\phi_{i,\,\mathrm{targ}}}\left(s',a'\left(s'\right)\right) \tag{4}$$

where r is the reward, s' the next state, d a discrete value that indicates if state s' is terminal and γ the discount factor.

Then both Q-policies are learned by regressing to this target:

$$L\left(\phi_i,\mathcal{D}\right) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}}\left[\left(Q_{\phi_i}(s,a) - y\left(r,s',d\right)\right)^2\right], i = 1,2 \tag{5}$$

Moreover, it uses a second policy network which contains a time-delayed version of the parameters of the first network to eliminate the instabilities caused by the need to optimize a target that depends on the same parameters to be optimized.

Like other algorithms which approximate the Q function, it uses an experience reply buffer which contains previous experiences. The policy is learned by maximizing Q_{ϕ_1}.

The Soft Actor-Critic (SAC, [4]) algorithm is similar to the TD3 algorithm. However, it uses a stochastic policy instead than a deterministic policy. Moreover, it determines the next state on the basis of the current policy instead than on the basis of a previous version of the policy.

The policy is trained to maximize a trade-off between the expected return and entropy, a measure of randomness of the policy.

$$\pi^* = \arg\underset{\pi}{max} \underset{\tau\sim\pi}{\mathrm{E}}\left[\sum_{t=0}^{\infty}\gamma^t\left(R\left(s_t,a_t,s_{t+1}\right) + \alpha H\left(\pi\left(\cdot\mid s_t\right)\right)\right)\right] \tag{6}$$

where H is the entropy of the policy computed from its distribution and α the trade-off coefficient.

The experiments reported in this article can be replicated by using evorobotpy2, available from https://github.

com/snolfi/evorobotpy2, and stable baseline available from https://github.com/hill-a/stable-baselines. The modified reward functions described in Sect. 5 are implemented in evorobotpy2 and can be used by specifying the desired version of the problems.

2 General efficacy

Comparing the general efficacy of EA and RLA is far from trivial since the performance depends on the problems considered, on the setting of the hyperparameters, and on the fitness or reward functions used, as we will also illustrate in the following sections.

In any case, overall, the existing comparative studies do not provide evidence indicating a general superiority of one class of methods over the other. Given the limited number of studies which compared the OpenAI-ES and the PPO algorithms we discuss here also the studies carried out with the CMA-ES [13] another state-of-the-art evolutionary algorithm similar to the OpenAI-ES and the TRPO [14], an algorithm similar to the PPO.

The results reported in [15] obtained with the CMA-ES and TRPO indicate a general superiority of the latter over the former. Indeed, the TRPO outperforms the CMA-ES on 11 out of 12 classic control and MuJoCo locomotors problems. On the other hand, this result is biased from the fact that the duration of the training is determined on the basis of the total number of evaluation steps without considering that the computation cost of the TRPO algorithm is higher than that of the CMA-ES algorithm. Evolutionary algorithms tend to be less sample efficient than reinforcement learning methods in general but are less computationally expensive and can benefit more from parallelization. Indeed, in a subsequent work [16] demonstrated that the OpenAI-ES evolutionary algorithm and the TRPO reinforcement learning algorithm achieve similar performance on the MuJoCo locomotors problems and on the Atari problems at equal computational cost. Moreover, they demonstrated that highly parallel implementations of evolutionary algorithms permit generating solutions in a remarkably short period of time.

The dependency of the relative performance on the problem considered can be appreciated in [17] which reports the superiority of one type of algorithm in certain problems and of the other type in the remaining problems.

The impact of the fitness or reward function is discussed in Sect. 5.

Reinforcement learning algorithms scale better to problems requiring the optimization of a large set of parameters than classical evolutionary algorithms. However, modern evolutionary strategies such as the OpenAI-ES algorithm appear to scale much better than classical methods [1].

3 Ability to cope with reward which are sparse in time

Another property which differentiates evolutionary and reinforcement learning algorithms is the ability to cope with rewards which are sparse over time, i.e. situations in which the offset in time between the execution of appropriate actions and the reception of the corresponding reward is high.

EAs do not suffer from sparsity of the reward over time since they operate on the basis of a fitness measure that encodes the sum of the rewards collected during evaluation episodes. RLAs instead struggle with the temporal credit assignment problem when rewards are sparse over time because they operate by estimating the efficacy of specific actions. Temporal difference in RLAs use bootstrapping to better handle this aspect but still struggle with the sparsity of the rewards when the time horizon is long. An example of this effect is reported in the next Section.

We do not analyze this qualitative difference in more detail in this paper since it is well documented in the literature (see for example [18, 19]).

Notice, however, that both EAs and RLAs suffer from problems in which the chance to obtain a positive reward is too small in general terms, irrespectively from the duration of the gap between the execution of appropriate actions and the collection of the reward. Problems of this kind may cause a bootstrap problem, i.e. the impossibility to start progressing caused by the impossibility to differentiate between adaptive and counter-adaptive variations. This problem can be alleviated by adding an intrinsic reward component in the reward function that encourages the robots to perform new behaviors and/or to experience new observations [20–23].

4 Propensity/capacity to discover minimal solutions

Both evolutionary and reinforcement learning algorithms introduce random variations to discover better policies. However, they differ in the way in which variations are introduced. Indeed, the OpenAI-ES evolutionary algorithm, and more generally evolutionary algorithms, introduce variations in the parameters of the policy during the generation of a new population of policies while the PPO reinforcement learning algorithm, and most reinforcement learning algorithms, introduce variations in the actions performed by the policy in each step (for alternative reinforcement learning methods which introduce variations in the parameters of the policy see [24, 25]).

Moreover, in the PPO algorithm the distribution of variations is encoded in adapting parameters, is initially high, and usually decreases during the course of the training. In the OpenAI-ES algorithm, instead, the distribution of variation is constant. These differences have important consequences.

A first consequence is that the set of solutions which are available to the two algorithms differ. By using deterministic or almost-deterministic policies (i.e. policies in which the actions are perturbed through the addition of random values selected within a tiny range), the OpenAI-ES algorithm can adopt simple solutions, i.e. solutions which permit to solve the problem on the basis of few simple control rules. These simple solutions cannot be realized through the usage of stochastic policies and consequently cannot be found by the PPO algorithm.

This difference can represent an advantage or a disadvantage depending on the circumstances. Indeed, in some cases simple solutions permit to achieve optimal performance in a clever and effective way. In other cases, simple solutions permit to achieve only sub-optimal performance and drive the learning process toward solutions which are qualitatively different from optimal solutions and which might correspond to local minima. In the former case the OpenAI-ES algorithm outperforms the PPO algorithm. In the latter case the OpenAI-ES algorithm is outperformed by the PPO algorithm.

This difference can be illustrated by analyzing the solutions discovered by the OpenAI-ES and PPO algorithms for the Centipede and Breakout Atari problems. The OpenAI-ES algorithm solves these two problems by moving the player in a specific location and by later keeping the player in that precise position (see Fig. 1 center, Table 1, and Videos 1 and 3). The links to the videos are included in the Supplementary Material). This minimal strategy enables the agent to collect a very high reward in some problems, like the Centipede game, but is suboptimal in other problems like the Breakout game. This strategy is clearly inaccessible for agents trained with the PPO algorithm which are exposed to strong action perturbations. Consequently, the PPO algorithm is

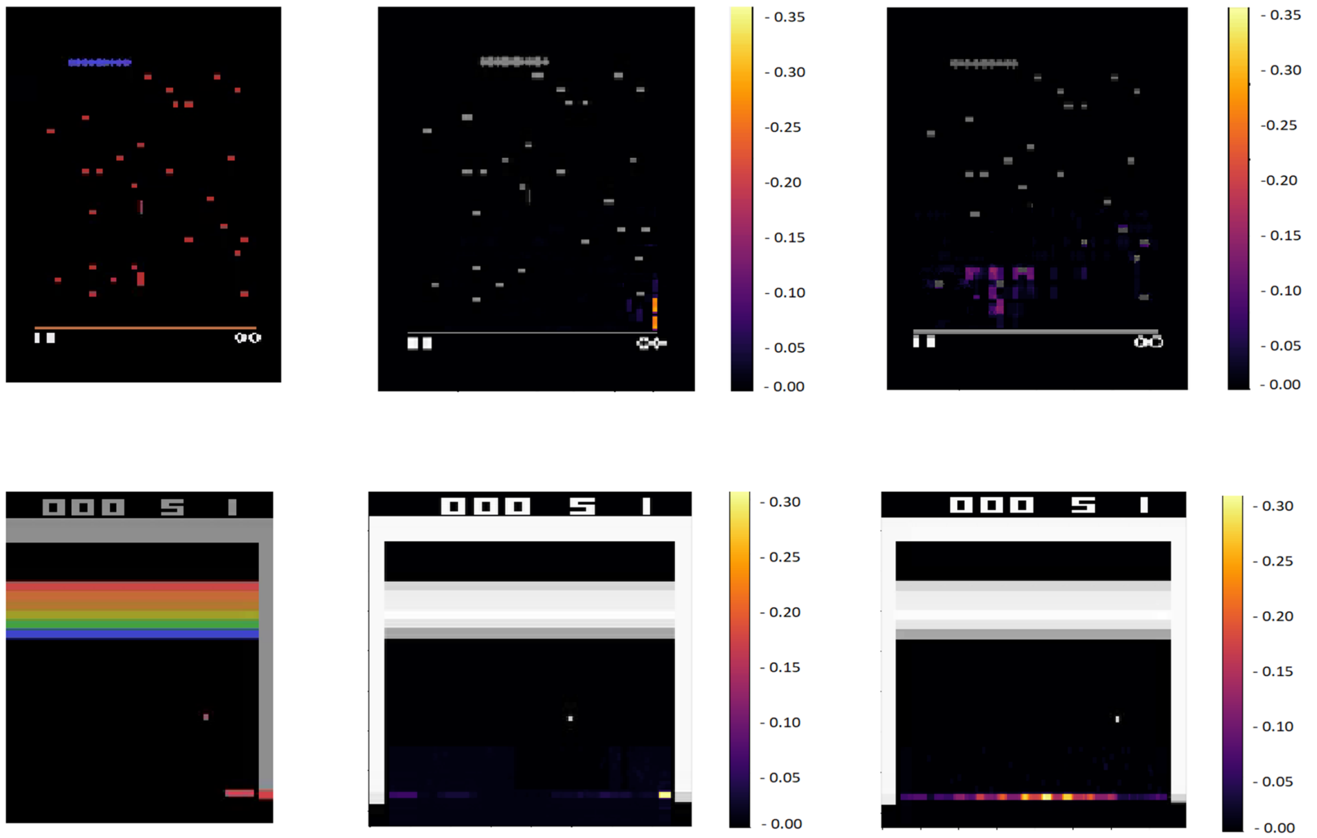

Fig. 1 Positions assumed by a typical player trained on the Centipede and Breakout Atari problems (top and bottom, respectively). The first column shows a screenshot of the game in color. The other columns show in color at the bottom the frequency with which the agent is situated in different locations at the bottom during the game. The second and third columns display the data of the agents trained with the OpenAI-ES and PPO algorithms, respectively, Data obtained by running 5 replications of each experiment. As can be seen, the agents trained with the PPO algorithm move in different locations during the game while the agents trained with the OpenAI-ES algorithm remain in the same location for most of the game. Hyperparameters are reported in the supplementary material

Table 1 Performance obtained on 21 Atari games with the PPO and OpenAI-ES algorithms

Game	PPO	OpenAI-ES
Amidar	370.755 ± 99.09	242.5 ± 43.9
Asterix	3234.0 ± 108.9*	1475.0 ± 43.3
Asteroids	1846.67 ± 176.19	3175.0 ± 258.5
BeamRider	3679.34 ± 377.9*	1320.0 ± 20.7
Boxing	92.395 ± 0.93*	29.5 ± 1.5
Breakout	350.4 ± 88.5*	30 ± 8
Gravitar	727.375 ± 74.0	2300.0 ± 0.0*
Krull	7431.93 ± 206.50	8773.5 ± 1879.5
Phoenix	10,124.67 ± 1199.5*	4860.0 ± 411.9
Qbert	5487.75 ± 1306.04*	3183.3 ± 1297.8
Solaris	2825.2 ± 98.5	2378.2 ± 102.5
Atlantis	1,819,341.3 ± 101,474.7*	90,650.0 ± 2250.0
Centipede	4182.05 ± 41.25	26,046.5 ± 2290.5*
FishingDerby	− 18.36 ± 20.80*	-51.0 ± 4.0
Pong	19.99 ± 0.34	20.3 ± 0.4
Skiing	− 11,319.67 ± 3322.67	-11,064.0 ± 4121.7
SpaceInvaders	608.30 ± 14.67	1218.3 ± 76.6*
Venture	67.66 ± 51.84	1150.0 ± 250.0*
Freeway	21.53 ± 15.23	28.5 ± 5.5*
Kangaroo	1350.0 ± 200.5	1100.0 ± 100.0
Frostbite	292.32 ± 18.10	1267.5 ± 575.0*

Each experiment was replicated 10 times. Each cell indicates the average performance and the standard deviation obtained in 10 replications of the experiment. The asterisks indicate the conditions in which one method produces significantly better performance than the alternative method (Wilcoxon rank sum test p-value < 0.001)

forced to develop more complex strategies which rely on the ability to intercept the ball from different positions (Fig. 1 right, Table 1, and Video 2 and 4). In the case of the Breakout problem, the possibility of the OpenAI-ES algorithm to select a minimal strategy is counter-productive since the strategy is suboptimal and constitutes a local minimum, i.e. prevents the possibilities to later select better strategies. Consequently, the PPO algorithm outperforms the OpenAI-ES algorithm on this problem. In the case of the Centipede problem, instead, the possibility to select this minimal strategy is advantageous since the strategy is more effective than alternative strategies. Consequently, the OpenAI-ES algorithm outperforms the PPO algorithm in this problem.

Table 1 displays the performance obtained with the PPO and the OpenAI-ES algorithms on 21 Atari games. The former outperforms the latter method in 8 games, the latter outperforms the former in 6 games. The performance does not differ statistically in the remaining cases. The games where an algorithm statistically outperforms the other are highlighted with an asterisk. As for the Breakout game, the advantage of the PPO agents in the Asterix, BeamRider Boxing, Phoenix, Atlantis, Fishing Derby and Qbert games is obtained through the development of behavioral strategies which are more elaborated than those displayed by OpenAI-ES agents. Moreover, as for the Centripede game, the advantage of the OpenAI-ES agents on the Gravitar, SpaceInvaders, Frostbite, Freeway, and games is obtained through the development of simple behavioral strategies which result effective in these games and which are not discovered by the PPO agents. The advantage of the OpenAI-ES agents on the Venture games, instead, can be explained by the large temporal offset between the execution of the appropriate action and the collection of the corresponding reward which characterizes this game.

The ability of EAs to discover simple and clever strategies is well-documented in the literature [26, 27]. The presence of a correlation between solutions which are simple and solutions which represent local minima is indicated by few preliminary studies [28, 29]. Moreover, the fact that solutions which are robust to variations are often more general than solutions which are fragile to variation is supported by several studies [28–30]. The fact that some of the strategies available to evolutionary algorithms are not available to reinforcement learning algorithms was not discussed before, as far as we know, and consequently constitutes an original contribution.

The qualitative difference between the OpenAI-ES and the PPO algorithms can be reduced by perturbing the actions also in the case of the former algorithm. This technique is usually used to evolve solutions which are robust to environmental variations [31] and/or to evolve solutions which can cross the reality gap, i.e. which keep operating effectively once they are moved from the simulated to the real environment [32].

Figure 2 displays the performance achieved by training deterministic policies, stochastic policies subjected to minor perturbations and stochastic policies subject to much larger perturbations with the OpenAI-ES algorithm on three different PyBullet continuous problem: walker2D, Hopper and Humanoid [33, 34]. In the former cases the perturbations are generated with the addition of Gaussian random values with an average of 0.0 and standard deviation of 0.01. In the latter case, the distribution of perturbation for each action value is parametric and is initially set to values close to 1.0 (as in the case of the policies used by the PPO algorithm). As can be seen, the addition of small perturbations to the action vector permits to generate solutions which are robust with respect to this form of perturbation. Moreover, it leads to better performance in part of the problems considered. On the other hand, the utilization of the parametric Gaussian policies normally used by the PPO leads to significantly worse performance in all cases.

Overall, this implies that the OpenAI-ES algorithm tolerates and can even benefit from the addition of small action

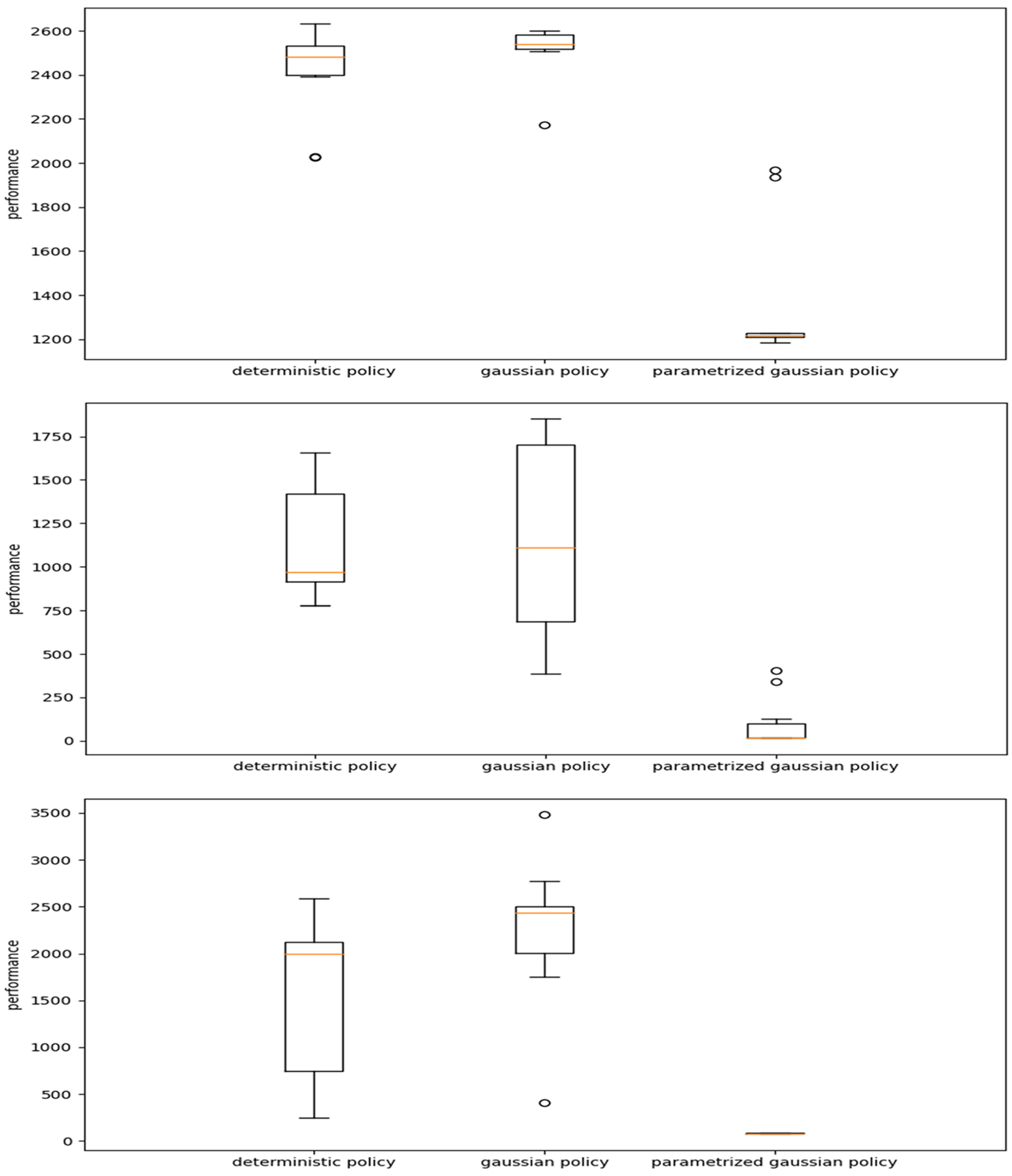

Fig. 2 Maximum performance obtained with the OpenAI-ES algorithm on the Pybullet Hopper (top), Walker2d (center) and Humanoid (bottom) environments. The boxplots display the performance obtained by using a deterministic policy, a Gaussian policy in which actions are perturbed with tiny random values (i.e. random values generated with an average of 0.0 and a distribution of 0.01), and a parametrized diagonal Gaussian policy. The second experimental condition produces significantly better performance than the first experimental condition in the case of the Hopper and Humanoid problems (Wilcoxon non parametric test p-value < 0.001) and equally good performance in the case of the Walker2D problem (Wilcoxon non parametric test p-value > 0.05). The third experimental conditions produce worse performance than the first experimental condition in all cases (Wilcoxon non parametric test p-value < 0.001). Hyperparameters are reported in the supplementary material

perturbations. However, unlike the PPO algorithm, it is unable to tolerate large perturbations.

5 Dependency on reward shaping

Evolutionary and reinforcement learning algorithms also differ with respect to their dependency on reward shaping.

Reward shaping refers to the attempt to facilitate the development of effective solutions by designing reward functions which rate the learning agents not only for their ability to solve the task but also for abilities which are instrumental to the solution of the problem. For example, the development of a walking ability can be obtained by rewarding the agent on the basis of its speed toward the destination only, i.e. on the basis of its ability to walk fast only. Alternatively, it can be obtained by also rewarding the learning agent for the ability to remain upright. This additional reward component (incentive) is intended to favor the development of an ability to stay upright which does not constitute a solution of the problem by itself but which can facilitate the development of the ability to walk. Unfortunately, however, identifying useful incentives can be challenging since the effect of the incentives is hard to predict and since the introduction of incentives can promote the development of non-effective solutions which maximize the incentives without maximizing the primary component of the reward function [35, 36].

Many benchmark problems include incentives in their reward functions. For example, the Pybullet locomotors problems and the Bipedal hardcore problem include a bonus for staying upright. In this section we analyze the effect that the incentive has on agents trained on the Hopper and Walker2d Pybullet problems and on the Bipedal Hardcore problem with the OpenAI-ES algorithm and with the PPO, T3D and SAC reinforcement learning algorithms.

The comparison of the performance obtained with and without the incentive (Figs. 3 and 4, respectively) and the visual inspection of the behavior of the trained agents indicates that the incentive has a strong impact on the results. The impact is positive or negative depending on the problem and on the algorithm used. Overall, the effect of the incentive is positive in settings in which the agents fail to develop a walking ability by being rewarded for the distance travelled only and negative in the other settings.

In the case of the Bipedal Hardcore problem, the agents manage to develop an ability to walk without the incentive with all algorithms. Consequently, the distance travelled by the agents trained with the incentive is lower in all cases.

For the other two problems, the OpenAI-ES algorithm manages to develop a walking behavior without the incentive and fails with the incentive. This confirms that the addition of the incentive is counterproductive in settings in which the agents manage to solve the task also without it (see also [37]). The low performance obtained with the incentive is due to the fact that in this experimental condition the agents develop an ability to avoid falling by staying still which maximizes the reward obtained through the incentive only. The PPO algorithm is unable to develop a walking ability without the incentive (i.e. the trained agents just fall down after few steps) and benefit from the incentive. The TD3 also benefits from the incentive since it fails to develop an

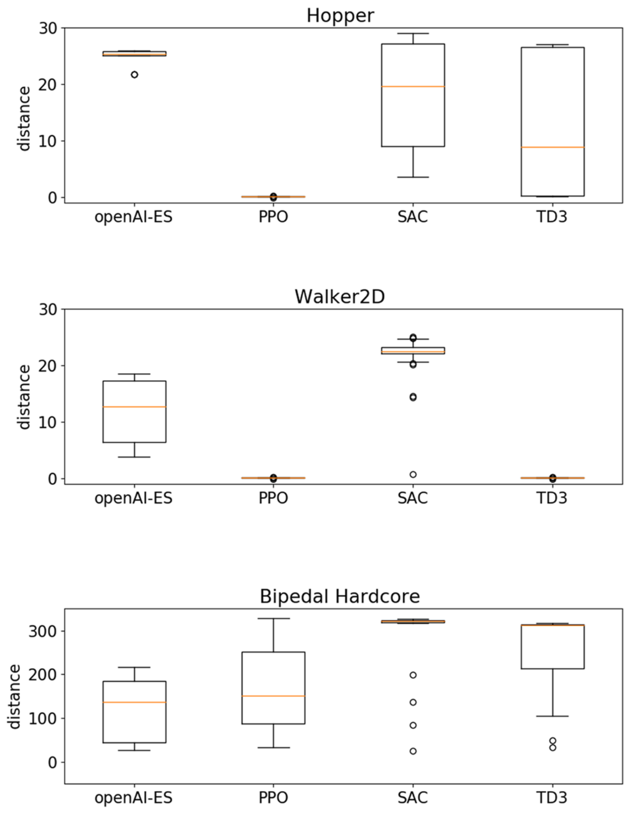

Fig. 3 Average distance traveled by Hopper, Walker2D, and Bipedal Hardcore agents trained with the OpenAI-ES, PPO, TD3 and SAC algorithms. Results obtained with the reward function which does not include the incentive for remaining upright. Each boxplot shows the results obtained post-evaluating for 5 episodes the best 10 trained agents of the 10 corresponding replications. Boxes represent the inter-quartile range of the data and horizontal lines inside the boxes mark the median values. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box. The hyperparameters used are described in the supplementary material

ability to walk in the case of the Walker2D problem and in some of the replications of the Hopper problem. The SAC algorithm manages to produce a walking behavior without the incentive in most of the replications and produces similar performance on the average with the incentive. The fact that the distribution of performance among replications is wider without the incentive and smaller with the incentive, indicates that in the case of SAC the incentive has a positive impact on the replications which fail producing a walking behavior and a negative impact on the other replications.

The differences among the algorithms are probably due to the usage of deterministic versus stochastic policies and to differences in the exploration abilities. The usage of policies which are highly stochastic, especially during the initial phase of the training process, increases the complexity of the problem to be solved since it requires to discover a set of parameters which enable the agent to walk in the presence of strong action perturbations. Consequently,

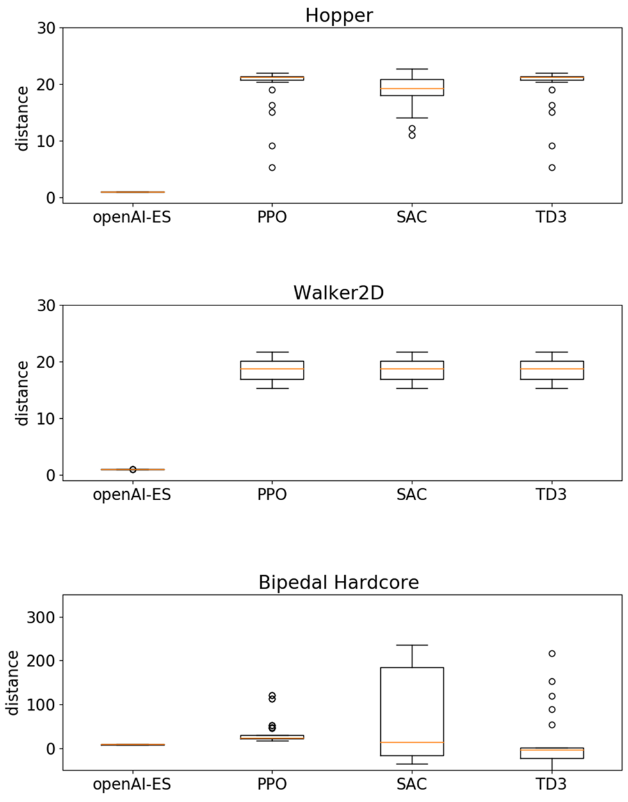

Fig. 4 Average distance traveled by Hopper, Walker2D, and Bipedal Hardcore agents trained with the OpenAI-ES, PPO, TD3 and SAC algorithms. Results obtained with the standard reward function which includes an incentive for remaining upright. Each boxplot shows the results obtained post-evaluating for 5 episodes the best 10 trained agents of the 10 corresponding replications. Boxes represent the inter-quartile range of the data and horizontal lines inside the boxes mark the median values. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box. The hyperparameters used are described in the supplementary material

the usage of stochastic policy increases the potential utility of the incentive. This consideration can be used to explain why the PPO algorithm, which uses a stochastic policy, is completely unable to discover a walking ability in the case of the Hopper and Walker2D problems without the incentive and benefit from the incentive. The results obtained with the TD3 and SAC algorithm, on the other hand, seem to imply that the algorithms differ also with respect to the ability to explore the search space independently from whether they use a stochastic or deterministic policy. This is not surprising since evolutionary and reinforcement learning methods differ with respect to the way in which they introduce variations and since the SAC algorithm, which is less dependent on the incentive, improves its exploration ability through the maximization of entropy.

More generally, the results reported in this section demonstrates that the performance of all algorithms are dramatically affected by the characteristics of the reward functions and that the reward functions optimized for an algorithm can be very sub-optimal for other algorithms. This implies that benchmarking alternative methods without optimizing the reward function can be useless, an issue which was neglected to date.

6 Ability to cope with environmental variations

Still another property which differentiates EAs and RLs is the ability to cope with environmental variations. Exposing the agents to variable environmental conditions is necessary to promote the development of solutions which are robust with respect to variations, i.e. to avoid the selection of solutions overfitted to the specific environmental conditions experienced during the evolutionary or learning process [31]. The environmental conditions can be varied by changing the initial position/orientation of the robot and of the objects present in the environment at the beginning of evaluation episodes and by perturbing the state of the robot and/or the environment during the episode. The introduction of environmental variations, however, makes the reward measure noisy. Indeed, in the presence of environmental variations, the fitness or the reward collected by an agent does not depend on the skill of the agent only but also on the environmental conditions encountered during the agent's evaluation. The fitness or reward will thus be overestimated or underestimated for agents which encountered favorable or unfavorable environmental conditions, respectively.

Reinforcement learning methods, like the PPO, can tolerate greater environmental variations than evolutionary methods such as the OpenAI-ES. This can be explained by considering that the latter algorithm, and more generally ESs, operate by introducing variations maximizing the total reward while the PPO, and many other RLAs, operate by introducing variations maximizing the advantage, i.e. the offset between the reward expected in a certain condition and the reward obtained in that condition. The expected reward is higher and smaller in easy and difficult environmental conditions, respectively. Consequently, the utilization of the advantage permits to filter out the effects of the variations of the environmental conditions, providing that the prediction of the expected reward is accurate.

The fact that the PPO algorithm can tolerate higher environmental variations than the OpenAI-ES algorithm can be illustrated with the Slime Volley environment [38]. This is a volley game in which two agents are situated in the two subparts of a field divided by a net. The goal of each agent is to send the ball into the ground of the opponent and to avoid that the ball touches the agent's field. In the version of the problem considered, the left agent is trained while the right agent moves on the basis of a pre-trained policy which remains constant. In a problem of this kind the orientation and velocity with which the ball is launched at the beginning of the episode has a strong impact on the fitness/reward collected by the agent, especially at the beginning of the evolutionary or learning process in which the ability of the agent to intercept the ball is poor. The reward obtained by an agent correlates primarily with the fraction of episodes in which the ball is launched toward the agent's own field and with the initial orientation and velocity with which the ball is launched — two factors which are independent from the agent's skill.

The fact that the PPO algorithm tolerates the effect of those environmental variations better than the OpenAI-ES algorithm is demonstrated by the fact the PPO manages to quickly develop effective agents while the OpenAI-ES fails (Fig. 5, top). Moreover, it is demonstrated by the fact that the OpenAI-ES algorithm manages to solve the symmetrical version of the Slime Volley problem in which the noise caused by the environmental variation is substantially reduced without reducing the overall range of variation of the environmental conditions (Fig. 5, bottom). In the symmetrical version of the problem, the orientation and velocity of the ball is generated randomly during even episodes while is generated by inverting the angle of 180 degrees and by maintaining the same velocity of the previous episode, during odd episodes. This ensures that the number of times in which the ball is launched in the field of the two players and the relative angle with which it is launched are the same. The modification of the problem introduced thus reduces the impact of the environmental variations without altering the complexity of the problem to be faced and the distribution of conditions to be faced.

The weakness of the OpenAI-ES algorithm in that respect can be reduced by using the super-symmetrical version of the original algorithm introduced here in which symmetrical individuals are exposed to the same environmental conditions (Fig. 6). The algorithm uses symmetrical sampling to improve the accuracy of the estimated fitness gradient. This means that the population is formed by couples of offspring which are generated by perturbing the parameters of the parent through the addition and subtraction of the same vector of random values. In the standard version of the algorithm, each individual is exposed to randomly different environmental conditions. In the super-symmetrical version of the algorithm introduced here, instead, each couple of symmetrical offspring is exposed to randomly different environmental conditions but the two symmetrical offspring are exposed to the same environmental conditions. This permits us to estimate the relation between the perturbations received by each couple of symmetrical offspring and the fitness obtained independently from the effect of environmental variations. As shown by Fig. 6, this method permits to achieve significantly better performance than the standard OpenAI-ES algorithm in the modified version of the task (Wilcoxon non parametric test p-value < 0.01).

Another method which can be used to reduce the noise of the fitness measure in evolutionary algorithms consists in estimating the relative difficulty of environmental conditions and choosing conditions which have similar levels of difficulty on the average (see [39]).

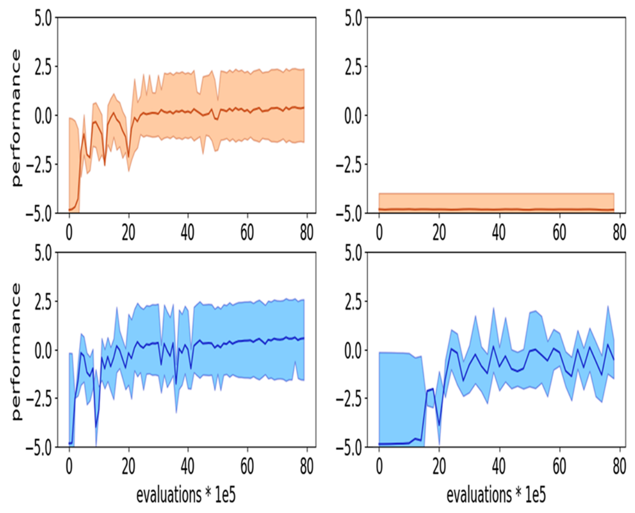

Fig. 5 Performance obtained by agents trained with the PPO and OpenAI-ES algorithms, left and right respectively, during the training process. Data obtained on the standard and symmetrical versions of the Slime Volley problem, top and bottom respectively. Mean and 90% bootstrapped confidence intervals of the mean (shadow area) across 10 replications per experiment

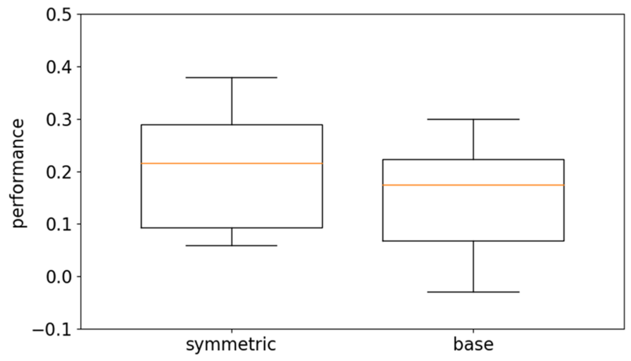

Fig. 6 Performance obtained by agents trained with the super-symmetrical version of the OpenAI-ES algorithm (symmetric) and with the standard version of OpenAI-ES algorithm (base) on the symmetrical version of the Slime Volley problem. Each boxplot shows the results obtained in 10 replications. Boxes represent the inter-quartile range of the data and horizontal lines inside the boxes mark the median values. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box

7 Discussion

In this article we compared the EAs and RLs by focusing in particular on the OpenAI-ES and PPO algorithms which are the most similar methods belonging to the two classes and which represent the state-of-the-art in their respective class. The results of the comparisons reported in the literature and the original results reported here do not indicate a general superiority of one algorithm over the other but rather that the two methods differ qualitatively with respect to several factors. The appreciation of these factors can be crucial to identify the method which is most promising for a particular problem and/or to identify how the weakness of the methods can be reduced.

Probably one of the most important factors is the variability of the environmental conditions. Unlike the OpenAI-ES, the PPO algorithm includes a mechanism for filtering out the noise affecting the reward measure caused by environmental variations. Such a mechanism consists in the calculation of the advantage which is based on an estimation of the expected reward. This enables the PPO algorithm to outperform the OpenAI-ES algorithm in problems in which the impact of environmental variations is large and in which the expected reward can be predicted with sufficient accuracy. Conversely, it enables the OpenAI-ES algorithm to outperform the PPO in problems in which the expected reward cannot be predicted accurately.

The weakness of OpenAI-ES algorithm in this respect can be reduced in different manners: (i) by using the super-symmetrical version of the algorithm introduced here, (ii) by estimating the complexity level of environmental conditions and by evaluating agents in environmental conditions which have a similar level of complexity ([39, 40], and (iii) by altering the problem in a way which reduce the impact of environmental variation (see Sect. 6). The weakness of the PPO algorithm with respect to problems in which estimating the expected reward is difficult can be reduced by enriching the observation of the critic with information which is available in simulation and which cannot be accessed in hardware (see for example [35]).

A second important factor is the reward function. The OpenAI-ES algorithm operates effectively also in problems in which the temporal offset between the execution of the appropriate actions and the collection of the corresponding reward is large and in which the reward function is simple. The PPO algorithm struggles more in problems in which the rewards are sparse in time and benefit more from the introduction of incentives in the reward function. Alternative reinforcement learning algorithms, such as the PPO, the SAC and the TD3 also differ in that respect among themselves. These findings imply that the reward function should be optimized to the particular method used. Moreover, they imply that benchmarking alternative algorithms without optimizing the reward function to each specific method provides little evidence on the relative efficacy of the compared methods.

A third important factor is the solution space which can be accessed by the two methods. The OpenAI-ES algorithm has access to a large set of solutions which includes minimal solutions, i.e. solutions which operate on the basis of few control rules. The PPO algorithm instead has access to a restricted solution space which includes only the solutions capable of copying with large action perturbations. This latter set of solutions often exclude minimal solutions. This qualitative difference represents a strength and a weakness for the OpenAI-ES and the PPO methods, respectively, for problems in which simple solutions achieve high performance. Instead, it represents a weakness and a strength for the OpenAI-ES and the PPO methods, respectively, for problems in which minimal solutions correspond to local minima. These qualitative differences also explain why evolutionary and reinforcement learning methods often discover qualitatively different behavioral solutions.

Expliciting the qualitative difference of alternative methods permits clarifing the relative weaknesses of each algorithm and identifing methods for ameliorating such weaknesses. For example, the realization of the inability of the OpenAI-ES method to deal with large environmental variations enabled us to propose the super-symmetrical version of this algorithm which is more effective in this respect. Moreover, understanding the characteristics of the particular method used can be used to tune the characteristics of the experimental setup in a more informed way.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12065-022-00801-3.

Author contributions All authors contributed equally

Declarations

Conflict of interests The authors declare the absence of any financial or non-financial competing interest.

References

1. Salimans T, Ho J, Chen X, Sidor S, Sutskever I (2017) Evolution strategies as a scalable alternative to reinforcement learning. arXiv:1703.03864v2
2. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347
3. Fujimoto S, Hoof H, Meger D (2018). Addressing function approximation error in actor-critic methods. In: International conference on machine learning. PMLR. pp 1587–1596

4. Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tan J, Kumar V, Zhu H, Gupta A, Abbeel P, Levine S (2018) Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905
5. Wierstra D, Schaul T, Glasmachers T, Sun Y, Peters J, Schmidhuber J (2014) Natural evolution strategies. J Mach Learn Res 15(1):949–980
6. Sehnke F, Osendorfer C, Rückstieß T, Graves A, Peters J, Schmidhuber J (2010) Parameter-exploring policy gradients. Neural Netw 23(4):551–559
7. Glasmachers T, Schaul T, Yi S, Wierstra D, Schmidhuber J (2010). Exponential natural evolution strategies. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation (pp 393–400)
8. Schaul T, Glasmachers T, Schmidhuber J (2011) High dimensions and heavy tails for natural evolution strategies. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation (pp 845–852)
9. Mnih V, Badia A P, Mirza M, Graves A, Lillicrap T, Harley T., Silver D Kavukcuoglu K (2016). Asynchronous methods for deep reinforcement learning. In: International conference on machine learning (pp 1928–1937). PMLR
10. Zhang Z, Wang D, Zhao D, Han Q, Song T (2018) A gradient-based reinforcement learning algorithm for multiple cooperative agents. IEEE Access 6:70223–70235
11. Konda V, Tsitsiklis J (1999). Actor-critic algorithms. Advances in neural information processing systems, 12
12. Haarnoja T, Zhou A, Abbeel P, Levine S (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning (pp 1861–1870). PMLR
13. Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9:159–195. https://doi.org/10.1162/106365601750190398
14. Schulman J, Levine S, Abbeel P, Jordan MI, Moritz P. (2015) Trust region policy optimization. In: ICML, pp 1889–1897
15. Duan Y, Chen X, Houthooft R, Schulman J, Abbeel P (2016). Benchmarking deep reinforcement learning for continuous control. In: International conference on machine learning. PMLR, pp 1329–1338
16. Salimans T, Ho J, Chen X, Sidor S Sutskever I (2017) Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864
17. Zhang S, Zaiane OR (2017) Comparing deep reinforcement learning and evolutionary methods in continuous control. arXiv preprint arXiv:1712.00006
18. Khadka S, Tumer K (2018). Evolution-guided policy gradient in reinforcement learning. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems (pp 1196–1208)
19. Zhu S, Belardinelli F, León B G (2021). Evolutionary reinforcement learning for sparse rewards. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp 1508–1512)
20. Badia A P, Piot B, Kapturowski S, Sprechmann P, Vitvitskyi A, Guo Z D, Blundell C. (2020). Agent57: Outperforming the atari human benchmark. In: International Conference on Machine Learning (pp 507–517). PMLR
21. Burda Y, Edwards H, Storkey A, Klimov O. (2018). Exploration by random network distillation. arXiv preprint arXiv:1810.12894
22. Schmidhuber J (2010) Formal theory of creativity, fun, and intrinsic motivation (1990–2010). IEEE Trans Auton Ment Dev 2(3):230–247
23. Lehman J, Stanley K O(2008). Exploiting open-endedness to solve problems through the search for novelty. Artificial Life, 329–336
24. Plappert M, Houthooft R, Dhariwal P, Sidor S, Chen Y, Chen X, Asfour T, Abbeel P, Andrychowicz M (2017). Parameter space noise for exploration. *arXiv preprint* arXiv:1706.01905
25. Raffin A, Stulp F (2020) Generalized state-dependent exploration for deep reinforcement learning in robotics. *arXiv preprint* arXiv:2005.05719
26. Lehman J, Clune J, Misevic D, Adami C, Altenberg L (2020) The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. Artif Life 26(2):274–306
27. Wilson DG, Cussat-Blanc S, Luga H, Miller JF (2018). Evolving simple programs for playing Atari games. In: Proceedings of the Genetic and Evolutionary Computation Conference (pp 229–236)
28. Milano N, Nolfi S (2018). Scaling up cartesian genetic programming through preferential selection of larger solutions. arXiv preprint arXiv:1810.09485 (2018)
29. Milano N, Pagliuca P, Nolfi S (2019) Robustness, evolvability and phenotypic complexity: insights from evolving digital circuits. Evol Intel 12(1):83–95
30. Wagner A (2013) Robustness and evolvability in living systems, vol 24. Princeton University Press
31. Pagliuca P, Nolfi S (2019) Robust optimization through neuroevolution PloS one 14(3):e0213193
32. Jakobi N, Husbands P, Harvey I (1995) Noise and the reality gap: the use of simulation in evolutionary robotics. In: Moran F, Moreno A, Merelo JJ, Chacon P (eds) European Conference on Artificial Life. Springer, Berlin
33. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) OpenAI Gym. arXiv:1606.01540
34. Coumans E, Bai Y (2016) Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019
35. Andrychowicz M, Baker B, Chociej M et al. (2018). Learning dexterous in-hand manipulation. arXiv:1808.00177v5
36. Nolfi S (2021) Behavioral and cognitive robotics: An adaptive perspective. roma, italy: institute of cognitive sciences and technologies, National Research Council (CNR-ISTC). ISBN 9791220082372
37. Pagliuca P, Milano N, Nolfi S (2020) Efficacy of modern neuro-evolutionary strategies for continuous control optimization. Front Robot A I:7
38. Ha D (2020) Slime volleyball gym environment. https://github.com/hardmaru/ slimevolleygym
39. Milano N, Nolfi S (2021) Automated curriculum learning for embodied agents a neuroevolutionary approach. Sci Rep 11(1):1–14
40. Milano N, Carvalho J T, Nolfi S. (2017). Environmental variations promotes adaptation in artificial evolution. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1–7). IEEE