Combining Evolution and Deep Reinforcement Learning for Policy Search: A Survey

OLIVIER SIGAUD, Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique
Paris, France

Deep neuroevolution and deep Reinforcement Learning have received a lot of attention over the past few years. Some works have compared them, highlighting their pros and cons, but an emerging trend combines them so as to benefit from the best of both worlds. In this article, we provide a survey of this emerging trend by organizing the literature into related groups of works and casting all the existing combinations in each group into a generic framework. We systematically cover all easily available papers irrespective of their publication status, focusing on the combination mechanisms rather than on the experimental results. In total, we cover 45 algorithms more recent than 2017. We hope this effort will favor the growth of the domain by facilitating the understanding of the relationships between the methods, leading to deeper analyses, outlining missing useful comparisons and suggesting new combinations of mechanisms.

CCS Concepts: • Computing methodologies → Reinforcement learning;

Additional Key Words and Phrases: Evolutionary algorithms

ACM Reference format:
Olivier Sigaud. 2023. Combining Evolution and Deep Reinforcement Learning for Policy Search: A Survey. ACM Trans. Evol. Learn. 3, 3, Article 10 (September 2023), 20 pages.
https://doi.org/10.1145/3569096

1 INTRODUCTION

The idea that the extraordinary adaptive capabilities of living species result from a combination of evolutionary mechanisms acting at the level of a population and learning mechanisms acting at the level of individuals is ancient in life sciences [91] and has inspired early work in Artificial Intelligence (AI) research [31]. This early starting point has led to the independent growth of two bodies of formal frameworks, evolutionary methods, and reinforcement learning (RL). The early history of the evolutionary side is well covered in Reference [2] and from the RL side in Reference [95]. Despite these independent developments, research dedicated to the combination has remained active, in particular around Learning Classifier Systems [43, 89] and studies of the Baldwin effect [105]. A broader perspective and survey on all the evolutionary and RL combinations anterior to the advent of the “deep learning” methods using large neural networks can be found in Reference [19].
In this article, we propose a survey of a renewed approach to this combination that builds on the unprecedented progress made possible in evolutionary and deep RL methods by the growth of computational power and the availability of efficient libraries to use deep neural networks. As this survey shows, the topic is rapidly gaining popularity with a wide variety of approaches and even emerging libraries dedicated to their implementation [98]. Thus, we believe it is the right time for laying solid foundations to this growing field by listing the approaches and providing a unified view that encompasses them. There are recent surveys about the comparison of evolutionary and RL methods [59, 77] that mention the emergence of some of these combinations. With respect to these surveys, ours is strictly focused on the combinations and attempts to provide a list of relevant papers as exhaustive as possible at the time of its publication, irrespective of their publication status.

This survey is organized into groups of algorithms using the evolutionary part for the same purpose. In Section 2, we first review algorithms where evolution is looking for efficient policies, that is, combining deep neuroevolution and deep RL. We then cover in Section 3 algorithms where evolution directly looks for efficient actions in a given state rather than for policies. In Section 4, we cover the combination of deep RL algorithms with diversity seeking methods. Finally, in Section 5, we cover various other uses of evolutionary methods, such as optimizing hyperparameters or the system’s morphology. To keep the survey as short as possible, we consider that the reader is familiar with evolutionary and RL methods in the context of policy search and has a good understanding of their respective advantages and weaknesses. We refer the reader to Reference [88] for an introduction of the methods and to surveys about comparisons to know more about their pros and cons [59, 77].

2 EVOLUTION OF POLICIES FOR PERFORMANCE
The methods in our first family are listed in Table 1, they combine a deep neuroevolution loop and a deep RL loop. Figure 1 provides a generic template to illustrate such combinations. The methods optimize the performance of a policy by searching in the space of policy parameters. The central question left open by the template is how both loops interact with each other. Note that this template is not much adapted to account for works where the combination is purely sequential, such as Reference [40] or the Goal Exploration Process Policy Gradient (gep-pg) algorithm [12]. Besides, to avoid any confusion with the multi-agent setting, note that agents are interacting in isolation with their own copy of the environment and cannot interact with each other.

The main motivation for combining evolution and deep RL is the improved performance that may result from the combination. For instance, through simple experiments with simple fitness landscapes and simplified versions of the components, combining evolution and RL can be shown to work better than using either of the two in isolation [100]. Why is this so? One of the explanations is the following. A weakness of policy gradient methods at the heart of deep RL is that they compute an estimate of the true gradient based on a limited set of samples. This gradient can be quite wrong due to the high variance of the estimation, but it is applied blindly to the current policy without checking that this actually improves performance. By contrast, variation-selection methods at the heart of evolutionary methods evaluate all the policies they generate and remove the poorly performing ones. Thus a first good reason to combine policy gradient and variation-selection methods is that the latter may remove policies that have been deteriorated by the gradient step. Below we list different approaches building on this idea. This perspective is the one that gives rise to the largest list of combinations. We further split this list into several groups of works in the following sections.

2.1 Deep RL Actor Injection
One of the main algorithms at the origin of the renewal of combining evolution and RL is Evolutionary Reinforcement Learning (erl) [38]; see Figure 2(a). It was published simultaneously
Table 1. Combinations Evolving Policies for Performance

| Algo.      | Prop. | RL algo. | Evo. algo. | Actor Injec. | + Comb. Mech. | Surr. Fitness | Soft Update | Buffer Filt. |
|------------|-------|----------|------------|--------------|---------------|---------------|-------------|---------------|
| ERL [58]   |       | DDPG     | GA         | •            | x             | x             | x           | x             |
| CERL [37]  |       | TD3      | GA         | •            | x             | x             | x           | x             |
| PDERL [4]  |       | TD3      | GA         | •            | x             | x             | x           | x             |
| ESAC [94]  |       | SAC      | ES         | •            | x             | x             | x           | x             |
| FIDI-RL [85]|      | DDPG     | ARS        | •            | x             | x             | x           | x             |
| X-DDPG [20]|       | DDPG     | GA         | •            | x             | x             | x           | x             |
| CEM-RL [75]|       | TD3      | CEM        | •            | x             | •             | x           | x             |
| CEM-ACER [96]|     | ACER     | CEM        | •            | •             | x             | x           | x             |
| SERL [102]|       | DDPG     | GA         | •            | x             | •             | x           | x             |
| SPDERL [102]|      | TD3      | GA         | •            | x             | •             | x           | x             |
| PGPS [41]  |       | TD3      | CEM        | •            | x             | •             | •           | x             |
| BNET [92]  |       | BBNE     | CGP        | •            | x             | •             | •           | x             |
| CSPC [108]|       | SAC + PPO| CEM        | •            | x             | x             | x           | •             |
| SUPE-RL [61]|      | RAINBOW or PPO | GA       | •            | x             | •             | •           | •             |
| G2AC [5]   |       | A2C      | GA         | •            | x             | •             | x           | x             |
| G2PPO [5]  |       | PPO      | GA         | •            | x             | •             | x           | x             |

The table states whether the algorithms in the rows use the mechanisms in the columns. The colors are as follows. In the column about other combination mechanisms (+ Comb. Mech.): Critic gradient addition • (green), Population from Actor ♠ (blue), None x (red). In all other columns: • (green): yes, x (red): no. In bnet, bbne stands for Behavior-Based NeuroEvolution and cgp stands for Cartesian Genetic Programming [62]. The different ga labels stand for various genetic algorithms, we do not go into the details.

Fig. 1. The general template of algorithms combining deep neuroevolution and deep RL. A population of agents interact with an environment and produce trajectories composed of states, actions, and rewards. From the left-hand side, an evolutionary loop selects and evolves these agents based on their fitness, which is computed holistically over trajectories. From the right-hand side, a deep RL loop improves one or several agents using a gradient computed over the elementary steps of trajectories stored into a replay buffer.

with the Genetic-Gated RL algorithms g2ac and g2ppo [5], but its impact was much greater. Its combination mechanism injects the RL actor into the evolutionary population.

The ERL algorithm was soon followed by Collaborative Evolutionary Reinforcement Learning (cerl) [37] that extends ERL from RL to distributed RL where several agents learn in parallel,
Fig. 2. The template architecture for ERL, ESAC, FIDI-RL, and CERL (a) and the PDERL architecture (b). In ERL, an actor learned by DDPG is periodically injected into the population and submitted to evolutionary selection. If DDPG performs better than the GA, then this will accelerate the evolutionary process. Otherwise the DDPG agent is just ignored. In ESAC, DDPG is replaced by SAC and in FIDI-RL, the GA is replaced by ARS [60]. In CERL, the DDPG agent is replaced by a set of TD3 actors sharing the same replay buffer but each using a different discount factor. Again, those of such actors that perform better than the rest of the population are kept and enhance the evolutionary process, whereas the rest are discarded by evolutionary selection. In PDERL, the genetic operators of ERL are replaced by operators using local replay buffers so as to better leverage the step-based experience of each agent.

and all these agents are injected into the evolutionary population. The main weakness of ERL and CERL is their reliance on a genetic algorithm that applies a standard \( n \)-point-based crossover and a Gaussian weight mutation operator to a direct encoding of the neural network architecture as a simple vector of parameters. This approach is known to require tedious hyperparameter tuning and generally perform worse than evolution strategies that are also mathematically more founded [9, 79]. In particular, the genetic operators used in ERL and CERL based on a direct encoding have been shown to induce a risk of catastrophic forgetting of the behavior of efficient individuals.

The Proximal Distilled Evolutionary Reinforcement Learning (PDERL) algorithm [4], see Figure 2(b), builds on this criticism and proposes two alternative evolution operators. Instead of standard crossover, all agents carry their own replay buffer and crossover selects the best experience in both parents to fill the buffer of the offspring, before applying behavioral cloning to get a new policy that behaves in accordance with the data in the buffer. This operator is inspired by the work of Reference [23]. For mutation, they take as is the improved operator proposed in Reference [45], which can be seen as applying a Gauss–Newton method to perform the policy gradient step [72].

Another follow-up of ERL is the Evolutionary Soft Actor Critic (ESAC) algorithm [94]. It uses the Soft Actor Critic (SAC) algorithm [27] instead of Deep Deterministic Policy Gradient (DDPG) [49] and a modified evolution strategy instead of a genetic algorithm, but the architecture follows the same template. Similarly, the Finite Difference RL (FIDI-RL) algorithm [85] combines DDPG with Augmented Random Search (ARS), a finite difference algorithm that can be seen as a simplified version of evolution strategies [60]. FIDI-RL uses the ERL architecture as is. The method is shown to outperform ARS alone and DDPG alone, but neither ESAC nor FIDI-RL are compared to any other combination listed in this survey. Finally, the X-DDPG algorithm is a version of ERL with several asynchronous DDPG actors where the buffers from the evolutionary agents and from
Combining Evolution and Deep Reinforcement Learning for Policy Search

Fig. 3. The CSPC (a) and CEM-RL (b) architectures. In CSPC, an on-policy and an off-policy algorithms are combined, together with two replay buffers and a performance-based actor injection rule, to improve the sample efficiency of ERL-like methods. In CEM-RL, gradient steps from the TD3 critic are applied to half the population of evolutionary agents. If applying this gradient is favorable, then the corresponding individuals are kept; otherwise, they are discarded.

The DDPG agents are separated, and the most recent DDGP agent is injected into the evolutionary population at each timestep [20].

The Behavior-based NeuroEvolutionary Training (bnet) algorithm [92] is borderline in this survey, as it does not truly use an RL algorithm but uses a Behavior-Based Neuroevolution (bbne) mechanism that is only loosely inspired from RL, without relying on gradient descent. BNet combines a robust selection method based on standard fitness, a second mechanism based on the advantage of the behavior of an agent, and a third mechanism based on a surrogate estimate of the return of policies. The BBNE mechanism is reminiscent of the Advantage Weighted Regression (awr) algorithm [70], but it uses an evolutionary approach to optimize this behavior-based criterion instead of standard gradient-based methods. The reasons for this choice is that the evolutionary part relies on Cartesian Genetic Programming (cgp) [62] that evolves the structure of the neural networks, but gradient descent operators cannot be applied to networks whose structure is evolving over episodes.

The Cooperative Heterogeneous Deep Reinforcement Learning (CHDRL) architecture [108] extends the ERL approach in several ways to improve the sample efficiency of the combination. First, it uses two levels of RL algorithms, one on-policy and one off-policy, to benefit from the higher sample efficiency of off-policy learning. Second, instead of injecting an actor periodically in the evolutionary population, it does so only when the actor to be injected performs substantially better than the evolutionary agents. Third, it combines the standard replay buffer with a smaller local one that is filled with filtered data to ensure using the most beneficial samples. The CSPC algorithm, depicted in Figure 3(a) is an instance of CHDRL using the CEM, SAC, and PPO [81] algorithms.

Note that if an RL actor is injected in an evolutionary population and if evolution uses a direct encoding, then the RL actor and evolution individuals need to share a common structure. Removing this constraint might be useful, as evolutionary methods are often applied to smaller policies than RL methods. For doing so, one might call upon any policy distillation mechanism that strives to obtain from a large policy a smaller policy with similar capabilities.
Fig. 4. In the g2n (a) and supe-rl (b) architectures, the evolutionary population is built locally from the RL actor. In g2n, the evolutionary part explores the structure of the central layer of the actor network. In supe-rl, more standard mutations are applied, the non-mutated actor is inserted in the evolutionary population and the actor is soft-updated toward its best offspring.

2.2 RL Gradient Addition

Instead of injecting an RL actor into the population, another approach applies gradient steps to some members of this population. This is the approach of the cem-rl algorithm [75], see Figure 3(b), which combines the Cross-Entropy Method (cem) [78] and Twin Delayed Deep Deterministic Policy Gradient (td3) [22]. This work was followed by cem-acer [96] that simply replaces td3 with Actor Critic with Experience Replay (acer) [103].

2.3 Evolution from the RL Actor

In the algorithms listed so far, the main loop is evolutionary and the RL loop is used at a slower pace to accelerate it. In the Genetic-Gated Networks (g2n) [5] and Soft Updates for Policy Evolution (supe-rl) [61] algorithms, by contrast, the main loop is the RL loop and evolution is used to favor exploration.

In g2n, shown in Figure 4(a), evolution is used to activate or deactivate neurons of the central layer in the architecture of the actor according to a binary genome. By sampling genomes using evolutionary operators, various actor architectures are evaluated, and the one that performs best benefits from a critic gradient step before its genome is used to generate a new population of architectures. This mechanism provides a fair amount of exploration both in the actor structures and in the generated trajectories and outperforms random sampling of the genomes. Two instances of the g2n approach are studied, g2ac based on Advantage Actor Critic (a2c) [63] and g2ppo based on Proximal Policy Optimization (ppo) [81], and they both outperform the RL algorithm they use.

The supe-rl algorithm, shown in Figure 4(b), is similar to g2n apart from the fact that evolving the structure of the central layer is replaced by performing standard Gaussian noise mutation of all the parameters of the actor. Besides, if one of the offspring is better than the current RL agent, then the latter is modified toward this better offspring through a soft update mechanism. Finally, the non-mutated actor is also inserted in the evolutionary population, which is not the case in g2n.

2.4 Using a Surrogate Fitness

A weakness of all the methods combining evolution and RL that we have listed so far is that they require evaluating the agents to perform the evolutionary selection step, which may impair sample
Fig. 5. The sc-erl (a) and pgps (b) architectures are two approaches to improve sample efficiency by using a critic network as a surrogate for evaluating evolutionary individuals. In sc-erl, the surrogate control part is generic and can be applied to several architectures such as erl, cerl, or cem-rl. It considers the critic as a surrogate model of fitness, making it possible to estimate the fitness of a new individual without generating additional samples. (b) The pgps uses the same idea but combines it with several other mechanisms, such as performing a soft update of the actor toward the best evolutionary agent or filling half the population using the surrogate fitness and the other half from cem generated agents.

In the Surrogate-Assisted Controller for erl (sc-erl) [102] and Coupling Policy Gradient with Population-based Search (pgps) [41] architectures, this concern is addressed by using a critic network as a surrogate for evaluating an agent. Importantly, the evaluation of individuals must initially rely on the true fitness but can call upon the critic more and more often as its accuracy gets better. As shown in Figure 5(a), the sc-erl architecture is generic and can be applied on top of any of the combinations we have listed so far. In practice, it is applied to erl, pderl, and cem-rl, resulting in the serl and spderl algorithms in the first two cases.

The pgps algorithm [41], shown in Figure 5(b), builds on the same idea but uses it in the context of a specific combination of evolutionary and RL mechanisms that borrows ideas from several of the previously described methods. In more detail, half of the population is filled with agents evaluated from the surrogate fitness, whereas the other half are generated with cem. Furthermore, the current td3 actor is injected into the population and benefits from a soft update toward the best agent in the population.

3 EVOLUTION OF ACTIONS FOR PERFORMANCE

In this section, we cover algorithms where evolution is used to optimize an action in a given state rather than optimizing policy parameters. The general idea is that variation-selection methods such as cem can optimize any vector of parameters given some performance function of these parameters. In the methods listed in the previous section, the parameters were those of a policy and the performance was the return of that policy. In the methods listed here, the parameters specify the action in a given state and the performance is the Q value of this action in that state.

In an RL algorithm like Q-learning [104], the agent needs to find the action with the highest value in a given state for two things: for performing critic updates, that is, updating its estimates of the action-value function using $Q(s_t, a_t) \leftarrow r(s_t, a_t) + \max_{a} Q(s_{t+1}, a) - Q(s_t, a_t)$, and for acting using $\arg \max_{a} Q(s_t, a)$. When the action space is continuous, this amounts to solving an expensive optimization problem, and this is required at each training step. The standard solution to this problem in actor–critic methods considers the action of the actor as a good proxy for the best
action. The estimated best action, that we note $\tilde{a}_t$, is taken to be the actor’s action $\tilde{a}_t = \pi(s_t)$, resulting in using $Q(s_t, a_t) \leftarrow r(s_t, a_t) + \max_a Q(s_{t+1}, \pi(s_{t+1})) - Q(s_t, a_t)$ for the critic update and using $\tilde{a}_t$ for acting.

But as an alternative, one can call upon a variation-selection method to find the best-performing action over a limited set of sampled actions. This approach is used in the QT-OPT algorithm [36], as well as in the Cross-Entropy Guided Policies (CGP) [90], Soft Actor-Critic with Cross-Entropy Policy Optimization (SAC-CEPO) [87], Self-Guided and Self-Regularized Actor-Critic (GRAC) [84], and Evolutionary Action Selection RL (EAS-RL) [57] algorithms. This is the approach we first cover in this section. The Zeroth-Order Supervised Policy Improvement (ZOSPI) algorithm [93] also benefits from optimizing actions with a variation-selection method, though it stems from a different perspective.

As Table 2 shows, the QT-OPT algorithm [36] simply samples 64 random actions in the action space and performs two iterations of cem to get a high-performing action, both for critic updates and action selection. It is striking that such a simple method can perform well even in large action spaces. This simple idea was then improved in the CGP algorithm [90] so as to avoid the computational cost of action inference. Instead of using cem to sample an action at each timestep, a policy network is learned based on the behavior of the cem. This network can be seen as a surrogate of the cem sampling process and is trained either from the sampled $\tilde{a}_t$ using Behavioral Cloning (BC) or following a Deterministic Policy Gradient (DPG) step from the critic.

The EAS-RL algorithm [57] is similar to CGP apart from the fact that it uses Particle Swarm Optimization (PSO) instead of CEM. Besides, depending on the sign of the advantage of the obtained action $\tilde{a}_t$, it uses either BC or DPG to update the policy for each sample.

Symmetrically to CGP, the SAC-CEPO algorithm [87] performs standard critic updates using SAC but selects actions using CEM. More precisely, it introduces the idea to sample the action from the current policy rather than randomly and updates this policy using BC from the sampled actions. Besides, the article investigates the effect of the CEM parameters but does not provide solid conclusions.

The GRAC algorithm [84] combines ideas from CGP and SAC-CEPO. A stochastic policy network outputs an initial Gaussian distribution for the action at each step. Then, a step of CEM drawing 256 actions out of this distribution is used to further optimize the choice of action both for critic updates and action selection. The policy itself is updated with a combination of two training losses.

Finally, the ZOSPI algorithm [93] calls upon variation-selection for updating the policy rather than for updating the critic or selecting the action. Its point is rather that gradient descent algorithms tend to get stuck into local minima and may miss the appropriate direction due to

| Algo.       | Prop.           | Critic update       | Action Selection     | Policy Update      |
|-------------|-----------------|---------------------|----------------------|--------------------|
| QT-OPT [36] | $\tilde{a}_t = \text{cem (random, 64, 6, 2)}$ | $\tilde{a}_t = \text{cem (random, 64, 6, 2)}$ | No policy          |
| CGP [90]    | $\tilde{a}_t = \text{cem (random, 64, 6, 2)}$ | $\tilde{a}_t = \pi(s_t)$ | BC or DPG          |
| EAS-RL [57] | $\tilde{a}_t = \text{psos (10,10)}$ | $\tilde{a}_t = \pi(s_t)$ | BC + DPG           |
| SAC-CEPO [87]| $\text{sac update}$ | $\tilde{a}_t = \text{cem (π, 60 → 140, 3% → 7%, 6 → 14)}$ | BC                 |
| GRAC [84]   | $\tilde{a}_t = \text{cem (π, 256, 5, 2)}$ | $\tilde{a}_t = \text{cem (π, 256, 5, 2)}$ | PG with two losses |
| ZOSPI [93]  | $\text{ddpg update}$ | $\tilde{a}_t = \pi(s_t) + \text{perturb. network}$ | BC($\tilde{a}_t = \text{argmax}(\text{random, 50})$) |

The cells in green denote where evolutionary optimization takes place. We specify the use of CEM for optimizing an action with $\tilde{a}_t = \text{cem(source, N, Ne, I)}$, where source is the source from which we sample initial actions, N is the size of this sample (the population), Ne is the number of elite solutions that are retained from a generation to the next and I is the number of iterations. For PSO, the shown parameters are the number of action N and the number of iterations T. And we use $\tilde{a}_t = \text{argmax(source, N)}$ for simply take the best action over N samples from a given source.
various approximations, whereas a variation-selection method is more robust. Thus, to update its main policy, ZOSPI simply samples a set of actions and performs BC toward the best of these actions, which can be seen as a trivial variation-selection method. The typical number of sampled actions is 50. It then adds a policy perturbation network to perform exploration, which is trained using gradient descent.

4 EVOLUTION OF POLICIES FOR DIVERSITY

The tradeoff between exploration and exploitation is central to RL. In particular, when the reward signal is sparse, efficient exploration becomes crucial. All the papers studied in this survey manage a population of agents, hence their capability to explore can benefit from maintaining behavioral diversity between the agents. This idea of maintaining behavioral diversity is central to two families of diversity seeking algorithms, the novelty search (NS) [46] algorithms that do not use the reward signal at all, see Figure 6(a), and the quality-diversity (QD) algorithms [15, 76], see Figure 6(b), which try to maximize both diversity and performance. As the NS approach only looks for diversity, it is better in the absence of reward, or when the reward signal is very sparse or deceptive as the best one can do in the absence of reward is try to cover a relevant space as uniformly as possible [18]. By contrast, the QD approach is more appropriate when the reward signal can contribute to the policy search process. In this section we cover both families separately.

4.1 Novelty Seeking Approaches

Maintaining a distance between agents in a population can be achieved in different spaces. For instance, the Stein Variational Policy Gradient (SVPG) algorithm [54] defines distances in a kernel space and adds to the policy gradient a loss term dedicated to increasing the pairwise distance between agents. Alternatively, the Diversity via Determinants (DvD) algorithm [68] defines distances in an action embedding space, corresponding to the actions specified by each agent in a large-enough set of random states. Then DvD optimizes a global distance between all policies by maximizing the volume of the space between them through the computation of a determinant. Despite their interest, these two methods depicted in Figure 7 do not appear in Table 3, as the former does not have an evolutionary component and the latter uses NSR-ES [14] but does not have an RL component.
Fig. 7. The svpg (a) and dvd (b) architectures. In svpg, individual policy gradients computed by each agent are combined so as to ensure both diversity between agents and performance improvement. In dvd, a purely evolutionary approach is combined with a diversity mechanism that seeks to maximize the volume between the behavioral characterization of agents in an action embedding space. Both architectures do not aim to combine evolution and RL, though they both try to maximize diversity and performance in a population of agents.

Table 3. Combinations Evolving Policies for Diversity

| Algo.        | Prop. | RL algo. | Diversity algo. | Distance space |
|--------------|-------|----------|-----------------|----------------|
| p3s-td3 [35] |       | TD3      | Find best       | Policy params. |
| deprl [52]   |       | TD3      | CEM             | Policy params. |
| arac [17]    |       | SAC      | NS-like         | Policy params. |
| ns-rl [86]   |       | GC-DDPG  | True NS         | Manual BC      |
| pns-rl [53]  |       | TD3      | True NS         | Manual BC      |

NS: Novelty Search. Policy params: distance is computed in the policy parameters space. GC-DDPG: goal-conditioned DDPG. Manual BC: distances are computed in a manually defined behavior characterization space.

A more borderline case with respect to the focus of this survey is the Population-guided Parallel Policy Search (p3s-td3) algorithm [35], depicted in Figure 8. Though p3s-td3 is used as a baseline in several of the papers mentioned in this survey, its equivalent of the evolutionary loop is limited to finding the best agent in the population, as shown in Figure 8(a). This implies evaluating all these agents but using neither variation nor selection. Besides, the mechanism to maintain a distance between solutions in p3s-td3 is ad hoc and acts in the space of policy parameters. This is also the case in the Diversity Evolutionary Policy Deep Reinforcement Learning (deprl) algorithm [52], which is just a variation of cem-rl where an ad hoc mechanism is added to enforce some distance between members of the evolutionary population.

The Attraction-Repulsion Actor-Critic (arac) algorithm [17] also uses a distance in the policy parameter space, but it truly qualifies as a combination of evolution and deep RL, see Figure 8(b). An original feature of arac is that it selects the data coming into the replay buffer based on the novelty of agents, which can result in saving a lot of poorly informative gradient computations. A similar idea is also present in Reference [6] where instead of filtering based on
Figure 8. The p3s-td3 (a) and arac (b) architectures. In p3s-td3, all agents are trained with RL and evaluated, then they all perform a soft update toward the best agent. The arac algorithm maintains a population of policies following a gradient from a common sac critic [27]. The critic itself is trained from trajectories of the most novel agents. Besides, diversity in the population is ensured by adding an attraction-repulsion loss $L_{AR}$ to the update of the agents. This loss is computed with respect to an archive of previous agents themselves selected using a novelty criterion.

In the Population-Guided Novelty Search for Reinforcement Learning (pns-rl) algorithm [53], a group of agents is following a leader combining a standard policy gradient update and a soft update toward the leader. Then, for any agent in the group, if its performance is high enough with respect to the mean performance in an archive, it is added to the archive. Crucially, the leader is selected as the one that maximizes novelty in the archive given a manually defined behavioral characterization. In addition, for efficient parallelization, the algorithm considers several groups instead of one but where all groups share the same leader.

The Novelty Search RL (ns-rl) algorithm [86] can be seen as a version of cem-rl whose RL part targets higher novelty by training less novel agents to minimize in each step the distance to the BC of the most novel agent. As the most novel agent and its BC change in each iteration, the RL part is implemented with goal-conditioned policies. This implies that the goal space is identical to the behavioral characterization space.

4.2 Quality-Diversity Approaches

By contrast with NS approaches that only try to optimize diversity in the population, QD approaches combine this first objective with optimize the performance of registered policies, their quality. As Figure 6(b) suggests, when combined with an RL loop, the QD loop can give rise to
Table 4. Quality-Diversity Algorithms Including an RL Component

| Algo.                  | Prop.                  | Type of Archive | Q. improvement | D. improvement |
|------------------------|------------------------|-----------------|----------------|---------------|
| PGA-ME [66]            | MAP-Elites             | TD3 or GA       | TD3 or GA      |               |
| QD-PG-PF [11]          | Pareto front           | TD3             | TD3            |               |
| QD-PG-ME [71]          | MAP-Elites             | TD3             | TD3            |               |
| CMA-MEGA-(TD3, ES) [99]| MAP-Elites             | TD3 + OPENAI-ES | OPENAI-ES      |               |

All these algorithms rely on the MAP-Elites approach and the BC space is defined manually. For each algorithm in the rows, the table states whether quality and diversity are optimized using an RL approach or an evolutionary approach.

various solutions depending on whether quality and diversity are improved with an evolutionary algorithm or a deep RL algorithm.

The space of resulting possibilities is covered in Table 4. In more detail, the **Policy Gradient Assisted MAP-Elites** (PGA-ME) algorithm [66] uses two optimization mechanisms, TD3 and a GA, to generate new solutions that are added to the archive if they are either novel enough or more efficient than previously registered ones with the same behavioral characterization. By contrast, in the **Quality-Diversity RL** (QD-RL) approach, the mechanisms to improve quality and diversity are explicitly separated and improve a quality critic and a diversity critic using TD3. Two implementations exist. First, the QD-PG-PF algorithm [11] maintains a Pareto front of high quality and diversity solutions. From its side, the QD-PG-ME algorithm [71] maintains a MAP-Elites archive and introduces an additional notion of state descriptor to justify learning a state-based quality critic. Finally, the **Covariance Matrix Adaptation MAP-Elites via a Gradient Arborescence** (CMA-MEGA) approach [99] uses the OPENAI-ES algorithm [79] to improve diversity and either OPENAI-ES or a combination of OPENAI-ES and RL to improve quality. Table 4 only shows the latter. Note that, in contrast to the other algorithms, the combination mechanism comes with additional parameters, which are themselves optimized with CMA-ES.

To summarize, one can see that both quality and diversity can be improved through RL, evolution, or both.

5 EVOLUTION OF SOMETHING ELSE

In all the architecture we have surveyed so far, the evolutionary part was used to optimize either policy parameters or a set of rewarding actions in a given state. In this section, we briefly cover combinations of evolution and deep RL where evolution is used to optimize something else that matters in the RL process, or where RL mechanisms are used to improve evolution without calling upon a full RL algorithm. We dedicate a separate part to optimizing hyperparameters, as it is an important and active domain.

5.1 Evolution in MBRL

The CEM algorithm can be used to optimize open-loop controllers to perform **Model Predictive Control** (MPC) on robotic systems in the Deep Planning Network (PLANET) [28] and Policy Planning (POPLIN) [101] algorithms, and an improved version of CEM for this specific context is proposed in References [73, 74]. Besides, this approach combining open-loop controllers and MPC is seen in the **Probabilistic Ensembles with Trajectory Sampling** (PETS) algorithm [10] as implementing a form of **Model-Based Reinforcement Learning** (MBRL), and CEM is used in PETS to choose the points from where to start MPC, improving over random shooting. Finally, in Reference [3], the authors propose to interleave CEM iterations and **Stochastic Gradient Descent (SGD)** iterations...
Table 5. Algorithms Where Evolution Is Applied to Something Else Than Action or Policy Parameters or to More Than Policy Parameters

| Algo.     | RL algo.       | Evo algo. | Evolves what?                      |
|-----------|----------------|------------|-----------------------------------|
| GA-DRL [82, 83] | DDPG (+HER)   | GA         | Hyper-parameters                  |
| PBT [34]  | Any            | Ad hoc     | Parameters and Hyper-parameters   |
| AAC [24]  | SAC            | Ad hoc     | Parameters and Hyper-parameters   |
| SEARL [21] | TD3            | GA         | Architecture, Parameters and Hyper-parameters |
| OHT-ES [97] | Any            | ES         | Hyper-parameters                  |
| EPQ [32]  | Ad hoc (~PPO)  | ES         | Reward-related functions          |
| PPO [48]  | ~DDPG          | ~CEM       | Critic                            |
| EVO-RL [30] | Q-LEARNING, DQN, PPO | BT | Partial policies                  |
| DRL [25]  | PPO            | GA         | System’s morphology               |
| * [67]    | PPO            | GA         | System’s morphology               |

All algorithms in the first half optimize hyperparameters. *: The algorithm in Reference [67] is given no name. BT stands for Behavior Tree.

Fig. 9. The PBT architecture. The evolution part consists of two operators, explore and exploit, that act both on the hyperparameters and the parameters of the agents.

To improve the efficiency of optimization of MPC plans, in a way reminiscent to CEM-RL combining policy gradient steps and CEM steps. But all these methods are applied to an open-loop control context where true reinforcement learning algorithms cannot be applied, hence they do not appear in Table 5.

5.2 Evolution of Hyper-parameters

Hyperparameter optimization (HPO) is notoriously hard and often critical in deep RL. The most straightforward way to leverage evolutionary methods in this context nests the deep RL algorithm within an evolutionary loop that tunes the hyper-parameters. This is the approach of the Genetic Algorithm Deep RL (GA-DRL) algorithm [82, 83], but this obviously suffers from a very high computational cost. Note that the authors write that GA-DRL uses DDPG + Hindsight Experience replay (HER) [1], but the use of HER is in no way clear as the algorithm does not seem to use goal-conditioned policies.

More interestingly, the Population-Based Training (PBT) architecture [34] depicted in Figure 9, is designed to solve this problem by combining distributed RL with an evolutionary mechanism that acts both on the parameters and hyperparameters within the RL training loop. It was
successfully used in several challenging applications [33] and benefits from an interesting capability to tune the hyperparameters according to the current training dynamics, which is an important meta-learning capability [39]. A follow-up of the PBT algorithm is the Automatic Actor-Critic (AAC) algorithm [24], which basically applies the same approach but with a better set of hyperparameters building on lessons learned in the recent deep RL literature.

A limitation of PBT is that each actor uses its own replay buffer. Instead, in the Sample-Efficient Automated Deep Reinforcement Learning (SEARL) algorithm [21], the experience of all agents is shared into a unique buffer. Furthermore, SEARL simultaneously performs HPO and Neural Architecture Search, resulting in better performance than PBT. Finally, the Online Hyper-parameter Tuning via Evolutionary Strategies (OHT-ES) algorithm [97] also uses a shared replay buffer but limits the role of evolution to optimizing hyperparameters and does so with an ES algorithm. Given the importance of the problem, there are many other HPO methods, most of which are not explicitly calling upon an evolutionary approach. For a wider survey of the topic, we refer the reader to Reference [69].

5.3 Evolution of Miscellaneous RL or Control Components

Finally, we briefly survey the rest of algorithms listed in Table 5. The Evolved Policy Gradient (EPG) algorithm [32] uses a meta-learning approach to evolve the parameters of a loss function that replaces the policy gradient surrogate loss in policy gradient algorithms. The goal is to find a reward function that will maximize the capability of an RL algorithm to achieve a given task. A consequence of its design is that it cannot be applied to an actor–critic approach.

Instead of evolving a population of agents, the Evolved Q-maps (EQ) algorithm [48] evolves a population of critics, which are fixed over the course of learning for a given agent. This is somewhat symmetric to the Zeroth-Order Actor-Critic (ZOAC) algorithm [47] that uses evolution to update an actor given a critic trained with RL.

The Evolutionary-driven RL (Evo-RL) algorithm [29] evolves partial policies. Evolution is performed in a discrete action context with a Genetic Programming approach [42] that only specifies a partial policy as Behavior Trees (BTs) [13]. An RL algorithm such as Deep Q-Network (DQN) [64] or PPO is then in charge of learning a policy for the states for which an action is not specified. The fitness of individuals is evaluated over their overall behavior combining the BT part and the learned part, but only the BT part is evolved to generate the next generation, benefiting from a Baldwin effect [91].

Finally, several works consider evolving the morphology of a mechanical system whose control is learned with RL. Table 5 only mentions two recent instances, one where the algorithm is not named [25] and Deep Evolutionary Reinforcement Learning (DERL) [67], but this idea has led to a larger body of works, e.g., References [26, 56].

5.4 Evolution Improved with RL Mechanisms

Without using a full RL part, a few algorithms augment an evolutionary approach with components taken from RL.

First, the Trust Region Evolution Strategies (TRES) algorithm [51] incorporates into an ES several ideas from the TRPO [80] and PPO [81] algorithms, such as introducing an importance sampling mechanism and using a clipped surrogate objective so as to enforce a natural gradient update. Unfortunately, TRES is neither compared to the NES algorithm [106] that also enforces a natural gradient update nor to the safe mutation mechanism of [45] that has similar properties.

Second, there are two perspectives about the previously mentioned ZOAC algorithm [47]. One can see it as close to the ZOSPI algorithm described in Section 3, that is an actor–critic method where gradient descent to update the actor given the critic is replaced by a more robust
derivative-free approach. But the more accurate perspective, as put forward by the authors, is that zoac is an es method where the standard es gradient estimator is replaced by a gradient estimator using the advantage function so as to benefit from the capabilities of the temporal difference methods to efficiently deal with the temporal credit assignment problem.

Finally, with their guided es algorithm [58], the authors study how a simple es gradient estimator can be improved by leveraging knowledge of an approximate gradient suffering from bias and variance. Though their study is general, it is natural to apply it to the context where the approximate gradient is a policy gradient, in which case guided es combines evolution and RL. This work is often cited in a very active recent trend that tries to improve the exploration capabilities of es algorithms by drawing better than Gaussian directions to get a more informative gradient approximator [7, 8, 16, 107]. In particular, the sgEs algorithm [50] leverages both the guided es idea and the improved exploration ideas to produce a competitive es-based policy search algorithm.

6 CONCLUSION

In this article, we have provided a list of all the algorithms combining evolutionary processes and deep reinforcement learning we could find, irrespective of the publication status of the corresponding papers. Our focus was on the mechanisms and our main contribution was to provide a categorization of these algorithms into several groups of methods, based on the role of evolutionary optimization in the architecture.

We have not covered related fields such as algorithms that combine deep RL and imitation learning, though at least one of them also includes evolution [55]. Besides, we have not covered works that focus on the implementation of evolution and deep RL combinations, such as Reference [44], which shows the importance of asynchronism in such combinations.

Despite these limitations, the scope of the survey was still too broad to enable deeper analyses of the different combination methods or a comparative evaluation of their performance. In the future, we intend to focus separately on the different categories so as to provide these more in-depth analyses and perform comparative evaluation of these algorithms between each other and with respect to state-of-the-art deep RL algorithms, based on a unified benchmark.

Our focus on elementary mechanisms also suggests the possibility to design new combinations of such mechanisms, that is, combining the combinations. For instance, one may include into a single architecture the idea of selecting samples sent to the replay buffer so as to maximize the efficiency of the RL component, more efficient crossover or mutation operators as in PDERL, soft policy updates, hyperparameter tuning, and so on. There is no doubt that such combinations will emerge in the future if they can result in additional performance gains, despite the additional implementation complexity.

ACKNOWLEDGMENTS

The author thanks three anonymous reviewers. In addition, the author thanks Giuseppe Paolo, Stéphane Doncieux, and Antonin Raffin for useful remarks about this article, as well as several colleagues from ISIR for their questions and remarks about the algorithms.

REFERENCES

[1] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NeurIPS’17), Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5048–5058.

[2] Thomas Bäck, Ulrich Hammel, and H.-P. Schwefel. 1997. Evolutionary computation: Comments on the history and current state. IEEE Trans. Evol. Comput. 1, 1 (1997), 3–17.
[3] Homanga Bharadhwaj, Kevin Xie, and Florian Shikurti. 2020. Model-predictive control via cross-entropy and gradient-based optimization. *In Learning for Dynamics and Control*. PMLR, 277–286.

[4] Cristian Bodnar, Ben Day, and Pietro Lió. 2020. Proximal distilled evolutionary reinforcement learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI’20), The 32nd Innovative Applications of Artificial Intelligence Conference (IAAI’20)*, the 10th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI’20). AAAI Press, 3283–3290.

[5] Simyung Chang, John Yang, Jaeseok Choi, and Nojun Kwak. 2018. Genetic-gated networks for deep reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS’18)*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 1754–1763.

[6] Gang Chen. 2019. Merging deterministic policy gradient estimations with varied bias-variance tradeoff for effective deep reinforcement learning. arXiv:1911.10527. Retrieved from https://arxiv.org/abs/1911.10527.

[7] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, and Vikas Sindhwani. 2019. From complexity to simplicity: Adaptive ES-active subspaces for blackbox optimization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS’19)*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 10299–10309. https://proceedings.neurips.cc/paper/2019/hash/88bade49e98db8790df275fceb37a13-Abstract.html.

[8] Krzysztof Choromanski, Mark Rowland, Vikas Sindhwani, Richard E. Turner, and Adrian Weller. 2018. Structured evolution with compact architectures for scalable policy optimization. In *Proceedings of the 35th International Conference on Machine Learning (ICML’18), Proceedings of Machine Learning Research, Vol. 80*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 969–977.

[9] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. 2018. Back to basics: Benchmarking canonical evolution strategies for playing atari. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI’18)*, Jérôme Lang (Ed.). 1419–1426. https://doi.org/10.24963/ijcai.2018/197.

[10] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems 31* (2018).

[11] Geoffrey Cideron, Thomas Pierrot, Nicolas Perrin, Karim Beguir, and Olivier Sigaud. 2020. QD-RL: Efficient mixing of quality and diversity in reinforcement learning. arXiv:2006.08505. Retrieved from https://arxiv.org/abs/2006.08505.

[12] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. 2018. GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms. In *Proceedings of the 35th International Conference on Machine Learning (ICML’18), Proceedings of Machine Learning Research, Vol. 80*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 1038–1047.

[13] Michele Colledanchise and Petter Ögren. 2018. *Behavior Trees in Robotics and AI: An Introduction*. CRC Press.

[14] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2018. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS’18)*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 5032–5043.

[15] Antoine Cully and Yiannis Demiris. 2017. Quality and diversity optimization: A unifying modular framework. *IEEE Trans. Evol. Comput.* 22, 2 (2017), 245–259.

[16] Anton Dereventsov, Clayton G. Webster, and Joseph Daws. 2022. An adaptive stochastic gradient-free approach for high-dimensional blackbox optimization. In *Proceedings of International Conference on Computational Intelligence*. Springer, 333–348.

[17] Thang Doan, Bogdan Mazoure, Audrey Durand, Joelle Pineau, and R. Devon Hjelm. 2019. Attraction-repulsion actor-critic for continuous control reinforcement learning. arXiv:1909.07543. Retrieved from https://arxiv.org/abs/1909.07543.

[18] Stephane Doncieux, Alban Laflaquière, and Alexandre Conanx. 2019. Novelty search: A theoretical perspective. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 99–106.

[19] Madalina M. Drugan. 2019. Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. *Swarm Evol. Comput.* 44 (2019), 228–246.

[20] Federico Espositi and Andrea Bonarini. 2020. Gradient bias to solve the generalization limit of genetic algorithms through hybridization with reinforcement learning. In *International Conference on Machine Learning, Optimization, and Data Science*. Springer, 273–284.

[21] Jörg K. H. Franke, Gregor Köhler, André Biedenkapp, and Frank Hutter. 2021. Sample-efficient automated deep reinforcement learning. In *Proceedings of the 9th International Conference on Learning Representations (ICLR’21)*. OpenReview.net.
Combining Evolution and Deep Reinforcement Learning for Policy Search

[22] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML ’18)*, Proceedings of Machine Learning Research, Vol. 80. Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 1582–1591.

[23] Tanmay Gangwani and Jian Peng. 2018. Policy optimization by genetic distillation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR ’18)*. OpenReview.net. https://openreview.net/forum?id=ByOmniWwC-

[24] Jake Grigsby, Jin Yong Yoo, and Yanjun Qi. 2021. Towards automatic actor-critic solutions to continuous control. arXiv:2106.08918. Retrieved from https://arxiv.org/abs/2106.08918.

[25] Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. 2021. Embodied intelligence via learning and evolution. arXiv:2102.02202. Retrieved from https://arxiv.org/abs/2102.02202.

[26] David Ha. 2019. Reinforcement learning for improving agent design. *Artif. Life* 25, 4 (2019), 352–365.

[27] Tuomas Haarnoja, Auret Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft actor-critic algorithms and applications. arXiv:1812.05905. Retrieved from https://arxiv.org/abs/1812.05905.

[28] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. 2019. Learning latent dynamics for planning from pixels. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2555–2565.

[29] Ahmed Hallawa, Thorsten Born, Anke Schmeink, Guido Dartmann, Arne Peine, Lukas Martin, Giovanni Iacca, AE Eiben, and Gerd Ascheid. 2021. Evo-RL: Evolutionary-driven reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 153–154.

[30] Ahmed Hallawa, Jaro De Roose, Martin Andraud, Marian Verhelst, and Gerd Ascheid. 2017. Instinct-driven dynamic hardware reconfiguration: Evolutionary algorithm optimized compression for autonomous sensory agents. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1727–1734.

[31] John H. Holland and Judith S. Reitman. 1978. Cognitive systems based on adaptive algorithms. In *Pattern-directed Inference Systems*. Elsevier, 313–329.

[32] Reih Houthooft, Yuhua Chen, Phillip Isola, Bradly C. Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. 2018. Evolved policy gradients. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS ’18)*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 5405–5414.

[33] Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, et al. 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* 364, 6443 (2019), 859–865.

[34] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. 2017. Population-based training of neural networks. arXiv:1711.09846.

[35] Whiyoung Jung, Giseung Park, and Youngchul Sung. 2020. Population-guided parallel policy search for reinforcement learning. In *Proceedings of the 8th International Conference on Learning Representations (ICLR ’20)*. OpenReview.net.

[36] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. 2018. QT-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. arXiv:1806.10293. Retrieved from https://arxiv.org/abs/1806.10293.

[37] Shauharda Khadka, Somdeb Majumdar, Tarek Nassar, Zach Dwiel, Evren Tumer, Santiago Miret, Yinyin Liu, and Kagan Tumer. 2019. Collaborative evolutionary reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML ’19)*, Proceedings of Machine Learning Research, Vol. 97. Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3341–3350.

[38] Shauharda Khadka and Kagan Tumer. 2018. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS ’18)*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 1196–1208.

[39] Mehdi Khamassi, George Velentzas, Theodore Tsitsipis, and Costas Tzafestas. 2017. Active exploration and parameterized reinforcement learning applied to a simulated human-robot interaction task. In *Proceedings of the 1st IEEE International Conference on Robotic Computing (IRC ’17)*. IEEE, 28–35.

[40] Kyung-Joong Kim, Heejin Choi, and Sung-Bae Cho. 2007. Hybrid of evolution and reinforcement learning for othello players. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. IEEE, 203–209.

[41] Namyoung Kim, Hyunsuk Baek, and Hayong Shin. 2020. PGPS: Coupling policy gradient with population-based search (unpublished).

[42] John R. Koza et al. 1994. *Genetic Programming II*. Vol. 17. MIT Press, Cambridge, MA.
[43] Pier Luca Lanzi. 1999. An analysis of generalization in the XCS classifier system. *Evol. Comput.*, 7, 2 (1999), 125–149.

[44] Kyunghyun Lee, Byeong-Uk Lee, Ukcheol Shin, and In So Kweon. 2020. An efficient asynchronous method for integrating evolutionary and gradient-based policy search. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems* (NeurIPS’20), Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Husnu-Tien Lin (Eds.).

[45] Joel Lehman, Jay Chen, Jeff Clune, and Kenneth O. Stanley. 2018. Safe mutations for deep and recurrent neural networks through output gradients. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 117–124.

[46] Joel Lehman and Kenneth O. Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evol. Comput.*, 19, 2 (2011), 189–223.

[47] Yuheng Lei, Jianyu Chen, Shengbo Eben Li, and Sifa Zheng. 2022. Zeroth-order actor-critic. arXiv:2201.12518. Retrieved from https://arxiv.org/abs/2201.12518.

[48] Abe Leite, Madhavun Candadai, and Eduardo J. Izquierdo. 2020. Reinforcement learning beyond the Bellman equation: Exploring critic objectives using evolution. In *Proceedings of the Conference on Artificial Life* (ALIFE’20).

[49] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations* (ICLR’16), Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1509.02971

[50] Fei-Yu Liu, Zi-Niu Li, and Chao Qian. 2020. Self-guided evolution strategies with historical estimated gradients. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI’20)*, Christian Bessiere (Ed.). ijcai.org, 1474–1480. https://doi.org/10.24963/ijcai.2020/205

[51] Guoqing Liu, Li Zhao, Feidiao Yang, Jiang Bian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2019. Trust region evolution strategies. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence* (AAAI’19), the 31st Innovations Applications of Artificial Intelligence Conference (EAAI’19), the 9th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI’19). AAAI Press. 4352–4359. https://doi.org/10.1609/aaai.v33i01.33014352

[52] Jian Liu and Liming Feng. 2021. Diversity evolutionary policy deep reinforcement learning. *Comput. Intell. Neurosci.* (2021).

[53] Qihao Liu, Yujia Wang, and Xiaofeng Liu. 2018. PNS: Population-guided novelty search for reinforcement learning in hard exploration environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS’18). IEEE, 5627–5634.

[54] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. 2017. Stein variational policy gradient. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence* (UAI’17), Gal Elidan, Kristian Kersting, and Alexander T. Iler (Eds.). AUAI Press.

[55] Shuai Lü, Shuai Han, Wenbo Zhou, and Junwei Zhang. 2021. Recruitment-imitation mechanism for evolutionary reinforcement learning. *Inf. Sci.* 553 (2021), 172–188.

[56] Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. 2020. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Proceedings of the Conference on Robot Learning*. PMLR. 854–869.

[57] Yan Ma, Tianxing Liu, Bingsheng Wei, Yi Liu, Kang Xu, and Wei Li. 2022. Evolutionary action selection for gradient-based policy learning. arXiv:2201.04286. Retrieved from https://arxiv.org/abs/2201.04286.

[58] Niru Maheswaranathan, Luke Metz, George Tucker, Dami Choi, and Jascha Sohl-Dickstein. 2019. Guided evolutionary-ary search with surrogate gradients. In *Proceedings of the 36th International Conference on Machine Learning* (ICML’19), Proceedings of Machine Learning Research, Vol. 97, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 4264–4273.

[59] Amjad Yousef Majid, Serge Saaybi, Tomas van Rietbergen, Vincent Francois-Lavet, R. Venkatesha Prasad, and Chris Verhoeven. 2021. Deep reinforcement learning versus evolution strategies: A comparative survey. arXiv:2110.01411. Retrieved from https://arxiv.org/abs/2110.01411.

[60] Horia Mania, Aurelia Guy, and Benjamin Recht. 2018. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems* (NeurIPS’18), Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 1805–1814.

[61] Enrico Marchesini, Davide Corsi, and Alessandro Farinelli. 2021. Genetic soft updates for policy evolution in deep reinforcement learning. In *Proceedings of the 9th International Conference on Learning Representations* (ICLR’21). OpenReview.net.

[62] Julian Francis Miller and Simon L. Harding. 2009. Cartesian genetic programming. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. 3489–3512.

[63] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1928–1937.
[87] Zhenyang Shi and Surya P. N. Singh. 2021. Soft actor-critic with cross-entropy policy optimization. arXiv:2112.11115. Retrieved from https://arxiv.org/abs/2112.11115.

[88] Olivier Sigaud and Freek Stulp. 2019. Policy search in continuous action domains: An overview. Neural Netw. 113 (2019), 28–40.

[89] Olivier Sigaud and S. W. Wilson. 2007. Learning classifier systems: A survey. J. Soft Comput. 11, 11 (2007), 1065–1078.

[90] Riley Simmons-Edler, Ben Eisner, Eric Mitchell, Sebastian Seung, and Daniel Lee. 2019. Q-learning for continuous actions with cross-entropy guided policies. arXiv:1903.10605. Retrieved from https://arxiv.org/abs/1903.10605.

[91] George Gaylord Simpson. 1953. The baldwin effect. Evolution 7, 2 (1953), 110–117.

[92] Jörg Stork, Martin Zaefferer, Nils Eisler, Patrick Tichelmann, Thomas Bartz-Beielstein, and A. E. Eiben. 2021. Behavior-based neuroevolutionary training in reinforcement learning. In Proceedings of the Genetic and Evolutionary Computation Conference Companion. 1753–1761.

[93] Hao Sun, Ziping Xu, Yuhang Song, Meng Fang, Jiechao Xiong, Bo Dai, and Bolei Zhou. 2020. Zeroth-order supervised policy improvement. arXiv:2006.06600. Retrieved from https://arxiv.org/abs/2006.06600.

[94] Karush Suri, Xiao Qi Shi, Konstantinos N. Plataniotis, and Yuri A. Lawryshyn. 2020. Maximum mutation reinforcement learning for scalable control. arXiv:2007.13690. Retrieved from https://arxiv.org/abs/2007.13690.

[95] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. Mach. Learn. 8, 3 (1992), 279–292.

[96] Bruce H. Weber and David J. Depew. 2003. Evolution and Learning: The Baldwin Effect Reconsidered. MIT Press.

[97] Han Zheng, Pengfei Wei, Jing Jiang, Guodong Long, Qinghua Lu, and Chengqi Zhang. 2020. Cooperative heterogeneous deep reinforcement learning. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS’20), Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

Received 17 June 2022; revised 3 October 2022; accepted 18 October 2022

ACM Transactions on Evolutionary Learning, Vol. 3, No. 3, Article 10. Publication date: September 2023.