Bridging Evolutionary Algorithms and Reinforcement Learning: A Comprehensive Survey on Hybrid Algorithms

Pengyi Li, Jianye Hao, *Senior Member, IEEE*, Hongyao Tang, Xian Fu, Yan Zheng, and Ke Tang, *Fellow, IEEE*

***Abstract*—Evolutionary reinforcement learning (ERL), which integrates the evolutionary algorithms (EAs) and reinforcement learning (RL) for optimization, has demonstrated remarkable performance advancements. By fusing both the approaches, ERL has emerged as a promising research direction. This survey offers a comprehensive overview of the diverse research branches in ERL. Specifically, we systematically summarize the recent advancements in related algorithms and identify three primary research directions: 1) EA-assisted optimization of RL; 2) RL-assisted optimization of EA; and 3) synergistic optimization of EA and RL. Following that, we conduct an in-depth analysis of each research direction, organizing multiple research branches. We elucidate the problems that each branch aims to tackle and how the integration of EAs and RL addresses these challenges. In conclusion, we discuss potential challenges and prospective future research directions across various research directions. To facilitate researchers in delving into ERL, we organize the algorithms and codes involved on https://github.com/yeshenpy/Awesome-Evolutionary-Reinforcement-Learning.**

***Index Terms*—Evolutionary algorithms (EAs), evolutionary reinforcement learning (ERL), reinforcement learning (RL).**

I. Introduction

REINFORCEMENT learning (RL) [1] is an important category of learning methods within the field of machine learning [2], specializing in solving various sequential decision-making problems (SDPs). By modeling SDPs as the Markov decision processes (MDPs) [3], RL learns an optimal policy via the iterative policy optimization and evaluation with various optimization techniques (e.g., gradient descent [4]). Thanks to the development of deep learning, the capabilities of RL have been further enhanced. By leveraging the powerful expressive capabilities of neural networks and efficient gradient optimization, RL can approximate more complex value functions and demonstrate superior learning efficiency compared to the gradient-free methods [5]. In addition, RL, especially the off-policy RL, collects and reuses historical samples, significantly improving sample efficiency and making it more applicable to problems where samples are costly [6]. With these advancements, RL has achieved significant successes in diverse domains, including game AI [7], robotics [8], recommender system [9], and scheduling problems (SPs) [10]. However, RL still faces several inherent and long-standing challenges, including limited exploration abilities [11], poor convergence [12], sensitivity to hyperparameters [13], and suboptimality in gradient optimization [14]. These challenges hinder the application of RL in more complex real-world scenarios.

Evolutionary algorithms (EAs) [15], [16], [17], [18], [19] are a class of gradient-free, black-box optimization methods. By emulating the Darwin's theory of evolution, EAs iteratively evolve solutions. Due to the diversity within populations and the gradient-free random search, EAs have strong exploration ability. As a result, compared to conventional local search algorithms like gradient descent, EAs exhibit better global optimization capabilities within the solution space and are adept at solving multimodal problems [17], [20]. Moreover, EAs show good robustness and convergence properties, displaying resistance to noise and uncertainty [21]. With these characteristics, EAs have demonstrated formidable capabilities in various practical optimization problems [22], [23], including path planning [24], SPs [25], and circuit design [26]. Besides, EAs have also demonstrated the capacity to evolve a single policy for solving the multitask sequential decision problems [27], [28], [29]. Nonetheless, EAs also have weaknesses, including sensitivity to the design and selection of variation operators [30], as well as ineffective and redundant exploration arising from the random search [5]. Besides, EAs often demonstrate low sample efficiency in SDPs [12], [31], [32].

As discussed above, RL and EAs are two categories of methods based on different principles. Each has proven its efficiency in addressing specific problems but also faces

Manuscript received 23 January 2024; revised 7 June 2024; accepted 25 July 2024. Date of publication 14 August 2024; date of current version 8 October 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 92370132 and Grant 62106172; in part by the Xiaomi Young Talents Program of Xiaomi Foundation; in part by the National Project X of China under Grant JCKY2021204B104; and in part by the Science and Technology on Information Systems Engineering Laboratory under Grant WDZC20235250409 and Grant 6142101220304. This article was approved by Associate Editor G. Iacca. *(Corresponding author: Jianye Hao.)*

Pengyi Li, Jianye Hao, Xian Fu, and Yan Zheng are with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, China (e-mail: lipengyi@tju.edu.cn; jianye.hao@tju.edu.cn; xianfu@tju.edu.cn; yanzheng@tju.edu.cn.)

Hongyao Tang is with the Montreal Institute of Learning Algorithms, Montreal, QC H3A 0G4, Canada (e-mail: tang.hongyao@mlia.quebec).

Ke Tang is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: tangk3@sustech.edu.cn).

Digital Object Identifier 10.1109/TEVC.2024.3443913

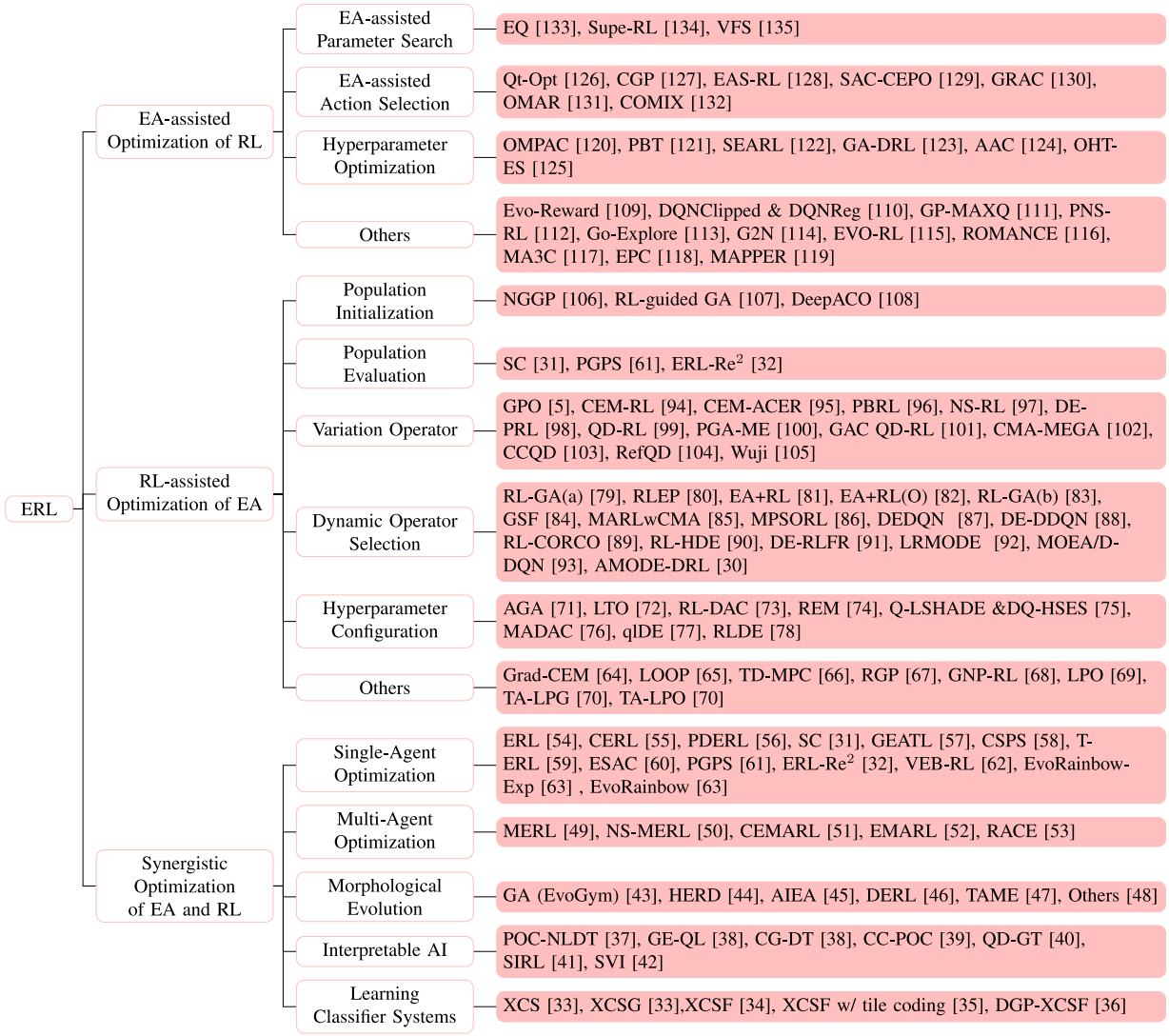


Fig. 1. Three major research directions in the ERL field. Each direction comprises multiple research branches.

different challenges. With the development of the EAs and RL communities, many researchers have found that these challenges can be addressed by combining advanced methods from both the fields, leading to the emergence of numerous hybrid methods. For brevity, we refer to these works as evolutionary RL (ERL). However, the technical pathways of these methods and the problems they aim to address are various and diverse. There is a lack of a comprehensive survey and systematic analysis in the ERL literature. In this article, we attempt to provide a systematic review of the existing ERL works. We first revisit these works from the following three primary perspectives.

1) *From the Perspective of RL:* RL is a class of learning algorithms used to tackle various SDPs. Apart from modeling problems as MDPs, RL involves configuring algorithms, interaction, and learning and optimizing network parameters. This process encounters many optimization challenges, such as hyperparameter optimization, action selection, and network parameter optimization. EAs exhibit strong search capabilities and global optimization prowess, which can further enhance the quality of solving such optimization subproblems within RL.
2) *From the Perspective of EAs:* EAs are a category of optimization algorithms that require iterative searches in the solution space to obtain feasible solutions. This typically involves population initialization, evaluation, operator design and selection, and algorithm configuration. However, this process often confronts assorted hurdles, such as how to construct the initial population, which often determines the quality of the final solution; how to perform efficient mutation that avoids redundant and inefficient exploration; and how to dynamically select operators at different stages of optimization to improve the performance. The learning ability of RL enables it to develop strong discriminative guidance and decision-making capabilities. Incorporating RL into the optimization process of EAs has been proven to further enhance the efficiency of EAs.

3) *From the Perspective of Collaboration:* EAs and RL can collaborate to solve a problem, typically through two approaches: a) EAs and RL simultaneously address the same problem and collaborate through some mechanisms and b) EAs and RL each solve a part of the problem and eventually integrate to form a complete solution. The former approach aims to complement the strengths and weaknesses of EAs and RL in problem-solving: EAs' exploration capability compensates for RL's exploration limitations, while RL's experience reuse and fine grained learning address EAs' sample inefficiency. The latter approach involves EAs and RL tackling tasks they excel in individually; for instance, EAs optimizing topology and RL learning control policies.

Based on the insights mentioned above, we categorize the ERL works into three main directions: 1) *EA-assisted Optimization of RL*. This integration approach incorporates EAs into the learning process of RL, mainly applied to address SDPs. It leverages diverse exploration, global optimization capabilities, and strong convergence and robustness of EAs to address challenges in RL such as limited exploration capabilities, suboptimality in gradient optimization, and sensitivity to hyperparameters. 2) *RL-assisted Optimization of EA*. Opposite to the previous category, this integration approach incorporates RL into the optimization flow of EAs, mainly applied to address various optimization problems and SDPs. It leverages efficient experience utilization and learning capabilities, along with the discriminative abilities of RL to address challenges in EAs such as population initialization, algorithm configuration, uncontrollable mutation, and high sample cost in fitness evaluation. 3) *Synergistic Optimization of EA and RL*. This integrated approach maintains complete processes for both EAs and RL to collaboratively address the same problem simultaneously or independently tackle partial solutions, which are subsequently integrated into a complete solution. This allows each of them to leverage their respective strengths in problem-solving and mutually enhance each other through complementary features, ultimately achieving better performance. We provide an overview in Fig. 1.

There have been some efforts to review works related to EAs and RL, including comparing EAs with RL algorithms [136], exploring the integration of EAs and RL for the policy search [12], [137], reviewing the hybrid algorithms within an unified framework, including motivation, natural models, subalgorithms, techniques, and properties [138], reviewing hybrid algorithms based on the challenges they address in RL [139], and reviewing the hybrid algorithms according to the key research areas of RL [140]. However, these surveys lack comprehensive and systematic investigation into the hybrid algorithms. Thus, we aim to furnish a more systematic and comprehensive survey to fill this gap. We summarize our contributions as follows.

1) We conduct a thorough and systematic analysis of the works in the ERL domain, leading to the establishment of three research directions: a) EA-assisted optimization of RL, b) RL-assisted optimization of EA, and c) synergistic optimization of EA and RL.

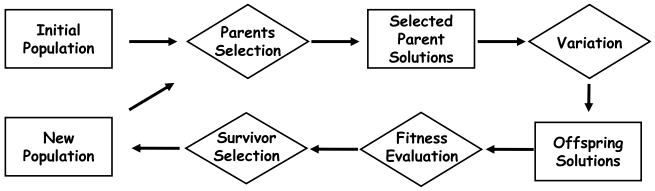


Fig. 2. EA optimization process. Diamond-shaped blocks represent the actions taken by the algorithm, while the rectangular blocks represent instances generated by the actions.

2) In each direction, we further subdivide the works into distinct research branches. Subsequently, we conduct an in-depth analysis of the fundamental issues to be addressed and the related algorithms for each branch.
3) Furthermore, we point out open challenges within each domain and propose potential research directions to address these challenges.

We begin by providing an overview of the basic algorithms in EAs and RL and definitions of involved problems in Section II. Subsequently, we delve into each direction to categorize the related works and present them from various branches, outlining the specific problems they address and the proposed approaches. After presenting the various branches within each research direction, we summarize open challenges and discuss the potential future directions.

II. Background

A. Evolutionary Algorithms

EAs are a class of biologically inspired gradient-free optimization methods that emulate the biological evolution processes [15], [17], [19]. Fig. 2 illustrates the entire evolutionary process. EAs begin by initializing the population with an initial set of individuals $\mathbb{P} = \{I_1, I_2, \ldots, I_N\}$. These individuals can have various forms in different problems, such as vectors, neural networks, and so on. Subsequently, EAs select parents using a selection operator, and offspring individuals are generated through variation. The selection operator typically involves choosing the individuals with the highest fitness scores determined through evaluation. Following this, the offspring are evaluated for their fitness, followed by survivor selection. The selected individuals form the new population for the next generation. All the EAs can be abstracted into the aforementioned process. In this survey, we cover various EAs [19], such as genetic algorithm (GA) [23], [141], differential evolution (DE) [142], evolution strategy (ES) [143], novelty search (NS) [144], MAP-elites [100], [145], genetic programming (GP) [146], and others. Fixed genomes are a common feature across GA, DE, and ES. In ES and DE, genomes are typically represented as real-valued vectors, whereas GA utilizes the binary strings. Recently some works have expanded the forms of these methods' genomes, allowing them to employ neural networks as genomes [12]. While NS and MAP-elites are the two diversity mechanisms. GP stands out for its ability to handle variable-length genomes, such as tree structures, which enables the exploration of the complex topology. It is worth noting that in SDPs, existing

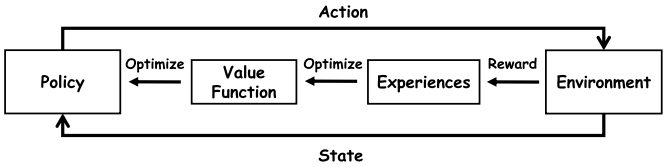


Fig. 3. RL process.

EAs typically utilize the policies, i.e., neural networks, as individuals in the population, and rely on these policies for action decision-making and interaction. Ultimately, they evaluate the population based on the averaged cumulative rewards obtained throughout several episodes of games, followed by evolution in the parameter space [32], [94]. We can observe that EAs struggle to utilize the finer-grained information, such as states, actions, and step-level rewards. This is one major factor leading to sample inefficiency.

B. Reinforcement Learning

RL needs to formalize the target problem as MDPs [1], where the agent interacts with the environment over several finite time steps. The MDP can be defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, T)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, γ is the discount factor, and T is the maximum episode length. At each time step t, the agent chooses an action $a_t \in \mathcal{A}$ according to the state $s_t \in \mathcal{S}$ and its policy $\pi(s) : \mathcal{S} \rightarrow \mathcal{A}$, receives a reward $r_t \in \mathcal{R}(s_t, a_t)$, and gets the next state s_{t+1} based on $\mathcal{P}(s_{t+1}|s_t, a_t)$. RL aims to find a policy that maximizes the cumulative discounted rewards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ at each time step t. The learning process is shown in Fig. 3. Q-learning [147] is a classical value-based RL algorithm that uses a Q-function $Q(s, a)$ to learn the Q-value, i.e., the cumulative reward obtained from taking a specific action a in a given state s. Q-learning selects actions based on the maximum Q-value $\pi = \arg\max_a Q(s, a)$ and employs the Bellman equation to update the Q-function toward the target y using $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(y - Q(s_t, a_t))$ and $y = \sum_{t=i}^{i+n-1} \gamma^{t-i} r_t + \gamma^n \max_a Q(s_{t+n}, a)$, where n denotes the number of steps considered for the future rewards, larger values of n improve the accuracy of the value function approximation by incorporating more reward signals. However, this also leads to increased variance. This approach is referred to as n-step TD, where n is often set to 1, considering only the one-step reward. the policy gradient methods [148], [149] are a class of the policy-based RL algorithms that directly optimize the policy of an agent. The basic idea is to update these parameters in the direction that increases the expected cumulative reward $\bar{R}_\theta = \mathbb{E}_\tau[R(\tau)]$, which is often done using techniques like gradient ascent. One classical algorithm is REINFORCE [148], which computes the gradient of $\bar{R}_\theta$ for the policy parameters using $\nabla \bar{R}_\theta = (1/M) \sum_{k=1}^{M} \sum_{t=1}^{T} G_t^k \nabla \log \pi_\theta(a_t^k|s_t^k)$ and $G_t^k = \sum_{i=t}^{T} \gamma^{i-t} r_i^k$, where G_t^k represents the cumulative discounted rewards from a specific time step t until the end of episode k. However, REINFORCE can only conduct the policy improvement after completing one episode, which leads to inefficiency. To solve the problem, the actor–critic method (AC) [150] integrates value-based RL into the policy gradient updates. This is accomplished by training a critic to supply one-step gradients for the policy, i.e., the actor. The development of deep learning has further enhanced the capabilities of RL, leading to the proposal of many deep RL algorithms, including DQN [151], [152], DDPG [153], TD3 [154], SAC [155], TRPO [156], PPO [157], and others.

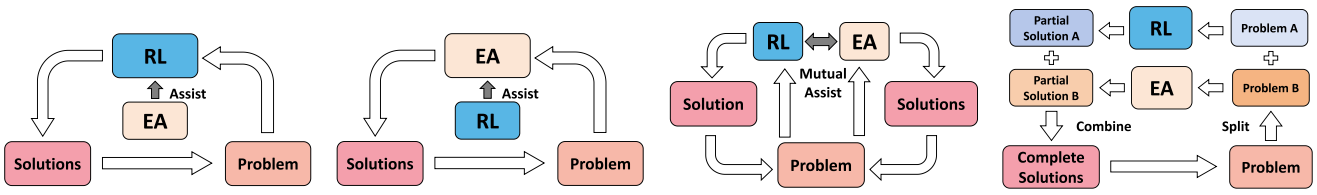
C. Problem Definition

Based on the tasks addressed by the hybrid algorithms, we categorize them into five major classes: 1) SDPs; 2) continuous optimization problems (CTOPs); 3) combinatorial optimization problems (COPs); 4) multiobjective optimization problems (MOOPs); and 5) multimodal optimization problems (MMOPs). *SDPs* involves modeling the task as an MDP. Through the policy control, the agent interacts with the environment and receives rewards. The ultimate goal is to obtain a policy that maximizes the cumulative rewards. Note that, other optimization problems can also be modeled as sequential decision problems. For better distinction, we specify the primary tasks involved here: maze navigation and robot control problems, including MuJoCo [158] and DMC [159]. Additionally, we also explore multiagent settings, which require training multiple policies to control multiple agents interacting in the environment, aiming to maximize the team rewards. The related tasks covered in this article include SMAC [160], MAMuJoCo [161], MPE [162], and flocking tasks [52]. In addition, this survey involves the other four types of optimization problems, including *CTOPs* [163], *COPs* [164], *MOOPs*, and *MMOPs*. CTOP and COP require finding a set of variables x in continuous or discrete spaces to either maximize or minimize the objective function $f(x)$. Typically, such problems come with constraints. CTOP primarily involves the CEC benchmark [165], [166], [167], while COP involves the traveling salesman problem (TSP) [168], vehicle routing problem (VRP) [169], SPs [170], and others. MOOP involves multiple objective functions, rather than a single one. In MOOP, our goal shifts from seeking a solution to minimize or maximize a single objective function, to finding a set of solutions where each represents an optimal solution across different preferences. Unlike MOOP, MMOP deals with situations in single or multiobjective optimization where a single objective function has multiple local optima. The objective of MMOP is to find all or as many local suboptimal solutions as possible.

III. EA-Assisted Optimization of RL

This section offers a comprehensive analysis of how EAs can enhance RL. In this optimization process, the related algorithms revolve around RL, with EAs playing a supporting role in refining this process. EAs cannot solve the problem independently. The optimization schematic is illustrated in Fig. 4(a). These works primarily focus on SDPs, which are the focus of most RL algorithms.

According to the optimization objective of EA, we classify the related methods into four branches: 1) EA-assisted parameter search; 2) EA-assisted action selection;



(a) (b) (c) (d)

Fig. 4. Schematic of four integration approaches. (a) EA-assisted optimization of RL. RL conducts search and improvement of the solution, with EAs playing a supporting role. EAs cannot independently optimize solutions. (b) RL-assisted optimization of EA. EAs conduct search and improvement of the solution, with RL playing a supporting role; RL cannot independently optimize solutions. (c) and (d) Synergistic optimization of EA and RL.

TABLE I
EA-ASSISTED OPTIMIZATION OF RL

Class	Algorithm	Task	EA	RL	Detail
Assisted Parameter Search	EQ [133]	MuJoCo	GA	DDPG	Critic Parameter
	Supe-RL [134]	Navigation & MuJoCo	GA	PPO & Rainbow	Policy Parameter
	VFS [135]	MuJoCo	GA	DDPG & PPO & TD3	Critic Parameter
Assisted Action Selection	Qt-Opt [126]	Real Robot Control	CEM	Double Q	Critic Update; Interaction
	CGP [127]	MuJoCo	CEM	Double Q	Critic Update; Policy Update
	EAS-RL [128]	MuJoCo	PSO	TD3	Policy Update
	SAC-CEPO [129]	MuJoCo	CEM	SAC	Policy Update
	GRAC [130]	MuJoCo & Real Robot Control	CEM	Double Q-learning	Critic Update; Policy Update
	OMAR [131]	MPE & D4RL	CEM-like	MA-CQL & CQL	Critic Update; Interaction
	COMIX [132]	MPE & MA-MuJoCo	CEM	QMIX	Policy Update
Hyperparameter Optimization	OMPAC [120]	Atari & Tetris	GA	TD(λ) Sarsa(λ)	λ, lr, τ_a, γ
	PBT [121]	DM Lab & Atari & StarCraft II	GA-like	UNREAL & FuN & A3C	lr, Entropy Cost, Unroll Length, Intrinsic Reward Cost
	SEARL [122]	MuJoCo	GA	TD3	Parameters, γ, Architecture, Activation
	GA-DRL [123]	Gym Robotics & AuboReach	GA	DDPG	$\gamma, lr, \epsilon, \tau, \eta$ (noise std)
	AAC [124]	DMC & Industrial Benchmark	GA	SAC	γ, h, k, a,c
	OHT-ES [125]	DMC	ES	TD3	n, lr
Others	Evo-Reward [109]	Hungry–Thirsty	PushGP	Q-learning	Reward Function Search
	DQNClipped & DQNReg [110]	MiniGrid & Atari	GA-like	DQN	Loss Function Search
	GP-MAXQ [111]	Foraging Task	GP	MAXQ	Hierarchy Search
	PNS-RL [112]	Gym Robotics & MuJoCo	NS	TD3	Improve Exploration
	Go-Explore [113]	Atari	GA-like	Imitation Learning	Improve Exploration
	G2N [114]	Atari & MuJoCo	GA	A2C or PPO	Activation of neurons
	EVO-RL [115]	Gym Control	GP	Q-learning & DQN & PPO	Special Setting
	ROMANCE [116]	SMAC	QD	QMIX	Robust MARL
	MA3C [117]	SMAC, Hallway & Traffic Junction & Gold Panner	GA-like	CMARL	Robust Communication
	EPC [118]	Particle-world Environment	GA-like	MADDPG	Large-Scale MARL
	MAPPER [119]	Grid world	GA-like	A2C	Improve Stability
	LPO [69]	Brax & MinAtar	ES	PPO	Accelerate and Improve Meta RL
	TA [70]	Grid world & Invaders & Brax & MinAtar	ES	LPO & LPG	Aware Learning Time Remaining

3) hyperparameter optimization; and 4) others. Specifically, EA-assisted parameter search focuses on leveraging the optimization characteristics of EAs to further improve RL. EA-assisted action selection primarily focuses on the challenge of dealing with a vast action space in continuous action settings, where it is difficult to determine the optimal

actions. The goal is to utilize EAs for action search to improve the optimization and decision-making process of RL. Hyperparameter optimization focuses on using EAs to automatically adjust RL hyperparameters to mitigate RL hyperparameter sensitivity and improve RL convergence performance. Others encompass the works that leverage EAs to enhance RL from different perspectives, e.g., reward function search, loss function search, and exploration. Below, we provide a detailed introduction to each branch and the related works involved.

EA-Assisted Parameter Search: The ultimate objective of DRL is to train a policy network capable of selecting actions that maximize the cumulative rewards. However, RL with a single policy often exhibits weak exploration capabilities, and gradient-based optimization can easily lead neural networks into the local optima, impeding the ability to achieve better performance [171], To solve the problems, researchers propose integrating EAs with RL to assist the parameter optimization.

EQ [133] employs EAs to replace the conventional Bellman equation for the critic optimization. Specifically, EQ maintains a critic population, where each critic optimizes a corresponding actor with the policy gradients. The fitness of each critic is based on the scores derived from the interactions between its respective actor and the environment. The critics with high fitness are selected as parents and perturbed with the Gaussian noise to generate the offspring. The experiments on the inverted pendulum task demonstrate that using EAs can achieve better guidance capabilities than using the conventional Bellman optimization while maintaining comparable performance. Supe-RL [134] employs GA to the auxiliary policy parameter search. At regular intervals, a population of policies is initialized by introducing perturbations to the current RL policy. The population interacts with the environment, and elite individuals soft update their parameters to the RL policy. In the off-policy RL, Supe-RL incorporates the elite experiences collected in the genetic evaluation phase into the RL replay buffer. The experiments show that Supe-RL can enhance Rainbow and PPO on navigation tasks and the MuJoCo environment, respectively. VFS [135] employs the same idea as Supe-RL but differs by the periodic construction of a critic population. The population is evolved by introducing perturbations of differing scales to the critics. Eventually, the critic with the least deviation from the true value function is chosen to replace the RL critic. The true value function is approximated using unbiased Monte Carlo estimation. VFS demonstrates improvements across various algorithms on the MuJoCo tasks, including PPO, DDPG, TD3, and SAC.

EA-Assisted Action Selection: Action selection runs throughout the entire process of RL improvement and evaluation. It primarily involves optimizing the value function, i.e., computing target values, and interacting with the environment. However, on continuous action tasks, the action space can be vast, making it challenging to determine the optimal actions for optimization and execution, especially in situations involving multimodality or multiple peaks. Traditional RL often employs the greedy policies or random sampling from distributions for action selection. However, this approach struggles to accurately capture the best behaviors. Therefore, some works propose the concept of action evolution: initializing a population of actions, evaluating their quality using Q-values as fitness, and then selecting the elite action for optimization and interaction.

Qt-Opt [126] does not explicitly maintain a policy network, it initializes a population of random actions from the action space. Under the current state, Qt-Opt applies two iterations of the cross-entropy method (CEM) [172] to the population, guided by the Value Function. The best action is then selected for the critic optimization and interaction. Qt-Opt demonstrates its efficiency in real-world robotic visual grasping tasks. While retaining the advantages of Qt-Opt, CGP [127] reduces computational burdens by introducing a policy to mimic actions sampled by CEM. In CGP, the critic's training uses CEM-derived actions. Concurrently, a policy is trained using the behavior cloning or policy gradients. CGP validates its efficiency on the MuJoCo tasks. EAS-RL [128] uses a similar idea to CGP, yet it distinguishes itself in two aspects: 1) employing TD3 with the original optimization process and replacing CEM with PSO and 2) integrating both the behavior cloning and policy gradients to optimize the policy. EAS-RL outperforms many ERL-related works on the MuJoCo tasks [54], [55], [56], [94]. SAC-CEPO [129] employs a stochastic policy, SAC, instead of a deterministic policy in CGP. SAC-CEPO divides SAC into two parts: 1) a mean network and 2) a deviation network. CEM is employed to select the best mean actions, while the mean network is trained through the behavior cloning, and the deviation network is learned using the SAC policy gradient. Besides, the generation of the CEM population is no longer based on the random sampling, but on sampling from a normal distribution based on learned mean and deviation networks. SAC-CEPO demonstrates improvements over SAC in MuJoCo tasks. GRAC [130] introduces three mechanisms to improve double Q-learning. We primarily focus on two mechanisms closely aligned with EAs: 1) CEM policy improvement and 2) max-min double Q-learning. Similar to previous methods, CEM policy improvement employs CEM to search for the optimal actions, the difference is that the CEM policy improvement uses the Q-value differences between the optimal actions and the actions taken by the RL policy as advantages to increase the probability of the RL policy selecting the optimal actions. Max-min double Q-learning is proposed to address the underestimation problem of double Q-learning. Specifically, GRAC obtains the CEM action and RL action based on the next state. Subsequently, it utilizes double Q networks to estimate the minimum Q value for the CEM action and the minimum Q value for the RL action. The higher of the two values is selected as the target value. GRAC demonstrates significant performance gains in MuJoCo tasks and is also validated on the real robots.

In the offline setting, the same problem is demonstrated to exist: the policy gradient improvements tend to get stuck in local optima due to the complex nature of the value function landscape. To solve the problem, OMAR [131] employs a modified version of CEM to select the most optimal actions and uses behavior cloning to fine tune the policy. The efficiency of OMAR is validated through the

experiments on the MPE and D4RL benchmarks. In the multiagent setting, COMIX [132] aims to address the problem of QMIX's inapplicability in continuous action spaces [160]. Within COMIX, QMIX is utilized for the value function approximation, followed by CEM for the action selection. The experiments demonstrate that COMIX outperforms MADDPG in MPE and MA-MuJoCo tasks.

EA-Assisted Hyperparameter Optimization: While DRL has shown remarkable prowess across diverse domains, it still suffers from the notorious issue of hyperparameter sensitivity, resulting in a complex and expensive tuning process. Many works solve the problems by incorporating EAs to tune hyperparameters for RL.

OMPAC [120] employs GA for RL hyperparameter optimization, including the trace-decay rate λ, discount factor γ, learning rate (lr), and temperature τ_a of softmax action selection. OMPAC trains a population of RL individuals with different hyperparameter configurations and evaluates individuals based on cumulative rewards. After a certain number of iterations, the hyperparameters of nonelite individuals are perturbed by adding Gaussian noise to generate offspring. OMPAC demonstrates significant performance improvements over the basic RL algorithms in Atari and Tetris tasks. PBT [121] introduces a more generalized optimization framework that can be applied to any neural network training process. Here, we focus solely on the RL aspect. Similar to OMPAC, PBT trains a population of policies with different hyperparameters. After a certain training period, the individuals with high fitness directly replace inferior ones by perturbing their hyperparameters or by resampling hyperparameters from predefined distributions. The efficiency of PBT is demonstrated across diverse domains encompassing DM Lab, Atari, and StarCraft II. SEARL [122] utilizes GA to dynamically adjust the parameters of RL, network architectures (layers, nodes, activation functions), and the lrs of both actor and critic. Similar to PBT, SEARL trains a population of RL individuals and stores experiences generated during the population evaluation phase in a shared replay buffer. Subsequently, SEARL employs GA to add Gaussian noise to network parameters and modify network architecture, and hyperparameters to form a new population. Then, SEARL optimizes the population through DRL based on the shared replay buffer. In four out of five MuJoCo tasks, SEARL outperforms PBT in terms of performance. GA-DRL [123] focuses on adjusting a wider range of hyperparameters, including the discount factor γ, lrs for both the actor and critic, the soft update coefficient τ, the probability of selecting random actions ϵ, and the variance of the noise η. These hyperparameters are encoded using an 11-bit binary representation. GA-DRL ranks individuals based on the minimum number of episodes required for the robotic arm to achieve an 85% success rate. The experiments demonstrate that the algorithms optimized with hyperparameters discovered by GA-DRL can achieve better performance in most of the robotic arm control tasks. AAC [124] dynamically adjusts five hyperparameters: the discount factor γ, the coefficient for SAC entropy h, action duration k, and the number of single-step updates a and c for both the actor and critic. AAC maintains a population of AC individuals with different hyperparameter configurations and a shared replay buffer. Fitness is defined as the mean of cumulative rewards over multiple episodes. Based on the fitness, the top 20% best and worst individuals are selected. Superior individuals replace inferior ones with hyperparameter perturbations. Ultimately, AAC demonstrates its efficiency on the DMC benchmark and Industrial benchmark [173]. OHT-ES [125] employs ES to dynamically adjust RL hyperparameters, including the n parameter in the n-step TD and the lr. OHT-ES samples hyperparameters from distributions maintained by ES and employs these hyperparameters to train several off-policy policies. These policies control agents interacting with the environment for evaluation, and the generated experiences are stored in a shared replay buffer for RL learning. Subsequently, the distributions of hyperparameters are updated. OHT-ES demonstrates significant improvements over the basic algorithm TD3 on the DMC benchmark.

Others: Some works use EAs or the principles of EAs to assist in other aspects of RL, which cannot be categorized into the three categories mentioned above.

Evo-Reward [109] constructs a population of reward functions and employs PushGP [174], a variant of GP, for population evolution to search for more efficient reward functions. Experimental results indicate that Evo-Reward can discover more efficient reward functions than the original reward function on the hungry–thirsty task. Evo-Meta RL [110] uses EAs to search for the RL loss function capable of generalizing across the diverse environments. Specifically, it transforms the computation process of the loss function into a computational graph and introduces mutations to the operation nodes within it. Evo-Meta RL operates in both the inner and outer loops; the outer loop identifies the parent computational structures based on cumulative rewards to generate offspring, while the inner loop conducts gradient optimization based on the structures identified in the outer loop. Evo-Meta RL constructs DQNClipped and DQNReg based on the two discovered loss functions and demonstrates their superiority over DQN and DDQN on Atari and MiniGrid tasks.

GP-MAXQ [111] employs GP to explore the hierarchical structure of the hierarchical RL method MAXQ [175]. In GP-MAXQ, MAXQ learns policies based on the hierarchies derived from GP, GP explores appropriate hierarchies using MAXQ's outputs. Experimental results on the forgiveness tasks indicate that GP can search for more efficient hierarchical structures. PNS-RL [112] aims to improve the exploration capacity of RL. PNS-RL consists of multiple populations, each comprising multiple exploration policies and one guiding policy. Each exploration policy maintains an actor, critic, and replay buffer, and is optimized through the policy gradients along with soft updates toward the guiding policy. Additionally, PNS-RL maintains an archive for selecting the guiding policy. The exploration policy that outperforms the average performance of the policies in the archive is added, whereas the policies in the archive performing notably worse than the added policy are removed. The most novel individual in the archive is selected as the guiding policy and shared across different populations. Novelty is measured based on the distance in the predefined behavioral descriptor space

between the agent and its nearest k neighbors. The experiments on the MuJoCo tasks show that PNS-RL outperforms PBT-TD3, P3S-TD3, CEM-TD3, and others. Go-Explore [113] also focuses on the exploration issues. Although it does not utilize RL, it addresses the challenging exploration problem in RL and provides significant inspiration. Go-Explore builds an archive of trajectories, recording trajectories reaching different states. Then, it selects the state from the archive that most likely leads to a new state, replicates the trajectory in the environment based on the archive, reaches that state, and starts the random exploration from the state. If the newly reached state is not in the archive or reaches an existing state with a more optimal trajectory, the archive is updated. Then, the policy is learned directly through imitation learning. Go-Explore outperforms the other RL algorithms and surpasses the human performance in challenging the exploration problems Montezuma and Pitfall. G2N [114] employs a binary GA population to control the activation of hidden neurons in the RL policy network, aiming to enhance the exploration capability. The experiments demonstrate that G2N can improve PPO and A2C on the MuJoCo and Atari tasks. EvoRL [115] simulates an evolutionary process, considering that some behaviors are innate and can only be obtained through evolution, while others are learnable. Therefore, EvoRL defines part of the policy's behavior as learnable and employs RL for learning, while another part is represented using the behavior trees and can only be evolved through GP.

In addition to the aforementioned works in the single-agent settings, some works focus on improving multiagent RL (MARL) with EAs. ROMANCE [116] focuses on the robust MARL. ROMANCE utilizes the QD algorithm to maintain a population of attackers that sporadically attack some collaborators within the team. The attackers employ the policy gradients for the individual mutation, incorporating regularization terms for maintaining population diversity and attack frequency. ROMANCE demonstrates its superiority over other robust MARL algorithms on the SMAC benchmark. Following a similar idea, MA3C [117] maintains a population of agents to attack the communication channel of cooperative MARL. The individuals are improved by MATD3 and the most novel individual based on the policy representation is retained. MA3C demonstrates robust communication across various tasks. EPC [118] addresses large-scale multiagent problems by progressively expanding from the small to large-scale scenarios in a curriculum-based manner. Larger-scale policies are directly cloned from the policies of the previous scale, but the policies that perform well in one scale may not necessarily be suitable for the next larger scale. Thus, EPC trains multiple parallel policies at new scales and uses MADDPG as a mutation operator for improvement, ultimately retaining the best-performing policies. In the particle-world environment, EPC demonstrates its ability to efficiently solve the large-scale MARL problems. MAPPER [119] focuses on the multiagent path planning problem in mixed dynamic environments. MAPPER employs EAs to enhance the stability of RL training by maintaining superior individuals and eliminating inferior ones. The mutation operator uses RL to improve the individuals, each policy can be replaced by another policy with a certain probability that decreases as the score increases. MAPPER uses A2C instead of the MARL algorithms.

A related line of work to Evo-Meta RL uses hardware acceleration to make the inner loop training far faster and more tractable [69]. By doing this, "learned policy optimization" [69] and "temporally aware learned policy optimization" [70] discover the alternative on-policy RL algorithms that significantly outperform PPO on the evaluation tasks. Hardware-accelerated evolutionary meta-RL has also been applied to evolve the reward functions [176], offline datasets [177], and adversarial environments [178].

Challenges and Future Directions: The utilization of EAs to assist RL through parameter search, action selection, hyperparameter tuning, and other aspects has demonstrated the potential to further improve the performance of RL. However, utilizing EAs for RL optimization still faces several challenges as follows.

1) Researchers require the domain knowledge to define the individual form, fitness function, variation, and selection operators.
2) EAs often require a substantial number of evolutionary iterations. This takes additional computational costs and may introduce extra sample costs.
3) EAs introduce additional hyperparameters, complicating the application and tuning.
4) The works involving EA-assisted RL primarily demonstrate their efficiency in SDP. However, there is a lack of validation and systematic analysis in other problems, e.g., CTOP.

In the following, we outline several prospective research directions for the future as follows.

1) Establishing an automated configuration mechanism for EAs to enhance their usability.
2) Enhancing the efficiency of EAs, such as constructing a more sample-efficient population evaluation method.
3) Introducing EAs that are more efficient, robust, and less sensitive to hyperparameters.
4) The role of EAs in RL, beyond the aforementioned branches, remains open for further exploration.
5) There is a need for deeper investigation into the potential of EA-assisted RL in various other optimization problems.

IV. RL-Assisted Optimization of EA

In this section, we move on to the opposite side and present a comprehensive overview of RL-assisted EA. The optimization schematic is illustrated in Fig. 4(b). In this optimization process, the related algorithms revolve around EAs, with RL playing a supporting role in refining this process. RL cannot optimize independently. We categorize the related works into six branches based on the impact of RL on different stages of EAs. These branches include RL-assisted population initialization, RL-assisted population evaluation, RL-assisted variation operators, RL-assisted operator selection, RL-assisted dynamic hyperparameter configuration, and others. Each branch focuses on utilizing RL to improve specific stages of EAs, with a dedicated focus on solving

TABLE II
RL-ASSISTED OPTIMIZATION OF EA

Class	Algorithm	Problems	EA	RL	Detail
Population Initialization	NGGP [106]	SRP	GP	PG	Provide individuals with PG
	RL-guided GA [107]	COP	GA	PPO	Provide individuals with PG
	DeepACO [108]	COP	ACO	REINFORCE	Provide individuals with PG
Population Evaluation	SC [31]	SDP	EA	DDPG	Use Critic and Replay Buffer
	PGPS [61]	SDP	CEM	TD3	Use Critic and Replay Buffer
	ERL-Re^2 [32]	SDP	EA	DDPG, TD3	H-step Bootstrap
Variation Operator	GPO [5]	SDP	GA	PPO or A2C	Apply PG for mutation
	CEM-RL [94]	SDP	CEM	TD3	Apply PG for mutation
	CEM-ACER [95]	SDP	CEM	ACER	Apply PG for mutation
	PBRL [96]	SDP	GA	DDPG	Apply PG for mutation
	NS-RL [97]	SDP	NS	DDPG	Apply PG for mutation
	DEPRL [98]	SDP	CEM	TD3	Apply PG for mutation
	QD-RL [99]	MOOP & SDP	Map-Elites-like	TD3	Guide search with QD Critics
	PGA-ME [100]	MOOP & SDP	Map-Elites	TD3	Half with PG and Half with EA
	GAC QD-RL [101]	MOOP & SDP	Map-Elites	SAC & DroQ	Half with PG and Half with EA
	CMA-MEGA [102]	MOOP & SDP	Map-Elites	TD3	Optimize the fitness with PG
	CCQD [103]	MOOP & SDP	Map-Elites	TD3	Half with PG and Half with EA & Construct the shared representations
	RefQD [104]	MOOP & SDP	Map-Elites	TD3	Half with PG and Half with EA & Construct the shared representations
	Wuji [105]	MOOP	MOEA	A2C	Apply PG to offspring
Dynamic Operator Selection	RL-GA(a) [79]	COP	GA	Q-learning	Variation operators, parent types
	RLEP [80]	CTOP	ES	Q-learning	Variation operators
	EA+RL [81]	COP	EP	Q-learning	Fitness Function
	EA+RL(O) [82]	COP	GA	Q-learning	Variation operators
	RL-GA(b) [83]	COP	GA	Q-learning	Variation operators
	GSF [84]	COP	GA	DQN, PPO	All operators during all stages
	MARLwCMA [85]	CTOP	DE CMA-ES	Q-learning	Variation operators
	MPSORL [86]	CTOP	PSO	Q-learning	Variation operators
	DEDQN [87]	CTOP	DE	DQN	Variation operators
	DE-DDQN [88]	CTOP	DE	DDQN	Variation operators
	RL-CORCO [89]	CTOP	DE	Q-learning	Variation operators
	RL-HDE [90]	CTOP	DE	Q-learning	Variation operators, parameters
	DE-RLFR [91]	MOOP	MODE	Q-learning	Variation operators
	LRMODE [92]	MOOP	MODE	Q-learning	Variation operators
	MOEA/D-DQN [93]	MOOP	MOEAs	DQN	Variation operators
	AMODE-DRL [30]	MOSP	MODE	DDQN,DDPG	Variation operators, parameter
Dynamic Hyperparameter Configuration	AGA [71]	MMOP	GA	Q-learning , SARSA	Crossover rate, mutation rate, tournament size, population size
	LTO [72]	BBOP	CMA-ES	GPS [179]	Mutation step-size parameter
	RL-DAC [73]	WBOP	-	Q-learning, DDQN	Formalize DAC as MDP
	REM [74]	CTOP	DE	VPG	Scale factor, crossover rate
	Q-LSHADE & DQ-HSES [75]	CTOP	LSHADE HSES	Q-learning , DQL	The switching time
	MADAC [76]	MOOP	MOEA/D	VDN	Hyperparameter, operators
	qlDE [77]	CTOP	DE	Q-learning	Scale factor, crossover rate
	RLDE [78]	CTOP	DE	Q-learning	Scale factor
Others	RGP [67]	SDP	GP	Q-learning	Improve efficiency
	GNP-RL [68]	SDP	GNP	Q-learning	Improve efficiency
	Grad-CEM [64]	SDP	CEM	SGD	Improve CEM efficiency
	LOOP [65]	SDP	CEM	SAC	Enhancing planning
	TD-MPC [66]	SDP	CEM	DDPG	Enhancing planning

distinct problems. Specifically, population evaluation, variation operators, and others primarily focus on SDP. Among these, population evaluation primarily focuses on leveraging the discriminative capability of the RL value functions to evaluate the individuals, aiming to enhance the sample efficiency of EAs. Variation operator concentrates on using RL value

functions to provide directional guidance for more efficient variation. Others primarily aim to enhance the evolutionary efficiency with RL or improve the accuracy of fitness for planning using the RL value function. Population initialization, dynamic operator selection, and dynamic hyperparameter configuration primarily focus on CTOP, COP, MOOP, etc. Among these, Population Initialization primarily aims to leverage RL's learning capability to provide initial solutions for the population. Dynamic operator selection focuses on the issue of operator sensitivity in EAs, i.e., no single operator can perfectly solve all the problems. The goal is to use RL to achieve the automated operator selection, enhancing the robustness of EAs. Dynamic hyperparameter configuration primarily focuses on utilizing RL to automatically configure the algorithm's hyperparameters in EAs. Below, we introduce each branch of this direction along with the related algorithms.

Population Initialization: Population initialization is the initial step for all EAs, where the solutions are randomly or heuristically provided as initial candidates. A well-designed population initialization can significantly enhance the search efficiency of EAs, while poor initialization can hinder the algorithms from finding the superior solutions. Some works have shown that leveraging known high-quality initial solutions can greatly improve the algorithm performance [180]. Consequently, several works propose to leverage RL to improve the quality of the initial population.

To address the symbolic regression problem (SRP), NGGP [106] employs a policy-gradient-guided sequence generator for the population initialization. Subsequently, a certain number of GP iterations are conducted, and the top E individuals from the population are combined with the initial population to train the sequence generator. Subsequently, the next iteration process begins. Experiments demonstrate that NGGP outperforms the previous algorithms on the SPR benchmark. Similarly, RL-guided GA [107] employs PPO to master and match the problem rules and constraints. Then, the trained policies are used for the population initialization. In each generation, RL learns to get new policies, which are added to the population. Experimental results demonstrate that RL efficiently learns the rules to generate the candidate solutions for the nuclear fuel assembly optimization problems.

Differing from the aforementioned approaches that directly employ RL to provide the initial solutions, DeepACO [108] utilizes a graph neural network trained with REINFORCE to initialize the heuristic measure of the ant colony optimization (ACO). In conventional ACO, the heuristic measure is typically predefined based on the expert knowledge. DeepACO uses the probability of RL selecting each node to initialize the heuristic measure, avoiding the introduction of the expert knowledge and simultaneously accelerating the solving efficiency. DeepACO brings significant performance gains on nine COPs, such as TSP, SOP, and others.

RL-Assisted Population Evaluation: The population evaluation is a crucial step in obtaining fitness. However, when applying EAs to address the sample-cost-sensitive problems, such as robot control, the evaluation process requires applying each solution in the population to the problem to obtain the fitness. This typically incurs a substantial sample cost. To solve the problem, several works propose using the RL value functions to evaluate the fitness of the EA individuals, thereby improving the sample efficiency.

SC [31] utilizes the expected Q values estimated using the RL critics and experiences as the fitness surrogate to evaluate the population. It proposes two approaches as follows.

1) Probabilistic evaluation using the surrogate.
2) Selecting a population twice the size of the original and filtering half using the surrogate.

PGPS [61] adopts the second approach of SC for the population evaluation. ERL-Re^2 [32] introduces H-step bootstrap for the population evaluation. Each interacts with the environment for H steps, and then the value function is used to estimate the Q value of the $(H+1)$th state, which are then summed with the extrinsic rewards in the form of cumulative discount to serve as the fitness. ERL-Re^2 probabilistically applies H-step bootstrap. All the above methods have shown significant improvements in the sample efficiency. Strictly speaking, the aforementioned works should be classified as instances of synergistic optimization of EA and RL. But in this context, we focus on the population evaluation using RL. We provide a detailed explanation of these methods in Section V.

RL-Assisted Variation Operators: Traditional variation operators are typically gradient-free and rely on the random search, which requires extensive exploration to find the feasible solutions, resulting in low exploration efficiency. To improve the efficiency, some works incorporate the policy gradient guidance into EAs to assist with variation operations. These works can be categorized into two main classes: 1) single-objective optimization and 2) quality-diversity optimization.

A. Single-Objective Optimization

This category of algorithms aims to find solutions that maximize a single objective. GPO [5] devises gradient-based crossover and mutation by the policy distillation and the policy gradient algorithm. CEM-RL [94] employs the TD3 critic to guide the CEM mutation. In CEM-RL, half of the population is randomly selected and used to optimize the TD3 critic. The policy gradient from the critic is then injected into these selected individuals for mutation. The other half of the population conducts the policy search by adding the Gaussian noise. CEM-RL outperforms CEM and TD3 in three tasks of OpenAI MuJoCo. Another algorithm in this category is CEM-ACER [95], which follows a similar framework as CEM-RL but replaces TD3 with ACER. PBRL [96] is similar to CEM-RL but with a key difference of replacing CEM with a GA while incorporating DDPG for mutations. Specifically, PBRL allows individuals to interact with the environment for some steps and performs individual gradient optimization using a blend of experiences from the current individual and others. Furthermore, PBRL presents an automated hyperparameter tuning version called hyperparameter tuning PBRL. In this version, each individual in the population is associated with a corresponding hyperparameter. Gaussian perturbations are added for the hyperparameter mutation. The improvement using the current hyperparameter is used as the fitness. PBRL and hyperparameter tuning PBRL demonstrate superior

performance over GPO and DDPG in four MuJoCo tasks. Instead of maximizing performance, NS-RL [97] integrates DDPG to improve the exploration capabilities. In NS-RL, the fitness is defined as the L2 distance between a policy and the k-nearest policies to it within the behavior characterization space. The most novel individual is selected as the elite, while the less novel individuals are improved by minimizing the difference in their novelty compared to the elite. Furthermore, NS-RL takes the goal as the additional input of the critic to enhance the generalization. NS-RL demonstrates its efficiency on maze tasks. DEPRL [98] considers the diversity of the policies, but the ultimate objective remains to maximize the rewards. DEPRL follows CEM-RL and employs the maximum mean discrepancy (MMD) to measure the distance among the policies as the diversity metric. By concurrently maximizing rewards and MMD with the gradient optimization and taking the rewards and MMD as the fitness metric, DEPRL improves the diversity and exploration capabilities of the population. DEPRL outperforms CEM-RL in some MuJoCo tasks.

B. Quality-Diversity Optimization

This category of algorithms diverges from the single-objective optimization, considering two ultimate objectives: 1) solution quality and 2) solution diversity. QD-RL [99] introduces two TD3 critics into the QD framework, one for quality and the other for diversity. The calculation method of diversity is the same as NS-RL. QD-RL maintains an archive to save all the past policies. At the start of each iteration, QD-RL selects individuals from the diversity-quality Pareto front constructed from the archive. Half of the selected individuals are optimized using the quality critic, and the other half are optimized using the diversity critic. Finally, the offspring are evaluated and inserted into the archive. QD-RL outperforms TD3 and other QD methods in exploration and deceptive reward maze tasks. PGA-ME [100] follows a similar process but differs in that half of the population is mutated using the original operator of Map-elites, while the other half employs the TD3 critic for mutation. PGA-ME outperforms QD-RL and other QD methods in QDMuJoCo tasks. Furthermore, another study [181] highlights the decisive role of the policy-gradient variation operator in PGA-ME, particularly in the early optimization stages. Moreover, the study shows that PGA-ME demonstrates robust performance in both the deterministic and stochastic environments, with solutions found in stochastic settings proving highly reproducible. GAC QD-RL [101] proposes a general framework by integrating SAC and DroQ [182] into PGA-ME. It reveals three insights as follows.

1) Enhancing the update-to-data ratio (UTD), which represents the frequency of updating the critic when new transitions are collected, leads to improved performance.
2) Unlike DroQ, the diverse data distribution in QD-RL makes it difficult for the critic to overfit, hence the regularization term in DroQ is not necessary.
3) PGA-ME (SAC) is more efficient than PGA-ME (TD3) in tasks with low-dimensional behavioral descriptor space, while its performance is inferior in tasks with high-dimensional behavioral descriptor space.

CMA-MEGA [102] maintains a distinct policy for training instead of sampling from the archive. It utilizes the features of the behavioral description space as diversity metrics. CMA-MEGA estimates the gradients of the diversity metric through OpenAI-ES, the gradients of the quality through TD3, and generates gradient coefficients for the population via CMA-ES to regulate the degree of optimization toward different objectives. Following this, it evaluates the individuals within the population optimized with varying coefficients, incorporates them into the archive, prioritizes individuals based on whether new grids are filled in the archive or if grids are elevated, and then updates the CMA-ES. Finally, optimize the TD3 critic. However, CMA-MEGA performs worse than PGA-ME on QDMuJoCo. CCQD [103] draws inspiration from ERL-Re^2 [32] by similarly decomposing the policy into shared representations and independent policy representations to facilitate the knowledge sharing. Unlike ERL-Re^2, CCQD maintains multiple shared state representations to construct different knowledge spaces, enhancing the algorithm through a cooperative evolution approach. Each policy requires an unique combination of the state and policy representations. Based on different shared representations, the behaviors of the policies may vary significantly. Policies discovered during the learning process are inserted into the archive. CCQD outperforms the previous QD algorithms, reaching a new state-of-the-art performance on QDMuJoCo. RefQD [104] also employs the shared state representation and attempts to address the mismatch between the old and new policies introduced by the shared state representation in Map-elites. It periodically re-evaluates the archive and weakens the elitist mechanism of QD by maintaining more decision parts in each archive cell. RefQD outperforms PGA-ME under limited resources. Wuji [105] uses A2C to further enhance the offspring produced by crossover and mutation in MOEA, which can be seen as an additional variation operator. Wuji demonstrates superior performance in game bug detection.

RL-Assisted Dynamic Algorithm Configuration: The utilization of EAs faces several significant challenges during both the configuration and application stages. First, no single EA operator can efficiently solve all the problems, leading to the need for a selection of EA operators based on the problem characteristics and expert insights. Second, EAs are highly sensitive to the hyperparameters, demanding meticulous adjustments. Even slight changes can lead to significant performance differences. To solve the problems, many works improve the usability and robustness of EAs by dynamically selecting the operators and tuning the hyperparameter configuration, which is commonly referred to as the dynamic algorithms configuration (DAC). In this context, our focus lies primarily on how RL can assist in the DAC process. We categorize these works into two major domains: 1) dynamic operator selection and 2) dynamic hyperparameter configuration.

Dynamic Operator Selection: The algorithms discussed here primarily focus on COP, MOP, MOOP, and CTOP. RL-GA(a) [79] employs $Q(\lambda)$ to enhance GA by dynamic operator and parent type selections. The population itself forms the

states and the rewards are defined as the improvement of the offspring compared to the parents. RL selects crossover and mutation operators along with specifying the parent types to which these operators should be applied. RL-GA(a) outperforms GA on the TSP. RLEP [80] employs Q-learning to dynamically select four mutation operators for EP. RLEP defines the rewards as the improvements of the offspring over the parents and directly approximate expected returns for the four mutation operators. RLEP outperforms or performs equivalently to the four basic mutation operators on the functional optimization problems. EA+RL [81] employs RL to dynamically select the fitness function to enhance the optimization efficiency of GA under the target fitness. The rewards are defined as the performance differences between the best individuals under the target fitness at sequential generations. The states are constructed based on the fitness values of the population. EA+RL demonstrates improvements over ES in the royal roads and H-IFF optimization problem. EA+RL(O) [82] dynamically employs Q-learning to select the crossover and mutation operators for the next generation. Similarly to EA+RL, the rewards are defined as the differences in performance between the best individual and its predecessor in the previous generation. The RL states are tailored for different tasks. The efficiency of EA+RL(O) is validated on the H-IFF optimization problem and the TSP. RL-GA(b) [83] employs RL to enhance GA in the electromagnetic detection satellite SPs. The definition of rewards is consistent with that of RL-GA(a), and the states are formulated by considering the fitness improvement and the original fitness values.

GSF [84] employs DQN and PPO to dynamically select appropriate combinations of algorithmic components (i.e., evolution operators) during different optimization stages in the capacitated VRP with time window (CVRPTW) [183]. GSF encodes essential information required to solve the CVRPTW problem as states, including fitness, the number of vehicles, and capacity. The rewards are determined by the performance improvement of the current population compared to the initial population. GSF demonstrates the efficiency of both PPO-GSF and DQN-GSF in the CVRPTW. MARLwCMA [85] proposes a framework that combines multiple optimization algorithms, including multioperator DE and CMA-ES. In this framework, multioperator DE dynamically selects mutation operators with the assistance of RL. The rewards are defined as the cumulative performance improvement of the offspring generated using the selected operators compared to their parents. The states contain two variables designed to reflect population diversity and quality. MARLwCMA outperforms other EAs on multiple CEC benchmarks. In MPSORL [86], the action space consists of four strategies, while the states are divided unevenly into five grades based on the fitness values. Subsequently, MPSORL selects the optimal strategy for each particle. To update the Q-table, a reward of 1 is returned if the particle improves; otherwise, a reward of 0 is returned. MPSORL demonstrates superior performance compared to the other PSO algorithms on the CEC benchmark. DEDQN [87] utilizes DQN to dynamically select from the three mutation operators in DE, primarily divided into two stages. In the first stage, DQN is trained, where the states are constructed based on the information from the fitness landscape theory [184], including fitness distance correlation and ruggedness of information entropy. The rewards are constructed from the live algebraic and the individual evolutionary efficiency. In the second stage, the trained DQN selects mutation operators to improve DE. DEDQN demonstrates its superiority over five well-known DE variants in the CEC2017 benchmark. DE-DDQN [88] utilizes DDQN to dynamically select mutation operators for each parent in DE. The RL state space comprises a 99-feature vector to capture the DE's current state. The reward function takes three forms: 1) R1, reflecting the fitness differences between offspring and parents; 2) R2, assigning a higher reward for improvements over the best solution compared to the improvements over the parents; and 3) R3, concurrently maximizing offspring's fitness differences while minimizing the gap between the offspring and the optimal solution. DE-DDQN demonstrates superior performance over the other DE methods and dynamic operator selection methods in the CEC benchmark. RL-CORCO [89] employs Q-learning to dynamically select two mutation operators for DE in constrained optimization problems. The population is divided into nine subpopulations based on the objective value and the degree of constraint violation, resulting in nine distinct states. Whenever a mutation strategy is applied and it either improves or maintains the performance of the population individuals, a reward of 1 is returned; otherwise, a reward of 0 is given. Additionally, RL-CORCO incorporates a population reinitialization mechanism to prevent it from becoming trapped in the local optima. RL-CORCO demonstrates its superiority over the other baseline methods on the CEC benchmark.

RL-HDE [90] employs Q-learning to dynamically select six mutation operators and adjust two trigger parameters for DE. To select mutation operators, RL-HDE divides the population into 20 states based on the diversity and average performance relative to the initial population. A reward 1 is given if a better solution is obtained with the selected mutation operator, 0 if there is no change, and -5 if the performance worsens. To balance global exploration and local exploitation, RL-HDE dynamically adjusts the two hyperparameters and constructs six states based on a similar definition for the operator selection. Regarding rewards, a better solution yields a reward of 1, no change results in a reward of 0, and deterioration leads to a reward of -1. Experimental results demonstrate that RL-HDE outperforms the other baselines in solving the complex interplanetary trajectory design problems, such as Cassini2 and Messenger-full. DE-RLFR [91] employs Q-learning to dynamically select one of three mutation operators for each individual in MOOP. Specifically, DE-RLFR categorizes the fitness of each objective into three levels based on their ranking, resulting in nine states for RL. A reward of 10 is assigned when offspring outperform their parents; otherwise, 0 is returned. Experiments across 11 multimodal MOOPs demonstrate that DE-RLFR can effectively construct the superior Pareto front. LRMODE [92] integrates the findings from a local landscape topology analysis with RL to approximate the optimal probability distribution for dynamically selecting the MODE's mutation operators. LRMODE demonstrates

superiority over other multiobjective optimization algorithms on the multiobjective optimization tasks. MOEA/D-DQN [93] utilizes the parent solutions and the weight vectors as RL states and constructs the RL rewards based on the fitness improvements. It employs DQN to choose variation operators for MOEAs, leading to superior performance compared to the other MOEAs across a diverse range of the MOP benchmarks. AMODE-DRL [30] dynamically selects five mutation operators and adjusts two continuous parameters in multiobjective SPs (MOSP). It leverages DQN to select mutation operators and DDPG to fine tune the continuous parameters. The RL states involve the current individual's fitness, fitness improvement, and population diversity. The RL rewards are defined by individual fitness and population diversity. Experimental results in both the randomly generated instances and real-world problem cases demonstrate that DRL significantly enhances the MODE's exploration and learning efficiency.

Dynamic Hyperparameter Configuration: In this context, we introduce the RL-assisted EA hyperparameter configuration, including crossover probability, mutation rate, population size, and others. AGA [71] leverages Q-learning to dynamically adjust the EA's crossover rate, mutation rate, tournament size, and population size. The RL states correspond to the population information, e.g., maximum fitness, mean fitness, and the previous action vector. The reward function is defined as the improvement of the best fitness. The experiments demonstrate that AGA outperforms GA on the multimodal problem generator introduced by Spears [185]. LTO [72] utilizes GPS [179] for dynamically adjusting the mutation step-size parameter of CMA-ES. The RL states include the current step-size value, the current cumulative path length, the history of objective value changes, and the step-size history from the previous h iterations. LTO constructs the RL rewards based on the objective value of the current solution. LTO demonstrates its efficiency in the BBOB-2009 benchmark [186]. RL-DAC [73] formalizes DAC as a contextual MDP to enable RL to learn across a set of instances. It also introduces the white-box benchmarks to demonstrate the efficiency of RL in hyperparameter tuning. Strictly speaking, RL-DAC is not limited to EAs; it can apply to all the optimization algorithms. REM [74] employs the variational policy gradient to continuously adapt the DE's scaling parameter and crossover rate. REM uses the present population information and the corresponding randomness as the state. The reward gives 0 if the algorithm reaches the maximum generation. Alternatively, it provides the negative logarithm of the smallest function value discovered by the EA. Experiments demonstrate that DE and adaptive DE, with tuned hyperparameters, outperform the counterparts and other methods.

In hybrid EAs, the timing of switching between different EA phases is crucial for the algorithm performance. Different from the rule-based switching methods, Q-LSHADE and DQ-HSES [75] propose an adaptive framework based on RL to adjust the switching time. Specifically, Q-LSHADE combines Q-learning and LSHADE [187], adaptively controlling when to use the linear population size reduction (LPSR) technique within L-SHADE. DQ-HSES combines DQN and HSES [188], adaptively controlling when to transition from the univariate sampling phase to the CMA-ES algorithm. The experiments in the CEC 2014 and 2018 benchmarks demonstrate that the proposed algorithms outperform SOTA EAs. MADAC [76] emphasizes the heterogeneity among various hyperparameters and recognizes that applying a single RL algorithm for configuring all the parameters can introduce complexities. Hence, MADAC applies a typical MARL method value-decomposition networks (VDNs) [189] to search for the appropriate settings for the multiobjective EA MOEA/D's [190] four categories of hyperparameters. The RL states incorporate characteristics from different aspects, e.g., the specific problem instance, attributes associated with the ongoing optimization process, and aspects concerning the evolving population. MADAC provides rewards when the algorithm discovers better solutions than the best so far and offers greater rewards to the agents that can find even better solutions in later stages. In multiobjective optimization challenges, MADAC demonstrates superior performance compared to the other methods. qlDE [77] uses Q-learning to dynamically adjust the two hyperparameters of DE, the scale factor F and crossover rate Cr. If the best individual in the population is better than the previous generation, the reward is 1; otherwise, it is -1. qlDE demonstrates comparable or superior performance to the other DE algorithms in five truss structural weight minimization problems. RLDE [78] employs Q-learning to dynamically adjust the scale factor F of DE. Actions are defined as 0.0, -0.1, and 0.1, which are added to the current F. If the offspring is superior to the parent, the reward is 1; otherwise, it is 0. RLDE outperforms the other algorithms in the parameter extraction problems involving various PV models.

Others: Here, we introduce several methods within RL-assisted optimization of EA in other aspects. Some algorithms are influenced by the Baldwin effect and the Lamarckian ideas [191], [192], which introduce learning into EAs to enhance the evolutionary efficiency. The early work experimentally validates that the introduction of learning can improve the efficiency of EAs [193]. Subsequently, many works attempt to incorporate RL into EAs. RGP [67] integrates Q-learning into tree-based GP to enhance evolutionary search. RGP utilizes GP to search trees, dividing the search space into coarse-grained regions. Q-learning is embedded at the leaf nodes of the tree for decision making. Ultimately, RGP demonstrates that incorporating Q-learning can further improve the efficiency of GP in maze tasks. GNP-RL [68] combines GNP [194] with Q-learning. GNP leverages the higher expressive power and more compact graph structure to address the bloat issue of tree structure. GNP-RL employs RL to more fully exploit state and reward information returned by the environment, thereby enhancing optimization efficiency. Ultimately, GNP-RL demonstrates the efficiency of the method in grid environments.

In addition, some works utilize RL to enhance EA-based planning methods. Model predictive control (MPC) [195] is a model-based control approach that begins by designing or learning a world model. Subsequently, it employs this model to plan a sequence of actions. To enhance efficiency, several works

replace the traditional random sampling planning methods with CEM, such as PETS [196], PlaNet [197], and POPLIN [198]. Building upon CEM, some works incorporate RL or gradient optimization into MPC to enhance the performance. In Grad-CEM [64], several random action sequences are generated, and stochastic gradients obtained from maximizing the rewards based on the dynamic model are used to update the generated sequences, which improves the efficiency of CEM. Experiments in MuJoCo and DMC benchmarks demonstrate that Grad-CEM outperforms CEM. LOOP [65] combines MPC and off-policy RL. To enhance estimation accuracy, LOOP augments the traditional H-step discounted rewards with Q-values. Additionally, trajectories generated by the RL policy based on the world model are combined with those generated by CEM to optimize the CEM distribution. LOOP outperforms the other model-based methods on the MuJoCo tasks. TD-MPC [66] employs the same framework as LOOP, with the distinction of encoding states into a latent space for modeling the world model, learning the policy, and approximating Q-values. Experiments show that TD-MPC outperforms LOOP and SAC on the DMC tasks.

Challenges and Future Directions: The above works demonstrate the efficiency of RL-assisted EAs in various aspects. Despite demonstrating the capability of RL to enhance EAs across various types of problems, RL-assisted EAs still face the following challenges.

1) Utilizing RL-assisted optimization of EA requires researchers to have a deep understanding of the target problem to formulate it as an MDP. Additionally, RL knowledge is necessary to select the suitable algorithms for learning.
2) RL introduces extra hyperparameters, which usually need to be adjusted based on the specific problem. This may entail additional trial-and-error overhead.
3) Although RL has demonstrated the ability to enhance EAs in experiments across different branches, this lacks the theoretical support and convergence guarantees.
4) Despite employing similar techniques, the absence of comparisons between different methods, especially in dynamic algorithm configuration, makes it challenging to determine which method currently outperforms in addressing the specific problems.

Based on the aforementioned challenges, we propose several future research directions as follows.

1) More advanced and stable RL algorithms within the EA process require further investigation, e.g., exploring more generalized modeling approaches and developing more robust and general RL algorithms.
2) Establish theoretical guarantees for RL-assisted EAs, including convergence and performance bounds.
3) For each research branch, further investigation can be conducted to address the existing limitations of the current methods, e.g., develop more efficient population evaluation methods and mutation operators.
4) Researchers can construct an unified framework and evaluate the related works in a consistent benchmark, offering more valuable insights.

V. Synergistic Optimization of EA and RL

The previous hybrid algorithms typically maintain only one of the approaches (EAs and RL) as the primary problem solver, while the other algorithm plays a supporting role. This section focuses on the synergistic optimization algorithms that integrate the complete learning and optimization processes of RL and EAs, either 1) to simultaneously solve the same problem with collaborative mechanisms or 2) independently optimize subproblems to obtain the partial solutions, which are subsequently combined to form a complete solution. The schematics are illustrated in Fig. 4(c) and (d). The related works in this direction focus on SDP. Below, we separately introduce two different collaboration approaches.

The first collaboration approach involves simultaneously solving the same problem using EAs and RL, with collaboration during the solving process. This collaboration approach is inspired by the complementary strengths demonstrated by EAs and RL. Specifically, EAs, based on the population and random exploration, offer excellent exploration and global optimization capabilities [54]. However, random search in vast parameter space often leads to low optimization efficiency. Additionally, EAs evaluate individuals based on the episodic rewards, which necessitate each individual to interact with the environment for fitness. These weaknesses result in significant sample costs, often ranging two to three orders of magnitude higher compared to RL [143]. While these coarse-grained episodic rewards make EAs more insensitive to the quality of the reward signals. In contrast, RL can leverage finer-grained information, e.g., states and rewards, and reuse the historical experiences, thereby providing higher sample efficiency, yet it suffers from the exploration challenges during the learning process, often prone to converging to the suboptimal solutions. Moreover, RL necessitates well-designed reward signals to ensure the final performance [12]. Through the comparison above, we observe that EAs and RL each have their strengths and weaknesses. The key point of this collaboration approach is how to establish a symbiotic relationship, maintaining their respective strengths while compensating for their weaknesses. Consequently, many works try to integrate EAs with RL for the synergistic optimization to enhance the search efficiency and solution quality.

Single-Agent Optimization: The earliest method is ERL [54], which establishes the foundational framework. In ERL, both EAs and RL engage in the policy search. EAs provide the diverse samples generated during the population evaluation to RL for the policy learning, thereby enhancing the sample efficiency. Conversely, RL incorporates its policy into the population to participate in the evolutionary process. If the RL policy achieves better performance, it guides and facilitates the population evolution. Through these mechanisms, ERL integrates the strengths of both EAs and RL. Experimental results demonstrate that ERL outperforms DDPG and GA on most OpenAI MuJoCo tasks. CERL [55] is a follow-up work to ERL, focusing on solving the sensitivity problem to the RL discount factors γ. It is important to note that the role of GA in CERL is not employed for the hyperparameter tuning but for the policy search, which is consistent with

TABLE III
SYNERGISTIC POLICY OPTIMIZATION WITH EA AND RL IN SINGLE-AGENT SETTINGS

Algorithm	Task	EA	RL	Fitness Surrogate	Policy Structure	RL Role	EA Role
ERL [54]	MuJoCo	GA	DDPG	N/A	Private	Policy Injection	Diverse Experiences For RL
CERL [55]	MuJoCo	GA	TD3	N/A	Private	Policy Injection	Diverse Experiences For RL
PDERL [56]	MuJoCo	PD-GA	DDPG	N/A	Private	Policy Injection	Diverse Experiences For RL
SC [31]	MuJoCo	GA & PD-GA	DDPG	Using Critic Estimates	Private	Policy Injection	Diverse Experiences For RL
GEATL [57]	Grid World	GA	A2C	N/A	Private	Policy Injection	Elite Policy Synchronisation
CSPS [58]	MuJoCo	CEM	PPO & SAC	N/A	Private	Policy Injection	Diverse Experiences For RL Two Separated Buffer
T-ERL [59]	MuJoCo	ES	TD3	N/A	Private	Policy Injection	Diverse Experiences For RL Two Separated Buffers
ESAC [60]	MuJoCo & DMC	A-ES	SAC	N/A	Private	Policy Crossover	Diverse Experiences For RL
PGPS [61]	MuJoCo	CEM	TD3	Using Critic Estimates	Private	Gradient Injection	Diverse Experiences For RL & Guided Policy Learning
ERL-Re^2 [32]	MuJoCo & DMC	B-GA	DDPG & TD3	H-Step Bootstrap (PeVFA)	Shared	Policy Injection	Diverse Experiences For RL
VEB-RL [62]	MinAtar & Atari	GA & CEM	DQN & Rainbow	The TD Error	Private	Value Function Injection	Diverse Experiences For RL
EvoRainbow-Exp [63]	MuJoCo & Maze & MinAtar & MetaWorld	CEM	TD3 & SAC	N/A	Private	Policy Injection	Diverse Experiences For RL Genetic Soft Update
EvoRainbow [63]	MuJoCo & Maze & MinAtar & MetaWorld	CEM	TD3 & SAC	H-Step Bootstrap (Critic)	Shared	Policy Injection	Diverse Experiences For RL Genetic Soft Update

that in ERL. Thus, we discuss it in this section. CERL maintains multiple RL learners with different gammas. Unlike dynamic adjustments, CERL predefines the gamma values without tuning them in the learning process. During training, resources are dynamically allocated based on the performances of learners. Experiments on the MuJoCo tasks demonstrate that CERL is more insensitive to the hyperparameters. Taking inspiration from GPO [5], PDERL [56] proposes proximal-distilled GA (PD-GA) to address the catastrophic forgetting issue associated with GA in ERL. Specifically, PD-GA encompasses novel crossover and mutation operators. The crossover operator distills desirable behaviors from the parents to offspring based on the Q values. The mutation operator adjusts the magnitude of mutations by taking into account parameter sensitivity to actions. PDERL demonstrates superior performance to ERL in OpenAI MuJoCo tasks. SC [31] focuses on mitigating the high sample cost associated with the population evaluation. It proposes leveraging RL critic as a surrogate for fitness and evaluating the individuals with the critic based on the samples from the replay buffer. Besides, SC introduces two mechanisms for the surrogate utilization: 1) employing the surrogate for population evaluation with a probability of P, while interacting with the environment with a probability of $1 - P$ and 2) generating a population larger than twice the original size and then using the surrogate model to filter half of the individuals. SC integrates with ERL and PDERL, demonstrating the performance improvements on the MuJoCo tasks.

Unlike the previous approaches that integrate off-policy RL with EA, GEATL [57] combines the on-policy RL with EA. Similar to ERL, in GEATL, RL influences EAs through the policy injection. However, the influence of EAs on RL operates differently: when the elite policy of the population outperforms the RL policy, the elite policy replaces the RL policy. Moreover, if their performances are comparable, there is a 50% chance that the elite policy replaces the RL policy. GEATL demonstrates its superiority over ERL in scenarios in grid world with sparse rewards. CSPC [58] incorporates the off-policy RL, on-policy RL, and EAs. Specifically, CSPS integrates SAC, PPO, and CEM. When the SAC policy outperforms PPO or the policies in the population, it replaces those individuals. Similarly, if the PPO policy excels, it replaces the population policies. Furthermore, CSPS introduces an additional local experience buffer for SAC to store recently generated experiences and incorporates several experience filtering mechanisms. These mechanisms ensure that the added local experiences are the most recent and superior to the minimum value among all the individuals at that time. SAC utilizes the local experience buffer for the policy optimization with a probability of P, while utilizing the global buffer with a probability of $1 - P$. CSPS outperforms the three basic algorithms on most of the MuJoCo tasks.

T-ERL [59] integrates ES with TD3 and constructs two replay buffers akin to CSPS. One buffer saves the experiences of all the individuals, while the other saves the recent RL experiences. T-ERL proportionally samples from both the buffers for RL training. T-ERL demonstrates superiority over TD3 on the three of four MuJoCo tasks ESAC [60] adopts the ERL framework while replacing DDPG with SAC and GA with a modified ES. The ES introduces an automatic adjustment mechanism to regulate the coefficient of the added Gaussian noise in ES, denoted as A-ES. The coefficient is updated based on the disparity between the best performance identified within the population and the average performance. Moreover, unlike GA, ESAC does not shield elite individuals from the mutation interference; instead, it employs crossover between the elites and the updated ES distribution to transmit the favorable traits from the elites to the offspring. ESAC demonstrates its superiority over the ES and other RL algorithms on the MuJoCo and DMC tasks. PGPS [61] follows the ERL framework to combine CEM and TD3. It is noteworthy that PGPS maintains a full life-cycle RL policy, distinguishing itself from CEM-RL. Within PGPS, the population of size N consists of the elite from the previous generation (index 0), individuals randomly sampled from the CEM distribution (index 1 to $[N/2]$), and individuals selected from the large CEM-sampled pool using the surrogate mechanism from SC (index $[N/2]$ to N). Moreover, PGPS introduces the guided policy learning. When the behavior difference between the elites in the population and the current RL actor exceeds a threshold, behavior cloning is employed to assist RL learning. Conversely, the constraints are relaxed if the difference is within the threshold. PGPS demonstrates superior or comparable performance to CEMRL, PDERL, CERL, CEM, and some RL algorithms in MuJoCo.

ERL-Re^2 [32] uncovers a primary problem prevalent in the existing ERL-related research: the wide use of isolated policy architectures, where each individual operates within its private policy network. However, this independent structure often hinders the efficient transfer of valuable knowledge. To solve the problem, ERL-Re^2 decomposes the policies into a shared state representation and independent linear policy representations. The policy structure facilitates knowledge sharing while simultaneously compacting the policy space. Moreover, ERL-Re^2 proposes behavioral-level genetic operators (B-GA) based on the linear policy representations, coupled with an H-step bootstrap fitness surrogate for the population evaluation. ERL-Re^2 achieves the state-of-the-art performance on the MuJoCo tasks.

VEB-RL [62] addresses the issue of the previous works overlooking value-based RL. VEB-RL constructs a population of value functions and corresponding target functions, using the negative TD error as the fitness metric for the value function evaluation. VEB-RL also introduces an elite interaction mechanism to avoid wasting interaction resources. VEB-RL significantly enhances DQN and rainbow on MinAtar and Atari.

EvoRainbow [63] and EvoRainbow-Exp [63] systematically review this branch of works from the five perspectives through the experiments, providing a detailed comparison of mechanisms with the same functionality. By integrating the most effective mechanisms, they construct EvoRainbow and its exploratory version, EvoRainbow-Exp. EvoRainbow incorporates parallel mode, shared architecture, CEM, genetic soft update, and H-step bootstrap (critic). EvoRainbow-Exp combines the parallel mode, private architecture, CEM, and genetic soft update. Both EvoRainbow and EvoRainbow-Exp have demonstrated superiority over the current state-of-the-art ERL methods across 20 tasks, including locomotion tasks, manipulation tasks, maze tasks, and Minatar.

Multiagent Optimization: In addition to the aforementioned single-agent optimization methods, the fusion of EAs and MARL has also made many advances. Here, the focus is on the cooperative settings, where we need to control multiple agents to complete the tasks. Compared to MARL, EAs offer additional advantages: EAs avoid the need to model the MARL problem as the MDP, thus circumventing the nonstationarity problem [53], [138]. Among these, MERL [49] is proposed to efficiently utilize both the team-level and agent-level rewards for collaboration. MERL maintains a team population and optimizes the team policies using EAs with team rewards, while simultaneously optimizing the individual policies using RL with the agent rewards. The overall optimization process is similar to ERL. MERL demonstrates superior performance to MATD3 and MADDPG on the MPE tasks. NS-MERL [50] extends MERL by considering two types of rewards during the optimization of the individual rewards. To encourage exploration, NS-MERL employs a count-based method to track the number of times the current observation has been visited, where higher counts result in lower rewards. This reward is then multiplied by the original heuristic reward. Additionally, a counterfactual mechanism is utilized to calculate the contribution of each individual, thereby enhancing the collaboration. The final reward is determined by multiplying the counterfactual reward with the two aforementioned rewards. Experimental results demonstrate that the constructed reward function outperforms the other reward functions in the multirover exploration domain. CEMARL [51] shares the same idea as MERL, primarily replacing GA with CEM. In each iteration, the population teams are sampled from the CEM distribution. Subsequently, a random individual is selected and optimized using MARL based on the individual rewards. Following individual optimization, all the teams interact with the environment to obtain team rewards, which are then used to optimize the CEM distribution. CEMARL also maintains a policy that is soft-updated to the team with the best performance in the population, enhancing stability. Experiments show that CEMARL outperforms MERL in MPE environments. CEMARL does not maintain a full life-cycle MARL policy. Instead, it samples the MARL policy from the CEM distribution each time. Therefore, we can view MARL as the variation operator, injecting gradients into one individual of the population. As CEMARL aligns more closely with MARL settings, we include it in this branch for discussion. Different from MERL and CEMARL, EMARL [52] focuses on more general task settings, i.e., only team-level rewards. EMARL combines GA with COMA, where the population individuals are first optimized using GA, and the optimized

TABLE IV
SYNERGISTIC POLICY OPTIMIZATION WITH EA AND RL IN MULTIAGENT SETTINGS

Algorithm	Task and Setting	EA	MARL	EA Role	RL Role	Other features
MERL [49]	MPE (Global Information & Dense agent reward & sparse team reward	GA	MATD3	Optimize Population with Team Reward & Provide Experiences	Optimize MARL with Agent Reward & Inject MARL Policy	Two Types of Rewards
NS-MERL [50]	Multi-rover Exploration Domain (Global Information & Dense agent reward & sparse team reward	GA	MATD3	Optimize Population with Team Reward & Provide Experiences	Optimize MARL with Agent Reward & Inject MARL Policy	Two Types of Rewards & Exploration
CEMARL [51]	MPE (Global Information & Dense agent reward & sparse team reward	CEM	MATD3	Optimize Population with Team Reward & Provide Experiences	Optimize MARL with Agent Reward	Two Types of Rewards
EMARL [52]	The Flocking Env (Only Team rewards Partial observation)	GA	COMA	Optimize Population with Team Reward & Provide Experiences	Optimize Population with Team Reward	Serial Optimization
RACE [53]	SMAC & MA-MuJoCo (Only Team rewards Partial observation)	A-GA	FACMAC or MATD3	Optimize Population with Team Reward & Provide Experiences	Optimize MARL with Team Reward & Inject MARL Policy	Shared Representation Architecture

population is further enhanced using the policy gradients. EMARL is evaluated on the flocking tasks and shows better performance compared to the benchmark MARL algorithms. The aforementioned algorithms are evaluated on simple tasks, whereas RACE [53] proposes a new hybrid framework and demonstrates its efficiency in facilitating collaboration within the complex tasks for the first time. Specifically. RACE introduces the concept of shared representations into the integration of MARL and EA. RACE divides the team policy into the shared observation encoders and independent linear policy representations. RACE maximizes the value function and value-aware mutual Information to inject collaboration-related information and superior global states into the shared representations. The agent-level crossover and mutation operations are operated on the linear representations to ensure the stable evolution. Finally, RACE achieves superior performance compared to the FACMAC, MATD3, and MERL on SMAC and MA-MuJoCo tasks.

In addition to the above collaboration approach, we can also decompose the problem into subproblems suited for EAs and subproblems suited for RL. A common pattern based on this collaboration approach is to utilize EAs for the structure search and RL for the policy learning. Next, we systematically introduce the methods involving this collaboration approach.

Morphological Evolution: Morphological Evolution continuously optimizes the robot morphology and the control policy. In such problems, the final solution consists of two components: 1) the optimal morphology and 2) its associated policy. In the morphological evolution, EAs and RL optimize different aspects of the objective. We briefly introduce related work in this area. Classic algorithms in this category involve EAs for evolving the morphologies and RL for the learning policies [43]. Some works attempt to simultaneously optimize both the morphology and policy using RL, such as CuCo [199] and Pre-Co [200], or simultaneously employ EAs to optimize the morphology and policy, such as HyperNEAT [201], or optimize the morphology with the other optimization techniques, such as Bayesian optimization [43], or explore the choice of genetic encoding for the morphology [48]. The hybrid algorithms in this area include EvoGym (GA) [43], HERD [44], AIEA [45], DERL [46], and TAME [47].

Interpretable AI: Policies derived from the deep neural networks often lack interpretability, making them difficult to analyse and impractical for application in real-world scenarios with potential risks. To solve the problems, many works integrate decision trees with EAs and RL for highly interpretable policies. Due to some methods lacking names in the original papers, we use abbreviations based on their characteristics to denote them. POC-NLDT [37] first collects a dataset using a policy pretrained with RL and then introduces two stages: 1) open-loop training and 2) closed-loop training. During the open-loop training, optimization is performed using a bi-level EA [202] based on the dataset. In this process, the upper level optimizes the tree structure, while the lower level seeks the optimal values for the tree's weights. In the closed-loop training, further reoptimization of the weights is conducted using the cumulative reward collected across several episodes. Finally, POC-NLDT demonstrates the interpretability and efficiency in four discrete action problems. GE-QL [38] evolves the tree structure using the grammatical evolution (GE [203]) and optimizes the leaf nodes using Q-learning. CG-DT [38] leverages GP to optimize structures of decision trees and employs CMA-ES [204] to optimize weights. CC-POC [39] extends POC-NLDT to continuous action spaces by constructing a population of actions. It utilizes GE for optimizing the tree structures and UMDA$_c^G$ [205] for the action optimization. Q-learning is employed to optimize the leaf nodes. CC-POC combines the two populations to obtain the complete solution. QD-GT [40] replaces GE with Map-elites and defines behavioral descriptors based on the action entropy and depth of decision trees. QD-GT demonstrates superior performance compared to the GE schemes on the cart pole and mountain car tasks. SIRL [41] proposes a collaborative framework that constructs a population of decision trees. Actions are chosen randomly from the population to interact with the environment, and experiences are shared among them. Subsequently, each

decision tree optimizes its leaf nodes using Q-learning and its tree structure using GE. SIRL demonstrates its efficiency across the six MUJOCO tasks. Besides, SVI [42] is proposed to use the symbolic regression to construct the smooth analytical expression-based value functions, introducing the symbolic value iteration to solve the Bellman equation. SVI offers higher interpretability compared to the black-box optimization of the neural networks.

Learning Classifier Systems (LCSs): LCSs [206] represent a class of methods that integrate learning with evolutionary principles to discover a set of rules capable of addressing a target task. LCS can also be referred to as population-based temporal-difference methods [207]. LCS consists of four key components as follows.

1) A population of classifiers, representing the current knowledge base. Each classifier consists of a condition, an action, and an associated fitness parameter.
2) A performance component, used to regulate the interaction between the environment and the population.
3) A reinforcement component, allocates rewards obtained to classifiers.
4) A discovery component, employed to discover new rules or refine existing ones.

LCS matches related classifiers based on the inputs each time. If no match is found, random classifiers are generated and added to the population. XCS [33] integrates Q-learning into LCS for learning, where each classifier represents an action-value function, and the associated parameters correspond to the weight matrix in the function approximation. Each classifier includes a condition, an action, and four main parameters. XCS utilizes fitness based on accuracy, employing GA to search in the action space for the classifier selection. To improve the system's robustness and reduce parameter dependency, researchers introduce the gradient descent methods into XCS, resulting in two approaches: 1) XCS with direct gradient (XCSG) and 2) XCS with residual gradient (XCSRG) [207]. XCSF evolves classifiers representing piecewise linear approximations of portions of the reward surface associated with the problem solution [208]. In XCSF, the classifier's prediction is calculated as a function, which is a linear combination of the classifier, rather than a scalar parameter. XCSF with tile coding [35] replaces the original classifier prediction function in XCSF with a tile coding approximation. DGP-XCSF [36] employs graph-based dynamical GP to represent the traditional condition-action production system rules for solving the continuous-valued RL problems. If you wish to explore further works related to LCS, you can refer to the work [206].

Challenges and Future Directions: Synergistic optimization has made significant progress in the recent years. For instance, the early works based on the first collaboration approach are only able to achieve the performance improvements on specific tasks [54], [55], [56]. With the development of this field, the recent works can consistently outperform both EAs and RL on a wide range of tasks [32]. Despite the significant progress made in this direction, there remains a need for further investigation into how to effectively integrate the strengths of EAs and RL. The direction of synergistic optimization of EA and RL faces challenges similar to those of EA-assisted optimization of RL and RL-assisted optimization of EA, such as the need for the domain knowledge, sensitivity to hyperparameters, and more. Concurrently, it presents a distinctive problem specific to this direction, namely how to integrate EAs and RL to maximize the advantages of both for various problems. Currently, this direction primarily focuses on SDP. Further research is required to explore how it can complement and provide advantages for addressing the other types of problems. In the future, researchers can explore in the following directions.

1) Explore how to integrate EAs and RL for synergistic optimization in addressing the other problems.
2) Replace the foundational algorithms in current ERL methods with more advanced EAs or RL algorithms to fully leverage the advanced methods of both the domains.
3) For the first collaboration approach, design more efficient mechanisms where EAs influence RL or RL influences EA, enhancing the positive impacts between EAs and RL.
4) For the second collaboration approach, how to better and more automatically decompose problems to fully leverage their respective advantages is also an important research direction.

In multiagent settings, combining EAs and MARL is still at a nascent stage but holds substantial potential for advancement. Researchers can further delve into investigating how to fuse the capabilities of EAs and MARL to drive efficient collaboration.

VI. Conclusion

Overall, this article systematically reviews different research directions within the field of ERL, along with the corresponding research branches within each direction. At the end of each section, the challenges faced by each direction and potential future research directions are summarized. We hope that this survey can comprehensively showcase the current development status of the ERL field to researchers, including the existing algorithms, technical details, research challenges, and future research directions.

It is worth emphasizing that, although this review indicates that EA-assisted RL and the synergistic optimization of EA and RL primarily focus on SDP, RL-assisted EA optimization is geared more toward addressing the other optimization problems, closely tied to the problem-solving strengths of EAs and RL. However, these directions should not be limited to specific problems. Fortunately, in recent years, there has been a considerable amount of work attempting to leverage RL to solve problems where EAs excel, such as COP [209] or employing EAs to address SDP [210], yielding significant positive outcomes. This further broadens the application boundaries of different technologies. With the development of ERL, solutions to various problems will gradually emerge in different directions, which is a development we eagerly anticipate. Therefore, our survey primarily takes a technical perspective to assist researchers in thoughtful consideration

and further expansion of the application boundaries in different research directions, driving the advancement of the field.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning—An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[2] M. I. Jordan and T. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, pp. 255–260, Jul. 2015.

[3] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 1990.

[4] M. Andrychowicz et al., "Learning to learn by gradient descent by gradient descent," in *Proc. NeurIPS*, 2016, pp. 1–9.

[5] T. Gangwani and J. Peng, "Policy optimization by genetic distillation," in *Proc. ICLR*, 2018, pp. 1–16.

[6] X. Chen, C. Wang, Z. Zhou, and K. W. Ross, "Randomized ensembled double *Q*-learning: Learning fast without a model," in *Proc. ICLR*, 2021, pp. 1–25.

[7] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, pp. 350–354, Oct. 2019.

[8] T. Johannink et al., "Residual reinforcement learning for robot control," in *Proc. ICRA*, 2019, pp. 6023–6029.

[9] L. Zou, L. Xia, Z. Ding, J. Song, W. Liu, and D. Yin, "Reinforcement learning to optimize long-term user engagement in recommender systems," in *Proc. KDD*, 2019, pp. 2810–2818.

[10] F. Ni et al., "A multi-graph attributed reinforcement learning based optimization algorithm for large-scale hybrid flow shop scheduling problem," in *Proc. 27th KDD*, 2021, pp. 3441–3451.

[11] J. Hao et al., "Exploration in deep reinforcement learning: From single-agent to multiagent domain," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 8762–8782, Jul. 2024.

[12] O. Sigaud, "Combining evolution and deep reinforcement learning for policy search: A survey," 2022, *arXiv:2203.14009*.

[13] T. Eimer, M. Lindauer, and R. Raileanu, "Hyperparameters in reinforcement learning and how to tune them," 2023, *arXiv:2306.01324*.

[14] E. Nikishin, M. Schwarzer, P. D'Oro, P. Bacon, and A. C. Courville, "The primacy bias in deep reinforcement learning," in *Proc. ICML*, 2022, pp. 1–20.

[15] T. Bäck and H. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evol. Comput.*, vol. 1, no. 1, pp. 1–23, Mar. 1993.

[16] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York, NY, USA: Oxford Univ. Press, Inc., 1996.

[17] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *Proc. ICGTSPICC*, 2016, pp. 261–265.

[18] Z. Zhou, Y. Yu, and C. Qian, *Evolutionary Learning: Advances in Theories and Algorithms*. Singapore: Springer, 2019.

[19] K. A. D. Jong, "Evolutionary computation: A unified approach," in *Proc. 10th GECCO*, 2020, pp. 2245–2258.

[20] J. Li, X. Li, and A. Wood, "Species based evolutionary algorithms for multimodal optimization: A brief review," in *Proc. CEC*, 2010, pp. 1–8.

[21] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.

[22] K.-H. Han and J.-H. Kim,, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 580–593, Dec. 2002.

[23] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," 2017, *arXiv: 1712.06567*.

[24] X. Yu, C. Li, and J. Zhou, "A constrained differential evolution algorithm to solve UAV path planning in disaster scenarios," *Knowl.-Based Syst.*, vol. 204, Sep. 2020, Art. no. 106209.

[25] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 4, pp. 904–916, Jul. 2019.

[26] D. Dasgupta and Z. Michalewicz, *Evolutionary Algorithms in Engineering Applications*. Berlin, Germany: Springer, 2013.

[27] S. Kelly and M. Heywood, "Emergent solutions to high-dimensional multitask reinforcement learning," *Evol. Comput.*, vol. 26, no. 3, pp. 347–380, 2018.

[28] R. J. Smith and M. I. Heywood, "A model of external memory for navigation in partially observable visual reinforcement learning tasks," in *Proc. EuroGP*, 2019, pp. 162–177.

[29] S. Kelly, T. Voegerl, W. Banzhaf, and C. Gondro, "Evolving hierarchical memory-prediction machines in multi-task reinforcement learning," *Genet. Program. Evol. Mach.*, vol. 22, pp. 573–605, Oct. 2021.

[30] T. Li, Y. Meng, and L. Tang, "Scheduling of continuous annealing with a multi-objective differential evolution algorithm based on deep reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 2, pp. 1767–1780, Apr. 2024.

[31] Y. Wang, T. Zhang, Y. Chang, B. Liang, X. Wang, and B. Yuan, "A surrogate-assisted controller for expensive evolutionary reinforcement learning," 2022, *arXiv:2201.00129*.

[32] J. Hao, P. Li, H. Tang, Y. Zheng, X. Fu, and Z. Meng, "ERL-RE^2: Efficient evolutionary reinforcement learning with shared state representation and individual policy representation," in *Proc. ICLR*, 2023, pp. 1–29.

[33] S. W. Wilson, "Classifier fitness based on accuracy," *Evol. Comput.*, vol. 3, no. 2, pp. 149–175, 1995.

[34] S. W. Wilson, "Classifiers that approximate functions," *Nat. Comput.*, vol. 1, nos. 2–3, pp. 211–234, 2002.

[35] P. L. Lanzi and D. Loiacono, "XCSF with tile coding in discontinuous action-value landscapes," *Evol. Intell.*, vol. 8, pp. 117–132, Apr. 2015.

[36] R. J. Preen and L. Bull, "Dynamical genetic programming in XCSF," *Evol. Comput.*, vol. 21, no. 3, pp. 361–87, 2013.

[37] Y. Dhebar, K. Deb, S. Nageshrao, L. Zhu, and D. Filev, "Interpretable-AI policies using evolutionary nonlinear decision trees for discrete action systems," 2020, *arXiv:2009.09521*.

[38] L. L. Custode and G. Iacca, "Interpretable AI for policy-making in pandemics," in *Proc. GECCO*, 2022, pp. 1763–1769.

[39] L. L. Custode and G. Iacca, "A co-evolutionary approach to interpretable reinforcement learning in environments with continuous action spaces," in *Proc. SSCI*, 2021, pp. 1–8.

[40] A. Ferigo, L. L. Custode, and G. Iacca, "Quality diversity evolutionary learning of decision trees," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.*, 2023, pp. 425–432.

[41] L. L. Custode and G. Iacca, "Social interpretable reinforcement learning," 2024, *arXiv:2401.15480*.

[42] J. Kubalík, E. Derner, J. Žegklitz, and R. Babuška, "Symbolic regression methods for reinforcement learning," *IEEE Access*, vol. 9, pp. 139697–139711, 2021.

[43] J. Bhatia, H. Jackson, Y. Tian, J. Xu, and W. Matusik, "Evolution gym: A large-scale benchmark for evolving soft robots," in *Proc. 35th NeurIPS*, 2021, pp. 1–14.

[44] H. Dong, J. Zhang, and C. Zhang, "Leveraging hyperbolic embeddings for coarse-to-fine robot design," in *Proc. CoRR*, 2023, pp. 1–28.

[45] S. Liu, W. Yao, H. Wang, W. Peng, and Y. Yang, "Rapidly evolving soft robots via action inheritance," *IEEE Trans. Evol. Comput.*, early access, Oct. 25, 2023, doi: 10.1109/TEVC.2023.3327459.

[46] A. Gupta, S. Savarese, S. Ganguli, and L. Fei-Fei, "Embodied intelligence via learning and evolution," *Nat. Commun.*, vol. 12, p. 5721, Oct. 2021.

[47] D. J. Hejna III, P. Abbeel, and L. Pinto, "Task-agnostic morphology evolution," 2021, *arXiv:2102.13100*.

[48] F. Pigozzi, F. J. C. Verdù, and E. Medvet, "How the morphology encoding influences the learning ability in body-brain co-optimization," in *Proc. GECCO*, 2023, pp. 1045–1054.

[49] S. Majumdar, S. Khadka, S. Miret, S. McAleer, and K. Tumer, "Evolutionary reinforcement learning for sample-efficient multiagent coordination," in *Proc. ICML*, 2020, pp. 6651–6660.

[50] A. A. Aydeniz, R. T. Loftin, and K. Tumer, "Novelty seeking multiagent evolutionary reinforcement learning," in *Proc. GECCO*, 2023, pp. 402–410.

[51] Y. Du, Y. Wang, Y. Cong, W. Jiang, and S. Pu, "Evolution strategies enhanced complex multiagent coordination," in *Proc. IJCNN*, 2023, pp. .

[52] Y. Guo, X. Xie, R. Zhao, C. Zhu, J. Yin, and H. Long, "Cooperation and competition: Flocking with evolutionary multi-agent reinforcement learning," in *Proc. ICONIP*, 2022, pp. 271–283.

[53] P. Li, J. Hao, H. Tang, Y. Zheng, and X. Fu, "RACE: Improve multi-agent reinforcement learning with representation asymmetry and collaborative evolution," in *Proc. ICML*, 2023, pp. 19490–19503.

[54] S. Khadka and K. Tumer, "Evolution-guided policy gradient in reinforcement learning," in *Proc. 32nd NeurIPS*, 2018, pp. 1–13.

[55] S. Khadka et al., "Collaborative evolutionary reinforcement learning," in *Proc. 36th ICML*, 2019, pp. 1–12.

[56] C. Bodnar, B. Day, and P. Lió, "Proximal distilled evolutionary reinforcement learning," in *Proc. AAAI*, 2020, pp. 1–10.

[57] S. Zhu, F. Belardinelli, and B. G. León, "Evolutionary reinforcement learning for sparse rewards," in *Proc. GECCO*, 2021, pp. 1508–1512.

[58] H. Zheng, P. Wei, J. Jiang, G. Long, Q. Lu, and C. Zhang, "Cooperative heterogeneous deep reinforcement learning," in *Proc. NeurIPS*, 2020, pp. 1–11.

[59] B. Zheng and R. Cheng, "Rethinking population-assisted off-policy reinforcement learning," in *Proc. GECCO*, 2023, pp. 1–11.

[60] K. Suri, "Off-policy evolutionary reinforcement learning with maximum mutations," in *Proc. 21st AAMAS*, 2022, pp. 1237–1245.

[61] N. Kim, H. Baek, and H. Shin, "PGPS: Coupling policy gradient with population-based search," unpublished.

[62] P. Li, H. Jianye, H. Tang, Y. Zheng, and F. Barez, "Value-evolutionary-based reinforcement learning," in *Proc. 41st ICML*, 2024, pp. 1–15.

[63] P. Li, Y. Zheng, H. Tang, X. Fu, and H. Jianye, "EvoRainbow: Combining improvements in evolutionary reinforcement learning for policy search," in *Proc. ICML*, 2024, pp. 1–21.

[64] H. Bharadhwaj, K. Xie, and F. Shkurti, "Model-predictive control via cross-entropy and gradient-based optimization," in *Proc. L4DC*, 2020, pp. 1–11.

[65] H. Sikchi, W. Zhou, and D. Held, "Learning off-policy with online planning," in *Proc. CORL*, 2021, pp. 1–30.

[66] N. Hansen, H. Su, and X. Wang, "Temporal difference learning for model predictive control," in *Proc. ICML*, 2022, pp. 1–20.

[67] K. L. Downing, "Reinforced genetic programming," *Genet. Program. Evol. Mach.*, vol. 2, pp. 259–288, Sep. 2001.

[68] S. Mabu, K. Hirasawa, and J. Hu, "A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning," *Evol. Comput.*, vol. 15, no. 3, pp. 369–398, 2007.

[69] C. Lu, J. Kuba, A. Letcher, L. Metz, C. S. de Witt, and J. Foerster, "Discovered policy optimisation," in *Proc. 36th Conf. Neural Inf. Process. Syst.*, 2022, pp. 1–18.

[70] M. T. Jackson, C. Lu, L. Kirsch, R. T. Lange, S. Whiteson, and J. N. Foerster, "Discovering temporally-aware reinforcement learning algorithms," in *Proc. ICLR*, 2024, pp. 1–19.

[71] A. E. Eiben, M. Horvath, W. Kowalczyk, and M. C. Schut, "Reinforcement learning for online control of evolutionary algorithms," in *Proc. ESOA*, 2006, pp. 151–160.

[72] G. Shala, A. Biedenkapp, N. H. Awad, S. Adriaensen, M. Lindauer, and F. Hutter, "Learning step-size adaptation in CMA-ES," in *Proc. PPSN*, 2020, pp. 1–30.

[73] A. Biedenkapp, H. F. B., T. Eimer, F. Hutter, and M. Lindauer, "Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework," in *Proc. 24th ECAI*, 2020, pp. 1–8.

[74] H. Zhang, J. Sun, Y. Wang, J. Shi, and Z. Xu, "Variational reinforcement learning for hyper-parameter tuning of adaptive evolutionary algorithm," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 5, pp. 1511–1526, Oct. 2023.

[75] H. Zhang, J. Sun, T. Bäck, Q. Zhang, and Z. Xu, "Controlling sequential hybrid evolutionary algorithm by *Q*-learning," *IEEE Comput. Intell. Mag.*, vol. 18, no. 1, pp. 84–103, Feb. 2023.

[76] K. Xue et al., "Multi-agent dynamic algorithm configuration," in *Proc. 36th NeurIPS*, 2022, pp. 1–28.

[77] T. N. Huynh, D. T. T. Do, and J. Lee, "*Q*-learning-based parameter control in differential evolution for structural optimization," *Appl. Soft Comput.*, vol. 107, Aug. 2021, Art. no. 107464.

[78] Z. Hu, W. Gong, and S. Li, "Reinforcement learning-based differential evolution for parameters extraction of photovoltaic models," *Energy Rep.*, vol. 7, pp. 916–928, Nov. 2021.

[79] J. E. Pettinger and R. M. Everson, "Controlling genetic algorithms with reinforcement learning," in *Proc. GECCO*, 2002, p. 692.

[80] H. Zhang and J. Lu, "Adaptive evolutionary programming based on reinforcement learning," *Inf. Sci.*, vol. 178, pp. 971–984, Feb. 2008.

[81] A. Buzdalova and M. Buzdalov, "Increasing efficiency of evolutionary algorithms by choosing between auxiliary fitness functions with reinforcement learning," in *Proc. 11th ICMLA*, 2012, pp. 150–155.

[82] A. Buzdalova, V. Kononov, and M. Buzdalov, "Selecting evolutionary operators using reinforcement learning: Initial explorations," in *Proc. GECCO*, 2014, pp. 1033–1036.

[83] Y. Song, L. Wei, Q. Yang, J. Wu, L. Xing, and Y. Chen, "RL-GA: A reinforcement learning-based genetic algorithm for electromagnetic detection satellite scheduling problem," *Swarm Evol. Comput.*, vol. 77, Mar. 2023, Art. no. 101236.

[84] W. Yi, R. Qu, L. Jiao, and B. Niu, "Automated design of metaheuristics using reinforcement learning within a novel general search framework," *IEEE Trans. Evol. Comput.*, vol. 27, no. 4, pp. 1072–1084, Aug. 2023.

[85] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Evolutionary framework with reinforcement learning-based mutation adaptation," *IEEE Access*, vol. 8, pp. 194045–194071, 2020.

[86] X. Meng, H. Li, and A. Chen, "Multi-strategy self-learning particle swarm optimization algorithm based on reinforcement learning," *Math. Biosci. Eng.*, vol. 20, no. 5, pp. 8498–8530, 2023.

[87] Z. Tan and K. Li, "Differential evolution with mixed mutation strategy based on deep reinforcement learning," *Appl. Soft Comput.*, vol. 111, Nov. 2021, Art. no. 107678.

[88] M. Sharma, A. Komninos, M. López-Ibáñez, and D. Kazakov, "Deep reinforcement learning based parameter control in differential evolution," in *Proc. GECCO*, 2019, pp. 1–9.

[89] Z. Hu and W. Gong, "Constrained evolutionary optimization based on reinforcement learning using the objective function and constraints," *Knowl. Based Syst.*, vol. 237, Feb. 2022, Art. no. 107731.

[90] L. Peng, Z. Yuan, G. Dai, M. Wang, and Z. Tang, "Reinforcement learning-based hybrid differential evolution for global optimization of interplanetary trajectory design," *Swarm Evol. Comput*, vol. 81, Aug. 2023, Art. no. 101351.

[91] Z. Li, L. Shi, C. Yue, Z. Shang, and B. Qu, "Differential evolution based on reinforcement learning with fitness ranking for solving multimodal multiobjective problems," *Swarm Evol. Comput.*, vol. 49, pp. 234–244, Sep. 2019.

[92] Y. Huang, W. Li, F. Tian, and X. Meng, "A fitness landscape ruggedness multiobjective differential evolution algorithm with a reinforcement learning strategy," *Appl. Soft Comput.*, vol. 96, Nov. 2020, Art. no. 106693.

[93] Y. Tian, X. Li, H. Ma, X. Zhang, K. C. Tan, and Y. Jin, "Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 4, pp. 1051–1064, Aug. 2023.

[94] A. Pourchot and O. Sigaud, "CEM-RL: Combining evolutionary and gradient-based methods for policy search," in *Proc. ICLR*, 2019, pp. 1–19.

[95] Y. Tang, "Guiding evolutionary strategies with off-policy actor-critic," in *Proc. 20th AAMAS*, 2021, pp. 1–9.

[96] K. W. Pretorius and N. Pillay, "Population based reinforcement learning," in *Proc. SSCI*, 2021, pp. 1–8.

[97] L. Shi, S. Li, Q. Zheng, M. Yao, and G. Pan, "Efficient novelty search through deep reinforcement learning," *IEEE Access*, vol. 8, pp. 128809–128818, 2020.

[98] J. Liu and L. Feng, "Diversity evolutionary policy deep reinforcement learning," *Comput. Intell. Neurosci.*, vol. 2021, no. 1, 2021, Art. no. 5300189.

[99] G. Cideron, T. Pierrot, N. Perrin, K. Beguir, and O. Sigaud, "QD-RL: Efficient mixing of quality and diversity in reinforcement learning," 2020, *arXiv:2006.08505*.

[100] O. Nilsson and A. Cully, "Policy gradient assisted MAP-elites," in *Proc. GECCO*, 2021, pp. 866–875.

[101] B. Lim, M. Flageat, and A. Cully, "Understanding the synergies between quality-diversity and deep reinforcement learning," in *Proc. GECCO*, 2023, pp. 1212–1220.

[102] B. Tjanaka, M. C. Fontaine, J. Togelius, and S. Nikolaidis, "Approximating gradients for differentiable quality diversity in reinforcement learning," in *Proc. GECCO*, 2022, pp. 1102–1111.

[103] K. Xue, R. Wang, P. Li, D. Li, H. Jianye, and C. Qian, "Sample-efficient quality-diversity by cooperative coevolution," in *Proc. ICLR*, 2023, pp. 1–26.

[104] R. Wang, K. Xue, C. Guan, and C. Qian, "Quality-diversity with limited resources," 2024, *arXiv:2406.03731*.

[105] Y. Zheng et al., "Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning," in *Proc. ASE*, 2019, pp. 772–784.

[106] T. N. Mundhenk, M. Landajuela, R. Glatt, C. P. Santiago, D. M. Faissol, and B. K. Petersen, "Symbolic regression via neural-guided genetic programming population seeding," 2021, *arXiv:2111.00053*.

[107] M. I. Radaideh and K. Shirvan, "Rule-based reinforcement learning methodology to inform evolutionary algorithms for constrained optimization of engineering applications," *Knowl. Based Syst.*, vol. 217, Apr. 2021, Art. no. 106836.

[108] H. Ye, J. Wang, Z. Cao, H. Liang, and Y. Li, "DeepACO: Neural-enhanced ant systems for combinatorial optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 1–23.

[109] S. Niekum, A. G. Barto, and L. Spector, "Genetic programming for reward function search," *IEEE Trans. Auton. Mental Develop.*, vol. 2, no. 2, pp. 83–90, Jun. 2010.
[110] J. D. Co-Reyes et al., "Evolving reinforcement learning algorithms," in *Proc. ICLR*, 2021, pp. 1–15.
[111] S. Elfwing, E. Uchibe, K. Doya, and H. I. Christensen, "Evolutionary development of hierarchical learning structures," *IEEE Trans. Evol. Comput.*, vol. 11, no. 2, pp. 249–264, Apr. 2007.
[112] Q. Liu, Y. Wang, and X. Liu, "PNS: Population-guided novelty search for reinforcement learning in hard exploration environments," in *Proc. IROS*, 2021, pp. 5627–5634.
[113] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "Go-explore: A new approach for hard-exploration problems," 2019, *arXiv:1901.10995*.
[114] S. Chang, J. Yang, J. Choi, and N. Kwak, "Genetic-gated networks for deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–10.
[115] A. Hallawa et al., "Evo-RL: Evolutionary-driven reinforcement learning," in *Proc. GECCO*, 2021, pp. 153–154.
[116] L. Yuan et al., "Robust multi-agent coordination via evolutionary generation of auxiliary adversarial attackers," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 1–10.
[117] L. Yuan, F. Chen, Z. Zhang, and Y. Yu, "Communication-robust multi-agent learning by adaptable auxiliary multi-agent adversary generation," 2023, *arXiv:2305.05116*.
[118] Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, and X. Wang, "Evolutionary population curriculum for scaling multi-agent reinforcement learning," in *Proc. ICLR*, 2020, pp. 1–18.
[119] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "MAPPER: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments," in *Proc. IROS*, 2020, pp. 11748–11754.
[120] S. Elfwing, E. Uchibe, and K. Doya, "Online meta-learning by parallel algorithm competition," in *Proc. GECCO*, 2018, pp. 426–433.
[121] M. Jaderberg et al., "Population based training of neural networks," 2017, *arXiv:1711.09846*.
[122] J. K. H. Franke, G. Köhler, A. Biedenkapp, and F. Hutter, "Sample-efficient automated deep reinforcement learning," in *Proc. ICLR*, 2021, pp. 1–23.
[123] A. Sehgal, N. Ward, H. M. La, C. Papachristos, and S. J. Louis, "GA+DDPG+HER: Genetic algorithm-based function optimizer in deep reinforcement learning for robotic manipulation tasks," in *Proc. IRC*, 2022, pp. 1–2.
[124] J. Grigsby, J. Y. Yoo, and Y. Qi, "Towards automatic actor-critic solutions to continuous control," 2021, *arXiv:2106.08918*.
[125] Y. Tang and K. Choromanski, "Online hyper-parameter tuning in off-policy learning via evolutionary strategies," 2020, *arXiv:2006.07554*.
[126] D. Kalashnikov et al., "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proc. 2nd Conf. Robot Learn.*, 2018, pp. 1–23.
[127] R. Simmons-Edler, B. Eisner, E. Mitchell, H. S. Seung, and D. D. Lee, "*Q*-learning for continuous actions with cross-entropy guided policies," 2019, *arXiv:1903.10605*.
[128] Y. Ma, T. Liu, B. Wei, Y. Liu, K. Xu, and W. Li, "Evolutionary action selection for gradient-based policy learning," 2022, *arXiv:2201.04286*.
[129] Z. Shi and S. P. N. Singh, "Soft actor-critic with cross-entropy policy optimization," 2021, *arXiv:2112.11115*.
[130] L. Shao, Y. You, M. Yan, S. Yuan, Q. Sun, and J. Bohg, "GRAC: Self-guided and self-regularized actor-critic," in *Proc. Conf. Robot Learn.*, 2021, pp. 1–16.
[131] L. Pan, L. Huang, T. Ma, and H. Xu, "Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification," in *Proc. ICML*, 2022, pp. 1–17.
[132] C. S. de Witt, B. Peng, P. Kamienny, P. H. S. Torr, W. Böhmer, and S. Whiteson, "Deep multi-agent reinforcement learning for decentralized continuous cooperative control," 2020, *arXiv:2003.06709v2*.
[133] A. Leite, M. Candadai, and E. J. Izquierdo, "Reinforcement learning beyond the bellman equation: Exploring critic objectives using evolution," in *Proc. ALIFE*, 2020, pp. 1–9.
[134] E. Marchesini, D. Corsi, and A. Farinelli, "Genetic soft updates for policy evolution in deep reinforcement learning," in *Proc. ICLR*, 2021, pp. 1–15.
[135] E. Marchesini and C. Amato, "Improving deep policy gradients with value function search," in *Proc. ICLR*, 2023, pp. 1–18.
[136] A. Y. Majid, S. Saaybi, T. Rietbergen, V. François-Lavet, R. V. Prasad, and C. J. M. Verhoeven, "Deep reinforcement learning versus evolution strategies: A comparative survey," 2021, *arXiv:2110.01411*.
[137] Q. Zhu et al., "A survey on evolutionary reinforcement learning algorithms," *Neurocomputing*, vol. 556, Nov. 2023, Art. no. 126628.
[138] M. M. Drugan, "Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms," *Swarm Evol. Comput.*, vol. 44, pp. 228–246, Feb. 2019.
[139] S. Kelly and J. Schossau, "Evolutionary computation and the reinforcement learning problem," in *Handbook of Evolutionary Machine Learning*. Singapore: Springer Nat., 2023.
[140] H. Bai, R. Cheng, and Y. Jin, "Evolutionary reinforcement learning: A survey," 2023, *arXiv:2303.04150*.
[141] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.
[142] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[143] T. Salimans, J. Ho, X. Chen, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*.
[144] J. Lehman and K. O. Stanley, "Novelty search and the problem with objectives," in *Genetic Programming Theory and Practice IX*. New York, NY, USA: Springer, 2011.
[145] J. Mouret and J. Clune, "Illuminating search spaces by mapping elites," 2015, *arXiv:1504.04909*.
[146] J. R. Koza et al., *Genetic Programming II*. Cambridge, MA, USA: MIT Press, 1994.
[147] C. Watkins and P. Dayan, "*Q*-learning," *Mach. Learn.*, vol. 8, pp. 279–292, May 1992.
[148] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, May. 1992.
[149] S. M. Kakade, "A natural policy gradient," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 1–8.
[150] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 1–7.
[151] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[152] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[153] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. ICLR*, 2016, pp. 1–14.
[154] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1–10.
[155] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1–10.
[156] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. ICML*, 2015, pp. 1–16.
[157] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[158] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.
[159] Y. Tassa et al., "Deepmind control suite," 2018, *arXiv:1801.00690*.
[160] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1–14.
[161] B. Peng et al., "FACMAC: Factored multi-agent centralised policy gradients," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, 2021, pp. 1–14.
[162] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.
[163] M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge, "Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges," *Inf. Sci.*, vol. 317, pp. 224-245, Oct. 2015.
[164] F. F. Peres and M. Castelli, "Combinatorial optimization problems and metaheuristics: Review, challenges, design, and development," *Appl. Sci.*, vol. 11, no. 14, p. 6449, 2021.
[165] K. Tang et al., "Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization," Dept. Nat. Inspir. Comput. Appl. Lab., Univ. Sci. Technol. China, Hefei, China, Rep. LSGO-CEC-2010, 2007.

[166] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, and H. China, "Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization," in *Proc. GENE*, 2013, pp. 1–23.
[167] J. Kudela, "A critical problem in benchmarking and analysis of evolutionary computation methods," *Nat. Mach. Intell.*, vol. 4, pp. 1238–1245, Dec. 2022.
[168] M. Flood, "The traveling-salesman problem," *Oper. Res.*, vol. 4, no. 1, pp. 61–75, 1956.
[169] P. Toth and D. Vigo, *The Vehicle Routing Problem*. Philadelphia, PA, USA: SIAM, 2002.
[170] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278-285, 1993.
[171] J. Hao et al., "Exploration in deep reinforcement learning: From single-agent to multiagent domain," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 8762–8782, Jul. 2024.
[172] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodol. Comput. Appl. Probab.*, vol. 1, pp. 127–190, Sep. 1999.
[173] D. Hein, A. Hentschel, V. Sterzing, M. Tokic, and S. Udluft, "Introduction to the 'industrial benchmark'," 2016, *arXiv:1610.03793*.
[174] L. Spector and A. Robinson, "Genetic programming and autoconstructive evolution with the push programming language," *Genet. Program. Evolv. Mach.*, vol. 3, pp. 7–40, Mar. 2002.
[175] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Artif. Intell. Res.*, vol. 13, pp. 227–303, Nov. 2000.
[176] S. Sapora, G. Swamy, C. Lu, Y. W. Teh, and J. N. Foerster, "EvIL: Evolution strategies for generalisable imitation learning," in *Proc. ICML*, 2024, pp. 1–17.
[177] A. Lupu, C. Lu, J. L. Liesen, R. T. Lange, and J. N. Foerster, "Behaviour distillation," in *Proc. ICLR*, 2024, pp. 1–17.
[178] C. Lu, T. Willi, A. Letcher, and J. N. Foerster, "Adversarial cheap talk," in *Proc. ICML*, 2023, pp. 1–25.
[179] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–9.
[180] D. Lobo and M. Levin, "Inferring regulatory networks from experimental morphological phenotypes: A computational method reverse-engineers planarian regeneration," *PLoS Comput. Biol.*, vol. 11, no. 6, 2015, Art. no. e1004295.
[181] M. Flageat, F. Chalumeau, and A. Cully, "Empirical analysis of PGA-MAP-elites for neuroevolution in uncertain domains," *ACM Trans. Evol. Learn.*, vol. 3, no. 1, pp. 1–32, 2023.
[182] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka, "Dropout q-functions for doubly efficient reinforcement learning," 2021, *arXiv:2110.02034*.
[183] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, part II: Metaheuristics," *Transp. Sci.*, vol. 39, no. 1, pp. 119–139, 2005.
[184] S. Wright et al., "The roles of mutation, inbreeding, crossbreeding, and selection in evolution," in *Proc. 6th Int. Congr. Genet.*, 1932, pp. 356–366.
[185] W. M. Spears, *Evolutionary Algorithms: The Role of Mutation and Recombination*. Berlin, Germany: Springer, 2000.
[186] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," Ph.D. dissertation, INRIA, Le Chesnay Cedex, France, 2009.
[187] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE CEC*, 2014, pp. 1658–1665.
[188] G. Zhang and Y. Shi, "Hybrid sampling evolution strategy for solving single objective bound constrained problems," in *Proc. IEEE CEC*, 2018, pp. 1–7.
[189] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. AAMAS*, 2018, pp. 2085–2087.
[190] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[191] J. M. Baldwin, "A new factor in evolution," *Diacronia*, vol. 30, 2018.
[192] C. L. Morgan, "On modification and variation," *Science*, vol. 4, no. 99, pp. 733–740, 1896.
[193] G. Hinton and S. Nowlan, "How learning can guide evolution," *Complex Syst.*, vol. 1, no. 3, pp. 1–8, 1987.
[194] K. Hirasawa, M. Okubo, H. Katagiri, J. Hu, and J. Murata, "Comparison between genetic network programming (GNP) and genetic programming (GP)," in *Proc. CEC*, 2001, pp. 1276–1282.
[195] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.
[196] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–12.
[197] D. Hafner et al., "Learning latent dynamics for planning from pixels," in *Proc. ICML*, 2019, pp. 1–11.
[198] T. Wang and J. Ba, "Exploring model-based planning with policy networks," in *Proc. ICLR*, 2020, pp. 1–20.
[199] Y. Wang et al., "Curriculum-based co-design of morphology and control of voxel-based soft robots," in *Proc. ICLR*, 2023, pp. 1–20.
[200] Y. Wang et al., "PreCo: Enhancing generalization in co-design of modular soft robots via brain-body pre-training," in *Proc. CORL*, 2023, pp. 478–498.
[201] F. Tanaka and C. Aranha, "Co-evolving morphology and control of soft robots using a single genome," in *Proc. SSCI*, 2022, pp. 1235–1242.
[202] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, Apr. 2018.
[203] C. Ryan, J. J. Collins, and M. O'Neill, "Grammatical evolution: Evolving programs for an arbitrary language," in *Proc. Eur. Conf. Genet. Program.*, 1998, pp. 83–96.
[204] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proc. ICEC*, 1996, pp. 312–317.
[205] P. Larranaga, "Optimization by learning and simulation of Bayesian and Gaussian networks," Univ. Basque Country, Leioa, Spain, Rep. EHU-KZAAIK-4/99, 1999.
[206] R. J. Urbanowicz and J. H. Moore, "Learning classifier systems: A complete introduction, review, and roadmap," *J. Artif. Evol. Appl.*, vol. 2009, no. 1, 2009, Art. no. 736398.
[207] M. V. Butz, D. E. Goldberg, and P. L. Lanzi, "Gradient descent methods in learning classifier systems: Improving XCS performance in multistep problems," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 452–473, Oct. 2005.
[208] P. L. Lanzi, "Learning classifier systems: Then and now," *Evol. Intell.*, vol. 1, pp. 63–82, Mar. 2008.
[209] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Comput. Oper. Res.*, vol. 134, Oct. 2021, Art. no. 105400.
[210] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. Cambridge, MA, USA: MIT Press, 2022.