Neural Computing and Applications
https://doi.org/10.1007/s00521-023-09124-5

S.I.: INNOVATIONS IN AI-BASED SYSTEMS AND SOFTWARE

Quality–diversity optimization of decision trees for interpretable reinforcement learning

Andrea Ferigo[1] · Leonardo Lucio Custode[1] · Giovanni Iacca[1]

Received: 30 July 2023 / Accepted: 16 October 2023

Abstract
In the current Artificial Intelligence (AI) landscape, addressing explainability and interpretability in Machine Learning (ML) is of critical importance. In fact, the vast majority of works on AI focus on Deep Neural Networks (DNNs), which are not interpretable, as they are extremely hard to inspect and understand for humans. This is a crucial disadvantage of these methods, which hinders their trustability in high-stakes scenarios. On the other hand, interpretable models are considerably easier to inspect, which allows humans to test them exhaustively, and thus trust them. While the fields of eXplainable Artificial Intelligence (XAI) and Interpretable Artificial Intelligence (IAI) are progressing in supervised settings, the field of Interpretable Reinforcement Learning (IRL) is falling behind. Several approaches leveraging Decision Trees (DTs) for IRL have been proposed in recent years. However, all of them use goal-directed optimization methods, which may have limited exploration capabilities. In this work, we extend a previous study on the applicability of Quality–Diversity (QD) algorithms to the optimization of DTs for IRL. We test the methods on two well-known Reinforcement Learning (RL) benchmark tasks from OpenAI Gym, comparing their results in terms of score and "illumination" patterns. We show that using QD algorithms is an effective way to explore the search space of IRL models. Moreover, we find that, in the context of DTs for IRL, QD approaches based on MAP-Elites (ME) and its variant Covariance Matrix Adaptation MAP-Elites (CMA-ME) can significantly improve convergence speed over the goal-directed approaches.

Keywords Quality–diversity optimization · Explainability · Decision trees · Reinforcement learning

1 Introduction

Thanks to the recent developments in Deep Learning (DL), AI is permeating our daily lives. However, there is a growing concern about the trustworthiness of DL-based systems [1]. This is especially evident in high-stakes applications (e.g., finance, healthcare, and industrial control systems), where a single unexpected behavior shown by the AI system can cause catastrophic damages. These issues led, in recent years, to the growing interest in XAI approaches [2–5].

However, the developments of the XAI field are seeing opposing points of view from the research community. On the one hand, some researchers think that DL is "hitting a wall" [6, 7]. Other researchers, instead, neglect the importance of XAI [8]. Regardless this dispute, what is clear, judging from the interest in this topic from public institutions (e.g., the EU[1] and UNESCO[2]), is that XAI will have a dominant role in the future of AI.

To add more fuel to this debate, some researchers are advocating for a particular subset of XAI, called IAI, which focuses on the use of transparent models (e.g., DTs and rule-based systems) [9]. In fact, these researchers point out that general XAI methods merely provide *a posteriori*

Andrea Ferigo and Leonardo Lucio Custode have contributed equally to this work.

✉ Giovanni Iacca
giovanni.iacca@unitn.it

Andrea Ferigo
andrea.ferigo@unitn.it

Leonardo Lucio Custode
leonardo.custode@unitn.it

[1] Department of Information Engineering and Computer Science, University of Trento, Via Sommarive 9, 38123 Povo, Trento, Italy

[1] https://eur-lex.europa.eu/resource.html?uri=cellar:e0649735-a372-11eb-9585-01aa75ed71a1.0001.02/DOC_1 &format=PDF.

[2] https://unesdoc.unesco.org/ark:/48223/pf0000380455/.

Published online: 14 November 2023

explanations of opaque ML models (e.g., DNNs), usually through simple model approximations, and as such they are not exact. Not having an exact explanation in high-stakes scenarios is however not acceptable, as a wrong (and unexplainable) model behavior can lead to significant damages. As described in [2], interpretability is instead a *structural* property of a model, which means that an interpretable model can be exactly understood and inspected by humans, while explainability is, in general, a *behavioral* property of a model, meaning that one can give "explanations" about the decisions taken by the model even without knowing its internal details.[3]

IRL has been recognized as one of the current grand challenges in the field of AI [9]. In fact, since RL is an extremely general methodology, it can be applied to a wide variety of problems, from the definition of taxation policies [10] to the control of tokamak plasmas in nuclear fusion plants [11]. These two application fields are also good examples of high-stakes domains where AI-based systems can be highly beneficial but, if controlled poorly, may have catastrophic effects. In these scenarios, users cannot just use opaque models in a plug-and-play manner, because even high-performing policies may be biased or have unpredictable behavior [9, 12]. Using an interpretable model, instead, would allow a thorough testing and inspection of the trained policy, to assess potentially unsafe behaviors.

Despite the importance and the potential application of IRL, this field is however currently falling behind the more established field of XAI. As discussed in [9], this is mainly because, compared to standard supervised settings (where a posteriori XAI models can be applied), IRL is much more challenging, mainly because: 1) in RL, the agent does not receive all the data at once, observations may be partial, and rewards may be delayed and depend on the long-term consequences of a series of interdependent state-actions transitions; 2) the search spaces in RL can be massive, with uncertain estimates. Yet, interpretability could be particularly valuable in RL, as it might help to reduce the RL search space, and possibly remove actions that might lead to harm. For these reasons, the research on IRL is highly relevant and timely, yet it presents several important challenges.

To partially address these challenges, recently some works proposed hybrid approaches for producing IRL models [13–15]. In these works, the authors combine Grammatical Evolution (GE) [16], i.e., a variant of Genetic Programming (GP) [17], and Q-learning [18], to produce interpretable DTs with RL-optimized behaviors. Another hybrid AI approach that makes use of DTs is presented in [19]. There, the authors combine deep RL models with DTs to obtain learned and instinctive behaviors.

Another direction that has attracted a growing interest in the recent literature focuses on RL algorithms that are capable of discovering not just one single policy, but rather a *set of good, diverse policies*. RL algorithms that focus on diversity are indeed able to explore better the policy space, and discover more complex (and possibly more robust) behaviors [20]. Examples of applications of RL where diversification may be needed are in the field of game playing, where agents capable of producing moves that go beyond known patterns can be more successful. Another area of applications may be in the context of intelligent control systems, such as smart industry or smart buildings, where a diversity-driven AI-based system could be able to propose diverse control strategies to produce different system trajectories with different dynamics, also reacting to unexpected conditions, such as changes in the task and/or environment.

In IRL, diversity-driven algorithms could be even more important, as they would allow the discovery of a set of solutions that can solve the task at hand not only with good performance but also using different strategies with different levels of complexity and interpretability. This could be extremely useful, as it would allow the users of the system to analyze and interpret the set of solutions and draw different insights on the problem at hand, and ultimately, improve their general understanding of the problem and their problem-solving skills.

An early attempt in this direction is presented in [20], where the authors propose a method called DOMiNO for discovering behaviorally different policies, without affecting performance in a significant way. However, it should be noted that DOMiNO does not focus on interpretability, as it is based on DNNs. Furthermore, it uses gradient-based optimization, differently from the present study where instead we use gradient-free optimization. Concerning DTs-based models, most of the previous works focus on goal-directed (i.e., not focused on diversity) optimization, while only limited work has been done aimed at searching for diversity in the optimization of DTs for IRL.

One possible way to achieve diversity in this context is through the use of the so-called QD algorithms [21], namely, optimization methods driven by an explicit search for diversity of solutions rather than the minimization of maximization of a given objective function (as done in traditional goal-directed approaches). QD methods allow in fact for a better exploration of the search space, potentially discovering better-performing policies than goal-directed optimization [22, 23]. Among the existing QD algorithms, of particular interest, especially (but not only) in the context of DTs, is ME [23], which as we will see later is one of

[3] Note that, in general, interpretability $\Rightarrow$ explainability, while explainability $\not\Rightarrow$ interpretability.

the main methods we use in the present study. This algorithm has in fact many desirable properties that can be particularly useful for IRL. First of all, differently from any other (either goal-directed or QD) algorithm, ME allows the user to define specific feature descriptors (as we will see in Sect. 2.2.3, in our case, these are the tree depth and the action entropy) and explicitly makes use of these descriptors to explore the search space. These descriptors can be either problem-dependent or problem-agnostic (as the ones chosen in our experiments), to provide different levels of abstraction and knowledge in the exploration process (potentially, also in an interactive way). Furthermore, the possibility of defining a user-specified feature space comes with a simple yet intuitive visualization in the form of heatmaps (such as the ones that we will show in Sect. 3), that easily allow the user to identify the most promising/effective areas of the search space. Another important advantage of ME is that it does not make use of gradients (differently from gradient-based, goal-oriented optimization algorithms, such as DOMiNO [20]), hence it can be applied also to non-differentiable objective functions. Finally, another important feature of ME is its ease of implementation and use.

In our previous work [24], we have made a first step toward QD optimization of DTs for IRL. However, that work was limited to a single Evolutionary Algorithm (EA), namely GE, and a single QD scheme, namely ME.

In the present work, we extend the analysis previously presented in [24] by significantly increasing the scope of the experimentation. In particular, we address two relevant research questions in the context of optimization of DTs for IRL, namely: (1) What are the effects of the introduction of QD schemes in different EAs? (2) How do different QD schemes affect the optimization process? To address these questions, we employ two different EAs, namely, GE [16] and GP [17], and two different QD schemes, i.e., ME [23] and its recent variant CMA-ME [25], comparing them with their corresponding baseline goal-directed (i.e., with no QD scheme) EAs. We study the performance and the "illumination" patterns of the methods shown above in two well-known benchmarks from the OpenAI Gym suite [26]: CartPole-v1 and MountainCar-v0.

To summarize, the main contributions of the present work are the following:

- We conduct an experimental study on two RL tasks, with two different EAs (GE and GP), combined with two different QD schemes, namely, ME and CMA-ME.
- We compare the QD approaches with vanilla (i.e., goal-directed) versions of GE and GP, used as baselines, thus for a total of 6 different algorithms for each task.
- For each task and algorithm, we analyze the fitness trend and the corresponding "illumination" capability w.r.t. a feature space described by the trees' depth and the entropy of their actions.

The rest of the paper are structured as follows. The next section describes the methods used in our comparison. Then, Sect. 3 shows the numerical results. Finally, Sect. 4 concludes this work and suggests possible future works. Please note that the list of acronyms used throughout the paper is reported in Table 1.

2 Methods

As mentioned above, our goal is to discover diverse IRL models for a given task. To do so, we evolve DTs by combining EAs with QD schemes, and RL (through Q-learning, see Sect. 2.3). More specifically, we compare different setups, involving two goal-directed EAs and two QD schemes. The two EAs we employ are GP [17] and GE [16]. The two QD schemes, instead, are the following:

- ME [23]: in this scheme, we employ the ME selection scheme on top of the EA used;
- CMA-ME [25]: in this scheme, the solutions coming from the ME algorithm are further refined by using Covariance Matrix Adaptation Evolution Strategies (CMAES) [27]. Finally, the ME scheme is applied to the refined solutions.

Table 1 List of acronyms used in the paper

Acronym	Definition
AI	Artificial intelligence
CMA	Covariance matrix adaptation
CMA-ES	Covariance matrix adaptation evolution strategies
CMA-ME	Covariance matrix adaptation MAP-Elites
DL	Deep learning
DNN	Deep neural network
DT	Decision tree
EA	Evolutionary algorithm
GE	Grammatical evolution
GP	Genetic programming
IAI	Interpretable artificial intelligence
IRL	Interpretable reinforcement learning
ME	MAP-Elites
ML	Machine learning
QD	Quality–Diversity
RL	Reinforcement learning
XAI	eXplainable artificial intelligence

Moreover, we compare the results obtained with the two QD schemes to two goal-directed EAs (based, respectively, on vanilla GE and GP), which act as baselines.

2.1 Evolutionary algorithm

Here, we briefly present the EAs used in the experimentation.

2.1.1 Grammatical evolution

In the GE setup, the genotype of a DT (i.e., its encoding) is a vector $\mathbf{g} = (g_0, \ldots, g_{\text{size}})$; $g_i \in [0, M]$, where M is an integer whose value must be significantly larger than the number of productions for each of the rules, to ensure uniform probabilities for all the productions of all the rules.

As a mutation operator, we use a uniform random mutation: given a genotype ($\mathbf{g}$), we replace a gene choosing a new value $g_{\text{new}} \sim \mathcal{U}(0, M)$, with probability p_g, where $\mathcal{U}(\cdot)$ denotes the uniform probability distribution.

As a crossover operator, instead, we use one-point crossover: given $\mathbf{p}_1$ and $\mathbf{p}_2$, we choose a random splitting point and create two new solutions by concatenating the two split genotypes.

The grammar used for GE in our experiments is shown in Table 2.

2.1.2 Genetic programming

When using GP, the genotype coincides with the DT. To enforce the interpretability of the solutions, we set a limit on the expression length.

The mutation operator works as follows. Initially, it randomly chooses a node. Then, if the chosen node is a leaf, it replaces it with a condition. Otherwise, if the chosen node is a condition, its expression is randomly modified.

To perform crossover between two trees, we randomly select two nodes, one for each tree, and swap them.

The functional set is composed of the following symbols: `if_then_else`, `gt`, `lt`, $+$, -, $\cdot$, /.

The terminal set, instead, is composed of leaves, input variables, and random constants.

Table 2 Oblique grammar used in the experiments

Rule	Production
Root	`if`
If	`if Condition then action else action`
Condition	$\sum_{i=0}^{n_{\text{inputs}}} \texttt{const} \cdot \texttt{input}_i < \texttt{const}$
Action	`leaf` or `if`
Const	$[-1, 1]$, with step of 0.001

2.2 Quality–diversity schemes

In this subsection, we briefly describe the QD schemes used in combination with the two EAs mentioned above.

2.2.1 MAP-Elites

ME (that stands for "Multi-dimensional Archive of Phenotypic Elites") [23] is a QD algorithm that tries to maximize the "illumination" of the search space (i.e., the balance between exploration and exploitation) by maintaining a multi-dimensional archive of the best solutions, which are indexed using the values of their "feature descriptors" (which are typically based on problem-dependent, user-defined properties of the solutions). To have meaningful illumination patterns, it is extremely important to have descriptors that are orthogonal to the solutions' fitnesses (i.e., the values of the objective function).

A descriptor can be defined as $\mathbf{d} \in \mathcal{D} = \{(d_0, \ldots, d_n) : d_i \in [\min_i, \max_i]; \forall i \in [0, n]\}$. A function $\mathcal{F} : S \rightarrow \mathcal{D}$ is defined to compute the descriptors associated with any solution $s \in S$.

The archive is structured as a multi-dimensional grid, where each dimension is divided into m equally-spaced bins.

When a new solution s is generated, its fitness and its descriptor $\mathcal{F}(s)$ are computed. Then, if the location of the archive is empty, the solution is inserted in the corresponding cell of the grid. Otherwise, the fitness of the solution present in the archive is checked. If it is worse than that of the current solution, the old solution is replaced by the new solution. Otherwise, if the fitness of the current solution is equal to or worse than that of the existing one, the current solution is discarded.

When using ME, we first initialize the map by randomly evaluating $init_{\text{pop}}$ random solutions. Then, we perform an iterative phase in which we sample solutions from the archive to generate new solutions through mutation and crossover (which depend on the use of GE or GP).

2.2.2 Covariance matrix adaptation MAP-Elites

CMA-ME [25] is a variant of ME that takes the benefit of the well-known CMA-ES algorithm [27]. The idea is to use the batch_n solutions sampled for the new batch (i.e., the new set of candidate solutions sampled from ME) as the initial population for batch_n parallel instances of CMA-ES. If CMA-ES is not able to improve the solution, at the next step of the algorithm, the new candidate solutions will be obtained through the mutation operator. This mechanism allows the algorithm to try escaping local optima.

Note that CMA-ES works on real-valued vectors, while ME works with GP or GE, as described before. To solve this discrepancy, we select from the phenotype (i.e., the DT) all the real values that are used in the DT, which are then optimized by CMA-ES.

2.2.3 Feature descriptor

As explained earlier, ME (and its derived algorithms) need a descriptor function $\mathcal{F}$ to map solutions to a location in the archive.

In the present study, we use a two-dimensional descriptor. The first dimension uses the entropy H of the actions taken by the agent: $H(s) = -\sum_{j=0}^{n_a} f(j) \cdot \log_{\mathrm{n_a}}(\mathrm{f(j)})$, where $f(j)$ is the frequency of the j-th action in the list of actions taken by the solution s. Entropy allows us to measure the diversity in terms of action distribution. Note that we use the number of actions n_a as the base for the logarithm, hence $H(s) \in [0, 1]$.

The second dimension of the descriptor, instead, measures a structural property of a solution: the depth of the DT. Note that, to make this descriptor more accurate, before computing the depth, we first execute a pruning on the DT, as described in [13].

It is important to note that the two features used for the descriptor are not to be considered as objectives, but as properties that are interesting for the study. For instance, one cannot say whether we want to maximize or minimize entropy, as the corresponding performance depends on the task at hand. On the other hand, one may need to have multiple solutions, each one with a different depth, to have the opportunity to choose the most appropriate model based on, e.g., hardware constraints.

2.3 Reinforcement learning

During the fitness evaluation phase, we perform RL on the leaves of the DTs, using ε-greedy Q-Learning [18], meaning that, with a probability of ε we take a random action, otherwise we choose the action with the best value.

More specifically, each leaf of the DT represents a "macro-state" σ, and the Q function is updated using the Bellman equation:

$$Q(\sigma, a) = (1 - \alpha)Q(\sigma, a) + \alpha(r + \gamma * \max_{\mathrm{a}}' \mathrm{Q}(\sigma', \mathrm{a}'))$$

where σ is a macro-state (i.e., a group of states that, navigating the DT, end in the same leaf), a is the action taken, α is a learning rate, r is the reward received by the environment, γ is the discount factor (that tunes the importance of future rewards w.r.t. the current ones), and σ' is the next macro-state, caused by the execution of action a in the current state.

The parameters used for the Q-Learning algorithm are shown in Table 3.

Table 3 Parameters used for Q-learning

Parameter	CartPole-v1	MountainCar-v0
ϵ	0.05	0.01
Initialization	$\sim\mathcal{U}(-1, 1)$	$\sim\mathcal{U}(-1, 1)$
Learning rate	0.001	0.001
Number of episodes	100	100

2.4 Fitness evaluation

To evaluate the quality of the evolved DTs, we use two well-known OpenAI Gym [26] environments: CartPole-v1 and MountainCar-v0.

We simulate m episodes for each solution, in order to: (1) allow the RL algorithm to converge to a well-performing policy, and (2) have a reliable estimate of the quality of the DT. A simulation ends when either the task is solved or a predefined time limit is hit. Once the m simulations have been completed, we compute the fitness of the DT by computing the mean of the scores across the m simulations.

Please note that, for the MountainCar-v0 environment, we normalize the values of each variable composing the observations since they have significantly different ranges of variation. To do so, we perform a min-max normalization (w.r.t. the ranges).

2.4.1 CartPole-v1

In the CartPole-v1 task,[4] the goal of the agent is to maintain a pole in equilibrium by moving the cart it lies on. The observations provided to the agent are: (1) x: the position of the cart, (2) v: the velocity of the cart, (3) θ: the angle of the pole, and (4): ω: the angular velocity of the pole. The agent can perform two actions: push the cart to the left, or push it to the right. The agent receives a reward of 1 for each timestep in which the pole is balanced and the cart is inside the bounds (i.e., $|\theta| < 12^\circ \wedge |x| < 2.4$), otherwise it receives a reward of 0. The simulation is ended whenever $|\theta| \geq 12^\circ$ or 500 timesteps have passed. This task is considered solved when the mean cumulative reward over 100 episodes is greater than or equal to 475.

[4] https://gym.openai.com/envs/CartPole-v1/.

2.4.2 MountainCar-v0

In the MountainCar-v0 task,[5] the goal of the agent is to drive a car on top of the right hill of a valley. To do so, the agent has to learn how to exploit the left hill to build momentum. The observation of the agent is composed of two variables: (1) x: the position of the car, and (2) v: the velocity of the car. The actions that the agent can perform are: (1) accelerate to the left, (2) do not accelerate, and (3) accelerate to the right. The agent receives a reward of -1 for each step, and it receives a reward of 0 when it reaches the top of the right hill. This reward function makes this problem hard to explore since the agent experiences a reward greater than -1 only when it completes the task. The simulation is terminated when the agent reaches the right hill or when the limit of 200 timesteps is reached. The task is considered solved when the mean cumulative reward, computed over 100 episodes, is greater than or equal to -110.

3 Results

In this section, we quantitatively compare the setups described in Sect. 2.

For each setup (i.e., each combination of an EA and a QD scheme), we performed 5 independent runs in order to statistically assess the significance of the results. This number was chosen to have sufficient statistical evidence about the comparison between the different setups while keeping the computational cost of the experiments limited.

The parameters used for all the methods are shown in Table 4. Note that the bounds for the behavioral feature (i.e., the entropy) are different across the two tasks. In fact, while in MountainCar-v0 we use the entire co-domain for the entropy, in CartPole-v1 we only consider the interval [0.8, 1]. This is because, in preliminary experiments, we observed that the region [0.0, 0.8] is scarcely populated with good solutions. We hypothesize that the reason underlying this phenomenon is that, being MountainCar-v0 a balancing task, the agent should frequently switch between the two actions, leading to high entropy.

Regarding the interpretability of the solutions, the authors in [28] proposed a quantitative metric to measure the interpretability of a mathematical formula. Here, we will use this metric to compare the interpretability of the solutions produced. Note, however, that instead of using the version of $\mathcal{M}$ proposed in [28], we will adopt the modified version from [13], as it is more general and can be used with any Machine Learning model. In fact, this version of $\mathcal{M}$ is essentially a proxy for the model complexity (meaning that the higher $\mathcal{M}$, the worse the interpretability), which is a general property of models. Moreover, it is worth mentioning that, for the schemes based on ME, we will compute the $\mathcal{M}$ value for the best-performing solution. In the case of ties, we choose the tree with minimum depth.

As for the "illumination" capability, we limit our analysis to a qualitative observation of how the two EAs fill the feature space.

[5] https://gym.openai.com/envs/MountainCar-v0/.

Table 4 Parameters used in the experiments

Parameter	CartPole-v1	MountainCar-v0
n_{pop}	200	200
init_{pop}	200	200
batch_n	20	20
$\text{total}_{\text{pop}}$	10,000	20,0000
T_s	2	2
p_{cx}	0.1	0.0
p_{mu}	1.0	1.0
G_s	100	100
M	40,000	40,000
Bins	10	10
Behavioral bounds	[0.8, 1.0]	[0.0, 1.0]
Structural bounds	[1, 10]	[1, 10]

3.1 CartPole-v1

As introduced before, we compare the results from both a performance and a diversity point of view.

For the GE setups, the top row of Fig. 1 shows the fitness trends of the best solutions found during the evolution. All the setups produce solutions capable of solving the task in less than 2000 fitness evaluations. Of note, GE+ME and GE+CMA-ME solve the task faster than GE, in terms of the number of fitness evaluations needed to converge.

A comparison of the results of our best DT (found across 5 runs) with the state-of-the-art is shown in Table 5. In the table, we can see that our method achieves the maximum score allowed by the environment, on par with most of the other methods (both interpretable and non-interpretable).

Concerning the illumination capability of the three setups, the bottom row of Fig. 1 shows the archives at the end of the evolution. Note that, in the case of GE, we consider all the solutions generated during the evolutionary process, rather than just the last generation, and fill the map a posteriori. In the case of GE+ME and GE+CMA-ME, instead, the map is filled during the evolutionary process, by construction. The results show that, while GE can find solutions that solve the task, its ability to illuminate the feature space is limited, as expected: in fact, the algorithm does not find a satisfactory number of diverse solutions. On

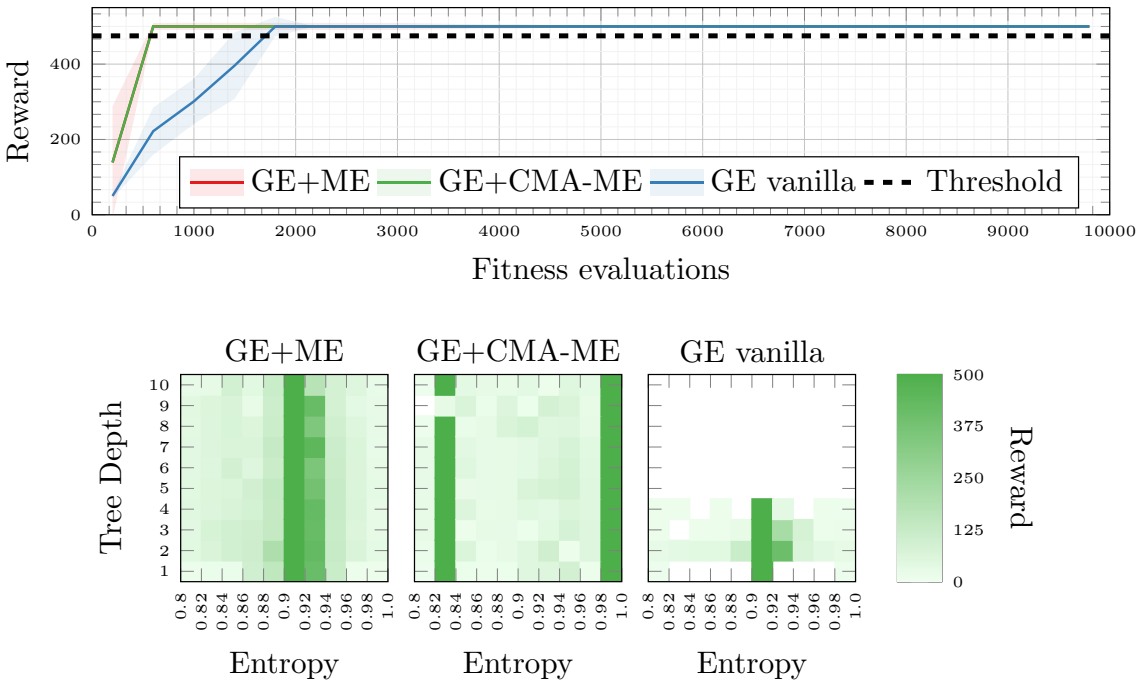


Fig. 1 Results on the CartPole-v1 task (GE setups). Top: fitness trends (mean ± std. dev. across 5 runs at each step of the algorithm). Bottom: maps obtained with the three setups. The results in each bin are averaged over 5 runs

Table 5 Comparison of our results with SOTA methods on CartPole-v1.

Source	Method	Score	$\mathcal{M}$
Meng et al. [29]	Policy discrepancy	**500.00**	1157.20
Meng et al. [29]	Policy discrepancy	**500.00**	1157.20
Meng et al. [29]	Policy discrepancy	**500.00**	1157.20
Silva et al. [30]	Differentiable DTs	388.76	89.20
Custode and Iacca [13]	Oblique DT	**500.00**	24.10
Ours	GE vanilla	**500.00**	24.1
Ours	GE+ME	**500.00**	24.1
Ours	GE+CMA-ME	**500.00**	24.1
Ours	GP vanilla	**500.00**	132.0
Ours	GP+ME	**500.00**	98.0
Ours	GP+CMA-ME	**500.00**	67.0

The boldface indicates the best results in terms of score

the other hand, GE+ME finds at least one solution for each possible DT depth and level of entropy.

Regarding the behavioral feature, while GE+ME still finds more different and high-performing solutions, GE vanilla and GE+ME seem to produce better results when the entropy values are in the range 0.9–0.92. This is probably due to the nature of the task, which requires high coordination between the two actions (*Push Left*/*Push Right*), leading to a similar frequency for the actions, and hence, high entropy. Interestingly, GE+CMA-ME found different solutions w.r.t. GE+ME, finding solutions either with a very high entropy level (between 0.98 and 1) or with relatively low entropy (in the interval 0.82–0.84). This suggests that this problem is multi-modal. Moreover, it appears that different QD schemes focus on different regions of the search space. Finally, it is worth noting that none of the considered QD schemes is able to illuminate well all the "promising" regions (i.e.,

$$H \in \{[0.82, 0.84), [0.9, 0.92), [0.9, 0.94), [0.98, 1.00]\}$$

.

The results obtained using GP are different. Regarding performance, see Fig. 2 (top row), we can observe that GP+ME converges slower than GP vanilla, while GP+CMA-ME converges faster than GP vanilla.

Regarding the illumination capabilities, Fig. 2 bottom row, we can observe that, while the general observation that GP vanilla explores the search space less effectively than GP+ME/CMA-ME is still valid, in this case, GP+ME and GP+CMA-ME are able to find solving solutions in most of the bins. Figure 3 shows some examples of DTs that solve the task, one for each GP setup.

3.2 MountainCar-v0

As for the MountainCar-v0 task, the top row of Fig. 4 shows the fitness trend for the GE setups. Similarly to the previous case, all the algorithms can solve the task.

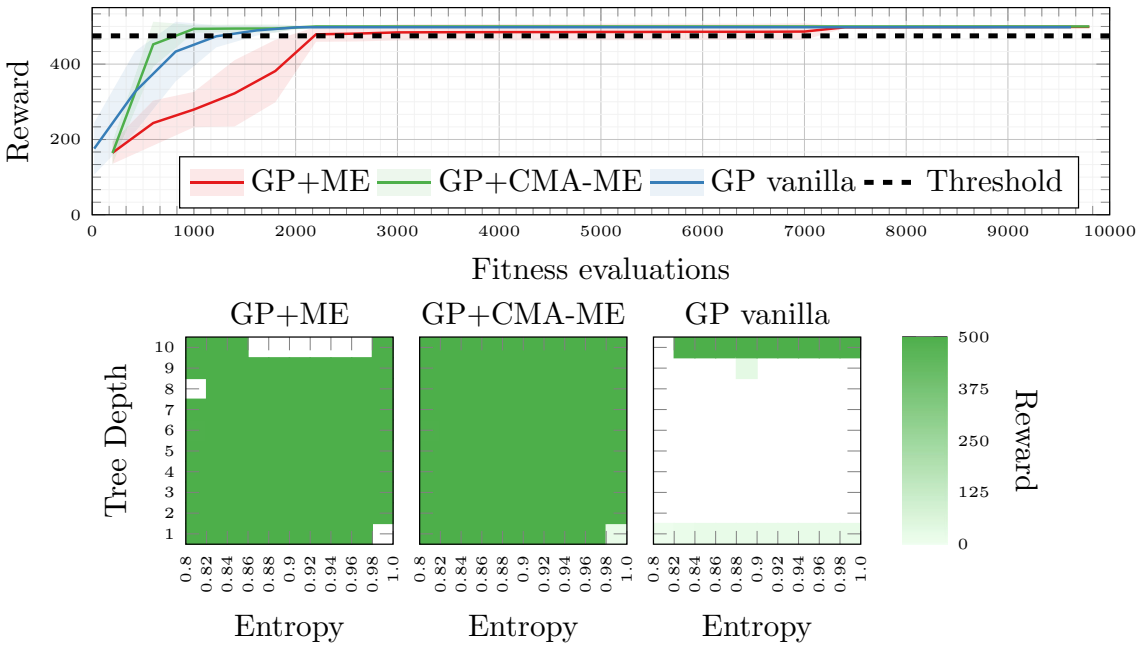
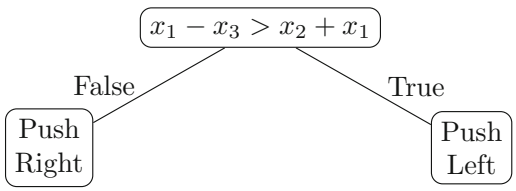
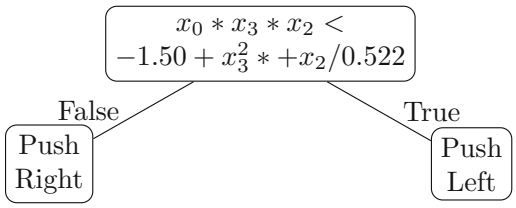


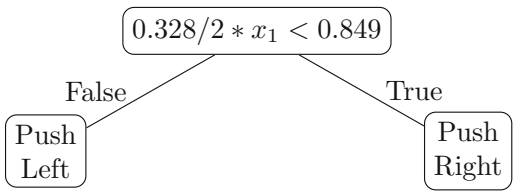
Fig. 2 Results on the CartPole-v1 task (GP setups). Top: fitness trends (mean ± std. dev. across 5 runs at each step of the algorithm). Bottom: maps obtained with the three setups. The results in each bin are averaged over 5 runs



(a) Example DT evolved with GP+ME.



(b) Example DT evolved with GP+CMA-ME.



(c) Example DT evolved with GP vanilla.

Fig. 3 Representation of three DTs that solve the CartPole-v1 task (after simplification). In this case, all setups (also those that are not shown in the figure) are able to find solutions that solve the task based on a single condition

However, GE vanilla and GE+ME solve the task in a comparable number of evaluations, with the former slightly faster (11×10^4 vs. 13×10^4 fitness evaluations). On the contrary, GE+CMA-ME is the fastest, finding a solution after only 10^4 fitness evaluations, probably due to the exploitation capabilities of the Covariance Matrix Adaptation (CMA) component. Note that, while GE+ME requires 110% of the fitness evaluations to reach the same performance of GE, GE+CMA-ME only requires 10%, which means that this scheme may significantly reduce the time needed to train DTs for IRL.

A comparison of the results of our best DTs (found across 5 runs) with the state-of-the-art is shown in Table 6. Here, we observe that both GP+ME and GP+CMA-ME achieve state-of-the-art performance. Moreover, a two-tailed Welch T-test (with confidence threshold $\alpha = 0.05$) between these two methods confirmed the statistical significance w.r.t. the previous state-of-the-art approach [13].

The bottom row of Fig. 4 shows the archive at the end of the evolution for the three GE setups. Similar to the CartPole-v1 case, GE+ME and GE+CMA-ME illuminate the feature space better than GE vanilla, covering 97% of bins in all 5 runs. On the other hand, GE vanilla concentrates on a small portion of the feature space. Overall, we can observe that the three setups find high-performing solutions in different areas of the feature space. Regarding the behavioral feature, while the DTs found by GE present a high entropy level (as in the CartPole-v1 task), GE+ME produces also DTs that have lower entropy. Hence, these DTs present behaviors in which at least one action is less frequent than the others. GE+CMA-ME pushes the search toward solutions with even lower entropy (smaller than

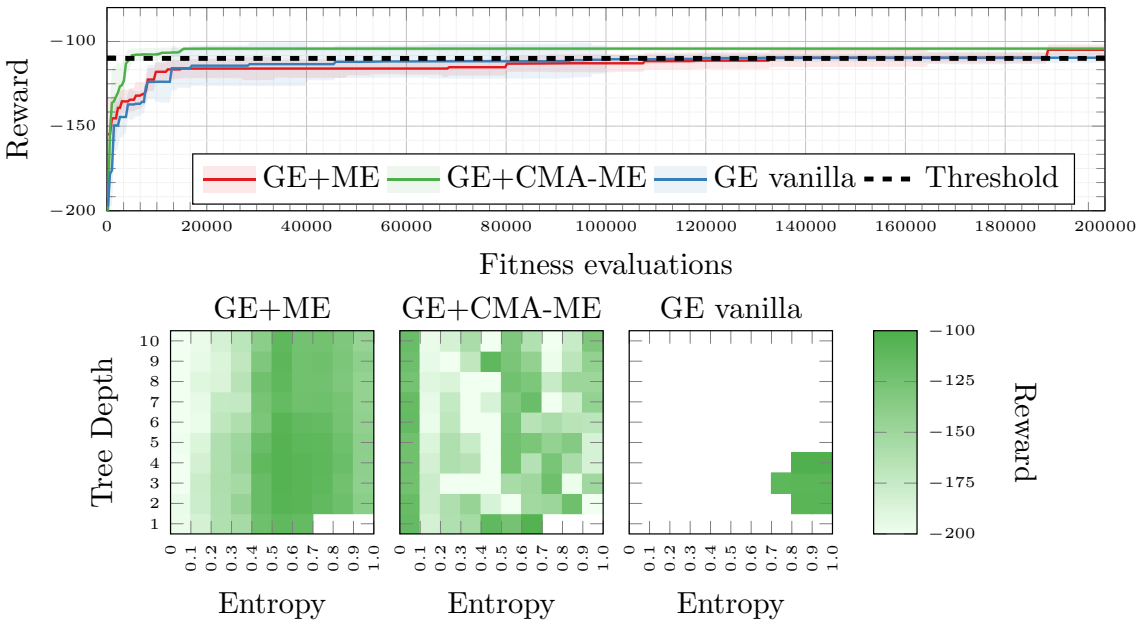


Fig. 4 Results on the MountainCar-v0 task (GE setups). Top: fitness trends (mean ± std. dev. across 5 runs at each step of the algorithm). Bottom: maps obtained with the three setups. The results in each bin are averaged over 5 runs

Table 6 Comparison of our results with SOTA methods on MountainCar-v0.

Source	Method	Score	$\mathcal{M}$
Zhiqing Xiao[a]	Closed-form policy	−102.61	54.70
Keavnn[b]	Soft Q networks	−104.58	31079.20
Harshit Singh[c]	Deep Q network	−108.85	984160.30
Colin M[d]	Double deep Q network	−107.83	46681.60
Amit[e]	Tabular SARSA	−105.99	381.50
Dhebar et al. [31]	NLDT (Open-loop)	−128.87	66.80
Custode and Iacca [13]	Orthogonal DT	−101.72	106.80
Ours	GE vanilla	−102.3	64.4
Ours	GE+ME	−103.3	13.6
Ours	GE+CMA-ME	−102.5	11.8
Ours	GP vanilla	−103.0	54.5
Ours	GP+ME	**−97.6**	59.0
Ours	GP+CMA-ME	**−98.0**	41.9

The boldface indicates the best results in terms of score

[a]https://github.com/ZhiqingXiao/OpenAIGymSolution

[b]https://github.com/StepNeverStop/RLs

[c]https://github.com/harshitandro/Deep-Q-Network

[d]https://github.com/CM-Data/Noisy-Dueling-Double-DQN-MountainCar

[e]https://github.com/amitkvikram/rl-agent

0.1), finding trees that use, for the majority of the time, just a single action (accelerate left). For the structural feature, we can observe that, as for the CartPole-v1 task, GE focuses only on small DTs (of depth 2 to 4), while GE+ME and GE+CMA-ME produce solutions that cover the entire range of tree depths [1, 10].

Of note, GE+ME and GE+CMA-ME produce also DTs with a depth equal to 1, meaning that the maximum number of leaves is 2. Hence, the entropy, in these cases, is bounded to be lower than 0.63, corresponding to the case in which the two actions have the same frequency (note that we calculate the entropy using as the base for the logarithm

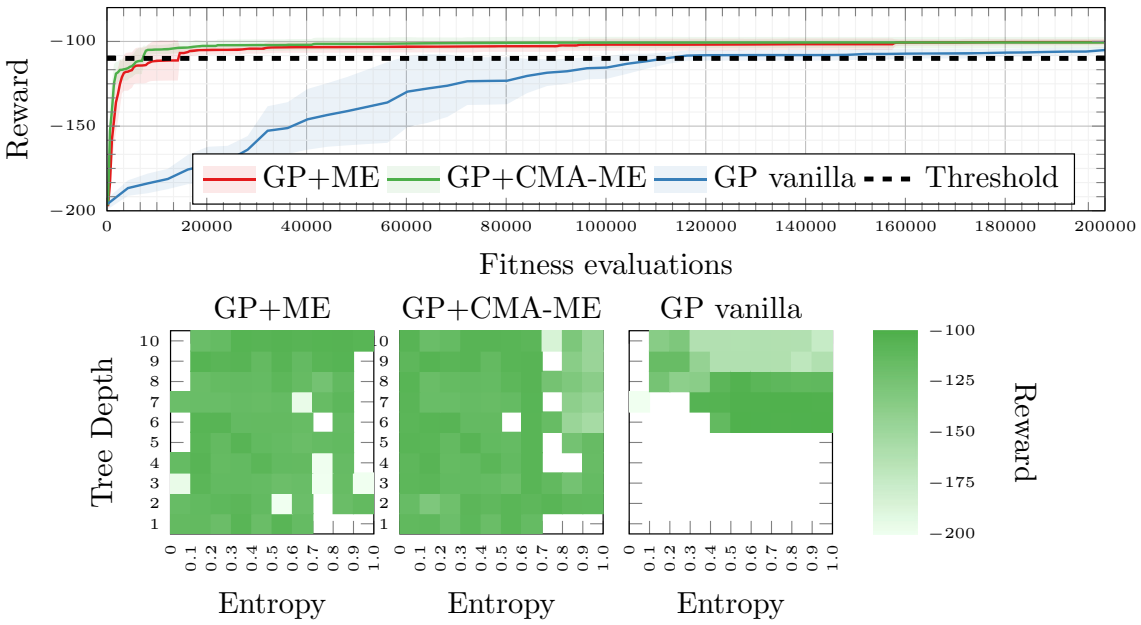


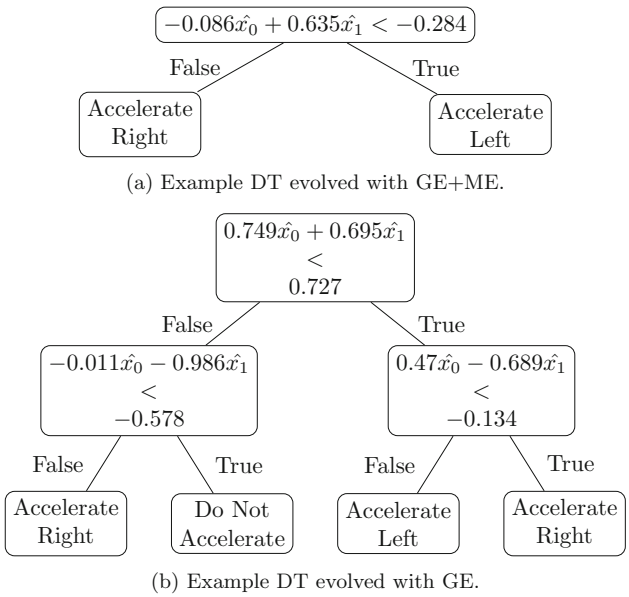
Fig. 5 Results on the MountainCar-v0 task (GP setups). Top: fitness trends (mean ± std. dev. across 5 runs at each step of the algorithm). Bottom: maps obtained with the three setups. The results in each bin are averaged over 5 runs

the number of actions, see Sect. 2.2.3). Figure 6 shows a representation of two example DTs.

With the GP setups, we can observe a similar behavior. However, in this case, both versions based on ME converge faster than GP vanilla, which slowly increases its performance, solving the task only after 12×10^4 fitness evaluations, see the first row of Fig. 5. Regarding the illumination capabilities, shown in the bottom row of Fig. 5, similar to the GE vanilla case, we have that GP+ME and GP+CMA-ME achieve a better exploration of the feature space, finding several high-performing solutions. However, GP vanilla shows better illumination performances than GE vanilla, probably due to the different mutation and crossover operators.

4 Conclusions and future works

In this paper, we have applied two QD schemes, namely, ME and CMA-ME, to a hybrid approach combining evolutionary optimization and RL for finding a diverse collection of interpretable models. In our experiments, we combined the two QD schemes with two EAs, namely, GE and GP. By testing all the combinations between the schemes and EAs on two tasks from OpenAI Gym, we draw insights into the capabilities of each setup, in terms of performance, efficiency, and exploration capabilities. Our experimental findings mainly suggest that different QD schemes achieve different illumination patterns, meaning that each algorithm explores the feature space differently.



(a) Example DT evolved with GE+ME.

(b) Example DT evolved with GE.

Fig. 6 Representation of two DTs that solve the MountainCar-v0 task (after simplification). GE finds solutions that use all the three actions (see Sect. 2.4.2). Hence, the depth of the DT is 2, while GE+ME finds also solutions that do not use the *Do Not Accelerate* action. Therefore, it is possible to produce a DT with a depth of 1

In summary, we observed that ME and CMA-ME find high-performing solutions while "illuminating" the feature space in a more efficient way w.r.t. the baseline approaches without QD. We also observed different behaviors, in terms of illumination capabilities, between GE and GP. This

suggests that the encoding and the mutation/crossover operators highly contribute to the illumination capabilities of the algorithm, as discussed in other works [32–34] outside the context of IRL.

In future works, we will extend this study to more recent variants of ME, such as those proposed in [25, 35], and to more challenging RL tasks, such as tasks with larger observation and action spaces, as well as tasks with delayed rewards and/or partial observations. Our intuition is that, by leveraging on better exploration, QD algorithms may be particularly effective in those scenarios. Moreover, we will investigate the scalability of ME schemes w.r.t. the number of features used in the descriptor. In fact, while in this work, we used only one behavioral and one structural feature, in some specific applications, one may need to define more than two features, e.g., to describe non-functional requirements of the solutions. Another interesting research direction would be to introduce interactions with the user during the search process, as done in [36]. In fact, incorporating the user feedback during the search may complement the natural tendency of QD algorithms to explore the feature space, while also allowing the optimization process to focus on the areas of the search space that are more interesting from the user perspective.

Acknowledgements Funded by the European Union (project no. 101071179). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or EISMEA. Neither the European Union nor the granting authority can be held responsible for them.

Funding Open access funding provided by Università degli Studi di Trento within the CRUI-CARE Agreement.

Data availability The data used in this study are available upon request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Gerlings J, Shollo A, Constantiou I (2020) Reviewing the need for explainable artificial intelligence (xAI). arXiv:2012.01007
2. Barredo Arrieta A, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, Garcia S, Gil-Lopez S, Molina D, Benjamins R, Chatila R, Herrera F (2020) Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. Inform Fusion 58:82–115
3. Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D (2018) A survey of methods for explaining black box models. ACM Comput Surv 51(5):1–42
4. Adadi A, Berrada M (2018) Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access 6:52138–52160
5. Bacardit J, Brownlee AEI, Cagnoni S, Iacca G, McCall J, Walker D (2022) The intersection of evolutionary computation and explainable AI. In: Genetic and evolutionary computation conference companion, pp 1757–1762. Association for Computing Machinery, New York, NY, USA
6. Marcus G (2018) Deep learning: A critical appraisal. arXiv:1801.00631
7. Langer M, Oster D, Speith T, Hermanns H, Kästner L, Schmidt E, Sesing A, Baum K (2021) What do we want from explainable artificial intelligence?-a stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research. Artif Intell 296:103473
8. Lipton ZC (2018) The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue 16(3):31–57
9. Rudin C, Chen C, Chen Z, Huang H, Semenova L, Zhong C (2021) Interpretable machine learning: fundamental principles and 10 grand challenges. arXiv:2103.11251
10. Zheng S, Trott A, Srinivasa S, Parkes DC, Socher R (2022) The AI economist: taxation policy design via two-level deep multiagent reinforcement learning. Sci Adv 8(18):2607
11. Degrave J, Felici F, Buchli J, Neunert M, Tracey B, Carpanese F, Ewalds T, Hafner R, Abdolmaleki A, Las Casas D (2022) Magnetic control of tokamak plasmas through deep reinforcement learning. Nature 602(7897):414–419
12. Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Mach Intell 1(5):206–215
13. Custode LL, Iacca G (2023) Evolutionary learning of interpretable decision trees. IEEE Access 11:6169–6184
14. Custode LL, Iacca G (2021) A co-evolutionary approach to interpretable reinforcement learning in environments with continuous action spaces. In: Symposium series on computational intelligence, pp 1–8. IEEE, New York, NY, USA
15. Custode LL, Iacca G (2022) Interpretable AI for Policy-making in pandemics. In: Genetic and evolutionary computation conference companion, pp 1763–1769. Association for Computing Machinery, New York, NY, USA
16. Ryan C, Collins J, Neill MO (1998) Grammatical evolution: evolving programs for an arbitrary language. In: European conference on genetic programming, pp 83–96. Springer, Berlin, Heidelberg
17. Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. Complex adaptive systems. MIT Press, Cambridge, MA, USA
18. Watkins CJCH (1989) Learning from delayed rewards. PhD thesis, King's College, Cambridge, UK
19. Hallawa A, Born T, Schmeink A, Dartmann G, Peine A, Martin L, Iacca G, Eiben AE, Ascheid G (2021) Evo-RL: Evolutionary-driven reinforcement learning. In: Genetic and evolutionary

computation conference—companion, pp 153–154. ACM, New York, NY, USA
20. Zahavy T, Schroecker Y, Behbahani F, Baumli K, Flennerhag S, Hou S, Singh S (2022) Discovering policies with DOMiNO: diversity optimization maintaining near optimality. arXiv:2205.13521
21. Cully A, Demiris Y (2017) Quality and diversity optimization: a unifying modular framework. IEEE Trans Evol Comput 22(2):245–259
22. Lehman J, Stanley KO (2011) Novelty search and the problem with objectives. In: genetic programming theory and practice, pp 37–56. Springer, New York, NY, USA
23. Mouret JB, Clune J (2015) Illuminating search spaces by mapping elites. arXiv:1504.04909
24. Ferigo A, Custode LL, Iacca G (2023) Quality diversity evolutionary learning of decision trees. In: 38th ACM/SIGAPP symposium on applied computing, pp 425–432
25. Fontaine MC, Togelius J, Nikolaidis S, Hoover AK (2020) Covariance matrix adaptation for the rapid illumination of behavior space. In: Genetic and evolutionary computation conference, pp 94–102. ACM, New York, NY, USA
26. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) OpenAI Gym. arXiv:1606.01540
27. Hansen N, Ostermeier A (1996) Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: International conference on evolutionary computation, pp 312–317. IEEE, New York, NY, USA
28. Virgolin M, De Lorenzo A, Medvet E, Randone F (2020) Learning a formula of interpretability to learn interpretable formulas. In: Parallel problem solving from nature, pp 79–93. Springer
29. Meng W, Zheng Q, Yang L, Li P, Pan G (2019) Qualitative measurements of policy discrepancy for return-based deep Q-network. IEEE Trans Neural Netw Learn Syst 31(10):4374–4380
30. Silva A, Gombolay M, Killian T, Jimenez I, Son SH (2020) Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In: International conference on artificial intelligence and statistics, pp 1855–1865. PMLR, Palermo, Italy
31. Dhebar Y, Deb K, Nageshrao S, Zhu L, Filev D (2020) Interpretable-AI policies using evolutionary nonlinear decision trees for discrete action systems. arXiv:2009.09521
32. Pigozzi F (2023) Camerota Verdù, F.J, Medvet E (2023) How the morphology encoding influences the learning ability in body-brain co-optimization. Genetic and evolutionary computation conference. GECCO '23. Association for Computing Machinery, New York, NY, USA, pp 1045–1054
33. Pigozzi F, Medvet E, Bartoli A, Rochelli M (2023) Factors impacting diversity and effectiveness of evolved modular robots. ACM Trans Evolut Learn 3(1):1–33
34. Ferigo A, Soros L, Medvet E, Iacca G (2022) On the entanglement between evolvability and fitness: An experimental study on voxel-based soft robots. In: ALIFE 2022: The 2022 Conference on artificial life. MIT press
35. Vassiliades V, Chatzilygeroudis K, Mouret J-B (2017) Using centroidal Voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. IEEE Trans Evol Comput 22(4):623–630
36. Urquhart N, Guckert M, Powers S (2019) Increasing trust in meta-heuristics by using MAP-Elites. In: Genetic and evolutionary computation conference companion, pp 1345–1348. ACM, New York, NY, USA