Quality Diversity Evolutionary Learning of Decision Trees

Andrea Ferigo
University of Trento
Trento, Italy
andrea.ferigo@unitn.it

Leonardo Lucio Custode
University of Trento
Trento, Italy
leonardo.custode@unitn.it

Giovanni Iacca
University of Trento
Trento, Italy
giovanni.iacca@unitn.it

ABSTRACT

Addressing the need for explainable Machine Learning has emerged as one of the most important research directions in modern Artificial Intelligence (AI). While the current dominant paradigm in the field is based on black-box models, typically in the form of (deep) neural networks, these models lack direct interpretability for human users, i.e., their outcomes (and, even more so, their inner working) are opaque and hard to understand. This is hindering the adoption of AI in safety-critical applications, where high interests are at stake. In these applications, explainable by design models, such as decision trees, may be more suitable, as they provide interpretability. Recent works have proposed the hybridization of decision trees and Reinforcement Learning, to combine the advantages of the two approaches. So far, however, these works have focused on the optimization of those hybrid models. Here, we apply MAP-Elites for diversifying hybrid models over a feature space that captures both the model complexity and its behavioral variability. We apply our method on two well-known control problems from the OpenAI Gym library, on which we discuss the "illumination" patterns projected by MAP-Elites, comparing its results against existing similar approaches.

CCS CONCEPTS

• **Theory of computation** → **Reinforcement learning**; • **Computing methodologies** → *Genetic programming*; *Hybrid symbolic-numeric methods*;

KEYWORDS

Quality diversity, Explainability, Decision trees, Reinforcement Learning, Grammatical Evolution

ACM Reference Format:
Andrea Ferigo, Leonardo Lucio Custode, and Giovanni Iacca. 2023. Quality Diversity Evolutionary Learning of Decision Trees. In *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-March 31, 2023, Tallinn, Estonia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3555776.3577591

SAC '23, March 27-March 31, 2023, Tallinn, Estonia

ACM ISBN 978-1-4503-9517-5/23/03…$15.00
https://doi.org/10.1145/3555776.3577591

1 INTRODUCTION

As Artificial Intelligence (AI) has become more pervasive in real-world applications, major concerns have arisen regarding the need for explanations of its outcomes and, possibly, its inner working [18]. This need is especially relevant in safety-critical applications, such as (but not limited to) healthcare, control systems, or financial regulatory systems, where the opaqueness of modern AI, mostly based on Deep Learning (DL), may pose serious issues. As such, the field of eXplainable Artificial Intelligence (XAI) has produced considerable research efforts in the past two decades [1–3, 19].

While there is currently a rather heated debate between those who believe that, also because of lack of explanations, DL is "hitting a wall" [27, 28], and those who instead rightly highlight the many successes of modern DL—especially in Computer Vision and Natural Language Processing—it is however quite clear that, to some extent, and especially in some domains, explanations are necessary and stakeholders do really need them to fully trust AI models [24].

As an alternative to the dominant paradigm of black-box DL-based models, some researchers have recently advocated the use of white-box (also called glass-box) models, such as, for instance, decision trees (DTs) and rule-based systems [34], noting that in some cases they can obtain similar or even better performance [33, 35]. Moreover, while often in black-box models only a posteriori explanations are possible, white-box models are "explainable by design", i.e., their transparent structure makes it possible to interpret (and, possibly, understand) their inner working and thus their outcomes.

Given the importance of interpretability, interpretable Reinforcement Learning (RL) has thus been identified as one of the current grand challenges in AI [34]. In fact, several modern applications of AI are modelled as RL problems. For example, deep RL has recently been used for the magnetic control of tokamak plasmas in a nuclear fusion plant [11], and for the definition of optimal taxation policies [51]. These two are, clearly, high-risk domains where the decisions made by the AI can have serious consequences on people's lives and, hence, interpretable models may be more appropriate. However, the complexity of some real-world problems may be too high to be captured by simple white-box models. For this reason, a promising direction in current AI attempts to break the dichotomy between black-box and glass-box models, by proposing hybrid models [30], e.g., based on neuro-symbolic AI [17, 37, 42].

While seminal works on hybrid AI date back to the late '90s-early 2000s, see, e.g., the works by Sun et al. [40, 41], or the studies on Learning Classifier Systems [22], there is nowadays a resurgence of interest in those models. In this sense, it is worth noting that many recent successes of DL, such as DeepMind's MuZero [38] and its predecessors, are actually based on hybrid models.

Previous research has mainly focused on the optimization of hybrid models, i.e., the goal of those studies was to find their optimal configuration to improve performance. Some instances of hybrid

models have been obtained by combining Q-learning [49] with DTs induced by Genetic Programming, as in [10], or by Grammatical Evolution (in the following "GE"), as in [7–9]. Another recent work [21] combined instead behavior trees with RL. However, when one wants to analyze this kind of models, other features—different from their performance—can be of interest. For instance, two important dimensions can be the *model complexity*, i.e., a (static) measure of the model's structure, and its *behavioral variability*, i.e., a (dynamic) measure of the model's capability of showing different behaviors during the execution of a given task. The first aspect can be relevant because, in general, simpler models can be easier to interpret [33, 35]. The second aspect can be relevant because a higher behavioral variability may indicate a better adaptation and a higher robustness of the model [20].

In this paper, we apply for the first time a quality diversity (QD) algorithm, namely the Multi-dimensional Archive of Phenotypic Elites (in the following, MAP-Elites, or just "ME") [31], to "disentangle" the relation between a hybrid model's performance, its complexity, and its behavioral variability. QD is an emergent trend in Evolutionary Computation that posits that some specific tasks, especially those that are characterized by deceptive objectives, can be solved more efficiently by algorithms that explicitly look for a diversification of the solutions found during the evolutionary process, rather than an explicit optimization of a given objective function [32]. Successful examples of QD algorithms are Novelty Search [25], and indeed ME. The latter, in particular, has been designed with the goal of "illuminating" (w.r.t. the objective quality) a given feature space defined by some specific features of interest: this is the use of the algorithm that we make here. Originally devised for robotic applications [6], ME has been successfully applied to various problems related to games [15, 23], logistics and scheduling [45, 46], neuroevolution [5], and constrained optimization [14]. Other works tried to use ME for interactive optimization [44], or to automatically derive rules to describe the relationships between features and objective quality [47]. In [13], ME has been used for the first time to analyze programs evolved by means of Genetic Programming w.r.t. two aspects of program architecture, namely the scope count (to measure program modularity) and the instruction entropy (to measure instruction diversity).

Here, we use ME to obtain a diverse collection of interpretable hybrid models composed of a DT combined with Q-learning on the leaves. These models are similar to the ones used in previous works [7–10] that, however, focus on the model optimization and analyze, a posteriori, the model complexity. In the present work, instead, we explicitly define as features for ME: 1) a measure of behavioral variability (i.e., the entropy of the actions taken by the model during the episode) and 2) a measure of model complexity (i.e., the depth of the DT). To the best of our knowledge, the only works that addressed the quest for diversity in RL tasks are the recent papers [50] and [43]. In [50], in particular, authors state that finding diverse solutions to a same RL problem can improve exploration, transfer, hierarchy, and robustness of agents. In that work, diversity is explicitly measured as the distance between the state occupancies of the policies in the obtained policy set, with agents controlled by actor-critic neural networks. Here, instead, we implicitly measure diversity in the aforementioned two-feature space, and we focus on DTs rather than neural networks. In [43], authors also identify the policy diversity as the key for robust RL agents. Similarly to our work, they also use ME; however, differently from us, they use gradient approximations and, most importantly, they project the policies onto a feature space made of domain-specific features (for a simulated locomotion task). In our case, instead, we consider domain-agnostic features, and as such our method can be of more general applicability. Despite these differences, our work shares the same motivation of these two previous works, i.e., we consider the search for diverse policies as a way to reach higher robustness. On top of that, we add however the important consideration about the interpretability of such models. In this regard, it is important to rule out a possible misconception: neither diversity nor interpretability are, *per se* objectives of the search (after all, we do not know *a priori* if these aspects are in conflict or not with the model performance). On the contrary, we consider them as *features*, hence the need for disentangling their relation with respect to the performance, and the use of ME.

We apply the proposed method on two well-known classic control problems from OpenAI Gym [4], namely Cart Pole and Mountain Car, and compare the results of ME and GE. We show that, by leveraging the exploration capability of ME, we are able to "illuminate" the relationship between model performance, complexity, and behavioral variability much more effectively than GE[1]. Moreover, our results are comparable with the state-of-the-art.

The rest of the paper is structured as follows. The next section describes the proposed method. The numerical results are presented in Section 3. Finally, Section 4 concludes this work.

2 METHOD

As introduced before, we aim to evolve DTs using a combination of an evolutionary algorithm (EA) and RL. While the EA evolves the structure of the DT, the RL algorithm optimizes the actions taken by the leaves, as in [7]. In the following, we describe the individual encoding, the EAs used to evolve the DTs, the RL technique that optimizes the action of the leaves, and how we evaluate the DTs, describing also the tasks performed.

2.1 Individual Encoding

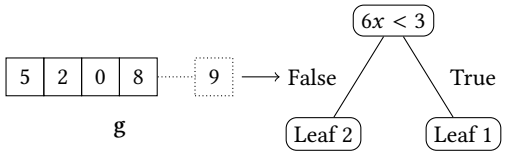
While the two EAs (namely, GE and ME, as described below) used in this study are different, in both cases we encode the genotype of an individual (i.e., a candidate solution representing a DT) as a vector $\mathbf{g} = (g_0, \ldots, g_{size})$ with $g_i \in [0, maxValue]$, where $maxValue$ is an integer value which must be greater than the number of possible choices for each production rule. We obtain the relative DT translating the genotype in the phenotype using an associate grammar [36]. This translation procedure operates as follows: given l as the number of possible choices for a given production rule in the grammar, the value $c = \mathbf{g}_i \mod l$ indicates that the c-th value will be taken as value. Note that we consider only oblique DTs, i.e., DTs in which each condition tests a linear combination of all the input variables. In Figure 1 we show an example of such mapping with a simplified grammar. The example of translation works as follows:

[1] In the remaining of the paper, we refer to the "illumination" concept originally introduced in [31], where ME is defined as an "illumination" algorithm "*because it illuminates the fitness potential of each area of the feature space, including tradeoffs between performance and the features of interest*". Here, the features of interest are the model complexity and its behavioral variability, while the fitness potential refers to the model performance (on the task at hand) in each area of the defined feature space.

the first rule *root* always produces an *if* node, which is composed of a condition and two actions. The condition rule requires 2 *const* nodes. Each *const* rule selects an integer value between 1 and 10, hence in this case $l = 10$. In the example shown in Figure 1 (a), the first two values of **g** are 5 and 2, which correspond, respectively, to the fifth and second element, i.e., 6 and 3. The next 2 values of **g**, used for the action rule, are 0 and 8. The action rule can produce a *leaf* or an *action*, hence in this case $l = 2$. Therefore, the calculation performed to select the production is $i = 0 \mod 2$ and $j = 8 \mod 2$: in both cases, the first element of the *action* production rule (a *leaf*) is produced. As both nodes are *leaf* nodes, no other nodes are produced and the rest of the genotype is not used. With this procedure, the genotype is translated into the DT shown in Figure 1 (b). Note that the final action performed by each leaf is not encoded in the genotype, but is optimized using RL during the task.

Rule	Production
Root	`if`
If	`if Condition then action else action`
Condition	$\sum_{i=0}^{n_inputs}$ `const` $\cdot$ `input`$_i$ < `const`
Action	`leaf or if`
const	$[1, 10]$, with step of 1

(a) Schema of the simplified grammar used in the example.



(b) How a given genotype g (left) is translated into a DT (right).

Figure 1: Illustration of the individual encoding. (a) A simplified grammar; (b) example of translation of a genotype g into a DT, using the grammar shown in (a), with $maxValue = 10$, and, for simplicity, a single input.

2.2 Evolutionary Algorithms

The first EA we consider is a simple form of GE [36]. The second is a QD algorithm, ME [31], that aims to find an archive of different solutions rather than producing a single optimal solution.

For the two EAs, we use the same mutation operator and the same computational budget, to make a fair comparison.

2.2.1 Grammatical Evolution. Following the basic form of GE [36], we initially create a population of n_{pop} randomly initialized solutions, then until we evaluate a total of $total_{pop}$ solutions, we iteratively create n_{pop} new solutions. These are created as follows:

(1) we select n_{pop} parents using tournament selection with size k;
(2) we group the solutions in pairs, and, with a probability of p_{cx}, we apply a crossover operator to each pair generating 2 offspring that substitute the parents in the selection process;
(3) each of the n_{pop} solutions is mutated with probability p_{mu}.

Then, the new solutions are evaluated, and the best n_{pop} solutions between the previous and the new ones are stored as the population for the next generation. After $total_{pop}$ solutions are evaluated, the algorithm returns the best solution in the final population. Algorithm 1 shows the structure of the algorithm.

Algorithm 1

```
1:  procedure GrammaticalEvolution
2:      n_pop ← randomInit()
3:      f_pop ← evaluate(n_pop)
4:      f_eval ← |n_pop|
5:      while f_eval < total_pop do
6:          new_pop ← crossover(n_pop, p_cx)
7:          new_pop ← mutation(new_pop)
8:          f_new ← evaluate(new_pop)
9:          f_eval ← f_eval + |new_pop|
10:         n_pop, f ← tournamentSelection(n_pop, f, new_pop, f_new, k)
        return best(n_pop)
```

2.2.2 MAP-Elites. Multi-dimensional Archive of Phenotypic Elites [31], commonly known as MAP-Elites, is a QD algorithm that maintains an archive of the best solutions that differ w.r.t. a given feature descriptor. The descriptor is necessary for ME to categorize each solution and, hence, store in the archive different solutions. Note that, while the descriptor needs to characterize a solution considering the problem being faced, it should be orthogonal to the solution's fitness (in fact, if the selected features are highly correlated to fitness, the illumination pattern would be of little interest).

A descriptor is generally defined as a vector $\mathbf{d} = (d_0, \ldots, d_n)$, with $d_i \in [min_i, max_i]$ and $\mathcal{D} : S \rightarrow \mathcal{R}^n$ being the function that, given a solution, returns its descriptor.

The archive is an n-dimensional grid, with each dimension divided in m bins. Thus, to find the coordinates $\mathbf{c} = (c_0, c_1, \ldots, c_n)$ of a solution s in the archive, we divide each dimension i of the descriptor in m equally wide bins, then we take the index of the bin in which the d_i values fall as the c_i coordinate.

During the evolution, to add a new solution s to the archive, we calculate the coordinates $\mathbf{c_s}$ and, if the position in the map is empty, we insert the solution into the archive. Otherwise, if that position already contains a solution, we keep the one with the best fitness.

As each dimension is divided into m bins, at the end of the evolution process the archive can store at most m^n solutions. We populate the map in two steps: the initialization and the iterative phases. In the initialization phase, we randomly create $init_{pop}$ solutions and try to insert them into the archive. During the iterative phase, we repeatedly generate $batch_n$ solutions and try to add them to the archive, until we generate a total of $total_{pop}$ solutions (also including $init_{pop}$ solutions). We create the new solutions as follows:

(1) we randomly select $batch_n$ solutions from the archive;
(2) we mutate the $batch_n$ solutions.

Algorithm 2 shows the structure of the algorithm.

2.2.3 Random Initialization. To initialize ME and GE, we populate the initial population with random genomes. More specifically, for each gene we uniformly sample a value in the range $[0, maxvalue]$, namely, $g_i \sim U(0, maxValue) \; \forall \; g_i \in \mathbf{g}$.

2.2.4 Mutation. We perform a uniform random mutation of each gene of an individual as follows: given the genotype ($\mathbf{g}$) of an individual, each gene has the same probability to be replaced with a new random gene with $g_{new} \in [0, maxValue]$.

2.2.5 Crossover. To perform crossover between a pair of parents $\mathbf{p}_1$ and $\mathbf{p}_2$, we randomly select a point $i \in [0, size]$ in the genotype, split the parent vectors into two parts around the selected point, and then mix the parts into two new genotypes.

2.2.6 Descriptor. As introduced before, ME stores a solution using a descriptor that indicates its position in the grid. Here we use two features to describe a DT. The first is a behavioral characterization of the DT, while the second represents the DT by its complexity.

To characterize the behavior of a solution, we use the entropy E of the actions taken by the agent, calculated as follows: be $n\_actions$ the number of possible actions that the individual can perform and be $\mathbf{actions} = (a_0, \ldots, a_{n\_actions})$ the vector of possible actions. During the fitness evaluation, we store in $actions_i$ how many times the i-th action is performed by the DT. Then, we calculate the vector of relative frequencies $\mathbf{f} = (f_0, \ldots, f_{n\_actions})$ such that $f_i = \frac{actions_i}{\sum_{j=0}^{n\_actions} actions_j}$, from which we calculate the entropy $E = -\sum_{i=0}^{n\_actions}(f_i \cdot log_{n\_actions} f_i)$. Note that, using as base for the logarithm the value $n\_actions$, E takes values in $[0, 1]$. In this way a policy that makes always the same action will have an entropy of 0, while a random policy, where $f_i = \frac{1}{n\_actions}$ for $i \in [0, n\_actions]$, will have an entropy of 1.

The second feature in the descriptor analyzes the structure of the DT, to characterize the solutions w.r.t. their complexity and, hence, their interpretability. To this aim, we use the depth of the DT, calculated after a simplification procedure carried out as in [7]. This procedure simply consists in removing all the nodes that are not visited during the fitness evaluation. In this way, we produce a smaller DT pruned of all the nodes (including leaves) that are not used. Then, we calculate the depth of the simplified version of the DT. Note that simplifying the DT does not influence the values of the behavioral feature, as the actions that are pruned do not contribute to the entropy calculation since their frequency is null.

As mentioned in the introduction, we should stress once again that the two features defined above are not, *per se*, objectives. As for entropy, it is not possible to state *a priori* if this quantity should be minimized or maximized. In fact, it may well be that in some specific tasks a higher behavioral variability is to be preferred, while in others a less variable behavior may be better. For this reason, we consider entropy as a feature rather than an explicit objective. Likewise, one may in principle aim to explicitly minimize the DT depth, to favor simpler models. However, it is difficult to state *a priori* if the model complexity and its performance are conflicting goals or not: once again, it could be that in some cases simpler models actually perform better. For these reasons, using a multi-objective approach on these two quantities may be misleading, at best, or not appropriate at all. On the contrary, modeling them as features, and analyze a posteriori, through the illumination capability of MAP-Elites, their correlation with performance, appears to be a more suitable approach.

Algorithm 2

procedure MAPElites
  $archive \leftarrow initArchive(n_{bins})$
  $init_{pop} \leftarrow randomInit()$
  $f_{pop} \leftarrow evaluate(init_{pop})$
  $descriptor_{pop} \leftarrow descriptor(init_{pop})$
  $archiveAdd(init_{pop}, f_{pop}, descriptor_{pop})$
  $f_{eval} \leftarrow |init_{pop}|$
  while $f_{eval} < total_{pop}$ **do**
    $new_{pop} \leftarrow archiveGetBatch(batch_n)$
    $new_{pop} \leftarrow mutation(new_{pop})$
    $f_{new} \leftarrow evaluate(new_{pop})$
    $descriptor_{new} \leftarrow descriptor(new_{pop})$
    $f_{eval} \leftarrow f_{eval} + |new_{pop}|$
    $archiveAdd(new_{pop}, f_{new}, descriptor_{new})$
  return *archive*

2.3 Reinforcement Learning

To optimize the action of the leaves, we use RL in the form of ϵ-greedy Q-Learning [49], with a fixed learning rate and a uniform random initialization.

2.4 Fitness evaluation

The evolved DTs are used to solve control tasks. We test two OpenAI Gym [4] environments, namely Cart Pole and Mountain Car.

For both tasks, the procedure used to evaluate the DT is the following: the genotype is translated into the corresponding DT, then it is evaluated on m independent episodes, where each episode uses a different seed for the random number generator.

Each episode is simulated until the task is solved or the time limit is reached. At each timestep, the reward from the environment is used to update the reinforcement model and it is accumulated for each episode. Moreover, in the Mountain Car environment, we normalize the observations in the range $[0, 1]$ using the following formula: $\hat{x_i} = \frac{x_i - min_i}{max_i - min_i}$. The bounds used for the normalization are $[-1.2, 0.6]$ and $[-0.07, 0.07]$. We have found indeed that normalization is needed to solve the Mountain Car task, while the Cart Pole task can be solved without.

When all the episodes have been simulated, the fitness of the individual is calculated as the average cumulative reward.

2.4.1 Cart Pole. In the Cart Pole task[2] the agent has to maintain in equilibrium a pole over a cart. At each timestep, the agent takes as input 4 pieces of information: the position of the cart x_c, the velocity of the cart v_c, the pole angle θ_p, and the pole angular velocity ω_p. The agent can take 2 actions: push the cart to the left or to the right. The reward is +1 for each timestep; each episode terminates after 500 timesteps, or if $|\theta_p| > 12^\circ$ or if $|x_c| > 2.4$. This task is solved if the cumulative reward for the agent has an average (on 100 episodes) greater than or equal to 475.

2.4.2 Mountain Car. In the Mountain Car task[3] the agent has to move a car up a hill building up momentum thanks to another hill positioned before the car. The information available to the agent

[2] https://gym.openai.com/envs/CartPole-v1/
[3] https://gym.openai.com/envs/MountainCar-v0/

at each timestep is: the position along the x-axis of the car (x_c), and its velocity (v_c). At each step, the agent has 3 possible actions: accelerate to the left, accelerate to the right, or do not accelerate. The task ends when the car reaches the top of the hill, or after 200 timesteps. Until the car does not reach the top of the hill, the reward is -1 for each timestep; the task is considered solved if the average reward on 100 episodes is greater than -110.

3 RESULTS

In this section we present the results obtained by GE and ME on the two different tasks. We are foremost interested in comparing two aspects of the two EAs: performance and "illumination" capability.

For both EAs, we performed 5 independent runs to statistically verify the results. In Table 1 and Table 2 we indicate the parameters used in the two environments with GE and ME respectively. Note that on the two tasks we use two different bounds for the entropy. In Mountain car, we set the bounds in the range $[0, 1]$, since three actions are possible. In Cart Pole, we instead set them in the range $[0.8, 1]$, as there are only two possible actions, and equilibrium between them is required to solve the task, i.e., solutions with lower entropy are quickly discarded. Table 3 describes the oblique grammar, which is common to all tasks and EAs. Finally, Table 4 shows the parameters used by Q-learning.

As regards the interpretability of the solutions, previous works [7, 8, 10] evaluate the complexity of the solutions based on the following factors: the number of symbols, the number of operations, the number of non-arithmetical operations, and how many times the non-arithmetical operations are consecutively composed. However, since in this work we use oblique DTs, the complexity of each node is the same (as they all evaluate a linear combination of inputs, see the Condition rule in Table 3). Hence, since total complexity of our evolved DTs depends only on their depth, we use the latter as measure of complexity.

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

Parameter	Cart Pole	Mountain Car
n_{pop}	200	200
$total_{pop}$	10000	200000
Tournament size	2	2
p_{cx}	0.1	0.1
p_{mu}	1.0	1.0
Genotype size	100	100
Genotype max value	40000	40000

Table 1: Parameters used for GE.

3.1 Cart Pole

As introduced before, we compare the results from both a performance and a diversity point of view. Figure 2 shows the trends of the best solutions found during the evolution. Both EAs produce solutions capable to solve the task in less than 2000 fitness evaluations. Of note, ME solves the task faster than GE, in terms of number of fitness evaluations.

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).

Parameter	Cart Pole	Mountain Car
Bins for dimension	10	10
Behavioral bounds	$[0.8, 1.0]$	$[0, 1]$
Structural bounds	$[1, 10]$	$[1, 10]$
$total_{pop}$	10000	200000
$batch_n$	20	20
$init_{pop}$	200	200
Tournament size	2	2
p_{cx}	0	0
p_{mu}	1.0	1.0
Genotype size	100	100
Genotype max value	40000	40000

Table 2: Parameters used for ME.

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

Table 3: Oblique grammar used in both EAs.

Parameter	Cart Pole	Mountain Car
ϵ	0.05	0.01
Initialization	Uniform $\in [-1, 1]$	Uniform $\in [-1, 1]$
Learning Rate	0.001	0.001
Number of episodes	100	100

Table 4: Parameters used for ϵ-greedy Q-learning.

Concerning the illumination capability of the two EAs, Figure 3 shows the archives at the end of the evolution for ME and GE. Note that, in the case of GE, we consider all the individuals generated during the evolutionary process, rather than just the last generation, and fill the map a posteriori. In the case of ME, instead, the map is filled during the evolutionary process, by construction of this algorithm. 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 allow to find a sufficient number of diverse solutions. On the other hand, ME finds at least one solution for each possible DT depth and level of entropy. Figure 4 shows two example DTs that solve the task.

Regarding the behavioral feature, while ME still finds more different and high-performing solutions, both EAs 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.

3.2 Mountain Car

As for the Mountain Car task, Figure 5 shows the fitness trend for the two EAs. As in the previous case, both algorithms can solve the task. However in this case GE is faster than ME at doing that: the former

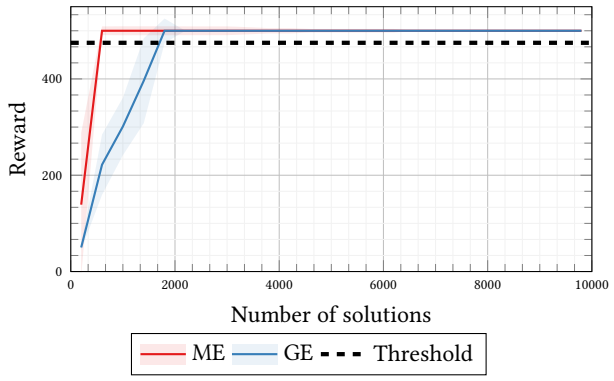


Figure 2: Fitness trends on the Cart Pole task with ME and GE. The dashed line indicates the "solved" threshold.

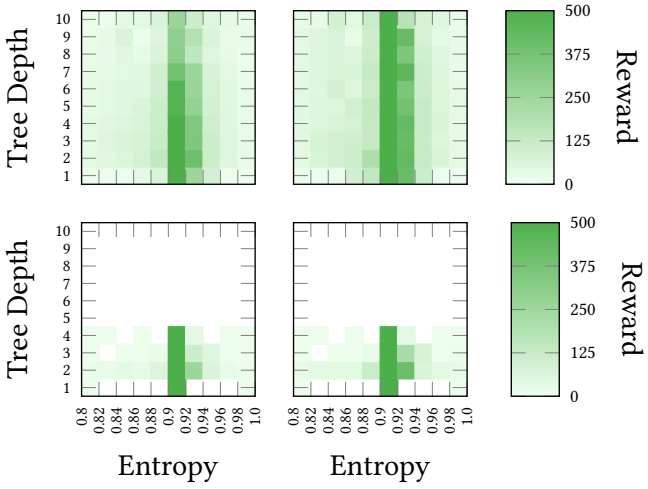
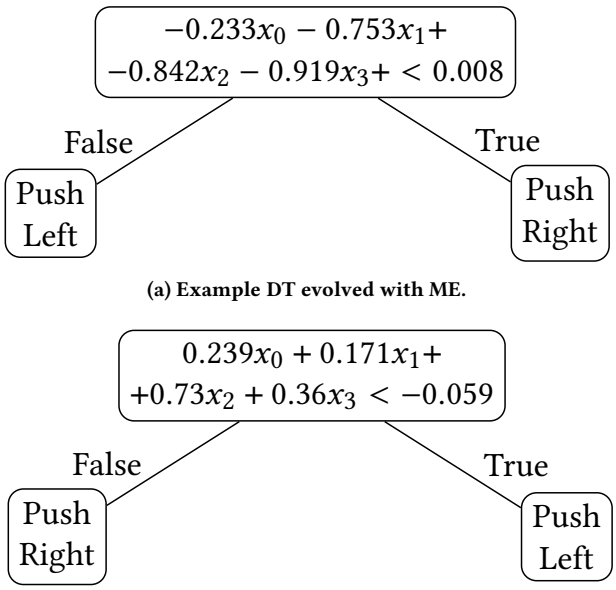


Figure 3: Maps obtained with ME (top row) and GE (bottom row) on the Cart Pole task. In the left column the results in each bin are averaged over 5 runs. Instead, in the right column each bin shows the maximum fitness over 5 runs.

needs around 110000 fitness evaluations; the latter, instead, finds the first solution that solves the task after around 130000 fitness evaluations. In other words, while eventually reaching slightly better performance, ME requires 10% of the total fitness evaluations budget more than GE to solve the task.

A comparison of the results of our best DT (found across 5 runs) with the state-of-the-art is shown in Table 6. While our method does not achieve the best score in this task, it is important to note that it is ranked second. However, since these results regard the best individual on multiple runs, there is no guarantee that these small differences are statistically significant.

Figure 6 shows the archive at the end of the evolution for the two EAs. Similar to the Cart Pole case, ME illuminates the feature space better than GE, covering 97% of bins in all 5 runs. On the other hand, GE concentrates on a small portion of the feature space. Overall, we can observe that the two EAs find solutions that solve



(a) Example DT evolved with ME.

(b) Example DT evolved with GE.

Figure 4: Representation of two DTs that solve the Cart Pole task (after simplification). Both EAs are able to find solutions that solve the task based on a single condition.

the problem in different areas of the feature space. Regarding the behavioral feature, while GE DTs present a high entropy level as in the Cart Pole task, 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. For the structural feature, we can observe that, as for the Cart Pole task, GE focuses only on small DTs (of depth 2 to 4), while ME produces solutions that cover the entire range of depths $[1, 10]$.

Of note, ME produces also DTs with a depth equal to 1, meaning that the maximum number of leaves is 2. Hence, the entropy in this case is limited to a maximum of circa 0.63, corresponding to the case in which the two actions are executed an equal number of times (we remember that we calculate the entropy using as the base for the logarithm the number of actions, see Section 2.2.6). Figure 7 shows a representation of two example DTs.

Table 5: Comparison of our results with state-of-the-art approaches on the Cart Pole task.

Source	Method	Score
Meng et al. [29]	Policy discrepancy	**500.00**
Meng et al. [29]	Policy discrepancy	**500.00**
Meng et al. [29]	Policy discrepancy	**500.00**
Silva et al. [39]	Differentiable DTs	388.76
Custode and Iacca [7]	Orthogonal DT	**500.00**
Ours	Oblique DT	**500.00**

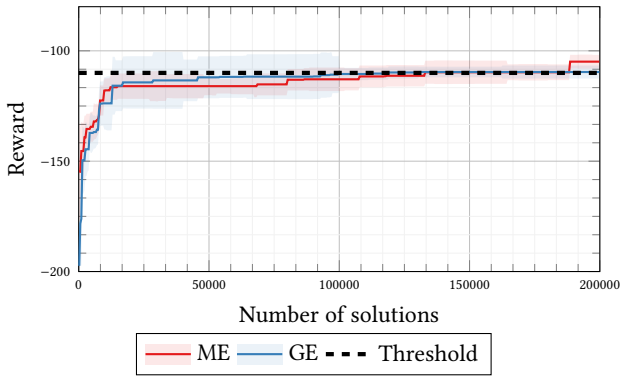


Figure 5: Fitness trends on the Mountain Car task with ME and GE. The dashed line indicates the "solved" threshold.

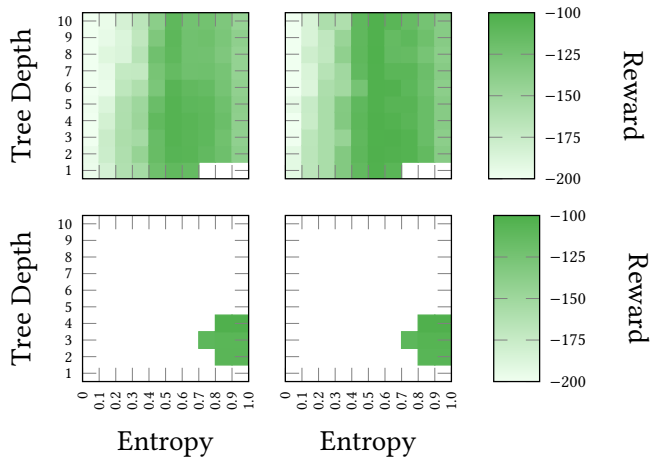
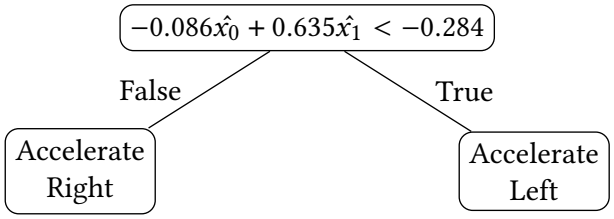


Figure 6: Maps obtained with ME (top row) and GE (bottom row) on the Mountain Car task. In the left column the results in each bin are averaged over 5 runs. Instead, in the right column each bin shows the maximum fitness over 5 runs.

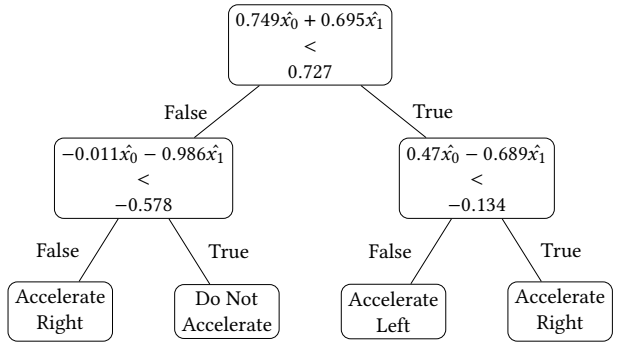
4 CONCLUSION

In this paper, we have applied a QD algorithm, namely ME, for finding a diverse collection of interpretable hybrid models composed of a DT combined with Q-learning on the leaves. We have tested the method on two tasks from OpenAI Gym library, namely Cart Pole and Mountain Car, and compared the results of ME with those obtained by GE. We have then discussed the results of the two EAs in terms of performance and "illumination" capability, given a feature space defined by model complexity and behavioral variability.

Summarizing, we observed that, in both tasks, ME finds solutions that solve the task, "illuminating" at the same time the feature space in a more efficient way w.r.t. GE. Moreover, while both EAs produced models with low complexity, hence good interpretability, in the Mountain Car task ME found that one action is not necessary to solve the task.



(a) Example DT evolved with ME.



(b) Example DT evolved with GE.

Figure 7: Representation of two DTs that solve the Mountain Car task (after simplification). GE finds solutions that use all the three actions (see Section 2.4.2). Hence, the depth of the DT is 2, while 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 future works, we will extend this study to more recent variants of ME, such as those proposed in [16, 48], and to more challenging RL tasks. Moreover, we will investigate the scalability of ME w.r.t. the number of features used in the descriptor. Another interesting direction would be to introduce interactions with the user during the search process, as done in [44].

Table 6: Comparison of our results with state-of-the-art approaches on the Mountain Car task.

Source	Method	Score
Zhiqing Xiao[4]	Closed-form policy	-102.61
Keavnn[5]	Soft Q Networks [26]	-104.58
Harshit Singh[6]	Deep Q Network	-108.85
Colin M[7]	Double Deep Q Network	-107.83
Amit[8]	Tabular SARSA	-105.99
Dhebar et al. [12]	NLDT (Open-loop)	-128.87
Custode & Iacca [7]	Orthogonal DT	**-101.72**
Ours	Oblique DT	-102.6

[4]https://github.com/ZhiqingXiao/OpenAIGymSolution
[5]https://github.com/StepNeverStop/RLs
[6]https://github.com/harshitandro/Deep-Q-Network
[7]https://github.com/CM-Data/Noisy-Dueling-Double-DQN-MountainCar
[8]https://github.com/amitkvikram/rl-agent

REFERENCES

[1] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160.

[2] Jaume Bacardit, Alexander E. I. Brownlee, Stefano Cagnoni, Giovanni Iacca, John McCall, and David Walker. 2022. The Intersection of Evolutionary Computation and Explainable AI. In *Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, New York, NY, USA, 1757–1762.

[3] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:1606.01540.

[5] Cédric Colas, Vashisht Madhavan, Joost Huizinga, and Jeff Clune. 2020. Scaling MAP-Elites to deep neuroevolution. In *Genetic and Evolutionary Computation Conference*. ACM, New York, NY, USA, 67–75.

[6] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* 521, 7553 (2015), 503–507.

[7] Leonardo Lucio Custode and Giovanni Iacca. 2020. Evolutionary learning of interpretable decision trees. arXiv:2012.07723.

[8] Leonardo Lucio Custode and Giovanni Iacca. 2021. A co-evolutionary approach to interpretable reinforcement learning in environments with continuous action spaces. In *Symposium Series on Computational Intelligence*. IEEE, New York, NY, USA, 1–8.

[9] Leonardo Lucio Custode and Giovanni Iacca. 2022. Interpretable AI for Policy-Making in Pandemics. In *Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, New York, NY, USA, 1763–1769.

[10] Leonardo Lucio Custode and Giovanni Iacca. 2022. Interpretable Pipelines with Evolutionary Optimized Modules for Reinforcement Learning Tasks with Visual Inputs. In *Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, New York, NY, USA, 224–227.

[11] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. 2022. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* 602, 7897 (2022), 414–419.

[12] Yashesh Dhebar, Kalyanmoy Deb, Subramanya Nageshrao, Ling Zhu, and Dimitar Filev. 2020. Interpretable-AI Policies using Evolutionary Nonlinear Decision Trees for Discrete Action Systems. arXiv:2009.09521.

[13] Emily Dolson, Alexander Lalejini, and Charles Ofria. 2019. Exploring genetic programming systems with MAP-Elites. In *Genetic Programming Theory and Practice*. Springer, Cham, 1–16.

[14] Stefano Fioravanzo and Giovanni Iacca. 2021. MAP-Elites for Constrained Optimization. In *Constraint Handling in Metaheuristics and Applications*. Springer, Singapore, 151–173.

[15] Matthew C Fontaine, Scott Lee, Lisa B Soros, Fernando de Mesentier Silva, Julian Togelius, and Amy K Hoover. 2019. Mapping hearthstone deck spaces through MAP-Elites with sliding boundaries. In *Genetic and Evolutionary Computation Conference*. ACM, New York, NY, USA, 161–169.

[16] Matthew C Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. 2020. Covariance matrix adaptation for the rapid illumination of behavior space. In *Genetic and Evolutionary Computation Conference*. ACM, New York, NY, USA, 94–102.

[17] Artur d'Avila Garcez and Luis C Lamb. 2020. Neurosymbolic AI: the 3rd Wave. arXiv:2012.05876.

[18] Julie Gerlings, Arisa Shollo, and Ioanna Constantiou. 2020. Reviewing the need for explainable artificial intelligence (xAI). arXiv:2012.01007.

[19] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM computing surveys* 51, 5 (2018), 1–42.

[20] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, Stockholm, Sweden, 1861–1870.

[21] Ahmed Hallawa, Thorsten Born, Anke Schmeink, Guido Dartmann, Arne Peine, Lukas Martin, Giovanni Iacca, A. E. Eiben, and Gerd Ascheid. 2021. Evo-RL: Evolutionary-Driven Reinforcement Learning. In *Genetic and Evolutionary Computation Conference - Companion*. ACM, New York, NY, USA, 153–154.

[22] John H Holland, Lashon B Booker, Marco Colombetti, Marco Dorigo, David E Goldberg, Stephanie Forrest, Rick L Riolo, Robert E Smith, Pier Luca Lanzi, Wolfgang Stolzmann, et al. 1999. What is a learning classifier system?. In *International Workshop on Learning Classifier Systems*. Springer, Cham, 3–32.

[23] Ahmed Khalifa, Scott Lee, Andy Nealen, and Julian Togelius. 2018. Talakat: Bullet hell generation through constrained MAP-Elites. In *Genetic and Evolutionary Computation Conference*. ACM, New York, NY, USA, 1047–1054.

[24] Markus Langer, Daniel Oster, Timo Speith, Holger Hermanns, Lena Kästner, Eva Schmidt, Andreas Sesing, and Kevin Baum. 2021. What do we want from explainable artificial intelligence?–A stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research. *Artificial Intelligence* 296 (2021), 103473.

[25] Joel Lehman and Kenneth O Stanley. 2011. Novelty search and the problem with objectives. In *Genetic Programming Theory and Practice*. Springer, New York, NY, USA, 37–56.

[26] Jingbin Liu, Xinyang Gu, Shuai Liu, and Dexiang Zhang. 2019. Soft Q-network. arXiv:1912.10891.

[27] Gary Marcus. 2018. Deep learning: A critical appraisal. arXiv:1801.00631.

[28] Gary Marcus. 2022. Deep learning is hitting a wall. , 03–11 pages. Nautilus.

[29] Wenjia Meng, Qian Zheng, Long Yang, Pengfei Li, and Gang Pan. 2019. Qualitative measurements of policy discrepancy for return-based deep Q-network. *IEEE transactions on neural networks and learning systems* 31, 10 (2019), 4374–4380.

[30] André Meyer-Vitali, Roos Bakker, Michael van Bekkum, M de Boer, G Burghouts, J van Diggelen, J Dijk, C Grappiolo, J de Greeff, A Huizing, et al. 2019. Hybrid AI: white paper. TNO Reports.

[31] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. arXiv:1504.04909.

[32] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI* 3 (2016), 40.

[33] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (2019), 206–215.

[34] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2021. Interpretable Machine Learning: Fundamental Principles and 10 Grand Challenges. arXiv:2103.11251.

[35] Cynthia Rudin and Joanna Radin. 2019. Why Are We Using Black Box Models in AI When We Don't Need To? A Lesson From An Explainable AI Competition. Harvard Data Science Review.

[36] Conor Ryan, JJ Collins, and Michael O Neill. 1998. Grammatical evolution: Evolving programs for an arbitrary language. In *European Conference on Genetic Programming*. Springer, Berlin, Heidelberg, 83–96.

[37] Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. 2021. Neuro-Symbolic Artificial Intelligence Current Trends. arXiv:2105.05330.

[38] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609.

[39] Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. 2020. Optimization Methods for Interpretable Differentiable Decision Trees Applied to Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, Palermo, Italy, 1855–1865.

[40] Ron Sun. 1997. Learning, action and consciousness: A hybrid approach toward modelling consciousness. *Neural Networks* 10, 7 (1997), 1317–1331.

[41] Ron Sun. 2006. Connectionist Implementationalism and Hybrid Systems. Encyclopedia of Cognitive Science.

[42] Zachary Susskind, Bryce Arden, Lizy K John, Patrick Stockton, and Eugene B John. 2021. Neuro-Symbolic AI: An Emerging Class of AI Workloads and their Characterization. arXiv:2109.06133.

[43] Bryon Tjanaka, Matthew C Fontaine, Julian Togelius, and Stefanos Nikolaidis. 2022. Approximating gradients for differentiable quality diversity in reinforcement learning. arXiv:2202.03666.

[44] Neil Urquhart, Michael Guckert, and Simon Powers. 2019. Increasing trust in meta-heuristics by using MAP-elites. In *Genetic and Evolutionary Computation Conference Companion*. ACM, New York, NY, USA, 1345–1348.

[45] Neil Urquhart and Emma Hart. 2018. Optimisation and illumination of a real-world workforce scheduling and routing application (WSRP) via MAP-Elites. In *Parallel Problem Solving from Nature*. Springer, Cham, 488–499.

[46] Neil Urquhart, Silke Höhl, and Emma Hart. 2019. An illumination algorithm approach to solving the micro-depot routing problem. In *Genetic and Evolutionary Computation Conference*. ACM, New York, NY, USA, 1347–1355.

[47] Neil Urquhart, Silke Höhl, and Emma Hart. 2021. Automated, Explainable Rule Extraction from MAP-Elites Archives. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, Cham, 258–272.

[48] Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2017. Using centroidal Voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation* 22, 4 (2017), 623–630.

[49] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3 (1992), 279–292.

[50] Tom Zahavy, Yannick Schroecker, Feryal Behbahani, Kate Baumli, Sebastian Flennerhag, Shaobo Hou, and Satinder Singh. 2022. Discovering Policies with DOMiNO: Diversity Optimization Maintaining Near Optimality. arXiv:2205.13521.

[51] Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C. Parkes, and Richard Socher. 2022. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science Advances* 8, 18 (2022), eabk2607.