Regret and Restore – Enhancing Safety-critical Deep Reinforcement Learning Through Evaluation Stages[⋆]

Timo P. Gros, Nicola J. Müller, and Verena Wolf

Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
{timopgros,nmueller,wolf}@cs.uni-saarland.de

Abstract. Deep reinforcement learning (DRL) has succeeded tremendously in many complex decision-making tasks. However, reward structures of safety-critical applications are not well suited for standard DRL training because they are typically very sparse and undiscounted. Various exploration strategies have been proposed to address the problem of sparse rewards. However, they do not take information about the current safety performance into account; thus, they fail to systematically focus on the parts of the state space most relevant for training. Here, we propose ***r****egret* ***a****nd state* ***r****estoration in* ***e****valuation-based deep reinforcement learning* (RARE), an algorithm that introduces two innovations: (i) it combines deep statistical model checking evaluation stages with state restorations, i.e., restarting episodes in formerly visited states, and (ii) it exploits estimations of the regret, i.e., the gap between the policies' current and optimal performance. Besides others, we showcase that both innovations are beneficial. RARE outperforms baselines such as deep Q-learning and Go-Explore in an empirical evaluation.

1 Introduction

Over the past decade, deep reinforcement learning (DRL) has made remarkable advancements in various complex decision-making tasks. Notably, it has demonstrated exceptional success in domains such as board games, including chess and go [40,41,42]. Additionally, its practical applications have extended to domains such as vehicle routing [33], robotics [21], and autonomous driving [37].

Despite these successes, DRL still suffers from significant weaknesses, especially in safety-critical applications. Safety objectives typically yield degenerated reward structures. For instance, maximizing the probability of reaching a goal state without getting trapped in an unsafe (absorbing) state yields an extremely sparse reward: 1 in the goal state and 0 elsewhere. DRL is known to perform poorly with such sparse reward structures [2,22,30,36,39]. Hence, for (D)RL training to be able to identify a useful policy, proxy objectives are used, such as discounted cumulative rewards giving positive feedback for goal states and (highly) negative feedback for unsafe states [15].

There are sophisticated exploration strategies to handle sparse reward problems [4,8,11,12]. However, they perform poorly in safety-critical applications. While

⋆ This work was partially supported by the German Research Foundation (DFG) under grant No. 389792660, as part of TRR 248, see https://perspicuous-computing.science.

these strategies are good at exploring the state space, their training process is conducted without systematically including the current (safety) performance, thereby neglecting to focus on areas where training is more important.

We introduce a DRL algorithm capable of handling degenerated reward structures and addressing safety-critical applications. We propose two innovations: (i) we combine deep statistical model-checking (DSMC) evaluation stages [19,15] with state restorations, where we start new training episodes in carefully selected states based on the previously evaluated performance, and (ii) we exploit estimations of the regret [27], i.e., the gap between the current and optimal performance, to focus the training on high-regret states.

Our state restoration procedure (innovation (i)) draws inspiration from Go-Explore [11]. During training, we store states deemed interesting for learning and sample initial states for subsequent training episodes from them. We leverage recent work on evaluation stages to identify states that provide us with particularly valuable information for training. These evaluation stages use deep statistical model checking [18] to obtain information about the safety properties of the current policy.

Based on the evaluated performance, we estimate the regret (innovation (ii)). We propose two different techniques to focus the training on archived states where the regret is high, i.e., where the policy is far from optimal: starting more training episodes in such states or giving them a higher priority within the replay buffer that we use for batch updates of the agent's neural network. Using the results from the evaluation stages to obtain the regret approximation allows us to inform the algorithm also about the original safety objective, even when a proxy objective needs to be used for learning.

We consider an illustrative example. Figure 1 depicts a map of the Racetrack, one of our benchmarks, where the goal is marked in green. The agent starts in one of the two purple cells in the lower right corner and must navigate through the grid up to the goal cells. Due to uncertainty in the environment, the narrow connections between the bottom cells (visited with a low probability) and the rest of the map lead to a reduced maximum achievable goal reachability probability from these bottom cells.

Now consider Figure 2, which depicts how often states from each grid cell have been used to update the policy. Standard DRL algorithms only start training from the initial states and fail to explore the state space sufficiently.

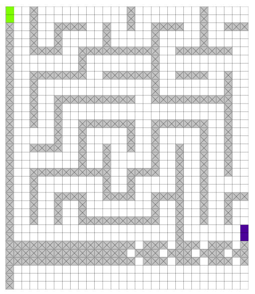
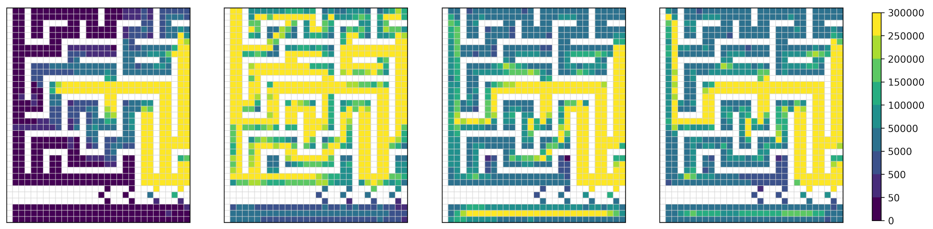


Fig. 1: Racetrack's extended Maze map.

Figure 2a shows that the states near the goal line have never been used to update the policy. Go-Explore, a DRL algorithm utilizing state restorations, thoroughly explores the map. However, it is unable to focus the learning on the most relevant parts, and further, it is unable to find a solution from the cells at the bottom part of the map, as it only rarely starts from there. In contrast, Figure 2c was created by using our proposed technique of state restoration combined with evaluation stages (innovation (i)). The agent reaches the goal and is able to solve the task, also from the more difficult part at the bottom. Hence, the corresponding policy's average return and maximum goal reaching probability are higher. Still, it visits the cells at the bottom of



(a) Plain Reinforcement Learning. (b) Go-Explore. (c) Using state restoration. (d) Using regret and state restoration.

Fig. 2: Heatmaps visualizing how often states from each grid cell have been used to update the policy during training.

the map unnecessarily often. Lastly, consider Figure 2d, which also includes regret estimation, i.e., both innovations (i) and (ii). As soon as the agent finds a solution for the bottom part, it shifts its focus (recognizable on the color scale, especially compared to 2c). Thus, it is able to train more in regions where the performance can still be improved, resulting in an even better overall performance.
To summarize, the contribution of our paper is as follows:
(i) We present our algorithm ***r**egret **a**nd state **r**estoration in **e**valuation-based deep reinforcement learning* (RARE) in Section 3.
(ii) We conduct an ablation study (Section 4) to demonstrate the effect of both state restoration and regret separately.
(iii) Section 5 provides an empirical evaluation of RARE, comparing it to the baselines deep Q-learning, deep Q-learning with prioritized replay, and Go-Explore on two benchmarks.

2 Background

Prior to presenting our contributions, this section covers foundational concepts.

2.1 Markov Decision Processes and Deep Q-Learning

We consider discrete Markov decision processes (MDPs) with finite sets of states $\mathcal{S}$, actions $\mathcal{A}$, and an initial distribution μ over the set of initial states $\mathcal{I} \subseteq \mathcal{S}$. For states $s_t, s_{t+1} \in \mathcal{S}$, a transition from state s_t to s_{t+1} when choosing action $a_t \in \mathcal{A}$ corresponds to an experience $(s_t, a_t, r_{t+1}, s_{t+1})$, where $r_{t+1} \in \mathbb{R}$ is the obtained reward. Our goal is to compute a deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the sum of the discounted accumulated rewards, also called the return, $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$, where R_k is the random variable of the k-th reward, T is the final time step, and $\gamma \in (0, 1]$ denotes the discount factor, which balances the importance of immediate and future rewards. For a fixed policy π and a given state s_t, we define the *Q-value* of state s_t and action a_t

$$q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[G_t | S_t = s_t, A_t = a_t \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s_t, A_t = a_t \right]$$

as the expected return G_t when taking action a_t in state s_t and following the policy π afterward.

Value-based algorithms, such as *deep Q-learning* [31] (DQN), approximate an optimal policy π_* by learning the Q-values. Deep Q-learning uses a neural network (NN) to learn $q_*(s_t, a_t)$ of the optimal policy π_*. Let θ_i denote the NN's parameters in the i-th training iteration and let $Q_{\theta_i}(s_t, a_t)$ be the corresponding estimated Q-values. The network is updated according to the loss function

$$L(\theta_i) = \mathbb{E}_{(s_t,a_t,r_{t+1},s_{t+1})\sim\mathcal{U}(\mathcal{B})} \left[\left(r_{t+1} + \gamma \cdot \max_{a'} Q_{\theta'}(s_{t+1}, a') - Q_{\theta_i}(s_t, a_t) \right)^2 \right],$$

where the expectations are taken over experiences $(s_t, a_t, r_{t+1}, s_{t+1})$ uniformly sampled from an experience replay buffer $\mathcal{B}$ [31]. To prevent unstable performance, it is common to optimize this network by using a network from a former iteration, the so-called target network with parameters θ' [31]. The soft update rule is $\theta' = (1-\tau)\cdot\theta_i + \tau\cdot\theta'$ with $\tau \in (0,1)$ [13,43].

The experiences are generated from an ϵ-greedy policy that chooses a random action with probability ϵ and an action yielding the highest Q-value with probability $1-\epsilon$. Starting from a high initial value, ϵ is exponentially reduced during training until it meets a specified threshold.

A popular extension of the DQN algorithm, called *deep Q-learning with prioritized experience replay* (DQNPR) [38], is based on the assumption that experiences with low individual losses do not contribute as much to the learning process as experiences with high losses since they carry less relevant information. Hence, for each experience $(s_t, a_t, s_{t+1}, r_{t+1})$, DQNPR computes a sampling priority δ proportional to its loss. During network updates, each experience gets sampled with a probability proportional to δ, such that experiences with high losses are used more frequently to update the policy than experiences with small losses.

2.2 DSMC Evaluation Stages

DSMC evaluation stages [19,15] leverage deep statistical model checking [17,18] to analyze the performance of DRL agents with regard to safety-critical properties.

DSMC uses the policy given by an NN to resolve the nondeterminism in the MDP constituting the environment. Subsequently, it performs Monte-Carlo-simulations until the estimated probability of the analyzed property is within a statistical confidence bound.

DSMC evaluation stages are conducted at regular intervals during training to evaluate any evaluation function E. The corresponding value for a state s is called the evaluation value $E(s)$. Previous work [19,15] proposes two different methods for incorporating the information gained through the ES into training, based on the assumption that the possible initial states for the problem at hand can be partitioned into a feasibly small set of state space regions:

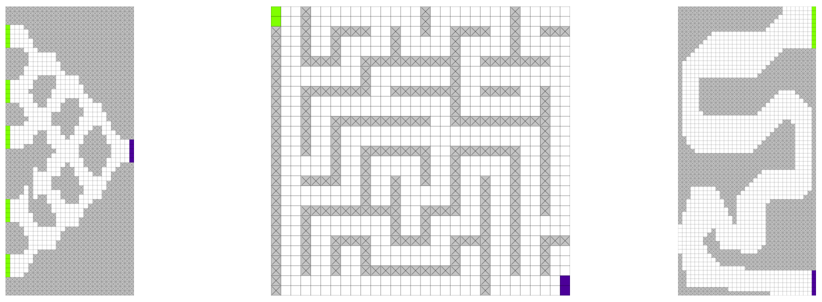
(1) *Evaluation-based initial distribution* (EID) shifts the initial distribution to start with a higher probability in areas with a lower evaluation value and vice versa.
(2) In many DRL algorithms, a replay buffer is used. Following the idea of DQNPR, this replay buffer can be prioritized. However, instead of using the TD-error as

the priority, *evaluation-based prioritized replay* (EPR) bases the priority on the evaluation value.

DSMC evaluation stages have been shown to be especially useful when utilizing evaluation functions for safety-critical objectives that are not directly suited as reward functions.

2.3 Benchmarks

Racetrack is a commonly used benchmark in the (D)RL community [7,16,20,44]. The task is to steer a car on a two-dimensional map from the starting line to reach a goal line without crashing into a wall or leaving the map. The agent is limited in its acceleration and deceleration of the current velocity, making foresighted decisions necessary. In addition, there is a certain probability that the selected acceleration will fail and, as a result, the velocity will remain unchanged. We will use one of the reward functions recommended by Gros [14], giving a positive reward (100) when reaching the goal, a negative one (-20) when crashing, and a zero reward else. Figure 3 shows three maps used in this paper besides the one displayed in Figure 1.



(a) River-deadend. (b) Maze. (c) Hansen-bigger.

Fig. 3: Three Racetrack maps used in this paper. The starting line is purple, the goal line is green, and the walls are gray.

MiniGrid is a benchmark widely used in the DRL community [9,10,12,27,35]. A MiniGrid environment corresponds to a discrete grid world where the agent needs to navigate through the grid and possibly interact with objects to solve the task. Figure 4 shows our custom DynObsDoor environment, where, starting at the top left-hand corner, the agent needs to walk past walls and avoid collisions with the randomly moving obstacles to reach the green goal cell. Furthermore, it must open the yellow door in the middle of the grid. The reward function used is similar to the one in Racetrack: positive (1) when winning, negative (-1) when losing, and zero else.

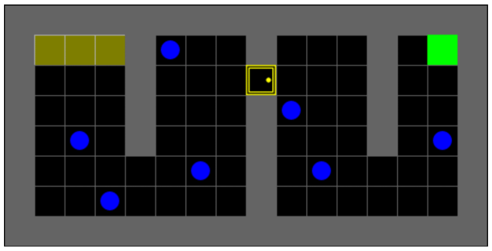


Fig. 4: The DynObsDoor environment. The starting cells are yellow, the goal cell is green, and the walls are gray. The blue dots are randomly moving obstacles.

3 Regret and State Restoration in Evaluation-based Deep Reinforcement Learning

This sections presents our algorithm ***r**egret **a**nd state **r**estoration in **e**valuation-based deep reinforcement learning* (RARE). We distinguish two variants: (i) ***r**egret **a**nd state **r**estoration in **e**valuation-based **i**nitial **d**istribution* (RAREID), and (ii) ***r**egret **a**nd state **r**estoration in **e**valuation-based **p**rioritized **r**eplay* (RAREPR). Both share the idea of conducting evaluation stages and exploiting this information for effective training in two different ways.

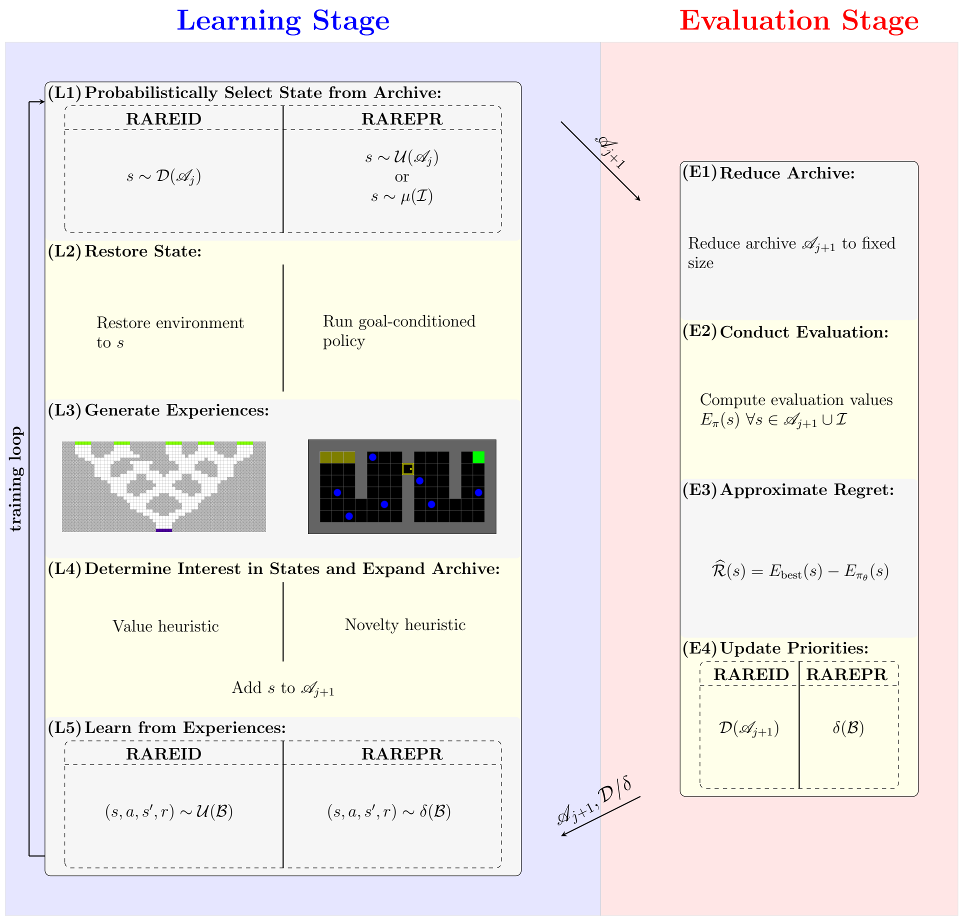
3.1 Idea

In a nutshell, our idea is to store states that seem *interesting* during the learning process in an *archive*. During a regular DSMC evaluation stage, we first reduce the archive $\mathscr{A}$ to its most relevant subset (to achieve a fixed size) and then *evaluate* all states $s \in \mathscr{A} \cup \mathcal{I}$. To identify states with suboptimal performance, we use the evaluation values to approximate the *regret*, i.e., the distance between the optimal and current performance when the agent starts in this state. Subsequently, we either shift the distribution of starting states in favor of high-regret states (RAREID) or prioritize the replay buffer $\mathcal{B}$ according to the estimated regret (RAREPR).

3.2 Details

As in prior work [15,19], we carry out evaluation stages at C-step intervals throughout training. In every learning stage, a new archive $\mathscr{A}_{j+1}$ is constructed and provided to the following evaluation stage. In every evaluation stage, we first reduce the archive, then use said reduced archive to compute a distribution $\mathcal{D}$ (RAREID) or priorities δ (RAREPR), and afterward provide both to the next learning stage.

We present our algorithm step by step, as illustrated in Figure 5. We begin with the learning stage, subdivided into five distinct components. Initially, in the first learning stage, we use the set of initial states as the archive, i.e., $\mathscr{A}_0 = \mathcal{I}$, and use the initial distribution to select a new starting state, i.e., $s \sim \mu(\mathcal{I})$. For RAREPR, we additionally use constant values for δ.



Learning Stage
Evaluation Stage
training loop
(L1) Probabilistically Select State from Archive:
RAREID
RAREPR
$s \sim \mathcal{D}(\mathscr{A}_j)$
$s \sim \mathcal{U}(\mathscr{A}_j)$ or $s \sim \mu(\mathcal{I})$
(L2) Restore State:
Restore environment to s
Run goal-conditioned policy
(L3) Generate Experiences:
(L4) Determine Interest in States and Expand Archive:
Value heuristic
Novelty heuristic
Add s to $\mathscr{A}_{j+1}$
(L5) Learn from Experiences:
RAREID
RAREPR
$(s, a, s', r) \sim \mathcal{U}(\mathcal{B})$
$(s, a, s', r) \sim \delta(\mathcal{B})$
$\mathscr{A}_{j+1}$
$\mathscr{A}_{j+1}, \mathcal{D}/\delta$
(E1) Reduce Archive:
Reduce archive $\mathscr{A}_{j+1}$ to fixed size
(E2) Conduct Evaluation:
Compute evaluation values $E_\pi(s)\ \forall s \in \mathscr{A}_{j+1} \cup \mathcal{I}$
(E3) Approximate Regret:
$\widehat{\mathcal{R}}(s) = E_{\text{best}}(s) - E_{\pi_\theta}(s)$
(E4) Update Priorities:
RAREID
RAREPR
$\mathcal{D}(\mathscr{A}_{j+1})$
$\delta(\mathcal{B})$

Fig. 5: Graphical Depiction of the RARE algorithm.

(L1) **Probabilistically Select State from Archive:** In the case of RAREID, we sample a state s according to $\mathcal{D}(\mathscr{A}_j)$. For RAREPR, we first choose with equal probability for either sampling s from the MDP's initial distribution μ or from archive $\mathscr{A}_j$ according to the uniform distribution $\mathcal{U}$.

(L2) **Restore State:** We restore the just sampled state s by either using the environment's restore option or running a goal-conditioned policy from an initial state until s is reached [11].

(L3) **Generate Experiences:** We generate new experiences by applying the DRL algorithm of choice starting from s.

(L4) **Determine Interest in States and Expand Archive:** For all states visited during the last episode, we check whether they are of *interest*, i.e., whether they contain information valuable for learning. If affirmed, we add those states to the new archive $\mathscr{A}_{j+1}$. We determine the interest by looking at two domain-independently applicable heuristics:

(i) *Value Heuristic*: We check for all episodes' states the smoothness of the corresponding state values, i.e., when transitioning from a state to its successor, we expect $|V_{\pi_\theta}(s_t) - (V_{\pi_\theta}(s_{t+1}) + r_t)|$ to be small, where π_θ is the current policy and the state values are estimated by the NN. A significant difference indicates that the agent might have made a poor decision in that state or that the state values of the current policy have not yet been propagated through the entire state space.

(ii) *Novelty Heuristic*: We leverage recent work of *random network distillation* (RND) [8]. At the beginning of training, we randomly initialize a neural network that returns a real-valued output for each state input. We use our agent's training observations to fit a second neural network to predict the output of the first one. As a result, for sufficiently explored states, the disparity between these networks is minimal. However, the difference is significant for infrequently or never encountered states. Consequently, this provides a reliable estimate of the novelty of a given state.

(L5) **Learn from Experiences**: We update the agent's network using observations in the replay buffer. In the case of RAREID, we sample uniformly, and for RAREPR, we sample based on the priorities δ determined by the most recent evaluation stage.

Note that the learning stage uses the current archive $\mathscr{A}_j$ jointly with the distribution $\mathcal{D}$ (RAREID) or the priorities δ (RAREPR), respectively, in stages (L1) and (L5). However, archive $\mathscr{A}_{j+1}$ contains the states stored for the next evaluation stage (L2). At the end of the learning stage, the new archive $\mathscr{A}_{j+1}$ is expanded by the current one $\mathscr{A}_j$. This resulting archive $\mathscr{A}_{j+1}$ is provided to the evaluation stage for further processing, which consists of four parts:

(E1) **Reduce Archive:** The number of interesting states may vary throughout the learning stages. Thus, we reduce the archive to a fixed size before performing the evaluation. We employ two different strategies to reduce the archive size, both rooted in difference measures of the provided states' descriptions. Both strategies strive to achieve the greatest possible coverage of the explored state space while retaining the most interesting states. For details, see Appendix A.

(E2) **Conduct Evaluation:** We evaluate each state contained in the archive or the set of initial states, i.e., $s \in \mathscr{A}_{j+1} \cup \mathcal{I}$, w.r.t the evaluation function using DSMC.

(E3) **Approximate Regret:** In this part, the regret is incorporated into RARE. For each state $s \in S$, the regret is defined as the difference between the state values of the optimal and the current policy [6,34], i.e.,

$$Regret(s) = v_*(s) - v_{\pi_\theta}(s),$$

where v_{π_θ} is the current policy with network weights θ.

As we are able to evaluate an arbitrary evaluation function and not just the value function, we here introduce the *evaluation regret*

$$\mathcal{R}(s) = E_*(s) - E_{\pi_\theta}(s),$$

where E_* is the evaluation of the optimal policy and E_{π_θ} the evaluation of the current policy π_θ. Naturally, the real value of E_* is unknown. Thus, we follow the idea of Jiang et al.'s *MaxMC* method [27] and approximate the regret as

$$\widehat{\mathcal{R}}(s) = E_{\text{best}}(s) - E_{\pi_\theta}(s),$$

where E_{best} denotes the best evaluation value encountered for all states in close Euclidean proximity to the given states' description in previous evaluation stages. If no such state has formerly been evaluated, E_{best} is set to 1, ensuring the agent's emphasis on this state during the next learning stage. Note that we linearly interpolate the evaluation values to $[0, 1]$ [19]. Further, we also linearly interpolate $\widehat{\mathcal{R}}$ to the same interval.

(E4) **Update Priorities:** We calculate a distribution for states in the archive (RAREID) or the priorities to be used for the replay buffer (RAREPR) based on the estimated regret, respectively. While we want the agent to focus on states with a high regret, the initial states, as the task's original objective, are of special interest. We define

$$\psi = clip\left(\frac{1}{|\mathcal{I}|}\sum_{s\in\mathcal{I}} E(s), 1 - \psi_{max}, \psi_{min}\right) \tag{1}$$

as the clipped average evaluation value of the initial states $\mathcal{I}$, where ψ_{max} and ψ_{min} are hyperparameters.

Considering RAREID, we set the distribution $\mathcal{D}$ such that the probability $p(s)$ to start in a certain state $s \in \mathscr{A}_{j+1}$ is given by

$$p(s) = \begin{cases} (1-\psi)\cdot \frac{\widehat{\mathcal{R}}(s)+\epsilon_p}{\sum_{s'\in\mathscr{A}\cup\mathcal{I}}(1-\widehat{\mathcal{R}}(s')+\epsilon_p)} & s \in \mathcal{I} \\ \psi \cdot \frac{\widehat{\mathcal{R}}(s)+\epsilon_p}{\sum_{s'\in\mathscr{A}\cup\mathcal{I}}(1-\widehat{\mathcal{R}}(s')+\epsilon_p)} & \text{else} \end{cases}, \tag{2}$$

where ϵ_p is a hyperparameter to ensure all samples have a non-zero probability. Moreover, $p(s)$ increases for states with a high regret and vice versa. The additional weighting with ψ ensures that the initial states are considered often enough to prevent catastrophic forgetting [29], depending on their current evaluation.

Considering RAREPR, the replay priority of each state is

$$\delta(s_t) = \begin{cases} (1-\psi)\cdot(\widehat{\mathcal{R}}(s_0)+\epsilon_p)^\alpha & \text{if } s_0 \in \mathcal{I} \\ \psi\cdot(\widehat{\mathcal{R}}(s_0)+\epsilon_p)^\alpha & \text{else} \end{cases}, \tag{3}$$

where ϵ_p is the minimal priority, and s_0 is the initial state of the corresponding episode of experience $(s_t, a_t, r_{t+1}, s_{t+1})$. The priority is higher if the regret is higher and vice versa. Again, the weighting with ψ considers the initial states particularly.

3.3 Regret and State Restoration in Evaluation-based Deep Q-learning

This paper uses Mnih et al.'s deep Q-learning as a base algorithm to implement RARE. We describe the complete algorithm in pseudo-code in Algorithm 1 (see Appendix B). The components of the RARE algorithm are highlighted in blue. However, note that RARE can easily be adapted to any kind of (deep) reinforcement learning; particularly RAREID to any such algorithm and RAREPR to any algorithm using a replay buffer.

4 Ablation Study

This section is dedicated to examining the two innovations of the RARE algorithm, (i) state restoration and (ii) regret, to highlight their impact.

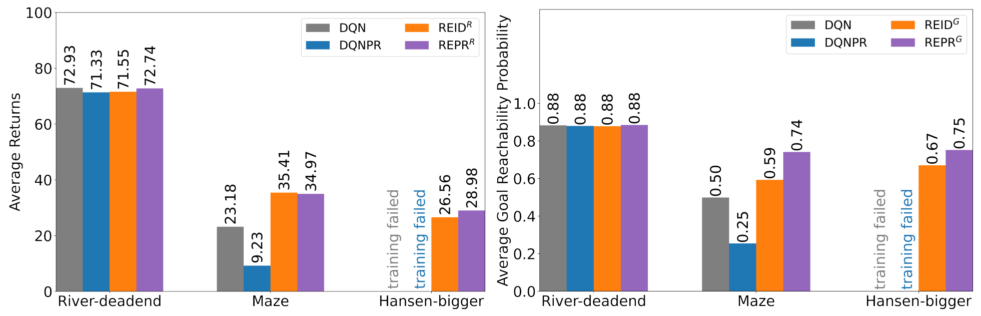
All reported results were computed by using DSMC with $\kappa = 0.05$ and $\epsilon = 0.01$ for the goal reachability probability, or $\epsilon = 1$ for the return, respectively, i.e., with a probability of 95% that the error is at most 0.01 (goal reachability probability) or 1 (return).

4.1 State Restoration

To demonstrate the effect of *state restoration*, we remove the regret estimation from RARE and only consider state restoration. Concretely, we replace $\widehat{\mathcal{R}}(s)$ with $(1 - E(s))$ in Equations (2) and (3), i.e., we compute the priorities and distribution only based on the evaluation values instead of computing them based on the regret.

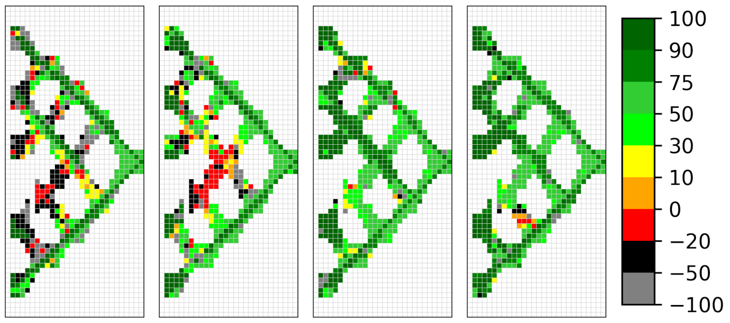
For the sake of clarity, we write REID (state **r**estoration in **e**valuation-based **i**nitial **d**istribution) or REPR (state **r**estoration in **e**valuation-based **e**xperience **r**eplay), respectively, when we refer to the algorithms without the regret estimation.

We conduct experiments where we compare the performance of two baselines, namely DQN and DQNPR, to both REID and REPR using two different evaluation functions: (i) the original objective, the return function (which we indicate with the



(a) Average return. (b) Average goal reachability probability.

Fig. 6: Average return and goal reachability probability achieved by DQN, DQNPR, REID, and REPR on Racetrack. Training failed means that the agent was unable to find the goal during training.



(a) DQN. (b) DQNPR. (c) REID^R. (d) REPR^R.

Fig. 7: Return achieved per map cell with zero velocity on the River-deadend map.

superscript ^{R}), and (ii) an objective not suited as a reward function, the goal reachability probability (which we indicate with the superscript ^{G}). As deep reinforcement learning is known to be sensitive to different random seeds, we perform multiple trainings and report the average result over all agents that were able to solve the task (here: find the goal).

Consider Figure 6a, which shows the achieved return for three different maps of the Racetrack for DQN, DQNPR, REID^R, and REPR^R when starting only in the original initial states, i.e., on the purple-colored starting line with zero velocity. For the River-deadend map, the performance is roughly equal, while for Maze and Hansen-bigger, both evaluation-based approaches clearly surpass the baselines.

Now additionally take into account Figure 7, displaying the achieved return throughout the map for River-deadend. Although the baselines perform similarly to the RARE algorithm when starting in the initial states, the latter clearly performs better throughout the complete map.

Using the goal reachability probability instead of the return as evaluation function E (see Figure 6b) gives the exact same observation: REID and REPR outperform the baselines. Thus, we demonstrated the benefits of using state restoration.

4.2 Regret

We now turn to the second novelty within RARE, regret. To analyze the impact of the regret, we compare the performance of RARE without employing the regret (as in Section 4.1) to the performance when regret is utilized.

Reconsider Figure 2 from the introduction. The third heatmap (Figure 2c) was conducted with REID^G, i.e., without using the regret but still by using state restoration; the last one on the right (Figure 2d) by RAREID^G, i.e., by also

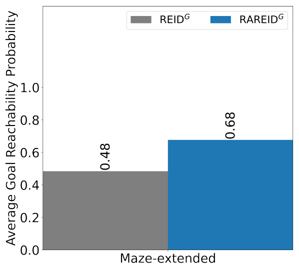


Fig. 8: Average goal reachability probability.

using the regret.[1] Both approaches used the goal reachability probability as their evaluation function. While they both ultimately supply good policies, Figure 8 shows a benefit when using the regret, and Figure 2 clearly indicates that without using the regret, the agent considers the map's bottom part exaggeratedly often.

5 Empirical Evaluation

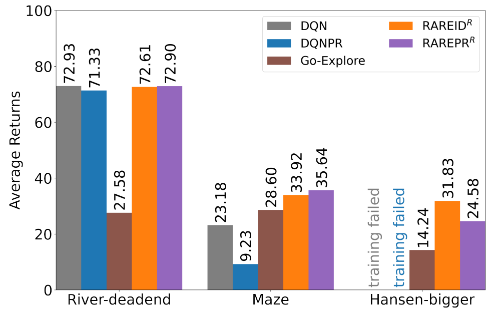
This section will provide a comprehensive empirical evaluation of the RARE algorithms with baselines. For the latter, we use DQN and DQNPR, since our RARE implementation is based on them, and Go-Explore, which is the approach that inspired our state restoration feature.

We conduct experiments on two benchmarks, namely the Racetrack and MiniGrid, using two different evaluation functions: (i) the original objective, i.e., the return, and (ii) an objective not suited as a reward function, the goal reachability probability.

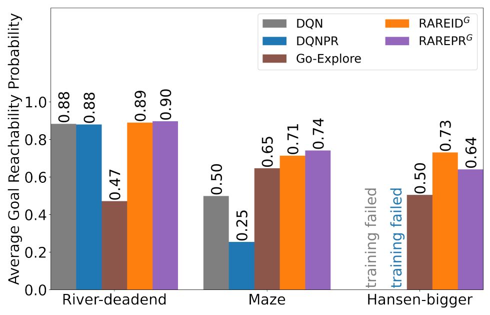
The following results were all obtained by using DSMC with $\kappa = 0.05$ and $\epsilon = 0.01$ (goal reachability probability), $\epsilon = 1$ (return - Racetrack), and $\epsilon = 0.01$ (return - MiniGrid). We again perform multiple trainings and report the average result over all agents that were able to solve the task. For further experiment details, we refer to Appendix C.

5.1 Racetrack

Consider Figure 9, which provides results for three different maps of the Racetrack. We report the performance from the initial states, i.e., the benchmark's original task.



(a) Average return.



(b) Average goal reachability probability.

Fig. 9: Average return and goal reachability probability achieved by DQN, DQNPR, Go-Explore, RAREID, and RAREPR on Racetrack. Training failed means that the agent was not able to find the goal during training.

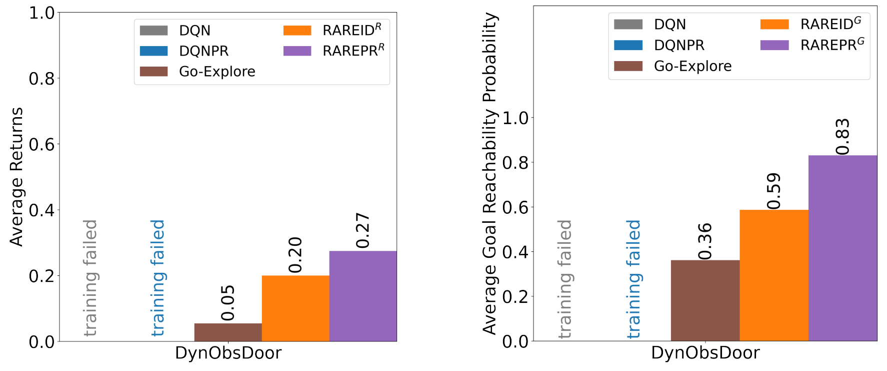
[1] The map was specifically designed to clearly highlight the advantages for easy visual discernment. Nonetheless, in the subsequent section, we will also showcase the advantages of employing regret on widely-used maps and an additional benchmark.

First, look at Figure 9a, where the return was used as an evaluation function and where we display the average obtained return. While on River-deadend, RARE performs roughly equal to the baselines (with an outlier by Go-Explore), both RAREID and RAREPR outperform the baselines on the more sophisticated maps Maze and Hansen-bigger. Now turn to Figure 9b, where the goal reachability probability was used as an evaluation function and what we also report about. The findings are similar: while on River-deadend the performance is roughly equal, RARE clearly performs best on all other maps.

5.2 MiniGrid

We now turn to our second benchmark, the well-established MiniGrid benchmark. Referring to Figure 10, the agents' performance gets compared in terms of return (Figure 10a) and goal reachability probability (Figure 10b). Accordingly, these were also the evaluation functions used for RARE.

In contrast to Racetrack, here, the primary advantage of RARE over DQN and DQNPR is that RARE is capable of solving the benchmark, while both DQN and DQNPR fail to find the goal. This distinction is evident regardless of whether the results are analyzed in terms of return or goal reachability probability. Go-Explore is able to find the goal, but still, both RARE algorithms clearly perform better.



(a) Average return.

(b) Average goal reachability probability.

Fig. 10: Average return and goal reachability probability achieved by DQN, DQNPR, Go-Explore, RAREID, and RAREPR on MiniGrid. Training failed means that the agent was not able to find the goal during training.

6 Related Work

Our work directly relates to *Go-Explore*, as our approach of state restorations combined with evaluation stages was inspired by the work of Ecoffet et al. [11]. Similarly to RARE, Go-Explore stores all visited states in its archive, yet instead of subsequently drawing states where safety performance is poor, as RARE does, Go-Explore draws the least frequently seen states. Thus, Go-Explore seeks to explore as much of the state space as possible, whereas our RARE method focuses the training on parts of the state space most relevant to safety objectives. Also, Go-Explore does not take the regret into account.

Further, state restorations can be compared to the well-established idea of *importance splitting*, where a restart is conducted from rare but relevant paths [32,26].

Recent work of Hasanbeig et al. [23] introduces a technique to include a property expressed as an LTL formula and synthesize policies to optimize the probability of fulfilling that LTL property. However, this method, while allowing for the specification of complex tasks, does not address the problems tied to safety-critical reward structures.

Similarly, Hasanbeig et al. [24] use LTL properties to derive meaningful reward functions for unknown environments. Applying their method to the property used in this paper (optimizing goal reachability probability without getting stuck in an unsafe state) yields the exact reward function we employ: positive when reaching the goal, negative when harming safety, and zero otherwise. Consequently, this otherwise effective approach proves unhelpful in our case.

Our research also connects to *safe reinforcement learning*. Various studies investigate the usage of *shields* [1,5,25] or *permissive schedulers* [28] to restrict the agent from reaching unsafe states, even during training. These methodologies, however, demand pre-computed shields or permissive schedulers. In contrast, our proposed method is model-free, and neither requires the computation of a shield or permissive scheduler in advance, nor do we have to restrict the action (and thus the state) space based on prior knowledge. Instead, the task is learned entirely through self-play and Monte-Carlo-based evaluation runs. Moreover, our approach is further applicable in more general scenarios, where states are not just differentiated into safe and unsafe, but exhibit more granular distinctions.

7 Conclusion and Future Work

We proposed two variants of the RARE algorithm, namely RAREID and RAREPR. We demonstrated that both innovations of RARE, (i) state restorations combined with DSMC evaluation stages, and (ii) utilizing the regret estimation, are beneficial when using deep reinforcement learning for safety-critical applications. Also, we provided an empirical evaluation showing that RARE outperforms the standard baseline of deep Q-learning and the related approach of Go-Explore.

In the future, we plan to incorporate a latent space representation into our algorithm. This will enable automatic clustering of the observed states. Therefore, we promise ourselves that this will help with (i) having a better selection of interesting states in the archives, and (ii) enabling an even better estimation of the maximal evaluation value and, thus, also improving our regret approximation.

In view of Anderson's recent work [3], we aim to investigate whether combining our algorithm with online shielding might improve the resulting policies' performance.

Also, even though this algorithm was specially designed to operate on sparse reward tasks, a comparison on dense reward benchmarks is of interest, as we expect our technique also to be beneficial in such settings.

References

1. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe Reinforcement Learning via Shielding. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
2. Amit, R., Meir, R., Ciosek, K.: Discount Factor as a Regularizer in Reinforcement Learning. In: International Conference on Machine Learning. pp. 269–278. PMLR (2020)
3. Anderson, G., Chaudhuri, S., Dillig, I.: Guiding safe exploration with weakest preconditions. arXiv preprint arXiv:2209.14148 (2022)
4. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., Zaremba, W.: Hindsight experience replay. Advances in neural information processing systems **30** (2017)
5. Avni, G., Bloem, R., Chatterjee, K., Henzinger, T.A., Könighofer, B., Pranger, S.: Run-time Optimization for Learned Controllers Through Quantitative Games. In: International Conference on Computer Aided Verification. pp. 630–649. Springer (2019)
6. Azar, M.G., Osband, I., Munos, R.: Minimax regret bounds for reinforcement learning. In: International Conference on Machine Learning. pp. 263–272. PMLR (2017)
7. Baier, C., Christakis, M., Gros, T.P., Groß, D., Gumhold, S., Hermanns, H., Hoffmann, J., Klauck, M.: Lab Conditions for Research on Explainable Automated Decisions. In: Trustworthy AI–Integrating Learning, Optimization and Reasoning: First International Workshop, TAILOR 2020. p. 83. Springer Nature (2020)
8. Burda, Y., Edwards, H., Storkey, A., Klimov, O.: Exploration by random network distillation. arXiv preprint arXiv:1810.12894 (2018)
9. Campero, A., Raileanu, R., Küttler, H., Tenenbaum, J.B., Rocktäschel, T., Grefenstette, E.: Learning with amigo: Adversarially motivated intrinsic goals. arXiv preprint arXiv:2006.12122 (2020)
10. Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T.H., Bengio, Y.: Babyai: A platform to study the sample efficiency of grounded language learning. arXiv preprint arXiv:1810.08272 (2018)
11. Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K.O., Clune, J.: First return, then explore. Nature **590**(7847), 580–586 (2021)
12. Flet-Berliac, Y., Ferret, J., Pietquin, O., Preux, P., Geist, M.: Adversarially guided actor-critic. arXiv preprint arXiv:2102.04376 (2021)
13. Fujita, Y., Nagarajan, P., Kataoka, T., Ishikawa, T.: Chainerrl: A Deep Reinforcement Learning Library. Journal of Machine Learning Research **22**(77), 1–14 (2021)
14. Gros, T.P.: Tracking the Race: Analyzing Racetrack Agents Trained with Imitation Learning and Deep Reinforcement Learning. Master's thesis, Saarland University, Saarland Informatics Campus, 66123 Saarbrücken (2021)
15. Gros, T.P., Groß, J., Höller, D., Hoffmann, J., Klauck, M., Meerkamp, H., Müller, N.J., Schaller, L., Wolf, V.: Dsmc evaluation stages: Fostering robust and safe behavior in deep reinforcement learning – extended version. In: Transactions on Modeling and Computer Simulation. Accepted for publication, not published yet.
16. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Köhl, M.A., Wolf, V.: Mogym: using formal models for training and verifying decision-making agents. In: Computer Aided Verification: 34th International Conference, CAV 2022, Haifa, Israel, August 7–10, 2022, Proceedings, Part II. pp. 430–443. Springer (2022)
17. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Steinmetz, M.: Analyzing neural network behavior through deep statistical model checking. International Journal on Software Tools for Technology Transfer pp. 1–20 (2022)

18. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Steinmetz, M.: Deep Statistical Model Checking. In: Proceedings of the 40th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE'20) (2020), available at https://doi.org/10.1007/978-3-030-50086-3_6
19. Gros, T.P., Höller, D., Hoffmann, J., Klauck, M., Meerkamp, H., Wolf, V.: Dsmc evaluation stages: Fostering robust and safe behavior in deep reinforcement learning. In: Quantitative Evaluation of Systems: 18th International Conference, QEST 2021, Paris, France, August 23–27, 2021, Proceedings 18. pp. 197–216. Springer (2021)
20. Gros, T.P., Höller, D., Hoffmann, J., Wolf, V.: Tracking the Race Between Deep Reinforcement Learning and Imitation Learning. In: International Conference on Quantitative Evaluation of Systems. pp. 11–17. Springer (2020)
21. Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-policy Updates. In: 2017 IEEE international conference on robotics and automation (ICRA). pp. 3389–3396. IEEE (2017)
22. Hare, J.: Dealing with Sparse Rewards in Reinforcement Learning. arXiv preprint arXiv:1910.09281 (2019)
23. Hasanbeig, M., Abate, A., Kroening, D.: Logically-constrained reinforcement learning. arXiv preprint arXiv:1801.08099 (2018)
24. Hasanbeig, M., Kroening, D., Abate, A.: Deep reinforcement learning with temporal logics. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 1–22. Springer (2020)
25. Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe Reinforcement Learning Using Probabilistic Shields (2020)
26. Jegourel, C., Legay, A., Sedwards, S.: Importance splitting for statistical model checking rare properties. In: Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25. pp. 576–591. Springer (2013)
27. Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J., Grefenstette, E., Rocktäschel, T.: Replay-guided adversarial environment design. Advances in Neural Information Processing Systems **34**, 1884–1897 (2021)
28. Junges, S., Jansen, N., Dehnert, C., Topcu, U., Katoen, J.P.: Safety-constrained Reinforcement Learning for MDPs. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 130–146. Springer (2016)
29. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
30. Knox, W.B., Stone, P.: Reinforcement Learning from Human Reward: Discounting in Episodic Tasks. In: 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication. pp. 878–885 (2012). https://doi.org/10.1109/ROMAN.2012.6343862
31. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level Control through Deep Reinforcement Learning. Nature **518**, 529–533 (2015)
32. Morio, J., Pastel, R., Le Gland, F.: An overview of importance splitting for rare event simulation. European Journal of Physics **31**(5), 1295 (2010)
33. Nazari, M., Oroojlooy, A., Snyder, L., Takac, M.: Reinforcement learning for solving the vehicle routing problem. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 9839–9849. Curran Associates, Inc. (2018)

34. Parker-Holder, J., Jiang, M., Dennis, M., Samvelyan, M., Foerster, J., Grefenstette, E., Rocktäschel, T.: Evolving curricula with regret-based environment design. In: International Conference on Machine Learning. pp. 17473–17498. PMLR (2022)
35. Raileanu, R., Rocktäschel, T.: Ride: Rewarding impact-driven exploration for procedurally-generated environments. arXiv preprint arXiv:2002.12292 (2020)
36. Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Wiele, T., Mnih, V., Heess, N., Springenberg, J.T.: Learning by Playing Solving Sparse Reward Tasks From Scratch. In: International Conference on Machine Learning. pp. 4344–4353. PMLR (2018)
37. Sallab, A.E., Abdou, M., Perot, E., Yogamani, S.: Deep Reinforcement Learning Framework for Autonomous Driving. Electronic Imaging **2017**(19), 70–76 (2017)
38. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized Experience Replay. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR (2016)
39. Schwartz, A.: A Reinforcement Learning Method for Maximizing Undiscounted Rewards. In: Proceedings of the Tenth International Conference on Machine Learning. vol. 298, pp. 298–305 (1993)
40. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature **529**(7587), 484–489 (2016)
41. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go Through Self-play. Science **362**(6419), 1140–1144 (2018)
42. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the Game of Go Without Human Knowledge. Nature **550**(7676), 354–359 (2017)
43. Stooke, A., Abbeel, P.: rlpyt: A Research Code Base for Deep Reinforcement Learning in Pytorch. arXiv preprint arXiv:1909.01500 (2019)
44. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. Adaptive computation and machine learning, The MIT Press, second edn. (2018)

A Archive Reduction Strategies

In general, states from the same parts of the state space are likely of similar interest. Hence, if we reduce an archive $\mathscr{A}_{j+1}$ by only keeping the states which are most interesting according to the utilized heuristic, the resulting states $s \in \mathscr{A}_{j+1}$ might be concentrated within close proximity of each other, preventing us from focusing the training on all relevant parts of the state space. For this reason, we utilize *archive reduction strategies* that aim to identify the most interesting states that are spread throughout the state space. We introduce two different archive reduction strategies:

The cluster strategy groups nearby states into clusters and then only considers the most interesting state $s \in \mathscr{A}_{j+1}$ from each cluster to be kept in $\mathscr{A}_{j+1}$. We realize this for Racetrack and MiniGrid by utilizing their numerical state representation.

The max distance strategy aims to reduce the archive to the states that cover the largest part of the state space possible. First, this strategy selects the most interesting state $s^* \in \mathscr{A}_{j+1}$ to be kept, meaning we initialize the reduced archive as $\mathscr{A}'_{j+1} = \{s^*\}$. Next, the strategy enforces that every further state s that is added to the reduced archive must fulfill

$$s = \underset{s \in \mathscr{A}_{j+1}}{\operatorname{argmax}} \min_{s' \in \mathscr{A}'_{j+1}} \|s, s'\|_2 \,,$$

i.e., s has the largest minimum Euclidean distance to all of the already selected states $s' \in \mathscr{A}'_{j+1}$. Afterward, we set $\mathscr{A}_{j+1} = \mathscr{A}'_{j+1}$.

B Pseudo-code

Algorithm 1 Regret and State Restoration in Evaluation-based Deep Reinforcement Learning

```
1: initialize archive $\mathscr{A}_0 = \mathcal{I}$
2: initialize $\psi = 0.0$
3: for episodes $e = 0$ to $E - 1$ do
4:     sample $s_0$ according to $\begin{cases} p(s_0) & \text{// [RAREID]} \\ \mu(s_0) & \text{// [RAREPR]} \end{cases}$
5:     for steps $t = 0$ to $T - 1$ do
6:         apply heuristic $h$ to $s_t$ & add $(s_t, h(s_t))$ to $\mathscr{A}_{j+1}$
7:         with probability $\epsilon$ select random action $a_t \in \mathcal{A}(s_t)$
8:         otherwise with probability $1 - \epsilon$ select $a_t = \underset{a \in \mathcal{A}(s_t)}{\text{argmax}}\, Q_{\theta_i}(s_t, a)$
9:         execute $a_t$; observe $s_{t+1}$ and $r_{t+1}$
10:        compute $\delta = \begin{cases} \text{constant} & \text{// [RAREID]} \\ \delta & \text{// [RAREPR]} \end{cases}$
11:        store $(s_t, a_t, r_{t+1}, s_{t+1}, \delta)$ in replay buffer $\mathcal{B}$
12:        every $K$ steps do
13:            sample mini-batch of experiences $(s_j, a_j, r_{j+1}, s_{j+1}, \delta)$ from $\mathcal{B}$ w.r.t. $\delta$
14:            set target $y_j = \begin{cases} r_{j+1} & s_{j+1} \text{ terminal} \\ r_{j+1} + \gamma \cdot \max_{a'} Q_{\theta'}(s_{j+1}, a') & \text{else} \end{cases}$
15:            perform gradient descent step on loss $(y_j - Q_\theta(s_j, a_j))^2$
16:            soft-update the network weights $\theta' = (1 - \tau) \cdot \theta_i + \tau \cdot \theta'$
17:        end every
18:    end for
19:    if $e > W$ then
20:        every $L$ episodes do
21:            $\mathscr{A}_{j+1} = \mathit{reduceArchive}(\mathscr{A}_{j+1})$
22:            compute $\widehat{\mathcal{R}}(s) = E_{\text{best}}(s) - E_{\pi_\theta}(s)$ for all $s \in \mathscr{A}_{j+1} \cup \mathcal{I}$
23:            update $\psi$
24:        end every
25:    end if
26: end for
```

C Hyperparameters

Hyperparameters that are used in multiple algorithms but only have one table entry have the same value in all instances.

Parameter	Description	Value
DQN:		
E	Number of episodes (Racetrack)	100.000
E	Number of episodes (MiniGrid)	40.000
T	Maximum episode length (Racetrack)	100
T	Maximum episode length (MiniGrid)	200
γ	Discount factor	0.99
K	Q-network update frequency	4
	Batch size	512
τ	Soft update coefficient	0.001
ϵ_{start}	Initial exploration coefficient of ϵ-greedy policy	1
ϵ_{decay}	Decay factor of ϵ in each episode	0.999
ϵ_{end}	Value of ϵ at the end of the training	0.05
$\|\mathcal{B}\|$	Size of replay buffer	10^8
α_{Adam}	Learning rate of Adam optimizer (Racetrack)	$8 \cdot 10^{-4}$
α_{Adam}	Learning rate of Adam optimizer (MiniGrid)	0.0001
	Probability of acceleration failing (Racetrack, River / Maze / Hansen)	0.5/ 0.25/ 0.25
DQNPR:		
α	Prioritization coefficient for sampling priorities	1
ϵ_p	Minimum priority	10^{-6}
RAREID & RAREPR:		
W	Number of pre-training episodes	10.000
L	Evaluation frequency	10.000
ψ_{min}		0.2
ψ_{max}		0.2
	Archive size after reduction (Racetrack, River / Maze / Hansen)	127/ 151/ 292
	Archive size after reduction (MiniGrid)	17
ϵ_p	Minimum priority	0.2
ϵ_{err}	Error in DSMC's evaluation during training (Racetrack, GRP / Return)	0.05/ 4
ϵ_{err}	Error in DSMC's evaluation during evaluation (Racetrack, GRP / Return)	0.01/ 1
ϵ_{err}	Error in DSMC's evaluation during training (MiniGrid, GRP / Return)	0.05/ 0.1
ϵ_{err}	Error in DSMC's evaluation during evaluation (MiniGrid, GRP / Return)	0.01/ 0.01
κ	Probability that DSMC's error is at most ϵ_{err}	0.05

Go-Explore:

	Number of demonstrations in robustification	10
	Number of episodes during each exploration phase	100
	Maximum number of archived states (Racetrack, River / Maze / Hansen)	509/ 606/ 1168
	Maximum number of archived states (MiniGrid)	69