Saarland University

Bachelor's Thesis

Lifting the RARE Algorithm to Continuous State and Action Spaces

Submitted by:

Michel Scherer

Submitted on:

April 11, 2025

Reviewers:

Prof. Dr. Verena Wolf

Prof. Dr. Jörg Hoffmann

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

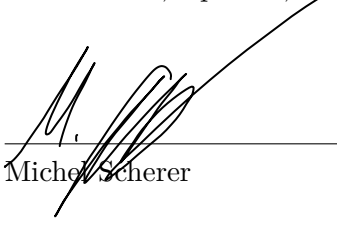
Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, April 11, 2025



Michel Scherer

Abstract

Deep Reinforcement Learning (DRL) finds increasing application in safety-critical domains like robotics and autonomous systems, demanding algorithms that ensure safety, especially during deployment. The *regret and state restoration in evaluation-based deep reinforcement learning* (RARE) framework offers a promising approach by leveraging deep statistical model checking (DSMC) and state restorations to guide exploration and provide statistical performance guarantees. However, its effectiveness has so far only been demonstrated in discrete state and action spaces, which leaves its potential in continuous domains unclear. This thesis addresses this gap by lifting the RARE framework to continuous state and action spaces. We integrate both the RAREID and RAREPR variants with two state-of-the-art continuous DRL algorithms, Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC), backed by the Stable-Baselines3 library that provides well-tested implementations of PPO and SAC. The adapted algorithms are evaluated against their respective baselines on various tasks within the Safety Gymnasium benchmark, which we modify to enable state restorations. The performance is assessed under different reward structures using high-confidence estimates of return, episode length, and survival probability obtained via DSMC. Our experiments, particularly seeded runs in the Circle task, demonstrate that RARE variants can enhance evaluation safety, often improving average survival probability and episode length or reducing performance variance compared to baselines. This is especially apparent under the original reward structure. Preliminary results in the Goal task suggest potential benefits as well. However, the extent of improvement is context-dependent, influenced by the agent, task complexity, baseline performance, and reward density. In conclusion, this work provides evidence that RARE is a viable framework for improving evaluation safety in continuous control tasks.

Acknowledgments

I would like to thank my advisors for their weekly feedback rounds that guided me throughout the writing of this thesis. I further would like to thank anybody that proof-read parts of my thesis, listened to difficulties encountered during writing or provided support in any other color, shape or form. Thank you.

Contents

1. Introduction

Deep Reinforcement Learning (DRL) is increasingly being utilized in real-world situations, complex and expensive tasks such as autonomously driving cars [1], robotics [2, 3], hardware design [4] or insulin pump regulations [5] use DRL algorithms. The importance of safe training procedures and safe algorithms is therefore not to be understated. The frequency of crashes in an autonomous driving agent or a robotics agent during training needs to be minimized to save costs and time, as repairing the agents is expensive in both. More gravely, malfunction in insulin pumps or autonomous vehicles can be fatal. Therefore, ensuring the safety of the fully trained agent is even more essential than training-time safety, as failure to reliably meet safety requirements during deployment can have catastrophic consequences.

Thus, many approaches to ensure safety in DRL training and deployment exist. Shielding approaches [6, 7, 8] aim to learn a "shield", which intercept actions that are deemed unsafe and thereby prevents the agent from taking unsafe actions. While these methods provide strong safety guarantees, they inherently risk inhibiting exploration of the state space, as they proactively restrict the agent from experiencing potentially informative, albeit risky, states [9].

Alternative approaches encourage the agent to safely explore the state space instead. Examples include methods such as "Leave No Trace" [10], which additionally to the original objective also learns how to safely return back to the initial state; "Go-Explore" [11, 12], which systematically expands the explored state space by restoring previously visited, promising states; and the RARE framework [9], which similarly uses systematic state restoration, but leverages statistical guarantees to prioritize the frequency of restoration for high-regret states.

The *regret and state restoration in evaluation-based deep reinforcement learning* (RARE) framework [9] stands out among these approaches, as it provides statistical guarantees on the agent's current estimated performance during exploration. The performance metric can be freely selected, for example the cumulative return or the goal-reaching probability. Statistical guarantees regarding the agent's performance in these metrics are then derived with *deep statistical model checking* [13] during training. RARE systematically restarts the training in previously visited states with a high *evaluation regret*, which is a measure of potential performance gain. This enables guided exploration of high-value states, where performance can be improved the most, and in turn reduces the potential of safety violations by avoiding unnecessary revisits to sufficiently explored states. Consequently, the trained agent efficiently develops an improved understanding and handling of the state space, leading to enhanced safety.

Two variations of RARE are introduced in the original paper, *regret and state restoration*

in evaluation-based initial distribution (RAREID), which prioritizes the initial states according to the evaluation regret, and *regret and state restoration in evaluation-based prioritized replay* (RAREPR), which adjusts the sampling priorities of a replay buffer according to the evaluation regret. While originally demonstrated on discrete state and action spaces, the RARE framework is algorithm-agnostic: RAREID can theoretically be integrated with any DRL algorithm, and RAREPR can be integrated with any DRL algorithm utilizing a replay buffer. Despite this generality and effectiveness, RARE has so far only been applied to *discrete state and action spaces*. Therefore, the potential in continuous state and action spaces, as they are frequently encountered in robotics or autonomous driving tasks, remains unexplored and presents the research gap addressed in this thesis.

To address this gap, this thesis adapts the RARE framework to continuous state and action spaces. Specifically, we integrate RARE with two widely-used DRL algorithms: *Proximal Policy Optimization* (PPO) [14] and *Soft Actor-Critic* (SAC) [15, 16]. Both have shown to achieve state-of-the-art performance in continuous control tasks [3, 5], and are therefore particularly suited for this adaptation. To showcase RARE's flexibility we extend the PPO and SAC implementations from the Stable-Baselines3 software package [17] without altering the underlying implementation details. The evaluation of the RARE-enhanced PPO and SAC algorithms, along with their baseline counterparts, is done using the *Safety Gymnasium* benchmark [18], which provides different continuous robotics tasks explicitly designed for safety evaluation. While the original RARE algorithm was specifically designed for sparse reward structures, Safety Gymnasium features a dense reward structure, where rewards are provided regularly throughout training.

The comparison of PPO and SAC to their RARE-enhanced counterparts aims answer the research question: 'Can RARE improve evaluation safety in continuous state and action spaces?' In order to accurately answer this research question the thesis is structured as follows: a detailed overview of related work is given in Chapter 2. The theoretical foundations of deep reinforcement learning and the algorithms applied in this thesis are outlined in Chapter 3. Chapter 4 contains explanations for the benchmark, the experimental design and the evaluation metrics. Implementation details for the RARE framework and required modifications to Safety Gymnasium are documented in Chapter 5. Chapter 6 presents the experimental findings. A discussion of the results and limitations, including an answer to the research question, and potential directions for future research are provided in Chapter 7.

2. Related Work

In this chapter we review literature relevant to the adaptation of the RARE algorithm for continuous state and action spaces. We first discuss recent progress in DRL, emphasizing the growing need for algorithms that ensure safety. Then we place existing algorithms in a classification framework for safe reinforcement learning approaches, especially positioning RARE in this landscape. Finally we motivate the baseline algorithms and the benchmark that we use.

2.1. Recent Advances In Deep Reinforcement Learning

DRL has been successfully applied to various real-world decision-making tasks across various domains. One such domain is healthcare, where DRL algorithms can be used in the automatic regulation of insulin pumps for diabetes type 1 patients [5]. Fox et al. [5] demonstrates that a DRL-based approach on simulated data leads to a reduction in median glycemic (too high or low blood glucose levels) risk by 50% and a reduction of time spent with low glucose levels by 99.8% compared to traditional methods. Another study [19] from 2023 investigates the use of DRL with human feedback in surgical robotics. The algorithm effectively trained policies for tracking moving objects or picking up gauze in a simulated environment. The algorithm outperformed the baselines in learning speed, safety and sample efficiency, while also improving exploration by leveraging human guidance.

Another large application lies in hardware engineering, where DRL is used to plan the physical layout of computer chips [4]. This task, which usually requires months of human effort can now be completed by a DRL-based algorithm in just six hours, achieving results that are comparable to or even better than those of human experts. DRL further enables autonomous robots to perform precise real-world interactions. OpenAI demonstrates this in a study [3] where a robotic hand, through precise finger joint manipulations, successfully rotates a cube between different configurations using only visual input. Beyond research labs, companies like Boston Dynamics [20] or ANYbotics [21] employ DRL methods to enhance the autonomy of quadruped robots. These systems are able to navigate complex terrains and are for instance used to increase the safety and productivity of steel plants [22] or for autonomous exploration [23].

While traditionally applied to sequential decision-making tasks, DRL also plays a key role in improving large language models (LLMs), which are subject to a lot of recent research. OpenAI's ChatGPT for instance is fine-tuned using *reinforcement learning from human feedback* (RLHF) [24], a technique that uses human preference signals to

better align the LLM's outputs to the prompts. This results in better responses and less undesired outputs.

2.2. The Need for Safe and Continuous Algorithms

These examples make it clear why trustworthy, safe algorithms are necessary. Fox et al. [5] highlights the critical importance that in closed-loop insulin control the DRL agent needs to correctly dose the insulin to prevent catastrophic failures. In surgical robotics, Ou and Tavakoli [19] emphasize the necessity of safety measures, as an uncontrolled agent could lead to serious harm in a medical environment. Errors in chip layout would result in increased production costs [4], while an incorrectly powered actuator in a robot could lead to mechanical failures, such as breaking a joint [3]. Moreover if the controller of an autonomous quadruped robots fails and the agent becomes immobile or damaged, it cannot complete its task and requires human intervention. Therefore methods that produce guarantees about the agent's performance after training are essential in these applications.

Additionally, many of the problems tackled in the papers mentioned above feature continuous state [3, 4, 5, 19] and action spaces [3, 5, 19]. While different techniques exist to discretize the action space, such as binning joint angles [3], it would preserve fine-grained control and offer greater flexibility to use the continuous actions directly [25].

2.3. Classification of Safe DRL Algorithms

To address the critical need for safety, researchers have developed various approaches to safe DRL. García and Fernández [26] provide a useful taxonomy for classifying these methods. They classify algorithms into two broad categories based on whether they change the optimization criterion or the exploration process. Each category has multiple subcategories for a more granular distinction.

The first group of algorithms changes the optimization criterion. Unlike classical (deep) RL, which maximizes the average cumulative return, these algorithms introduce constraints to limit unsafe behaviors and integrate them into the optimization procedure. Algorithms like conservative safety critics (CSC) [8] or constrained policy optimization (CPO) [27] belong to this category. Both methods introduce costs as an additional episodic quantity, similar to rewards, which quantify the severity of undesirable actions. CPO [27] uses primal-dual optimization on a modified trust-region optimization criterion that directly incorporates these costs. CSC [8] on the other hand trains a *safety critic* that conservatively estimates the costs of each state-action pair. The optimization is as well a primal-dual optimization, but instead of the costs it uses the safety critic's cost

estimate instead. Simultaneously the safety critic prevents the policy to act out actions that are evaluated to be unsafe during training.

The second category of safe RL algorithms defined by García and Fernández [26] contains algorithms that modify the exploration process. They further differentiate the category between algorithms that make use of external knowledge and algorithms that are risk-directed. Algorithms that contain external knowledge, can include this knowledge for instance in the form of a teacher component that advises the agent on action taking. This means that shielding approaches [6, 7] fall under this category, but also the safety critic in the CSC algorithm [8] can be understood as a teacher. Another method is called *leave no trace* (LNT) [10] and trains two policies: a "forward" policy, which learns the original objective and a "backward" policy that returns the agent to the starting position. The algorithm freely explores the state space as long as the backward policy is confident enough in its ability to return to the initial state. This method prioritizes real-world training, where resets require human intervention and should therefore be minimized. A major drawback of these kinds of algorithms is that the teacher could prevent access to parts of state space during training that would be beneficial in arriving at an optimal safe policy [9].

The Go-Explore [11, 12] and RARE [9] algorithms are also exploration-focused safe RL algorithms. Go-Explore and RARE both utilize state restorations during training, making them viable when an accurate simulation of the environment is available. They sample those states from an archive of previously visited states, but their sampling method differs. At each reset Go-Explore samples based on the state's visitation frequency, then the agent restores that state and explores from there. RARE performs a more robust statistical evaluation of archived states using *deep statistical model checking* (DSMC) periodically during training. Instead of the visitation frequency, it prioritizes states based on their *evaluation regret*, i.e. the difference between the best and current evaluation value. In the RAREID variant the evaluation regret is used to sample from the archive. The RAREPR variant samples the restoration state uniformly, but weights the priorities in the replay buffer according to the evaluation regret. By leveraging DSMC, RARE systematically prioritizes high-regret states, leading to a more efficient and safety-aware exploration than Go-Explore's visitation-based approach. Because of this new exploration process that puts the agent deliberately in unsafe positions during training, the authors of the original RARE paper coin the category of safe DRL of their algorithm *risk-directed exploration*. Notably, although Go-Explore shares similarities with RARE, it does not regard safety metrics and therefore is not placed in the safe DRL taxonomy.

2.4. Baseline Algorithms

To extend RARE to continuous state and action spaces, a base algorithm capable of handling such spaces is needed. Proximal Policy Optimization (PPO) [14] and Soft

Actor-Critic (SAC) [15, 16] are two state-of-the-art DRL algorithms which fulfill this requirement.

Proximal Policy Optimization (PPO) [14] has demonstrated remarkable performance in real-world applications. It has been used to achieve state-of-the-art results in the aforementioned dexterous finger joint manipulation [3] and chip placement for hardware design [4]. PPO is an on-policy actor-critic algorithm, making it suitable for RAREID, which modifies the initial state distribution.

Similarly, Soft Actor-Critic (SAC) [15, 16] has set new benchmarks in robotics [15, 16, 19]. Its entropy-regularized learning enhances exploration efficiency while maintaining robust performance. Beyond robotics, SAC has been successfully applied in closed-loop blood glucose control for type 1 diabetes patients as mentioned earlier [5]. Given SAC's use of a replay buffer, it serves as a natural foundation for adapting RAREPR, allowing prioritized replay based on evaluation regret.

The Stable-Baselines3 (SB3) [17] software package implements various well-tested DRL algorithms to provide a reliable and adaptable foundation for RL research. The package further supports 'callbacks', which allow to modify algorithm behavior at specific steps in training without changing the source code. By integrating RARE with SB3's PPO and SAC via callbacks, we ensure that RARE is integrated with well-established continuous DRL methods.

2.5. Common Benchmarks

Standardized benchmarks are crucial for evaluating and comparing DRL algorithms. The original RARE paper [9] demonstrates its effectiveness on the discrete benchmarks Racetrack [28] and MiniGrid [29]. The state space of Racetrack consists of the agents x and y position, as well as its current velocity, all in discrete units. For MiniGrid, the state space depends on the specific MiniGrid environment, but always contains a discrete direction vector, a rgb colored image tensor and environment specific details, which are also discrete. The action space as well is discrete in both benchmarks. In Racetrack the actions are discrete numbers which either signal no acceleration or a one unit acceleration in one of the eight cardinal directions. Similarly, in MiniGrid the actions are discrete and correspond to certain movements or environment-specific actions (e.g. picking up a key). This presents a gap in research where it is unclear how RARE performs in continuous environments.

To close this gap we utilize the Safety Gymnasium (SG) benchmark [18]. SG is extending the Safety Gym benchmark [30], which was deprecated in 2019. SG provides several robotics agents (e.g. Point and Car) and tasks (e.g. Goal and Circle) with continuous state and action spaces. In contrast to Racetrack or MiniGrid, SG incorporates cost signals associated with safety violations. While algorithms like CSC or CPO can directly

handle these, RARE provides no direct way of using cost signals. Additionally, there is no simple way to restore a state in this benchmark. We explain how we solve these problems in further detail in Chapter 4 and Chapter 5. Further SG predominantly features a dense reward structure compared to the sparse rewards in Racetrack or MiniGrid, contrasting the sparse reward settings that were used in the RARE paper. This allows us to investigate the performance of RARE in dense reward structures as well. The selection of Safety Gymnasium thus provides a suitable and challenging testbed for assessing the continuous RARE algorithm.

3. Theoretical Background

3.1. Fundamentals of Reinforcement Learning

At the foundation of DRL lay *Markov decision processes* (MDPs). A MDP is a tuple $(S, A, P, R, \mu, \gamma)$, where S is a set of possible *states*, which can be discrete or continuous. A is a set of actions and defines the action space, actions too can be discrete or continuous. P is the transition probability function, it describes the probability of arriving in state s' when applying action a in state s. R is the *reward function*, which describes the return when arriving in a state. μ is the initial state distribution, it describes the probability of starting in a certain state. We define $\mathcal{I} = \{s : \mu(s) \neq 0\}$ as the set of initial states. $\gamma \in (0, 1]$ denotes the discount factor balancing the immediate and future rewards

Reinforcement Learning has the goal to learn a policy $\pi(a|s)$ that assigns a probability of taking action a in state s. The *optimal policy* maximizes the expected cumulative discounted reward, also known as the *return*

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

where T represents the final time step.

3.1.1. Constrained Markov Decision Processes

Many safety scenarios can be modeled by using *constrained Markov decision processes* (CMDPs) [26, 31]. A CMDP extends the MDP tuple to $(S, A, P, R, C, \mu, \gamma, \mathbf{d})$, where C is a *cost function* analogous to the reward function R, assigning a cost $C(s, a, s')$ to transitions. The vector $\mathbf{d}$ specifies the maximum allowable expected cumulative cost d_i for each constraint. The objective within a CMDP is to find a policy π that maximizes the expected return, subject to the constraints on expected cumulative costs:

$$\begin{aligned} \max_{\pi} \quad & \mathbb{E}_{\pi}\left[G_0\right] \\ \text{subject to} \quad & \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t, s_{t+1})\right] \leq d_i \quad \forall i. \end{aligned}$$

This formalism allows explicit encoding of safety considerations. Algorithms like CPO [27] or CSC [8] directly address this constrained optimization problem. The RARE framework used in this thesis does not optimize the CMDP objective, although the benchmark we test against incorporates safety constraints via costs. Therefore, to integrate the notion of safety violation we treat any incurred cost as an episode-ending event.

3.1.2. Reward Shaping

While CMDPs provide a formal way to handle constraints using explicit cost functions, agent behavior can also be guided by modifying the reward signal itself, a technique known as *reward shaping*. This approach is particularly useful when the primary reward signal from the environment is sparse (e.g., only received at the end of an episode) or when we want to provide more direct feedback to encourage or discourage specific behaviors related to objectives like safety.

Reward shaping transforms the original MDP $M = (S, A, P, R, \mu, \gamma)$ into the shaped MDP $M' = (S, A, P, R', \mu, \gamma)$ where $R' = R + F$, with $F : S \times A \times S \to \mathbb{R}$ is called the *shaping reward function* [32]. The goal of F is to provide additional reward signals that guide the agent's exploration and learning process.

In the context of safe DRL it is often beneficial to use negative rewards to integrate safety considerations directly into the objective. This approach gives an immediate feedback signal for safety violations, and should therefore aid algorithms in improving the safety performance of agents.

In our work, since we treat cost incurrence as episode termination rather than an explicit constraint signal during learning, we additionally to the original reward structure, also experiment with reward shaping as a mechanism for discouraging unsafe actions.

3.1.3. Continuous State and Action Spaces

In DRL, state and action spaces can be either discrete or continuous. While discrete spaces consist of distinct, countable sets of states and actions, continuous spaces are characterized by states and actions that can take any value within a continuous range. Formally, in continuous environments, the state space S is typically a subset of a Euclidean space $\mathbb{R}^n$, and the action space A is a subset of $\mathbb{R}^m$, where n and m represent the dimensions of the state and action spaces, respectively.

One advantage of using continuous spaces is, that they allow a direct modeling of environment dynamics. If e.g. we have a robot with two sensors that measure distance to different objects in the environment, each of the sensors can be described with one

dimension of the observation space that depicts the distance measured by that sensor as a real-valued number. If the agent is equipped with 2 actuators that control the steering wheel an accelerator, then in continuous action spaces, each actuator can be directly controlled to arbitrary precision simultaneously.

Well researched models like Deep Q-Networks only work on discrete action spaces, because they take the maximum state-action value over all actions. This max operation is not possible in continuous spaces. One way to use these methods would usually consist of enumerating all combinations of actions and binning them into discrete ranges, which scales poorly with complex, multi-actuator settings and reduces the granularity of actions, by discarding information about the structure of the action domain [25].

3.2. Deep Statistical Model Checking

Deep statistical model checking (DSMC) [33] is a verification technique used to formally evaluate properties of a given policy within a MDP. Examples of such properties include the return or the goal-reaching probability of the agent in the environment. The key strength of DSMC lies in its ability to provide statistical guarantees on the accuracy of these estimates. Specifically, we can define an acceptable error tolerance $\varepsilon > 0$ and a confidence parameter $\kappa \in (0, 1)$. DSMC then performs sufficient simulations to guarantee that the probability of the estimation error for the property of interest exceeding the tolerance ε is less than the specified κ. This guarantee is formally expressed as

$$P(error > \varepsilon) < \kappa. \tag{I}$$

This probabilistic guarantee is achieved through established statistical techniques, like the construction of confidence intervals or the Chernoff bound. In the algorithm we use in this thesis, DSMC is used in regular intervals during the training procedure. This provides high-confidence assessments of the policy's return value. In the following, these will be referred to as evaluation values.

Additional to its usage in the training procedure of RARE, DSMC is also used to compute high-confidence means during our experiments' evaluations.

3.3. RARE

Regret and state restoration in evaluation-based deep reinforcement learning (RARE) [9] offers a framework to utilize DSMC to guide the training. The original authors distinguish

between two different variants of RARE: *regret and state restoration in evaluation-based initial distribution* (RAREID) and *regret and state restoration in evaluation-based prioritized replay* (RAREPR). They differ in how they guide the training using the evaluation values gathered through DSMC.

However, their training loops are very similar. RARE is split into two alternating stages: the *learning stage* and the *evaluation stage*. During the learning stage states are collected in an archive $\mathcal{A}_j$ and the algorithm learns from experiences. In the evaluation stage the initial and archived states are evaluated using DSMC and the sampling priorities are updated. As computing evaluation with DSMC is a time intensive operation, the evaluation stage is only conducted in a certain interval instead of after each learning stage.

3.3.1. Learning Stage

During the learning stage the policy is updated. For this, first a state is sampled either from the initial states or from the current archive $\mathcal{A}_j$. In the case of RAREID, the states are sampled according to a joint probability distribution that is generated during the evaluation stage. Therefore, we defer the explanation to the next section. After sampling, that state is then restored and an experience is generated with the underlying algorithm. Afterwards for each state that was visited the relevance is determined through the *value heuristic* or *novelty heuristic* and the state is added to the next archive $\mathcal{A}_{j+1}$.

The value heuristic computes the temporal difference of the value function

$$|V_{\pi_\theta}(s_t) - (V_{\pi_\theta}(s_{t+1}) + r_t)| \tag{II}$$

between the current state s_t and successor state s_{t+1}. Intuitively this measures how well the current state's value is estimated by our policy. A large value indicates a knowledge gap in this state and necessitates further training to improve the estimate. This makes the state relevant for training.

The novelty heuristic is based on *random network distillation* (RND) [34]. In the beginning of training two feed-forward neural networks (NN) are initialized randomly. One NN is the target network which weights will not be updated during training. The other network is the predictor network and is trained to, given a state, estimate the output of the target network. With small differences between model outputs, the state was already sufficiently explored, while with large differences the state was not encountered often enough.

Finally the underlying algorithm learns from the experience, using the its original procedure to do so. In the case of RAREID the original procedure is not changed at all. In the case of RAREPR, which requires a replay buffer, the sampling priorities are set according to the results of the last evaluation stage.

3.3.2. Evaluation Stage

In the evaluation stage the inital states and the archived states are evaluated with DSMC. Using the evaluation values the sampling priorities or the joint probability distribution are updated.

In the first step the next archive $\mathcal{A}_{j+1}$, which currently holds all states encountered during the learning stage, is merged with the previous $\mathcal{A}_j$ and reduced to a fixed size. This is done to reduce computational load and focus the training on the most important states. Two strategies are available for the reduction: the cluster strategy and the maximal distance strategy.

In the cluster strategy the archive is split into clusters according to e.g. the x-y position of the agent in the states. In each cluster only the most relevant state, according to the value or novelty heuristic, is kept.

In the maximal distance strategy the goal is to capture the largest part of the state space in the evaluation. For this the most relevant state s^* is selected and added to the reduced archive $\mathcal{A}'_{j+1}$. Iteratively the next states are added according to the largest minimum distance of that state to the already selected states, i.e. the state that fulfills

$$s = \text{argmax}_{s \in \mathcal{A}_{j+1}} \min_{s' \in \mathcal{A}'_{j+1}} \left\| s, s' \right\|_2 . \tag{III}$$

After the archive is reduced, all states in the initial states set $\mathcal{I}$ and the reduced archive $\mathcal{A}_{j+1}$ are evaluated using DSMC. This evaluation value is denoted by $E_{\pi_\theta}(s)$ for each state s. The evaluation value is used to compute an approximation of the *evaluation regret*

$$\hat{\mathcal{R}}(s) = E_{best}(s) - E_{\pi_\theta}(s), \tag{IV}$$

where $E_{best}(s)$ is the best evaluation value encountered thus far. In the special case where this is the first evaluation stage in the algorithm, this is set to $E_{best}(s) = 1$, which ensures that the priority for the state s will be high. The evaluation regret quantifies the

potential for improvement in that state. Both the evaluation values and the evaluation regret are linearly interpolated between 0 and 1.

After the computation of evaluation regrets, the priorities are updated. For this, first a weighing factor for the initial states is computed, it enforces a focus on the initial states compared to archived states during training because these are the original objective of the environment. The clipped average evaluation value is defined by

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

where $\psi_{\max}, \psi_{\min}$ are hyperparameters, which constrain the focus on initial states.

In the case of RAREID, using ψ and the evaluation values, a joint probability distribution

$$p(s) = \begin{cases} (1-\psi)\,\frac{\mathcal{R}(s)+\epsilon_p}{\sum_{s'\in\mathcal{A}_{j+1}\cup\mathcal{I}}\left(1-\mathcal{R}(s')+\epsilon_p\right)} & \text{if } s\in\mathcal{I}, \\ \psi\,\frac{\mathcal{R}(s)+\epsilon_p}{\sum_{s'\in\mathcal{A}_{j+1}\cup\mathcal{I}}\left(1-\mathcal{R}(s')+\epsilon_p\right)} & \text{else,} \end{cases} \tag{VI}$$

is constructed, using a small constant ϵ_p for numerical stability. Intuitively $p(s)$ is large for high-regret values and the weighing with ψ prevents *catastrophic forgetting* [9]. Catastrophic forgetting describes the loss of performance for already learned states during later training steps.

For RAREPR, the replay priority of each state is updated according to

$$\delta(s_t) = \begin{cases} (1-\psi)\cdot\left(\hat{\mathcal{R}}(s_0)+\epsilon_p\right)^\alpha & \text{if } s_0\in\mathcal{I} \\ \psi\cdot\left(\hat{\mathcal{R}}(s_0)+\epsilon_p\right)^\alpha & \text{else.} \end{cases} \tag{VII}$$

Again, ψ prevents catastrophic forgetting by increasing the sampling priority of initial states if they achieve low evaluation values, and ϵ_p is a constant for numerical stability. $\alpha\in[0,1]$ is a hyperparameter that controls the impact of higher and lower evaluation regret values on the priority. Intutively, a smaller α equalizes the priority of high and low regret states. s_0 is the starting state (either intial or archived) of the experience $(s_t, a_t, r_{t+1}, s_{t+1})$. This replay priority essentially increases the sampling rate of states that originated from a high regret state during policy updates.

3.4. Baseline Algorithms

3.4.1. Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) [14] is an on-policy actor-critic algorithm. In actor-critic methods, the *actor* network directly learns the policy function, whereas the *critic* network estimates the value function. Being on-policy implies that only experiences gathered under the current policy are used to update that policy, and all other experiences are discarded.

PPO builds upon Trust Region Policy Optimization (TRPO) [35], which employs the Kullback-Leibler (KL) Divergence as a constraint during policy updates. Although TRPO uses a more expensive second-order optimization procedure, it generally performs similarly to PPO's simpler clipped surrogate objective. Both algorithms explicitly ensure that the updated policy remains close to the old one, thereby mitigating the risk that a single adverse update can irreversibly harm performance.

During training, samples generated by the current policy are used to update both the actor and critic networks. To this end, a joint loss function is employed:

$$L_t^{\text{joint}} \;=\; L_t^{\text{actor}} \;+\; L_t^{\text{critic}} \;+\; L_t^{\text{entropy}}. \tag{VIII}$$

The *entropy loss* is given by

$$L^{\text{entropy}} \;=\; \hat{H}_\pi(s_t), \tag{IX}$$

which measures the entropy of the policy at state s_t. Including this term encourages the agent to distribute probability mass more evenly among similarly good actions, thus promoting better exploration and more stable training.

The *critic loss*

$$L_t^{\text{critic}} \;=\; R_{t+1} \;+\; \gamma\,\hat{V}(s_{t+1}) \;-\; \hat{V}(s_t) \tag{X}$$

corresponds to the *temporal difference error* (TD-error) at state s_t. Essentially, it quantifies how closely the critic's predicted value of s_t matches the actual return from that state.

Finally, the *actor loss*

$$L_t^{\text{actor}} \;=\; \hat{\mathbb{E}}_t\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\; \mathit{clip}\big(r_t(\theta),\,1-\epsilon,\,1+\epsilon\big)\,\hat{A}_t\Big)\Big], \tag{XI}$$

is the clipped surrogate objective proposed by Schulman et al. [14], where $\hat{A}_t$ denotes

the *generalized advantage estimate* (GAE) [36] and

$$r_t(\theta) \;=\; \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{XII}$$

is the ratio between the new and the old policy. By clipping the ratio to the interval $\big[1-\epsilon,\, 1+\epsilon\big]$, PPO ensures that the new policy does not deviate excessively from the old one.

Algorithm 1 outlines the PPO implementation used in this thesis. In each episode, N independent actors run the current policy $\pi_{\theta_{\text{old}}}$ in the environment until the goal is reached. After computing advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$ for each trajectory, the algorithm optimizes the joint loss L^{joint} for K epochs and then updates θ_{old}.

Algorithm 1 PPO

```
for episode = 1, 2, ... do
    for agent = 1, 2, ..., N do
        Run policy π_θold until goal is reached
        Compute advantage estimates Â_1, ..., Â_T
    end for
    Optimize L^joint w.r.t. θ for K epochs
    θ_old ← θ
end for
```

3.4.2. Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) [15, 16] is also an actor-critic algorithm. In contrast to PPO, it operates *off-policy*, enabling the use of experiences gathered from older policies for the current policy updates. A key feature of SAC is its *maximum entropy* objective, which differs from standard approaches (e.g., DQN, PPO) that merely maximize the average return. Specifically, SAC maximizes both return and policy entropy:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\big[R(s_t,a_t) + \alpha\, H(\pi(\cdot \mid s_t))\big], \tag{XIII}$$

where α is a hyperparameter that balances the trade-off between reward and entropy. In contrast to PPO, where entropy primarily serves as a regularizer, here it is explicitly part of the optimization objective. In the original SAC version [15], α was a fixed hyperparameter requiring manual tuning, whereas in a later variant [16] it is learned via constrained optimization. This revised version also introduces training *multiple critics* independently, which is reported to improve performance. This latter version is also used in our implementation.

The critic's optimization relies on repeated application of the *Bellman backup operator*,

$$\mathcal{T}^{\pi}Q(s_t, a_t) = R(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}\big[V(s_{t+1})\big], \tag{XIV}$$

where

$$V(s_t) = \mathbb{E}_{a_t\sim\pi}\big[\, Q(s_t, a_t) \;-\; \alpha \,\log \pi(a_t \mid s_t)\big] \tag{XV}$$

is the *soft state value function*. In practice, we use a neural network to approximate Q and apply stochastic gradient descent to minimize the *soft Bellman residual*:

$$L^{\text{critic}} = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\Big[\tfrac{1}{2}\Big(Q_\theta(s_t, a_t) - \big(R(s_t, a_t) + \gamma\, \mathbb{E}_{s_{t+1}\sim p}[V_{\bar{\theta}}(s_{t+1})]\big)\Big)\Big]. \tag{XVI}$$

Here, θ parameterizes the *local* Q-network, while $\bar{\theta}$ parameterizes the *target* Q-network, which stabilizes training. The gradient of this loss is given by

$$\hat{\nabla}_\theta L^{\text{critic}} = \nabla_\theta Q_\theta(s_t, a_t)\, \Big(Q_\theta(s_t, a_t) - \big(R(s_t, a_t) + \gamma\, V_{\bar{\theta}}(s_{t+1})\big)\Big). \tag{XVII}$$

SAC typically employs two Q-functions for increased robustness [16], taking the minimum of both when computing targets:

$$\hat{\nabla}_{\theta_i} L^{\text{critic}_i} = \nabla_{\theta_i} Q_{\theta_i}(s_t, a_t)\, \Big(Q_{\theta_i}(s_t, a_t) - \big(R(s_t, a_t) + \gamma(\min_n V_{\bar{\theta}_n}(s_{t+1}))\big)\Big). \tag{XVIII}$$

The *actor* is also represented by a neural network with parameters ϕ. Its loss is:

$$L^{\text{actor}} = \mathbb{E}_{s_t\sim\mathcal{D}}\Big[\mathbb{E}_{a_t\sim\pi_\phi}\big[\alpha\, \log \pi_\phi(a_t \mid s_t) \;-\; \min_n Q_{\theta_n}(s_t, a_t)\big]\Big]. \tag{XIX}$$

To minimize L^{actor}, we use the *reparameterization trick*,

$$a_t = f_\phi(\epsilon_t; s_t), \tag{XX}$$

where ϵ_t is a noise vector drawn from a Gaussian distribution. This approach transforms the expectation over actions into an expectation over ϵ_t, removing the direct dependence

of action sampling on ϕ. Hence, the gradient is

$$\begin{aligned}\hat{\nabla}_\phi L^{\text{actor}} = &\nabla_\phi \left(\alpha \log(\pi_\phi(a_t \mid s_t))\right) \\ &+ \left(\nabla_{a_t} \alpha \log(\pi_\phi(a_t \mid s_t)) - \nabla_{a_t} \min_n Q_{\theta_n}(s_t, a_t)\right) \nabla_\phi f_\phi(\epsilon_t; s_t). \end{aligned} \tag{XXI}$$

We similarly update α by minimizing

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t}\big[-\alpha \log \pi_t(a_t \mid s_t) \; - \; \alpha \bar{H}\big], \tag{XXII}$$

where $\bar{H}$ is a target entropy parameter. Although this approach does not theoretically guarantee convergence, its practical effectiveness was demonstrated [16] .

Finally, to boost stability, we maintain two soft Q-functions with independent local and target networks. The complete algorithm is presented in Algorithm 2.

Algorithm 2 SAC

Input: θ_1, θ_2, ϕ
$\bar{\theta}_1 \leftarrow \theta_1, \; \bar{\theta}_2 \leftarrow \theta_2$
$\mathcal{D} \leftarrow \emptyset$
for each iteration **do**
  for each environment step **do**
    $a_t \sim \pi_\phi(a_t \mid s_t)$
    $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, R(s_t, a_t), s_{t+1})\}$
  end for
  for each gradient step **do**
    $\theta_i \leftarrow \theta_i - \lambda_\theta \hat{\nabla}_{\theta_i} L^{\text{critic}_i} \quad \forall\, i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi L^{\text{actor}}$
    $\alpha \leftarrow \alpha - \lambda_\alpha \hat{\nabla}_\alpha J(\alpha)$
    $\bar{\theta}_i \leftarrow \tau\, \theta_i + (1 - \tau)\, \bar{\theta}_i \quad \forall\, i \in \{1, 2\}$
  end for
end for

4. Methodology and Benchmark

We use the safety gymnasium (SG) benchmark [18] to test our implementation of continuous RARE. To get a better understanding of the strengths and weaknesses of RARE different scenarios are investigated. We investigate two agents: the point agent and the car agent. Both agents differ in their action and observation space. We further train these agents in two different tasks: the circle task and the goal task. Both tasks come in three different difficulty levels, ranging from easy, unconstrained (level 0), over challenging with some obstacles (level 1) to hard with many obstacles (level 2).

All three versions of continuous RARE (RAREID-PPO, RAREID-SAC and REPR-SAC) are evaluated on all combinations of these two different agents, three different tasks and three different difficulty levels. This results in 12 different scenarios for each algorithm. We observe how during training the mean reward, mean episode length and crash amount progress.

After training, we evaluate the policy which achieved the best training reward using DSMC. This enables us to make high-confidence claims about the mean reward, mean episode length and mean survival probability of the policy. RARE's state restorations deliberately put the agent in critical situation during training, we therefore expect to see a smaller mean reward and smaller mean episode length during training time than during evaluation time.

Each RARE agent is compared against its baseline algorithm: RAREID-PPO is compared against PPO, while RAREID-SAC and REPR-SAC are compared against SAC. We expect that *during training* the reward, as well as the safety metrics of the baselines are superior to those of the RARE agent because of the restoration of critical states. However, during evaluation the initial states distribution for both is the same, which offers a fair ground for comparison. During the evaluation we expect RARE to either achieve similar results to the baselines, especially in the case where the baselines succeeds in finding an optimal policy, or exceed the baseline's performance.

In each task, SG uses costs to indicate a safety violation. RARE is not designed as a constrained safety algorithm as we pointed out in Chapter 2 and Chapter 3, which means that we must incorporate the safety violations differently. One way to do this is based on the realization that a safety violation in the two task we will look at in a bit, are interpretable in the real world as a possibly catastrophic failure that we need to avoid under any circumstance. For this reason we interpret any safety violation as an episode-ending event.

As this removes any notion of safety from the feedback given to the agent, it is potentially beneficial to reformulate the reward structure to include penalties for unsafe actions.

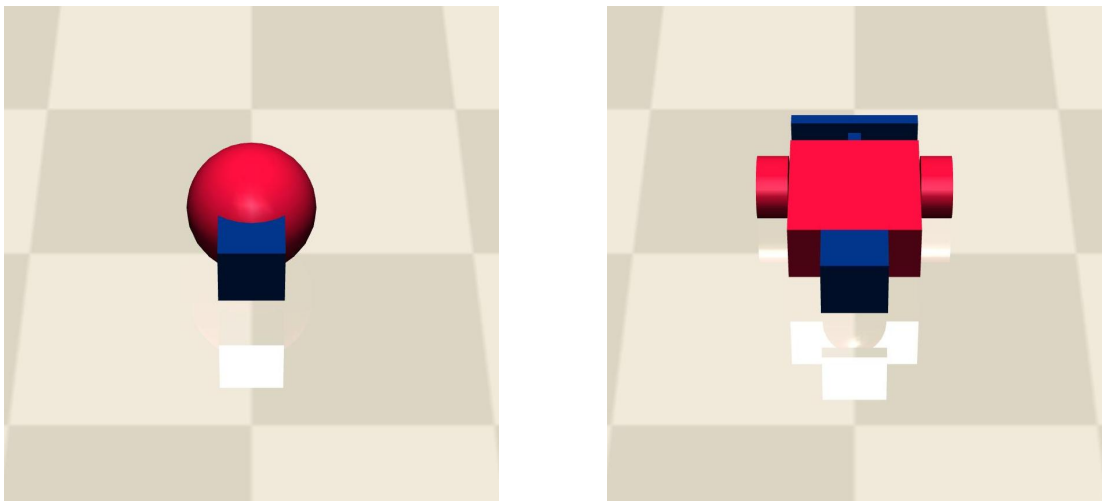
We therefore also include a reward shaping approach in our experiments.

Additionally, we change the reward structure of the goal task to create a sparse reward structure. This enables us to do an additional comparison to the original RARE paper [9], where the environments also have sparse rewards.

We now detail the agents and tasks that are used in the experiments. RARE's restore functionality should provide good exploration and performance even when the initial states do not cover the majority of the possible state space. We assume that the difference in performance to the baseline would therefore be less pronounced when the initial states already cover the majority of the state space. Therefore in the tasks the agent is placed solely in the coordinate (0,0) for each reset, as we assume that RARE provides stronger improvements in that scenario.

4.1. Agents

SG offers multiple different agents that simulate robots. These agents are equipped with different sensors that influence the shape of the observation space, as well as with different actuators that influence the shape of the action space. We use two agents of differing complexity in our experiments (Figure 4.1). We do this to improve the robustness of our findings.



(a) Point agent (b) Car agent

Figure 4.1.: Point and car agents, taken from safety gymnasium [18].

4.1.1. Point Agent

The *point agent* is composed of two simple geometric shapes: a ball, which depicts its main body, and a square, which indicates the front. There are two actuators on its body, one that moves it forward or backward, and another one that controls the rotational velocity of the agent. Both actuators are controlled simultaneously by the policy with values in the range $[-1, 1]$ each. It also contains different sensors which are part of the observation space: an accelerometer, a velocimeter, a gyrostat and a magnetometer that each measure the corresponding metric in x, y and z direction and can take on any values in $\mathbb{R}^3$ each. In total there are 12 different resulting observations at each timestep.

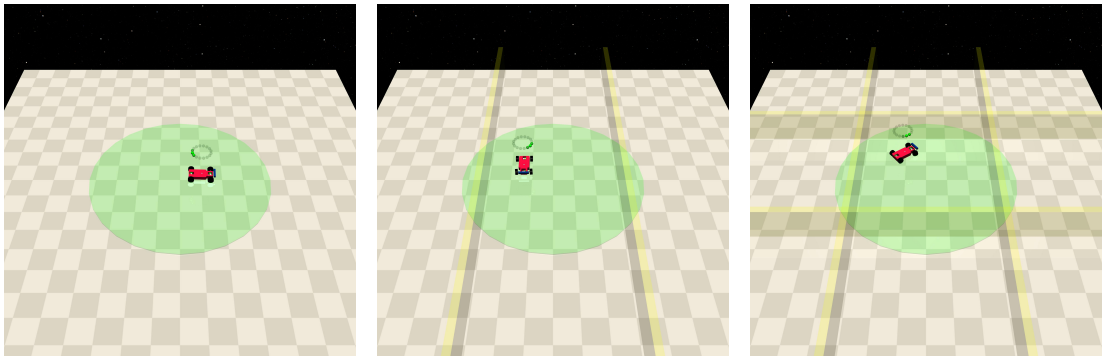
We assume that the Point agent should outperform the Car agent because of its simpler control scheme.

4.1.2. Car Agent

The *car agent* is more complex compared to the Point agent. Its main body is a square, is has two front wheels and a rear wheel that can take on any direction, in front it has a smaller square to determine the front, in the back a small shield to cover the rear wheel. It uses two actuators that each control one of the front wheels. This makes it harder to steer and accelerate the car compared to the point agent which has separate actuators for these functions. As the point agent, the car agent employs four sensors for acceleration, velocity, angular velocity and magnetic flux. Further it observes the current rotational position of the rear wheel, described by quaternions, and the angular velocity of the rear wheel. This results in 24 observations at each timestep.

4.2. Tasks

Tasks in SG define the objective, reward structure, cost function, environmental objects, and the required additional sensors of the agent. Although SG offers many task types, including vision-based and multi-agent scenarios, we restrict our experiments to two navigation-focused tasks. This selection maintains closer alignment with the benchmarks used in the original RARE paper [9], facilitating a more direct comparison than other task types would allow.



(a) Difficulty level 0 (b) Difficulty level 1 (c) Difficulty level 2

Figure 4.2.: Circle task at different difficulty levels, taken from safety gymnasium [18].

4.2.1. Circle Task

In the circle task (Figure 4.2), the agent must drive inside a circle with a radius of 1.5 in a counter-clockwise direction as fast as possible and as close as possible to the edge of the circle. In the easiest difficulty level the agent is presented with no additional obstacles. Level 1 adds two walls to the left and right side of the circle each at 1.125 units from the origin. Level 2 adds two more walls on the top and bottom of the circle.

At each time step, the agent's observation is extended with 16 sensors that simulate ***li****ght* ***d****etection* ***a****nd* ***r****anging* (LIDAR) sensors, equidistantly positioned around the agent's body. Each individual sensor's value is computed by $O_i = \frac{D_i}{D_{max}}$, where D_i is the distance to the closest object measured by the ith sensor and D_{max} is the maximal distance that the LIDAR sensors can measure. In the circle task $D_{max} = 6$.

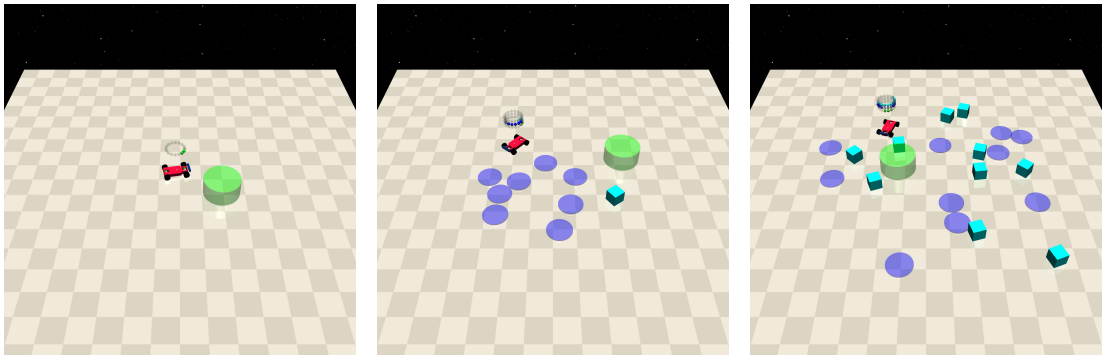
The original reward structure is described by

$$R_t = \frac{1}{1 + |r_{agent} - r_{circle}|} \times \frac{-uy + vx}{r_{agent}}, \tag{I}$$

where r_{agent} describes the Euclidean distance of the agent to the origin, r_{circle} the radius of the circle, u, v the velocity in x- and y-direction and x, y the x- and y-axis coordinate of the agent. This is scaled by a constant factor of 0.1. Intuitively, the agent maximizes the reward by driving forward in a counter-clockwise direction as fast and close to the circle's edge as possible.

The second relevant reward structure we use, slightly modifies Equation I by subtracting a constant of 2.5 from it if the agent crashed because of action a_t. Changing the reward structure in this manner is a reward shaping approach, as explained in the previous chapter.

4.2.2. Goal Task



(a) Difficulty level 0 (b) Difficulty level 1 (c) Difficulty level 2

Figure 4.3.: Goal task at different difficulty levels, taken from safety gymnasium [18].

The goal task (Figure 4.3) randomly creates at each environment reset a small, green goal circle on the map. The task of the agent is to drive into that circle. When it accomplishes that objective a new goal is randomly placed on the map. This continues until the agent accomplished 1000 steps or until it crashes. On difficulty level 0 there are only the agent and the goal on the map. The goal's location is uniformly sampled in a square area encapsulated by the points $(-1,-1),(-1,1),(1,-1),(1,1)$. It becomes slightly harder on level 1, where additionally eight hazards and one vase are placed on the map in the same square are as the goal, which is increased by 0.5. Difficulty level 2 is the hardest, it places ten hazards and ten vases on the map and the square area is increased by another 0.5.

The *hazard object* is a small, blue circle that creates costs when the agent touches it. The *vase object* is a small cyan box. In level 1 is has no costs associated with it, therefore the agent can freely touch it and is only slowed by the collision. However, in level 2 a collison also creates costs and therefore is episode-ending.

The agent's observation space is extended with 48 sensors: 16 LIDAR sensors for goal objects, 16 for hazard objects and 16 for vase objects. They work exactly as in the circle task, but the maximal distance is reduced to $D_{max} = 3$ for each sensor.

The original reward structure is split into two components. The first component is the distance reward,

$$R_t^{(distance)} = (D_{last} - D_{now}), \tag{II}$$

where D_{last} and D_{now} are the distance of the agent to the goal in timestep $t-1$ and t respectively. Intuitively this rewards the agent to get closer to the goal.

The second part of the reward is only relevant when the agent reaches a goal, it is defined by

$$R_t^{goal} = R_{goal}, \tag{III}$$

where $R_{goal} = 1$ in the original reward structure.

Similar to the circle task we also construct a reward shaping approach where we subtract 2.5 from the reward on crash. Further we can trivially create a sparse reward structure by setting $R_t^{(distance)} = 0$ for all timesteps t. Using this fact we create three more sparse reward approaches where we set $R_{goal} = 1$, $R_{goal} = 5$, $R_{goal} = 10$ and in each subtract 2.5 on crash.

4.3. Experiments

We use both Point and Car agent to solve the circle task with the two reward structures. For the original reward structure we deploy the training in each difficulty level for one million steps. This provides a perspective how RARE performs on cost-dependent environments, compared to the also cost-agnostic baselines. The second reward structure provides insight how a naïvely implemented constant negative reward influences the performance of our baselines, and in particular RARE's relative performance to those. We expect that a negative reward might discourage exploration and therefore necessitates longer training, which is why we train each agent for 5 million steps instead. As the lowest difficulty level does not present any hazards to the agent, we only train the agents on the second and third difficulty level in this case.

We use again both point and car agent to solve the goal task with all five reward structures. For the original reward structure we deploy the training in each difficulty level for five million steps, as we expect the task to be inherently more difficult to solve than the circle task. This, as in the cirlce task, provides insights in how RARE performs on a cost-dependent environment compared to the baselines. Given the absence of hazards in difficulty level 0, we perform the reward shaping approach and the three sparse reward approaches only on difficulty level 1 and 2, each for again 5 million timesteps. We expect that the reward shaping approach performs better than the original reward structure as it provides the agent with information about suboptimal moves too. The sparse reward structures provide an interesting angle of comparison to the original RARE paper, where both benchmarks used were also equipped with a sparse reward structure. We expect this to be harder to solve for the baseline algorithms, while the RARE variants could benefit due to their enhanced exploration capabilities.

Ultimately, as the goal of this thesis is not to optimize the SG benchmarks but rather to evaluate the capabilities of RARE in SG compared to its baselines, we deliberately experiment with three simple sparse reward configurations. This aims to mitigate the

risk that poor performance originate from an insufficient reward structure rather than from limitations of the method.

4.3.1. Evaluation

Each configuration is trained on the High Performance Computing (HPC) cluster provided by the university. Afterwards we plot the training rewards, training episode length, and crashes throughout the training. We save the model weights each 50,000 steps and select the best performing model checkpoint with respect to the training reward for evaluation. In the case of the circle task we were able to train five different seeds per algorithm and can therefore additionally report the 95% confidence interval of each metric. The more complex goal environment could not be tested with this statistical rigor because of time constraints.

In the evaluation we use the selected model weights to instantiate an agent in the environment. In this environment we utilize DSMC to obtain high-confidence estimations of the mean reward, mean episode length and survival probability of each algorithm. We plot each of the three metrics and note the best performing PPO and SAC based agents in each of the metrics. In the circle task we instead provide the average high-confidence mean of each metric for all seeds, as well as the standard deviation of those high-confidence means.

5. Implementation

In this chapter we explain the implementation of the RARE framework [9] into SB3 [17] using callbacks. We further detail the changes which were made to the safety gymnasium benchmark to enable state restorations directly in code.

5.1. Overview

We orientate our implementation on the implementation of RARE that the original authors provided. We use gymnasium wrappers to implement different environmental specifications for the tasks, which we discussed in the last chapter. For example, ending an episode prematurely if costs happen or changing the reward structure are implemented using wrappers. SB3 VecEnvWrappers and SB3 Callbacks are used to achieve our implementation of continuous RARE on top of SB3 algorithms. Additionally, we define a *Starter* object which encapsulates the necessary logic to restore specific states, depending on their type: initial state or archived state. In order to use this logic, certain changes to the source code of the SG benchmark are necessary, as well as small changes to the archive and DSMC implementation which the original authors provide.

5.2. Safety Gymnasium Changes

The SG benchmark, while providing many different safety relevant tasks, does not implement a restoration functionality. Thus we need to adapt the source code to enable finer control over the reset process. These changes primarily focus on exposing control over the simulator state and modifying the environment setup process during resets.

The first critical change involves the `World` class, where we introduce methods to retrieve and set the simulator state. This encompasses all simulation relevant variables: *time, joint position, joint velocities, applied actuator activations, warm-start acceleration and the last applied control mechanics.* Access to these variables is important to not only reset the position but also the simulation state that controls the observations of the agent.

Furthermore, in SG on each reset the objects' positions are sampled and placed randomly in the world. We adapt this so it becomes possible to set an agent's location or the whole position layout directly, before sampling the remaining positions. Having exact control

over all objects' positions becomes necessary when we restore archived states, which must not resample the objects positions because this would change the observations.

Specifically for the goal task we need an additional adjustment. The original code uses an extra method after the reset method to specifically randomly sample the goal position. As we already place the goal at the correct position during our modified reset method, we implement an attribute that is set during restoration and skips this resampling procedure.

Finally, we add a method that exposes the seed. This is needed to ensure that the restoration process creates the same environment as when it was originally seen. Notably, setting the seed during the reset is disconnected from the randomness in the policy which chooses the actions, so the DRL models are not seeded by this. These collective modifications allow for state restorations in the circle and goal task, which is elementary for the RARE framework.

5.3. Starter object

The RARE framework fundamentally relies on the ability to restore states, both initial states and archived states visited during exploration. Since the SG benchmark does not provide a built in state restoration feature, we developed a dedicated *Starter* object structure to manage this process and utilize the previously explained changes. This structure encapsulates the necessary information and logic to reset the environment to a specific configuration. It is important to note that the alternative would be to train a goal-conditioned policy, which is able to restore states. But as this was not used in the original paper, we deliberately not explore this approach further to improve the comparability of our results.

We define a base class, `BaseStarter`, to hold common information relevant to any saved state. This includes the raw MuJoCo simulator state, the random seed used to initialize the episode from which the state originates, environmental values like the cost and steps at that state, and the agent's position and rotation. The `BaseStarter` also provides utility methods for discretizing the agent's position into indices, calculating Euclidean distance between states, and defining equality and hashing based on these indices. Discretizing the indices is a simplification that is used in the archive. The Euclidean distance is needed for the maximal distance heuristic and the comparison methods are particularly useful for managing the archive of states within the RARE algorithm, e.g., when checking for duplicate states. The actual restoration logic is deferred to subclasses via an abstract restore method.

For initial states, the `InitialStarter` class defines the restoration logic. To restore specific initial states we set the agent's location parameter and then reset the environment. In the `ArchiveStarter` we define the restoration logic for archived states. In this case

we set the *initial* position and rotation of the agent, and the world layout in that state. In case of the goal task, we deactivate the resampling of the goal position. Then, resetting the environment with the archived state's seed and setting the saved simulator state in the environment results in the complete state restoration. Note that both restoration methods only work because we change parts of the SG source code.

5.4. Archive

During the learning stage RARE saves visited states together with their relevance inside of an archive. This class encapsulates the merge, reduction and sampling logic associated with archived states. It is slightly adapted from the original author's implementation to handle our starter objects.

Notably, while the RARE paper [9] describes an arbitrary archive form, the concrete implementation maps states into cells according to their x-y coordinates. For simplicity we similarly map states into cells. While in the case of Racetrack and MiniGrid these cells have a very close correspondence to the observation states, this is not necessarily true for SG. First, we need to discretize continuous x-y coordinates into bins. Second, when the placements of objects is random, as in the goal task, the same x-y coordinate can produce vastly different observations because the observations depend on the distance to objects but not on the coordinates alone.

We acknowledge that this might negatively influence the performance of RARE and discuss possible solutions in Chapter 7. Our implementation allows to define width and height of a map where cells are recorded, as well as the resolution. This is necessary because there is no restriction on the agent's position in the circle and goal tasks and we want to allow for arbitrarily small cells.

5.5. Evaluation Stage

The code that performs DSMC was provided by the original authors and only slightly adapted to work with our benchmark, SB3 policies and starter objects. The evaluation stage class is called inside the RAREID and RAREPR callbacks to perform DSMC on the states and during evaluation to compute high-confidence means of our evaluation metrics.

We note that that the notion of *goal-reaching probability* that was used in the original RARE paper, cannot be applied to the SG benchmark. In the circle task there is no goal to reach, while in the goal task the objective is to reach as many goals as possible compared to a singular goal. For this reason our changes to the evaluation stage code are only applicable with the return value as evaluation metric. During the evaluation we use

the survival probability, which is thematically similar to the goal reaching probability. With small changes this probability should also be possible to use as an evaluation function during the training of RARE.

5.6. Environment Wrappers

To implement task specific modifications and integrate RARE without altering the SG source code more than necessary we utilize wrappers. Wrappers are an interface provided by SG to provide additional functionality on top of the base environment. In particular they allow to extend the `step` and `reset` methods with custom behavior.

In order to end the episodes when a cost is registered, count crashes and implement the different reward structures (reward shaping and sparse rewards) we use the `SafeCostEndEpisode` wrapper. At each step it checks if the cost exceeds 0 and ends the episode by setting the `terminated` flag. This aligns with our methodological choice to treat safety violations as episode-ending events. Additionally, if the parameters for reward shaping is set, it also deducts the penalty from the reward if the cost exceeds 0. If the parameter for sparse rewards is set it further changes $R_t^{(distance)} = 0$ at each step and sets R_{goal} accordingly.

The `SafeRestoreWrapper` implements the restoration and part of the state saving logic. It intercepts the reset call and expects a starter object in the `options` dictionary. Then, the starter's restore method is called. Finally, it saves the starting position and rotation in a parameter. It also intercepts the step method and appends the current x and y coordinate to the info dictionary. This is used by the `VecRareIDWrapper` to create an `ArchiveStarter` object. The `SafeRestoreWrapper` is used for RAREID.

As RAREPR updates the priorities in a replay buffer after each evaluation stage, it is necessary to additionally keep track of the current starter's priority, as well as its x-y coordinate in the archive's cells. The replay buffer we use for RAREPR uses a very similar data structure to keep track of the priorities. This allow us to update the priorities for all experiences in the replay buffer efficiently. Therefore the `SafeRestoreWrapperPrioritizedReplay` extends `SafeRestoreWrapper` with this additional functionality during the reset and step methods.

Finally, for the baselines PPO and SAC we do not use starters. In order to specify the starting states we use another wrapper that randomly selects a coordinate from a fixed set. In our case this set consists solely of the coordinate $(0, 0)$. `SafeDiscretizeStartingStatesWrapper` implements the logic for this. It as well builds on our changes to the SG source code to specify the exact starting position. The same wrapper is also used for visualizing agents.

5.7. Vectorized Environment Wrappers

SB3's algorithms use vectorized environments for training, this means that multiple instances of the environment are created and used simultaneously. RARE uses different constructs, which require centralized management, for instance the archive between two evaluation stages is static, as well as all associated sampling probabilities. To improve comprehensiveness a central vectorized environment wrapper handles this overarching environment logic. In particular the sampling methods for initial states, the creation of `ArchiveStarter` objects and the counting of all crashes in all sub-environments is handled by these wrappers.

Similar to SG wrappers, SB3 also provides an interface for wrappers that can be used to modify environment behavior at certain points. In particular for both RAREID and RAREPR we implement their own wrapper. `VecRareIDWrapper` implements the logic necessary for RAREID, `VecRarePRWrapper` the logic for RAREPR.

Both wrappers handle the sampling of the initial states by injecting the respective starter object into the info dictionary at each reset. In the case of RAREPR the priority is also inserted and tracked internally using an array that is updated after each evaluation stage.

The core logic for creating `ArchiveStarter` objects lies here as well. After each step, if the agent's coordinate is still in the bounds defined by the archive, all relevant information is extracted and the starter object created. This is further used in the callbacks.

Finally, after each step, the sum of crashes over all sub-environments is computed and saved internally.

5.8. Callbacks

SB3 provides a callback system that allows to execute custom code at various points in their DRL algorithms, e.g. after each step or before the rollout collection. These callbacks are independent classes that can be passed to the learn method of any SB3 algorithm. We leverage that system to implement the core logic of RARE, i.e. the creation of the next archive, the computation of relevance and the evaluation stages, together with the associated priority updates. The callback is then handed to SB3's PPO and SAC implementations.

As for the vectorized environment wrappers, we developed two separate callbacks for RAREID and RAREPR. The `RAREIDCallback` implements the RAREID algorithm. During its initialization, it configures parameters related to the frequency of evaluations,

archive management settings including size and reduction strategy and the chosen relevance heuristic. This callback maintains the next archive, which accumulates states encountered between evaluation stages. It also tracks the best evaluation values observed for each state region to facilitate regret calculation. If the novelty heuristic is selected, the callback initializes a Random Network Distillation (RND) module as well.

After each step in the vectorized environment, the relevance of each starter is determined either with the novelty or value heuristic. In the case of the *value heuristic* we differentiate between the baseline algorithms. SB3's PPO version implements a value network as critic which we can directly use to compute the temporal difference error

$$|V_{\pi_\theta}(s_t) - (V_{\pi_\theta}(s_{t+1}) + r_t)|. \tag{I}$$

SB3's SAC version on the other hand does not use a value function directly, but instead uses multiple q-functions. In order to approximate the value function we set the policy evaluation to deterministic, which is possible via a flag in SB3's algorithms. The when we sample an action the policy always returns the one which has the largest probability mass. Using the identity of the value function

$$V_{\pi_\theta}(s_t) = Q_{\pi_\theta}(s_t, \pi_\theta(s_t)) \tag{II}$$

for deterministic policies, we compute Equation I. We note that this approximation is a limitation in the comparability to the original RARE implementation, where the value function was used directly.

If we use the *novelty heuristic* s_t is passed to the RND network and the mean squared error between predictor and target network constitutes the relevance.

After the relevance is computed in the step method, the callbacks add the stater, which was passed to the info dictionary by the vectorized environment wrapper, and its relevance to the archive.

Periodically, before the rollout collection process, an evaluation stage is conducted. For this the archive in the vectorized environment and the one in the callback are merged and reduced. Then the DSMC evaluation is done for each starter and regrets, evaluation values, psi and probabilities are updated. Afterwards the callback passes the reduced archive to the vectorized environment. Lastly a new, empty archive is created that collects states for the next evaluation stage.

The implementation details for RAREPR are largely the same. The main difference is that instead of starting probabilities the replay priorities are computed. The updating thereof is outsourced to the replay buffer.

5.9. Replay and Rollout Buffer

All algorithms implemented in SB3 use either a replay buffer or a rollout buffer to collect experiences for training the algorithms. We create our own subclass for replay buffer and rollout buffer in order to update the RND network if the novelty heuristic is used. This correctly updates the RND network only with samples used during training.

We implement another version of the replay buffer for RAREPR, which correctly adds states with their priority, samples states according to their priority, and updates the samples' priorities efficiently with vectorized operations.

5.10. Training Script

The training script `train_rare_safetygymnasium.py` allows us to centralize the training for the RARE agents. Using the command line interface (CLI) we can specify the concrete configuration we want to train on the HPC server. The workflow is as follows: First, we create the environment using the wrappers and vectorized wrappers on the standard safety gymnasium environment. Then, we build the callback and replay or rollout buffer according to the CLI arguments. Additionally, the CLI arguments are saved to `hyperparameters.json` for reproducibility. Finally we instantiate the baseline algorithm with the wrapped environment, and pass the callback to the learn method call. This starts the training. After training the script saves the model is saved to a folder specified with the CLI.

Similarly, the script `train_baselines_safetygymnasium.py` is used to train the baseline algorithms, PPO and SAC for the corresponding configuration. The workflow is the same as the RARE case, but callbacks and replay or rollout buffer are not modified.

These scripts allow us to flexibly declare our different training configurations for the server in submission files.

5.11. Evaluation Script

The evaluation is also conducted on the server. The corresponding logic is encapsulated in the `evaluate_comparisons.py` script. Using the CLI we specify the model checkpoint which we want to evaluate. The script then rebuilds the environment using the `hyperparameters.json` file and selects the correct baseline model and loads the model's weights. Finally the model evaluated in the environment with DSMC. After DSMC converges, the script saves the high-confidence means of return, episode length and survival probability in the model folder in a csv file.

5.12. Visualization

For rendering the model in the environment we provide a script called `visualize.py`, it enables us to specify a model checkpoint and render the models' performance in the environment. This is useful to interpret the model's statistics. A policy that always has the maximal episode length but low reward could for example just refuse to take any risk after an initial reward was gathered, rather than actually being cautious. Instead of plotting more metrics a visual inspection builds useful intuition about policy behavior.

For plotting the training and evaluation graphs, we provide the code as a jupyter notebook `create_plots.ipynb`. For the training curves we load the `progress.json` file using SB3's logger class. This file was created during training by said logger class automatically. This json file provides all the necessary training information. The evalution graph is plotted by loading the csv files with pandas, computing the mean and standard deviation for return, length and survival probability, and then using seaborn to plot a bar graph.

5.13. Hyperparameters

All hyperparameters related to PPO and SAC utilize the default hyperparameters from SB3. For the environment setup, the agent's initial position is fixed at coordinates (0, 0) and the training employs 5 parallel environments (`n_envs=5`). RARE-specific parameters include the state discretization `resolution` of 10 and an `archive_size` of 25. The boundary for inclusion in the archive (`max_coordinate`) is set to 2.0 for all but the goal environment at difficulty 2, where it is set to 3.0. This is done to accomodate for the larger creation area of goal and objects in this difficulty level. The number of RARE evaluation stages (`num_eval_stages`) is set to 20 for 1-million-step training runs and 100 for 5-million-step training runs, this achieves an evaluation stage at each 50,000th timestep. As relevance strategy, the value heuristic is used in every experiment.

The hyperparameters for the evaluation stage are `es_alpha=1`, `es_kappa=0.05` and `es_minimal_prio=0.01` for every experiment. In the circle task we use `es_epsilon=1.0`, while in the goal task we set it to `es_epsilon=0.5`. This is done to achieve stricter estimates, as the goal task is assumed harder. The minimal amount of simulations in DSMC (`es_initial_runs`) is set to 10. The hyperparameters for ψ_{min} and ψ_{max} are set to `psi_min=0.2` and `psi_max=0.8`. All policies choose the action stochastically (`deterministic=False`) during the episode collection in the evaluation stage. As archive reduction strategy we employ the cluster strategy in each experiment. The interpolation parameters of the reward (`negative_reward` and `positive_reward`) are -2.5 and 30 in the circle task, -5 and 40 in the goal task with the original and shaped reward structure, and -2.5 and $5 \times R_{goal}$ in the sparse reward structures. The reward shaping

penalty is -2.5 and `reward_shaping` is set to `True` in every but the original reward structure experiments. The sparse sparse goal reward is set to $1, 5$ or 10 in the respective sparse reward structure experiments with `sparse_reward = True`.

For the evaluation we set the admissible errors $\varepsilon_{return} = 1,\ \varepsilon_{length} = 25$ and $\varepsilon_{sp} = 0.1$, and use at least 100 episodes.

A complete table summarizing all hyperparameters can be found in Appendix A.

6. Experiments and Results

6.1. Circle Task

In the circle task we trained 5 different seeds of each of our algorithms under the original and shaped reward structures. The original reward structure includes all three difficulty levels, while the shaped reward structure was only trained on levels 1 and 2. We now report the metrics during training and during evaluation time for each algorithm, agent, task and reward structure.

6.1.1. Original Reward Structure

First we look at the original reward structure, where the reward of an action is calculated according to

$$R_t = \frac{1}{1 + |r_{agent} - r_{circle}|} \times \frac{-uy + vx}{r_{agent}}, \tag{I}$$

as explained in Chapter 4.

PPO Baseline - Training Results

We begin the report of the results with the PPO and RAREID-PPO training curves for the car agent. All referenced plots for the car agent are visible in Figure 6.1 and all plots for the point agent in Figure 6.2. In difficulty level 0 both algorithms perform equally well over the course of the whole training. As level 0 contains no dangers, it is unsurprising that the episode length is maximal and that no crashes occurred in either of the algorithms.

In difficulty level 1, the training curves follow our initial hypothesis that RARE during training slightly underperforms in all metrics. Because the restoration process puts the RARE agent deliberately into critical states, it comes as no surprise that the confidence interval for RAREID-PPO is also bigger.

Interestingly, difficulty level 2 does *not* follow our initial hypothesis. Even though the task contains the most hazards, RARE performs very similar to the baseline with a similar confidence interval. Moreover, RARE actually crashes *less often* during training than the baseline.

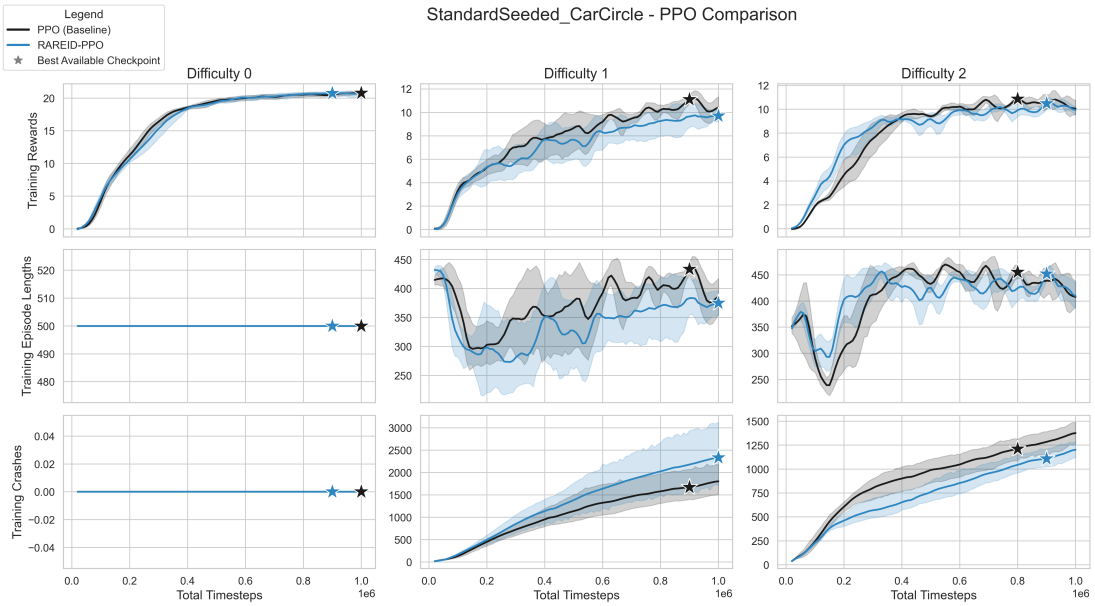


Figure 6.1.: Training mean reward, mean episode length and cumulative crashes for the PPO variants with car agent in the circle task on all difficulty levels for five seeds.

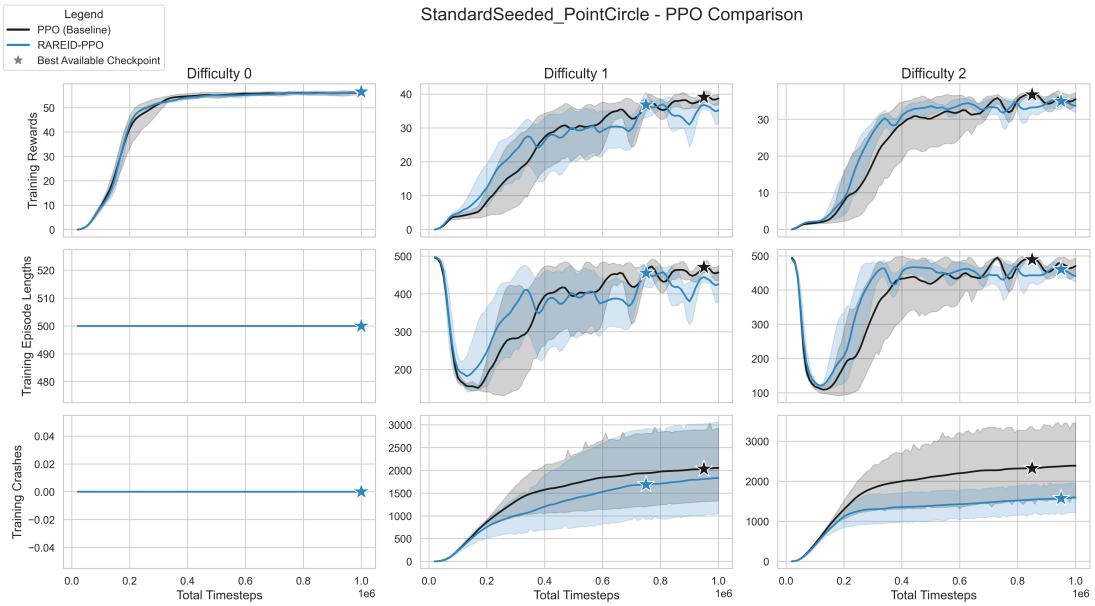


Figure 6.2.: Training mean reward, mean episode length and cumulative crashes for the PPO variants with point agent in the circle task on all difficulty levels for five seeds.

Looking at the point agent, we observe mostly similar performance for both RARE and PPO on all metrics for all three difficulty levels. Notably, the baseline achieves a slightly better reward in difficulty level 1, which follows our initial hypothesis of behavior during training time, but just as in the car agent case on level 2 RARE surprisingly crashes less often during training than the baseline.

PPO Baseline - Evaluation Results

After we now explained the training results for PPO and RAREID-PPO, we continue with the evaluation results. As explained we extracted the best model checkpoint for each seed with respect to the training reward, then we evaluated these models with DSMC and set $\varepsilon_{return} = 1$, $\varepsilon_{length} = 25$ and $\varepsilon_{sp} = 0.1$. We now report the average of these high-confidence means of reward, episode length and survival probability, together with their standard deviations as shown in Figure 6.3.

Starting with the car agent in level 0 (see Figure 6.3a), we can see in the evaluation data that the lack of difference between both algorithms during training also shows during evaluation. Both algorithms achieve a very similar return of 20.71 for PPO and 20.45 for RARE, with similar standard deviations (0.37 PPO/0.61 RARE). They also achieve the same episode length and survival probability.

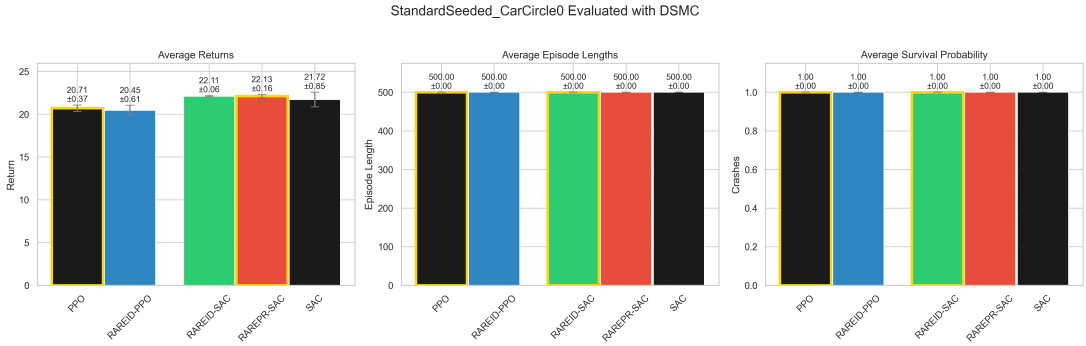
In difficulty level 1 (Figure 6.3b) the difference which we observed in the training data between both algorithms does not continue. PPO slightly outperforms RARE in each metric, however, RARE reduces the standard deviation in the safety relevant metrics: 43.02 standard deviation for episode lengths in PPO and 24.01 in the RARE variant, and for survival probability a much bigger difference of 0.22 in PPO and only 0.06 in RARE. For rewards both achieve a similar standard deviation. This result aligns with our hypothesis that RARE improves safety.

A similar result can also be observed in the evaluation data of level 2 (Figure 6.3c). Both algorithms achieve similar mean values in each metric, while RARE achieves a smaller standard deviation. In contrast to level 1, RARE also achieves on average better results in each metric. Thus RARE clearly improved safety for all car agents.

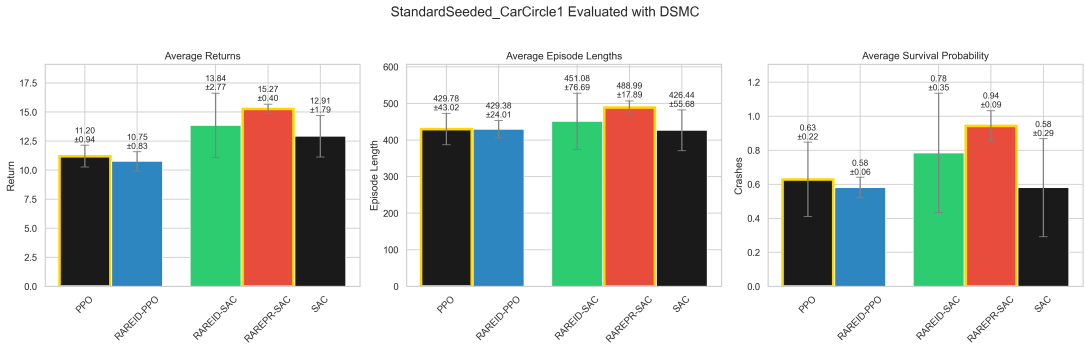
We continue the analysis of the evaluation data with the point agent. As with the car agent, the point agent shows no remarkable performance difference in the easiest difficulty level (Figure 6.4a). Because there are no obstacles the average episode length and survival probability stays the same for both algorithm types.

In difficulty level 1 we observe the same pattern as before for the car agent: RARE, on average, achieves slightly better performance in each metric and reduces the standard deviation greatly (Figure 6.4b).

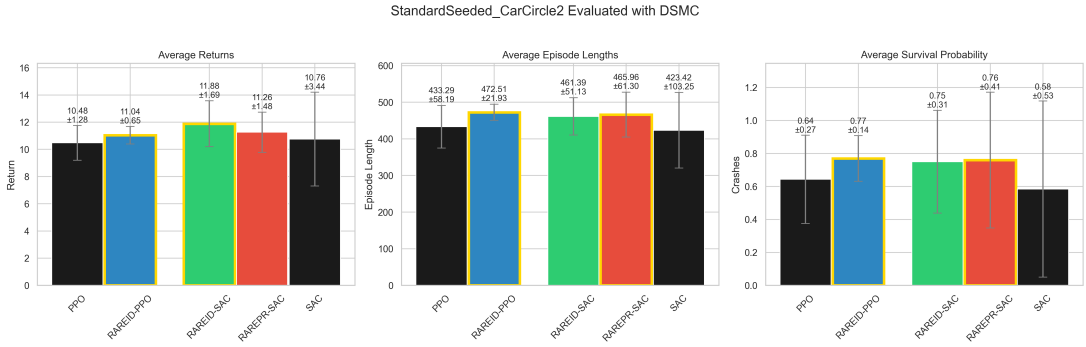
Interestingly, the evaluation data for the point agent in difficulty level 2 presents an



(a) Evaluation mean reward, mean episode length and survival probability for car agent on difficulty level 0 for five seeds evaluated with DSMC.



(b) Evaluation mean reward, mean episode length and survival probability for car agent on difficulty level 1 for five seeds evaluated with DSMC.



(c) Evaluation mean reward, mean episode length and survival probability for car agent on difficulty level 2 for five seeds evaluated with DSMC.

Figure 6.3.: Evaluation results for the car agent in the circle task on difficulty levels 0, 1, and 2 for five seeds evaluated with DSMC.

outlier (Figure 6.4c). Not only does the average RARE policy perform worse compared to the average PPO policy in each metric, but the RARE policies are also exhibiting a much bigger standard deviation than the baseline. This is especially interesting, as RARE seemed to perform much better than the baseline in the training data.

SAC Baseline - Training Results

We now turn our attention to the performance of the SAC baseline, and its RAREID and RAREPR counterparts. The corresponding plots can be found in Figure 6.5 and Figure 6.6.

During training on difficulty level 0, all three algorithms perform essentially identically. Rewards quickly plateau at a high level, episode lengths remain maximal throughout training, and no crashes occur, which is consistent with the lack of hazards in this environment.

For the car agent on difficulty level 1, the training dynamics stay similar: all three metrics for all three algorithms develop very similarly across the training. In particular our initial assumption that RARE would lead to worse performance in all metrics does *not* hold here neither.

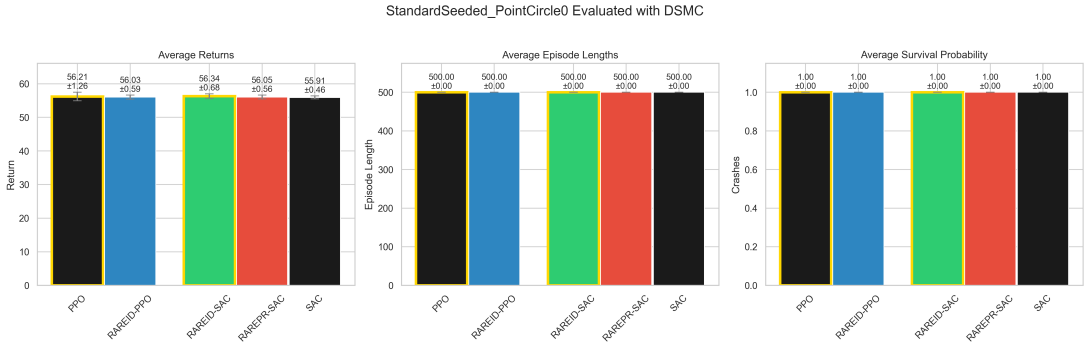
On difficulty level 2 for the car agent, the RARE variants show slightly less crashes on average during training. Besides this, the RARE variants and the baseline achieve similar metrics during training.

Turning to the point agent on difficulty level 1, the SAC baseline achieves slightly higher results in each metric compared to the RARE variants. This difference is most pronounced between the baseline and RAREPR, but RAREID is also outperformed by the baseline. In particular this training graph elicits the expected behavior of RARE during training again.

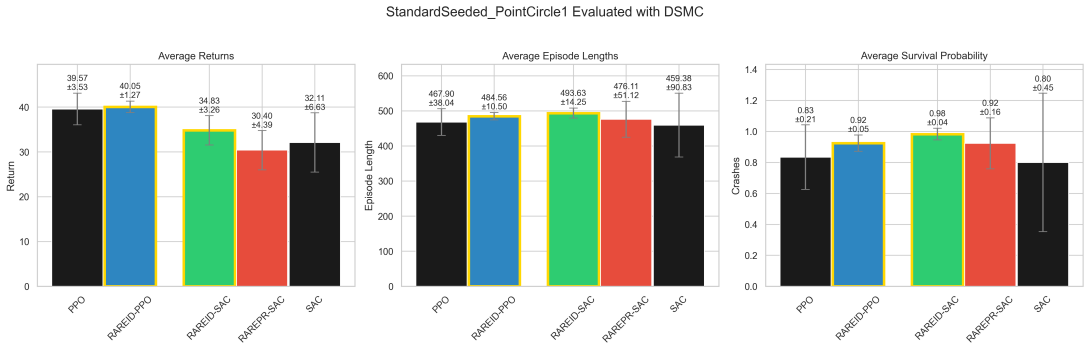
Finally, for the point agent on difficulty level 2, the SAC baseline achieves the highest rewards and maintains longer episode lengths throughout training, while also accumulating less crashes compared to the RARE variants. Both RARE variants show lower average rewards and more variable episode lengths compared to the baseline SAC. In summary our initial assumptions about RARE's training performance hold for the simple point agent, but not for the more complex car agent.

SAC Baseline - Evaluation Results

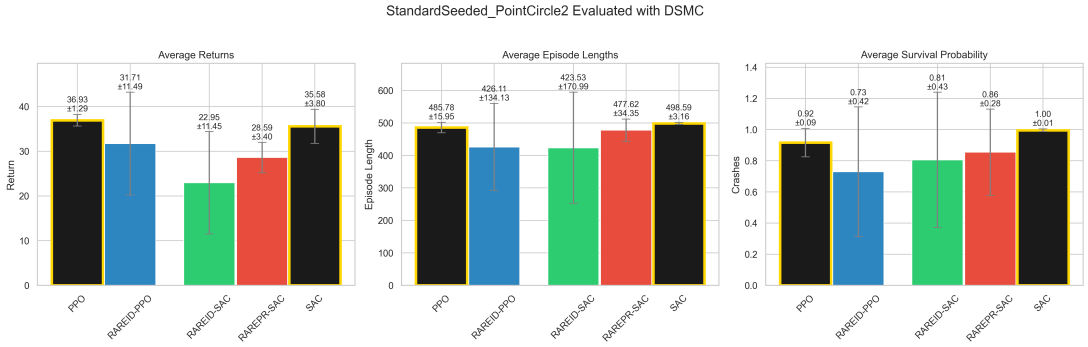
We now analyze the evaluation data for the SAC baselines, our protocol for evaluation is the same as in the PPO baseline.



(a) Evaluation mean reward, mean episode length and survival probability for point agent on difficulty level 0 for five seeds evaluated with DSMC.



(b) Evaluation mean reward, mean episode length and survival probability for point agent on difficulty level 1 for five seeds evaluated with DSMC.



(c) Evaluation mean reward, mean episode length and survival probability for point agent on difficulty level 2 for five seeds evaluated with DSMC.

Figure 6.4.: Evaluation results for the point agent in the circle task on difficulty levels 0, 1, and 2 for five seeds evaluated with DSMC.

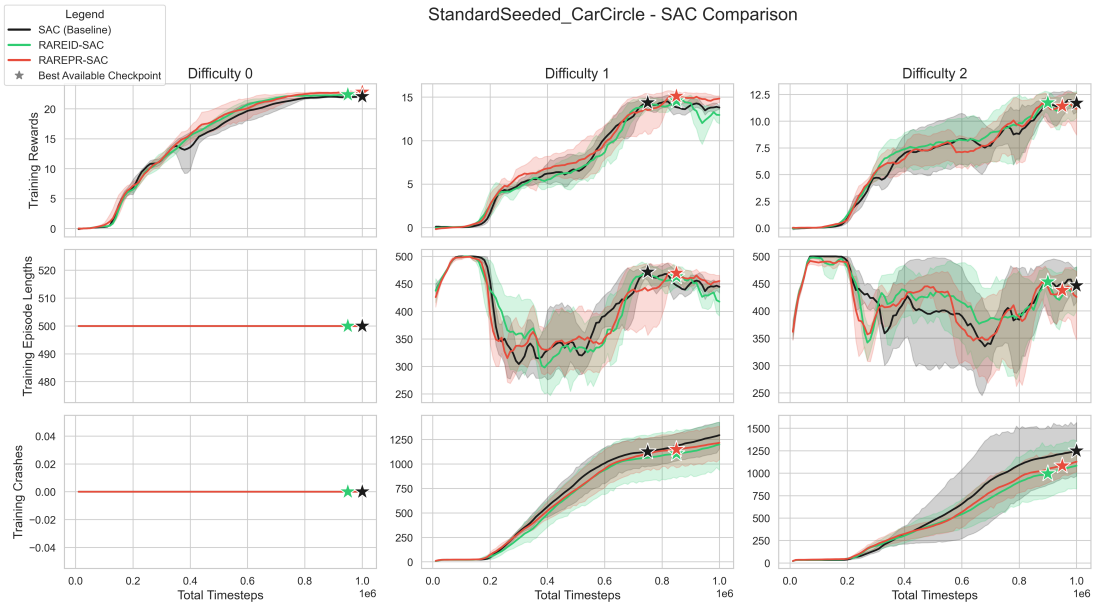


Figure 6.5.: Training mean reward, mean episode length and cumulative crashes for car agent in circle task with SAC variants on all difficulty levels for five seeds.

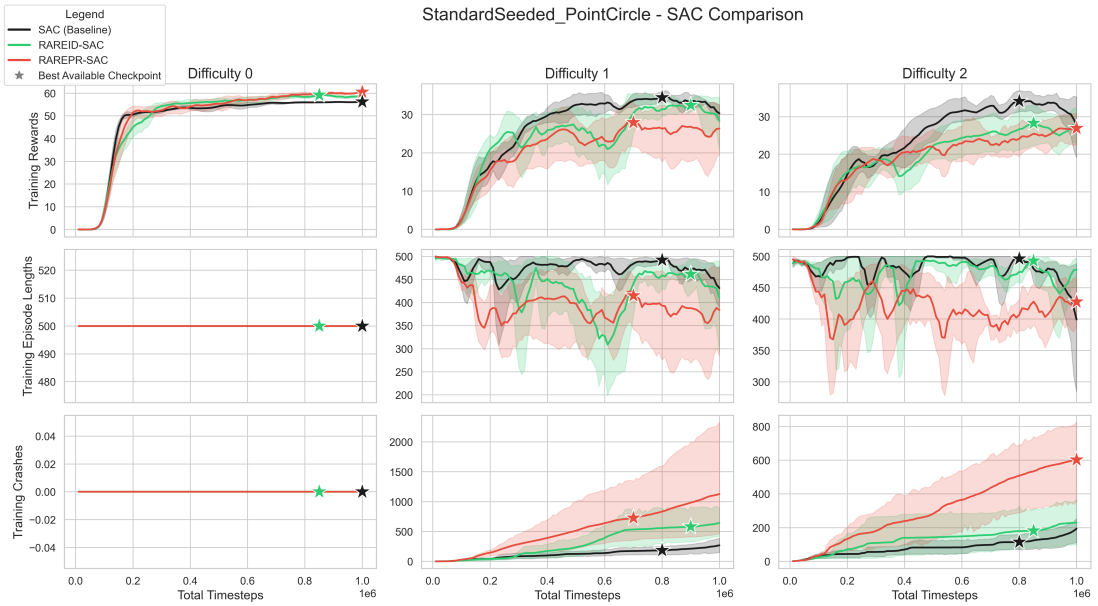


Figure 6.6.: Training mean reward, mean episode length and cumulative crashes for point agent in the circle task with SAC variants on all difficulty levels for five seeds.

On difficulty level 0 for both car (Figure 6.3a) and point (Figure 6.4a) agents, all three algorithms achieve essentially identical and perfect evaluation results. This shows that all algorithms are able to solve the simplest difficulty setting equally well.

For the car agent on difficulty level 1 (Figure 6.3b), the evaluation shows benefits from the RARE methods. Both RAREID-SAC and RAREPR-SAC achieve notably higher average survival probabilities compared to the SAC baseline. RAREPR-SAC also leads in average returns and episode lengths, indicating strong overall performance. RAREID-SAC also improved on average upon the baseline, but the standard deviation is large. This indicates high fluctuation of model performance over various seeds. RAREPR on the other hand reduced the standard deviation. This contrasts with the training phase where all three algorithms performed very similarly.

This advantage for the RARE variants continues on difficulty level 2 for the car agent (Figure 6.3c). Again, both RAREID-SAC and RAREPR-SAC demonstrate better average survival probabilities and episode lengths than the baseline SAC. The average returns are comparable across the three, though RAREID-SAC shows the highest average return. It is important to note, that the standard deviation for the surivial probability is very large for all three algorithms, while for the returns and episode length the RARE variants show a smaller standard deviation than the baseline.

The evaluation results for the point agent are similar. On difficulty level 1 (Figure 6.4b), RAREID-SAC provides the best safety performance with the highest survival probability and longest episode lengths, while simultaneously achieving the smallest standard deviation in all three metrics. RAREPR-SAC also shows better safety metrics than the SAC baseline, and reduces the standard deviation in all metrics. However, the baseline SAC achieves slightly higher average returns than RAREPR, aligning with the training results where the baseline slightly outperforms RARE variants, especially RAREPR-SAC.

Finally, and most surprisingly, on difficulty level 2 for the point agent (Figure 6.4c), the SAC baseline significantly outperforms its RARE counterparts in evaluation. It achieves perfect average survival probability (1.00 mean, 0.01 standard deviation) and the longest average episode length. Both RAREID-SAC and RAREPR-SAC show considerably lower survival probabilities with larger standard deviations and achieve lower average returns and average episode legnths. In particular RAREID-SAC also achieved large standard deviations in the return and episode length metrics. This outcome is particularly noteworthy as it contradicts the general trend and the training results for this specific scenario, where the baseline also leads in rewards/episode length but RAREID-SAC accumulates fewer crashes.

In summary, evaluating the SAC variants shows that RAREID and RAREPR generally enhance safety for the car agent and moderately for the point agent on level 1, at a slight cost to returns for RAREPR-SAC compared to the baseline. However, the standard SAC algorithm demonstrates superior return and safety performance for the point agent

on the most difficult level.

6.1.2. Shaped Reward Structure

The second reward structure we investigate for the circle task is the shaped reward structure,

$$R'_t = \begin{cases} R_t & \textit{if no costs occurred} \\ R_t - 2.5 & \textit{if costs occurred} \end{cases} \tag{II}$$

where R_t is the original reward function. This modification penalizes unsafe behavior by reducing the reward. Intuitively we expect this adjustment to improve the safety metrics of all algorithms by encouraging the policy to avoid unsafe action using the stronger association between unsafe behavior and the end of an episode.

As this modification is only relevant in the presence of cost-inducing objects, we ignore the easiest difficulty level and only investigate policy performance on difficulties 1 and 2.

PPO Baseline - Training Results

We begin with the PPO baseline under the shaped reward structure. The training graphs of the car agent can be found in Figure 6.7 and for the point agent in Figure 6.8. For the car agent on difficulty level 1, the training curves show that RAREID-PPO consistently underperforms compared to the baseline PPO across the safety metrics, while also staying slightly below the rewards. RAREID-PPO accumulates significantly more crashes, which aligns with our initial hypothesis that the restoration process might lead to inferior training performance.

This trend continues on difficulty level 2 for the car agent. RAREID-PPO achieves lower rewards, shorter episode lengths, and accumulates substantially more crashes throughout the training compared to the PPO baseline. The performance gap is more apparent on this difficulty level than the previous one, which again supports the hypothesis for training time performance.

Turning to the point agent with shaped rewards, on difficulty level 1 we observe a similar pattern. The PPO baseline consistently outperforms RAREID-PPO in terms of rewards and episode lengths, while also accumulating fewer crashes during training.

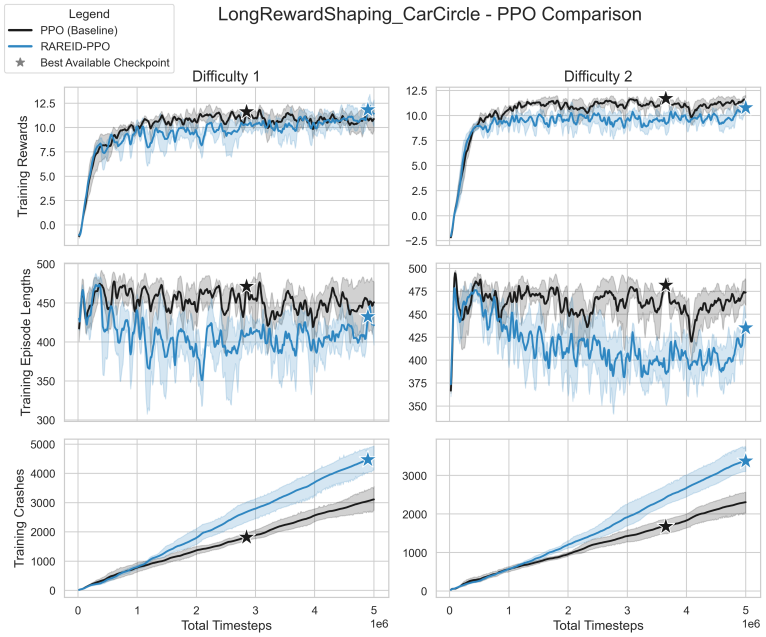


Figure 6.7.: Training mean reward, mean episode length and cumulative crashes for the PPO variants with the car agent in the circle task under the shaped reward structure on all difficulty levels for five seeds.

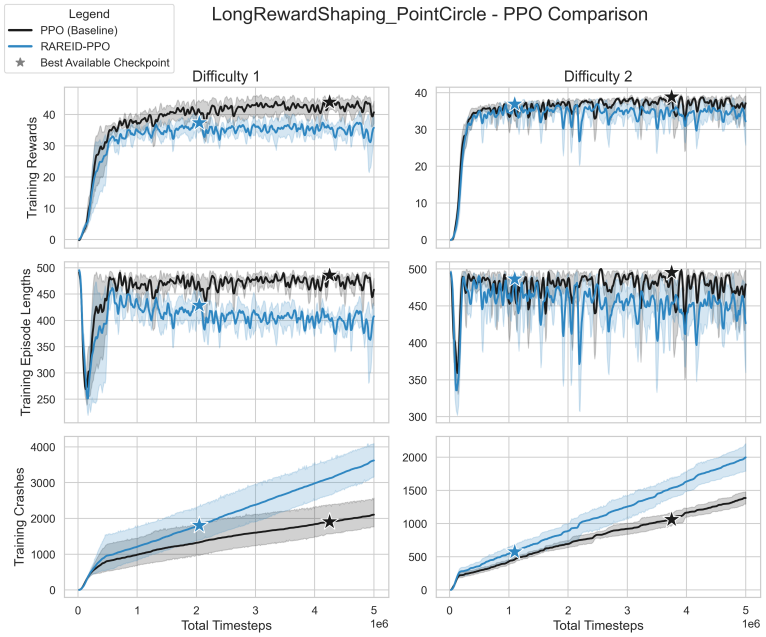


Figure 6.8.: Training mean reward, mean episode length and cumulative crashes for the PPO variants with the point agent in the circle task under the shaped reward structure on all difficulty levels for five seeds.

Similarly, on difficulty level 2 for the point agent, the PPO baseline maintains better performance across all metrics compared to RAREID-PPO throughout the training duration. Overall, under the shaped reward function, the training performance consistently aligns with the hypothesis that RAREID exhibits inferior training time performance compared to the standard baseline.

PPO Baseline - Evaluation Results

We evaluate the PPO agents trained with the shaped reward function using the same strategy as in the original reward function. For the car agent on difficulty level 1 (Figure 6.9a), the evaluation results mirror the training observations. PPO policies on average outperform RAREID-PPO policies in episode length and survival probability, while only slighlty underperform for the reward. In contrast to the original reward structure, RARE-PPO does not reduce the standard deviation of the means compared to PPO under the shaped reward structure.

On difficulty level 2 for the car agent (Figure 6.9b), RAREID-PPO shows a slight advantage in average survival probability and episode length compared to the PPO baseline, achieving higher safety with less variance. However, the baseline PPO achieves slightly better average returns. The safety benefit for RAREID-PPO here is less pronounced than observed under the original reward structure.

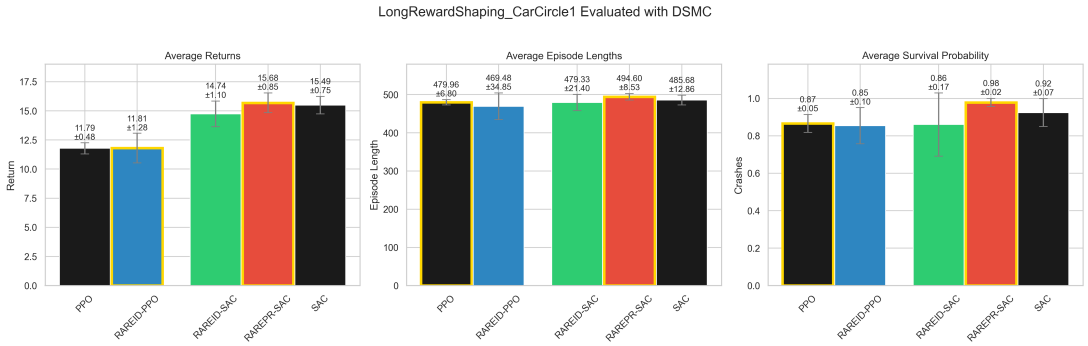
Looking at the point agent evaluations, on difficulty level 1 (Figure 6.10a), the PPO baseline achieves higher average returns, episode lengths and survival probability than RAREID-PPO. Most notably it also performs worse with regards to the standard deviation. Thus, RARE shows no improvement over the baseline in this environment.

Finally, for the point agent on difficulty level 2 (Figure 6.10b), both algorithms achieve similar results in each metric. Unlike under the original reward structure, RAREID-PPO does not perform significantly worse here.

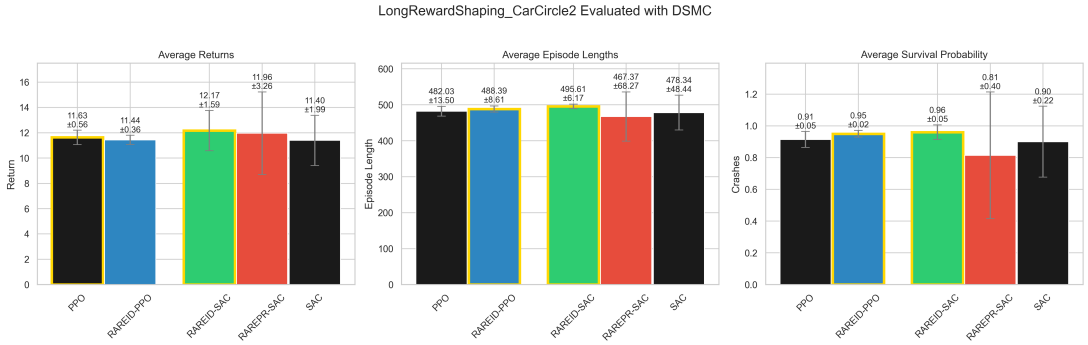
Overall, with the shaped reward function, the training performance of RAREID-PPO is consistently worse than the baseline, which matches our expectation, but the evaluation results are mixed. RAREID sometimes offers safety benefits by reducing the variance of resulting policies (car circle 2), but in other environments decreases safety while increasing variety (point circle 1). In the remaining environments RARE performs on par with the baseline.

SAC Baseline - Training Results

Now we examine the SAC algorithms under the shaped reward structure. We depict the training curves of the car agent in Figure 6.11 and of point agent in Figure 6.12.



(a) Evaluation mean reward, mean episode length and survival probability for car agent with shaped rewards on difficulty level 1 for five seeds evaluated with DSMC.



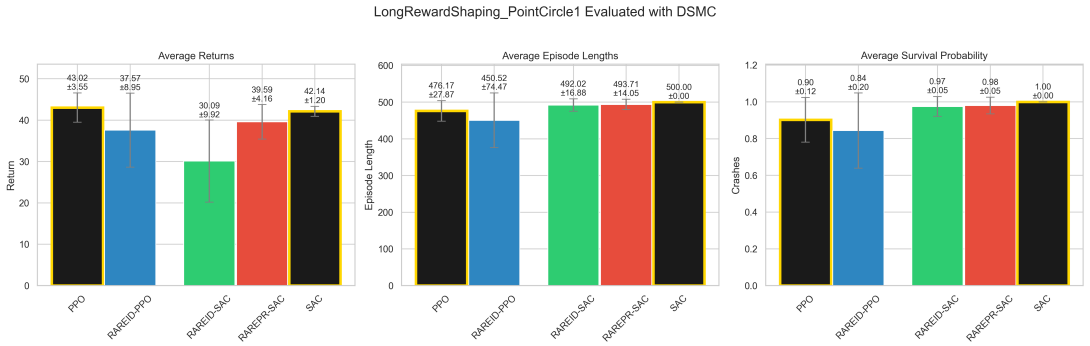
(b) Evaluation mean reward, mean episode length and survival probability for car agent with shaped rewards on difficulty level 2 for five seeds evaluated with DSMC.

Figure 6.9.: Evaluation results for the car agent in the circle task with shaped rewards on difficulty levels 1 and 2 for five seeds evaluated with DSMC.

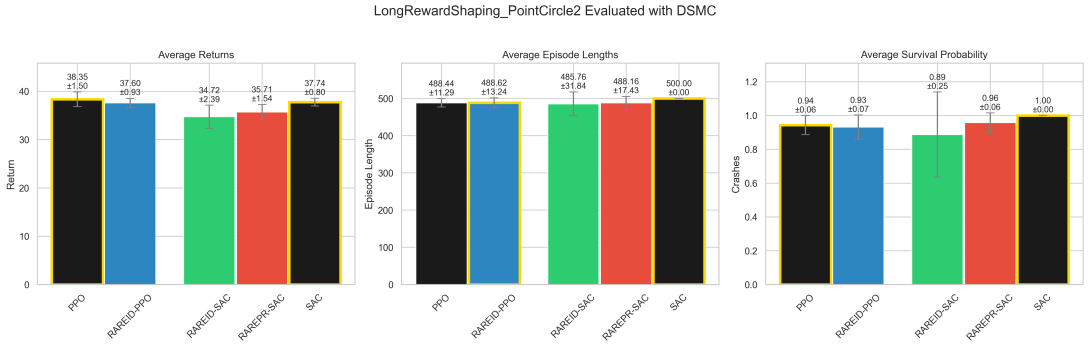
For the car agent on difficulty level 1, the training curves for SAC, RAREID-SAC, and RAREPR-SAC are remarkably similar for rewards, while episode lengths and cumulative crashes show slightly worse average performance for both RARE algorithms. This does not align with our expectations, as we would expect the reward to also be consistently below the baseline during training.

A similar pattern holds for the car agent on difficulty level 2. All three algorithms exhibit very similar performance during training. Rewards track closely, while RAREID and RAREPR are slightly worse in episode length and cumulative crashes. Interestingly, RAREID is closer to the baseline than RAREPR, in particular RAREID shows a very similar development of crashes during training. Again, our hypothesis of worse performance is not confirmed.

For the point agent on difficulty level 1, the training dynamics do not differ. The rewards are comparable for all three algorithms, and while RAREID and the baseline



(a) Evaluation mean reward, mean episode length and survival probability for point agent with shaped rewards on difficulty level 1 for five seeds evaluated with DSMC.



(b) Evaluation mean reward, mean episode length and survival probability for point agent with shaped rewards on difficulty level 2 for five seeds evaluated with DSMC.

Figure 6.10.: Evaluation results for the point agent in the circle task with shaped rewards on difficulty levels 1 and 2 for five seeds evaluated with DSMC.

are close in performance for episode length and cumulative crashes, RAREPR performs worse in the two safety metrics, especially consistently worse than the baseline.

On difficulty level 2 for the point agent, the SAC baseline achieves slightly higher rewards, longer episode lengths and lower crashes than the RARE variants. This fits our initial hypothesis again.

In summary, the training performance under the shaped reward function for SAC variants does not show a consistent trend relative to the baseline. For the car agent the performance is very similar, while for the point agent sometimes the baseline performs better, particularly compared to RAREPR-SAC. Our assumption that RARE would obtain lower rewards, shorter episodes and more crashes during training does generally not hold.

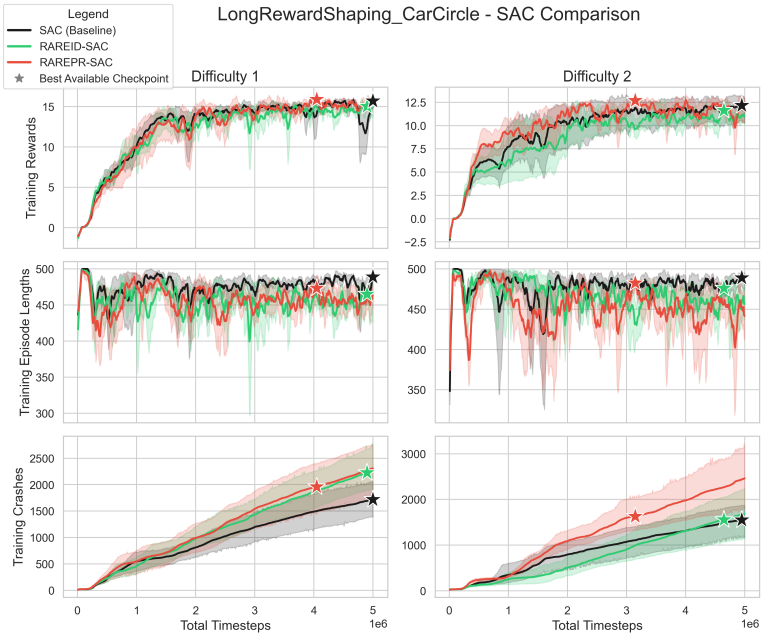


Figure 6.11.: Training mean reward, mean episode length and cumulative crashes for the car agent in the circle task with SAC variants and shaped rewards on all difficulty levels for five seeds.

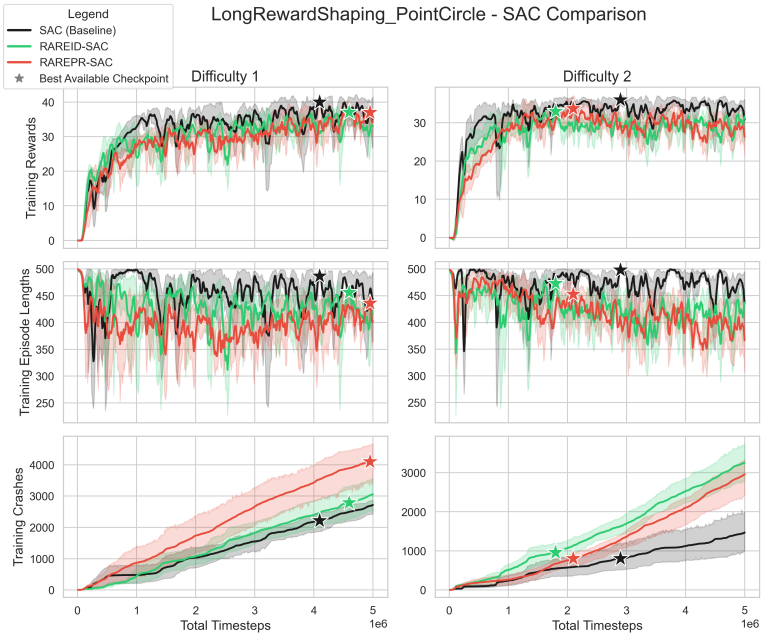


Figure 6.12.: Training mean reward, mean episode length and cumulative crashes for the point agent in the circle task with SAC variants and shaped rewards on all difficulty levels for five seeds.

SAC Baseline - Evaluation Results

Finally, we evaluate the SAC agents trained with the shaped reward function.

For the car agent on difficulty level 1 (Figure 6.9a), the evaluation results show a clear safety benefit for the RAREPR variant. RAREPR achieves the best survival probability and the best episode length while reducing the variance. In particular it does so without a reduction in reward, compared to the SAC baseline. RAREID on the other hand performs similar to the baseline in regards to return and episode lengths, but obtains a lower survival probability with a higher standard deviation.

On difficulty level 2 for the car agent (Figure 6.9b), RAREID-SAC stands out with the highest average survival probability, the longest average episode length but also the highest return. All of those with a smaller standard deviation than the baseline. RAREPR-SAC performs worse than the baseline in terms of survival probability, where it also shows a larger variance. But it achieves similar episode length and reward to the baseline. Notably, it also exhibits a larger standard deviation than the baseline in the returns.

Turning to the point agent evaluation on difficulty level 1 (Figure 6.10a), the safety metrics are very close to optimal for all three algorithms, however, the baseline achieves optimal results while the RARE variants are imperfect. Return-wise SAC also performs best, closely followed by RAREPR, although RAREPR has a greater standard deviation. RAREID greatly lacks behind in this metric with a smaller mean and larger variance altogether.

For the point agent on difficulty level 2 (Figure 6.10b), standard SAC again achieves perfect survival probability and maximal episode length, clearly showing the best safety performance. While both RARE variants perform close to perfect in terms of episode length, the average survival probability of RAREID lacks behind RAREPR and SAC with a larger standard deviation. SAC also achieves higher average returns compared to RAREID and RAREPR, although the difference is less pronounced than in the previous difficulty level.

In summary, the evaluation of SAC variants under the shaped reward function shows mixed results. RARE methods improve our safety metrics for the car agent compared to the baseline. However, for the point agent, they did not manage to improve upon the baseline performance and in the case of RAREID even *degrade* the safety performance.

6.2. Goal Task

In the goal task we trained one algorithm for each reward structure defined in Chapter 4. We examine the original reward structure, shaped reward structure and sparse reward with goal reaching rewards 1, 5 and 10. Under the original reward structure we investigate each difficulty level, while we constrain ourselves in the rest to the difficulties 1 and 2, which introduce hazards in the environment. We do this to focus the analysis on the safety aspects while providing a trivial environment without hazards to show RARE's principal ability to solve the goal task.

Similar as before we present each PPO and RAREID-PPO, and SAC, RAREID-SAC and RAREPR-SAC group together and analyze their training performance in the return, episode length and cumulative crashes, and the DSMC high-confidence mean of return, episode length and survival probability for the best policy encountered during training.

However, in contrast to the circle task, we were unable to train multiple seeds per algorithm due to time constraints. Therefore we can reason about the best performance we can observe, but not about the average performance of RARE.

6.2.1. Original Reward Structure

The original reward structure of the goal task is two fold. The agent is rewarded for reducing the distance to the goal according to

$$R_t^{distance} = D_{last} - D_{now} \tag{III}$$

where D is the distance before and after the last action respectively. If the agent arrives in the goal area, the goal reward

$$R_{goal} = 1 \tag{IV}$$

is added to the overall reward. This further incentives the agent to pursue the goal area.

PPO Baseline - Training Results

We begin by analyzing the training performance of PPO and RAREID-PPO in the goal task under the original reward structure. The curves can be seen in Figure 6.13

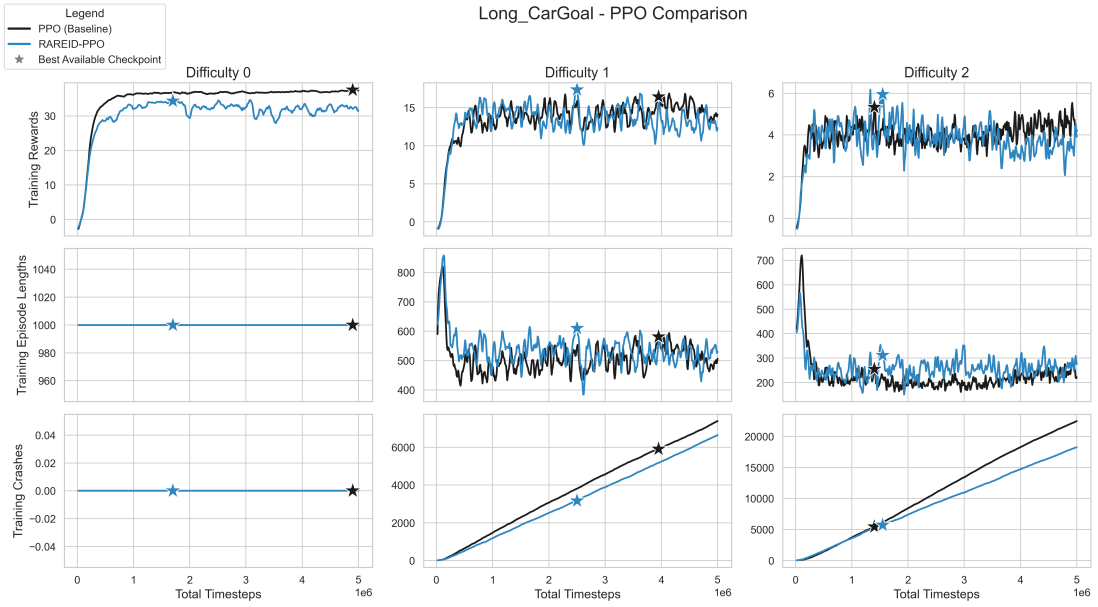


Figure 6.13.: Training mean reward, mean episode length and cumulative crashes for the car agent in the goal task with PPO variants and original rewards on all difficulty levels.

and Figure 6.14. On difficulty level 0 both algorithms demonstrate similar training curves for both the car and point agents. While the curves are effectively the same for the point agent, for the car agent our hypothesis that RARE performs worse than the baseline is observable. The rewards rapidly converge to their maximum, and the safety metrics are optimal, as we expect in this safe environment. Interestingly, the car agent, which we assume to be harder to learn due to its action space, achieves *better* rewards than the point agent.

On difficulty level 1, the performance diverges. For the car agent neither of the algorithms consistently outperforms the other, and RAREID-PPO achieves notably less accumulated crashes during training than the baseline. Therefore we cannot confirm our hypothesis that the training performance of RARE are below those of the baseline. This hypothesis *does* show, however, for the point agent: most of the time, RARE performs below the baseline in all three metrics, and only towards the end of training approaches the baseline performance. While the car agent again performs slightly better, the difference is not as pronounced as in the previous difficulty setting.

On difficutly level 2 the curves of the PPO baseline and RAREID-PPO do not differ significantly. For both agents RAREID achieves on average slightly longer episodes and slightly less crashes, while the rewards develop similarly.

In summary, during training under the original reward structure, PPO and RAREID-PPO perform identically in the absence of hazards. In hazardous environments, the

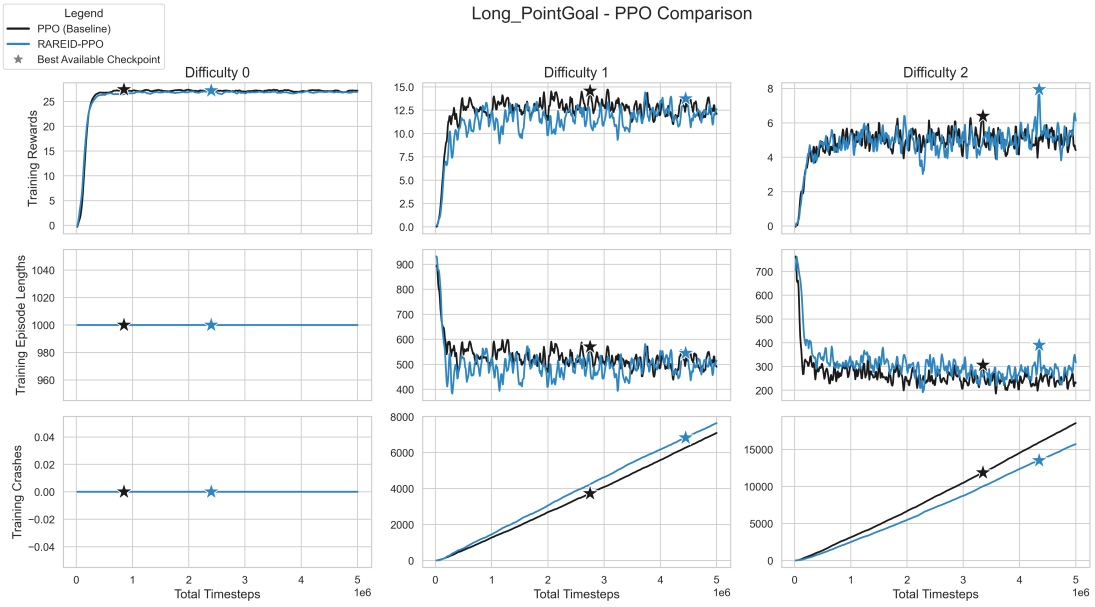


Figure 6.14.: Training mean reward, mean episode length and cumulative crashes for the car agent in the goal task with PPO variants and original rewards on all difficulty levels.

PPO baseline generally shows slightly better or comparable training rewards, while the episode lengths show no preference. In three of the four hazard tasks, PPO actually accumulates *more* crashes than RAREID-PPO. This is surprising as the restoration process restores critical states more often.

PPO Baseline - Evaluation Results

Following the training phase, we evaluate the best-performing model checkpoint for each algorithm using DSMC. On difficulty level 0 (Figure 6.15a and Figure 6.16a), for the car agent PPO achieves a better average reward than RAREID. However, in the point agent scenario both algorithms achieve very similar rewards. The performance gap we observed in the training data, is also present in the DSMC evaluation.

On difficulties 1 and 2 (Figure 6.15b, Figure 6.16b Figure 6.15c and Figure 6.16c), and for both agents, RARE performs slightly better than PPO in all metrics. We must stress however, that the survival probabilities in these tasks are generally low and that our admissible error in the evaluation is 10%. Therefore, while these advantages look promising, it is unclear if they remain under stricter guarantees or multiple seeds.

In conclusion, the evaluation results indicate a benefit of using RAREID-PPO in the goal task with hazards under the original reward structure. Despite showing similar performance during training compared to the baseline, RAREID-PPO leads to policies

with higher returns and improved safety at evaluation time. Nonetheless, this is merely a hint at performance improvements, as we only evaluate on seed for each algorithm.

SAC Baseline - Training Results

We now examine the training performance of the SAC baseline compared to its RAREID-SAC and RAREPR-SAC counterparts in the goal task under the original reward structure. We present the corresponding graphs in Figure 6.17 and Figure 6.18. As observed with PPO, on difficulty level 0, all three algorithms exhibit identical performance for both car and point agents. The rewards are noticeably higher for the car agent again, too.

On difficulty levels 1 and 2 the observations remain similar to before. Our hypothesis about the training performance is only observable in the RAREPR algorithm on difficulty level 2 for the car agent and difficulty level 1 for the point agent. For RAREID the performance are similiar to the baseline. We cannot observe the pattern that the RARE variants achieve less crashes than the baseline during training for the SAC algorithms.

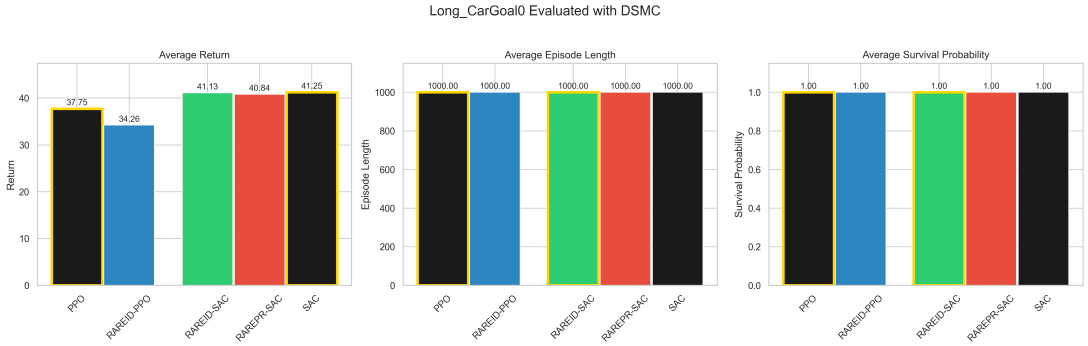
Overall, the training results for SAC variants under the original reward structure show that the baseline SAC generally performs similar or better during training.

SAC Baseline - Evaluation Results

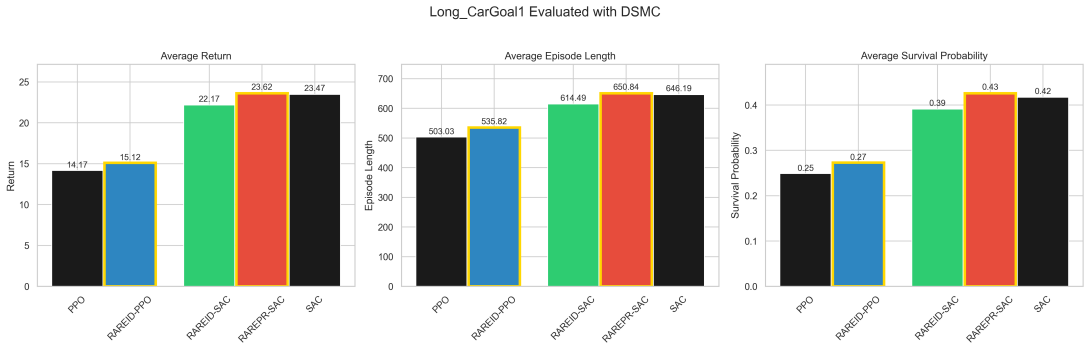
Finally, we assess the evaluation performance of the SAC, RAREID-SAC, and RAREPR-SAC algorithms using DSMC on the best model checkpoints. On difficulty level 0 (Figure 6.15a and Figure 6.16a), the average returns are high and very close, with RAREID-SAC achieving the highest return for the car agent and the baseline achieving the highest for the point agent (29.20). The safety metrics are maximal, as no hazards are present in level 0.

On difficulty level 1, the results become more varied. While for the car agent (Figure 6.15b), all results are very similar again: RAREPR slightly outperforms the baseline in all metrics, RAREID on the other hand performs slightly worse than the baseline. For the point agent (Figure 6.16b), the baseline outperforms both RARE algorithms: RAREPR still performs close to the baseline, but RAREID shows noticably worse performance in all three metrics.

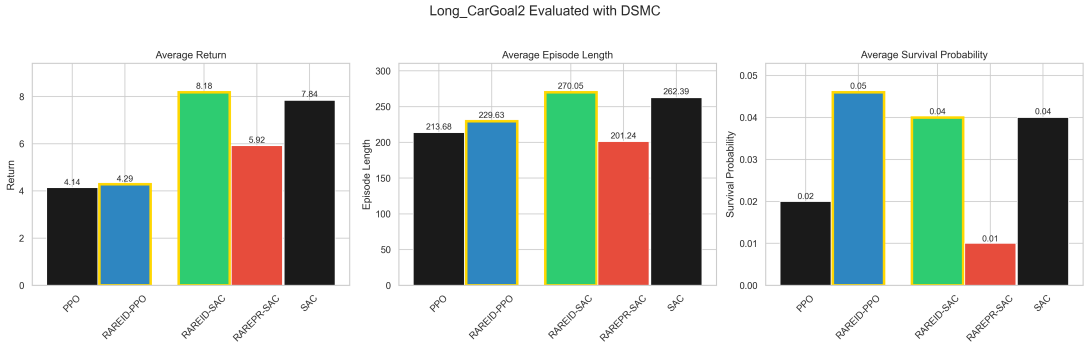
On difficulty level 2, the performance pattern diverges significantly between the agent types. For the car agent (Figure 6.15c), RAREID-SAC achieves the highest average return, average episode length and average survival probability. The baseline SAC performs second best with only a small performance gap, and in the case of survival probability equally well. RAREPR-SAC in contrast is significantly worse in all metrics.



(a) Evaluation mean reward, mean episode length and survival probability for car agent with original rewards on difficulty level 0 evaluated with DSMC.

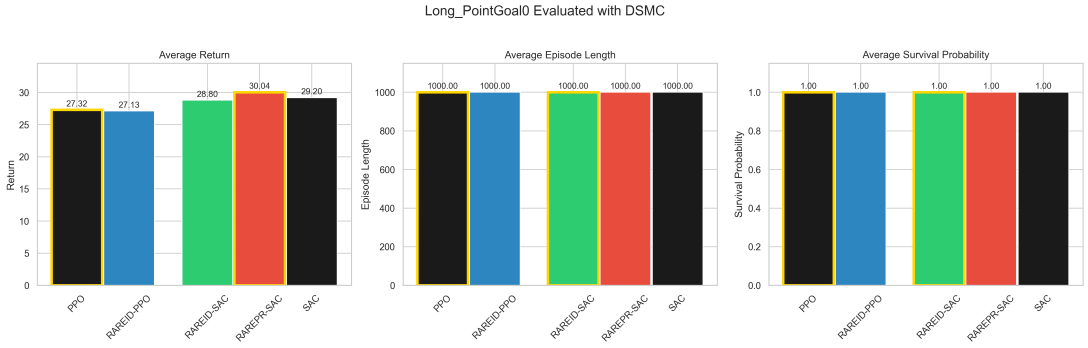


(b) Evaluation mean reward, mean episode length and survival probability for car agent with original rewards on difficulty level 1 evaluated with DSMC.

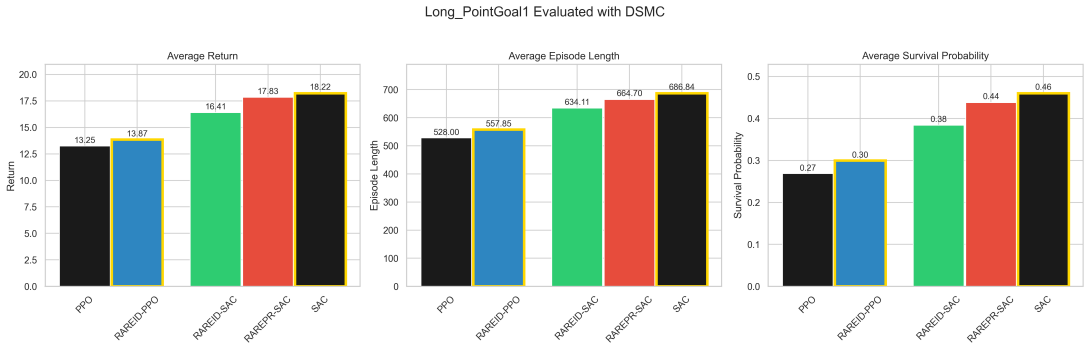


(c) Evaluation mean reward, mean episode length and survival probability for car agent with original rewards on difficulty level 2 evaluated with DSMC.

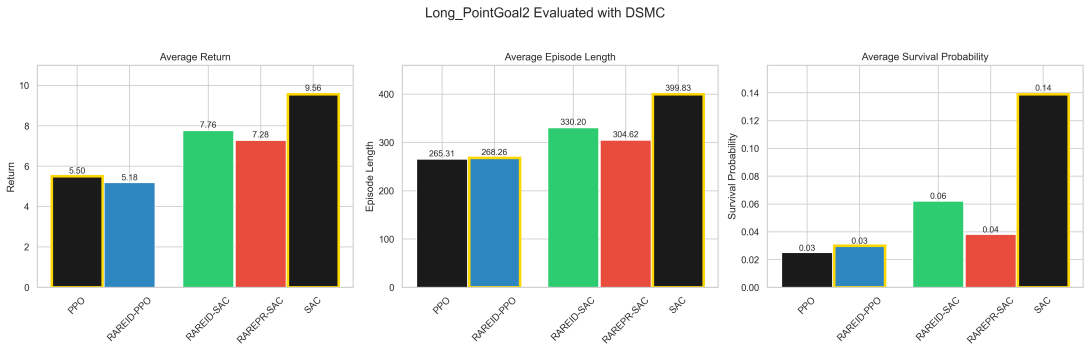
Figure 6.15.: Evaluation results for the car agent in the goal task with original rewards on difficulty levels 0, 1, and 2 evaluated with DSMC.



(a) Evaluation mean reward, mean episode length and survival probability for point agent with original rewards on difficulty level 0 evaluated with DSMC.



(b) Evaluation mean reward, mean episode length and survival probability for point agent with original rewards on difficulty level 1 evaluated with DSMC.



(c) Evaluation mean reward, mean episode length and survival probability for point agent with original rewards on difficulty level 2 evaluated with DSMC.

Figure 6.16.: Evaluation results for the point agent in the goal task with original rewards on difficulty levels 0, 1, and 2 evaluated with DSMC.

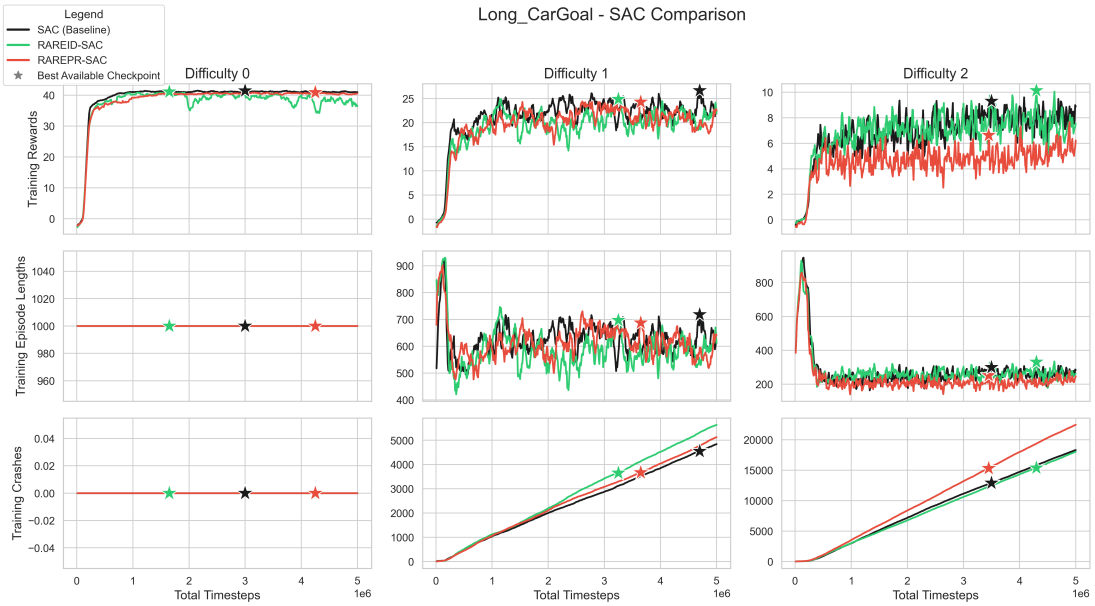


Figure 6.17.: Training mean reward, mean episode length and cumulative crashes for the car agent in the goal task with SAC variants and original rewards on all difficulty levels.

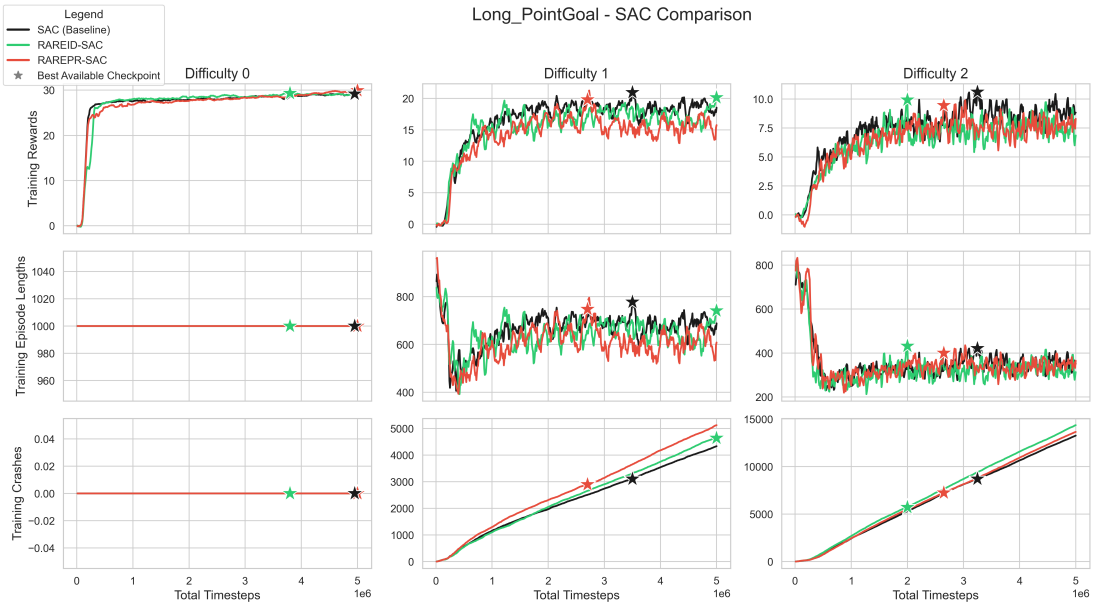


Figure 6.18.: Training mean reward, mean episode length and cumulative crashes for the point agent in the goal task with SAC variants and original rewards on all difficulty levels.

However, for the point agent (Figure 6.16c), the SAC baseline outperforms both RARE variants across all metrics significantly, achieving the highest average return, average episode length and average survival probability.

In conclusion, the evaluation data presents mixed results. For the more complex car agent, RARE methods show only small improvements to the baseline SAC. Either RAREID or RAREPR are able to match or marginally improve the baseline, however, on difficulty 2 we observe that RAREPR underperforms. Conversely, for the simpler point agent, the standard SAC baseline achieves superior performance. While both RARE variants perform similar to each other and not far from the baseline, they cannot surpass its performance. Especially in the hardest difficulty level for the point agent the difference between RARE and the baseline in all three metrics is significant.

6.2.2. Shaped Reward Structure

Under the shaped reward structure we investigate whether RARE performs better compared to the baseline if direct feedback signals about unsafe actions are available. For this we penalize actions which lead to crashes by 2.5, while keeping the original reward structure intact. As this only changes the reward structure in the presence of crashes, we investigate just difficulty levels 1 and 2. As before we first explain the results for the PPO based algorithms, and then those of the SAC based algorithms.

PPO Baseline - Training Results

We first examine the training dynamics of PPO and RAREID-PPO under the shaped reward structure. We show the plots in Figure 6.19 and Figure 6.20. For the car agent on difficulty level 1, the reward stays below the baseline for a majority of the training duration. However, in terms of episode length and accumulated crashes both develop very similar. On difficulty level 2, the performance is again very close, with no discernible differences between both algorithms.

Turning to the point agent, on difficulty level 1, we can again not detect any notable differences between both algorithms' training curves. On difficulty level 2, both algorithms start similarly, but towards the middle of the training the baseline is consistently achieving higher rewards and episode lengths, while RARE starts accumulating more crashes.

In summary, the training results under the shaped reward structure are inconsistent regarding our hypothesis. Besides difficulty level 2 of the point agent, the algorithms match each other in their performance. A clear benefit of the baseline during training due to less critical initial states is not observable.

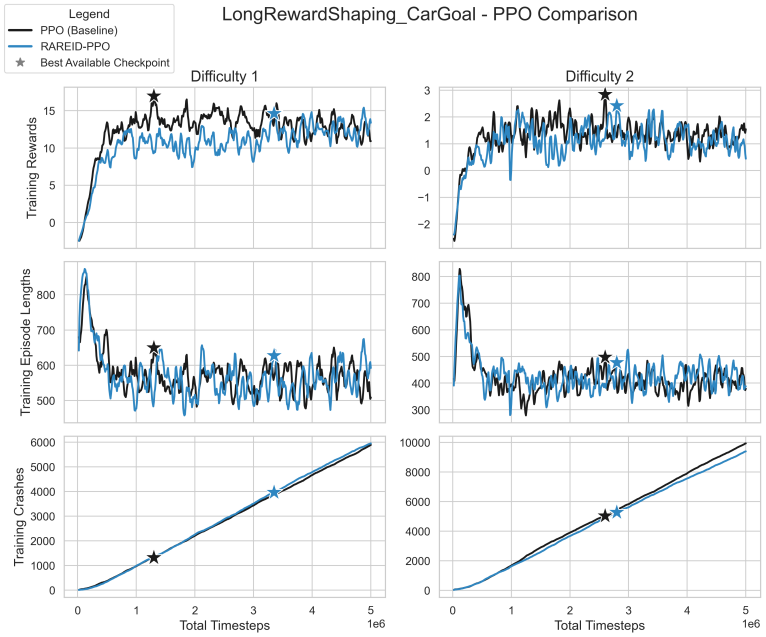


Figure 6.19.: Training mean reward, mean episode length and cumulative crashes for the car agent in the goal task with PPO variants and shaped rewards on all difficulty levels.

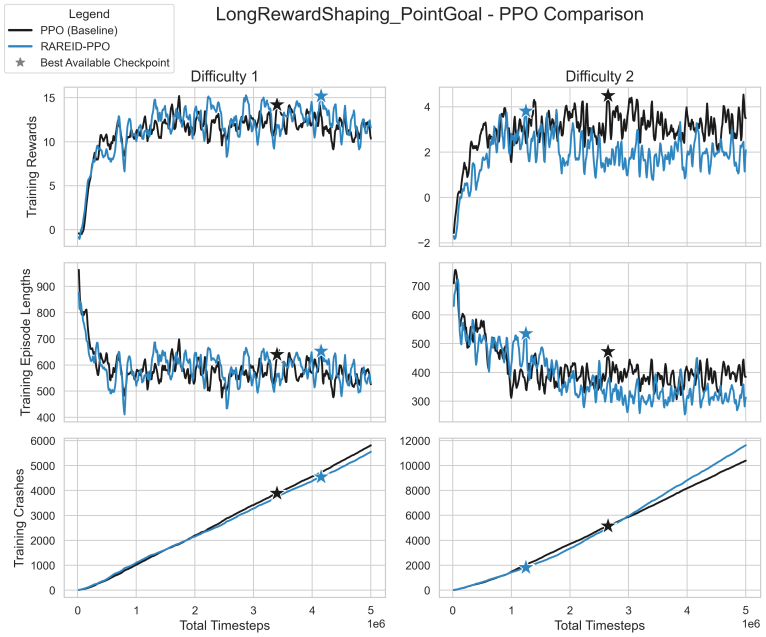


Figure 6.20.: Training mean reward, mean episode length and cumulative crashes for the point agent in the goal task with PPO variants and shaped rewards on all difficulty levels.

PPO Baseline - Evaluation Results

Evaluating the best model checkpoints with DSMC provides further insights. For the car agent on difficulty level 1 (Figure 6.21a) RARE achieves a comparable reward to the baseline, but outperforms it in episode length and survival probability. On level 2 (Figure 6.21b), both algorithms achieve the same survival probability, and very similar return and episode lengths.

For the point agent on difficulty level 1 (Figure 6.22a), the baseline PPO outperforms RAREID-PPO in all three metrics, although the return and survival probability are very similar. On difficulty level 2 (Figure 6.22b), the RARE algorithm managed to increase the mean survival probability and episode length, at the cost of a lower, but comparable return.

Overall, the evaluation results for PPO and RAREID-PPO under the shaped reward function show similar performance, and in the case of the point agent on difficulty 2 a safety tradeoff.

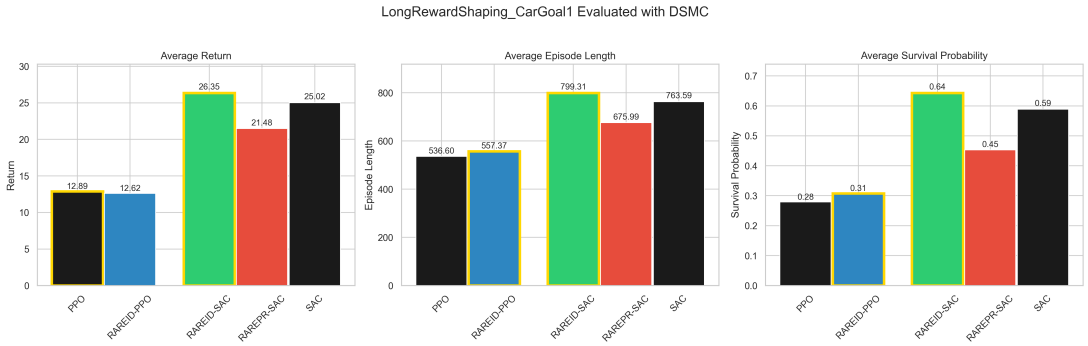
SAC Baseline - Training Results

Now we analyze SAC, RAREID-SAC and RAREPR-SAC under the shaped reward structure during training. The plots are presented in Figure 6.23 and Figure 6.24. For the car agent on difficulty level 1, all three algorithms develop similarly for the return and the episode length. While RAREID accumulates more crashes than the baseline, RAREPR accumulates less.

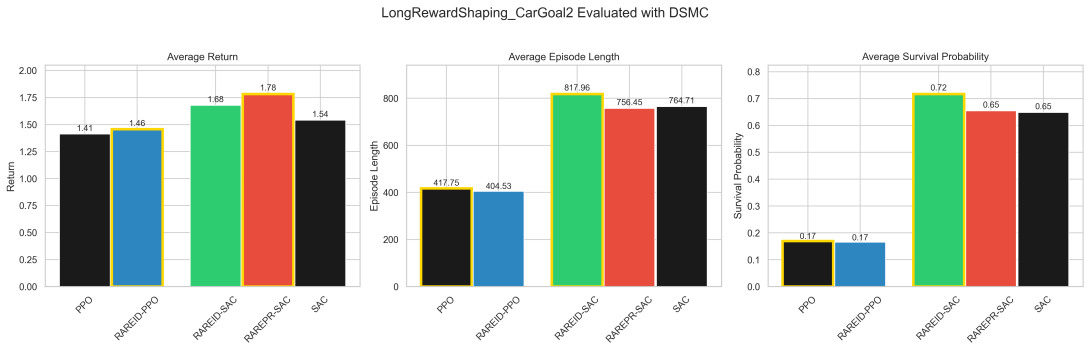
On difficulty level 2, the curves exhibit a slight advantage for the RARE variants with regards to the rewards. The curves of the episode lengths for RAREID and SAC are similar, but RAREPR performs consistently below the baseline in this metric. Concerning the crashes RAREID and SAC again develop equally, while RAREPR accumulates more crashes during training.

For the point agent on difficulty level 1, the baseline SAC clearly outperforms the RAREPR variant in terms of rewards, while RAREID shows very close rewards to the baseline. All three algorithm develop equally in the episode length metric. RAREID-SAC accumulates more crashes than the baseline, while RAREPR-SAC mirrors the baseline performance.

On difficulty level 2, the baseline SAC clearly outperforms RARE in the first half of training on the reward, but then decreases such that in the end all three algorithms perform similarly. Notably all policies yield rewards near 0, therefore we can assume that the algorithms do not manage to learn a acceptable policy. In terms of episode length and accumulated crashes both RARE policies perform better than the baseline.

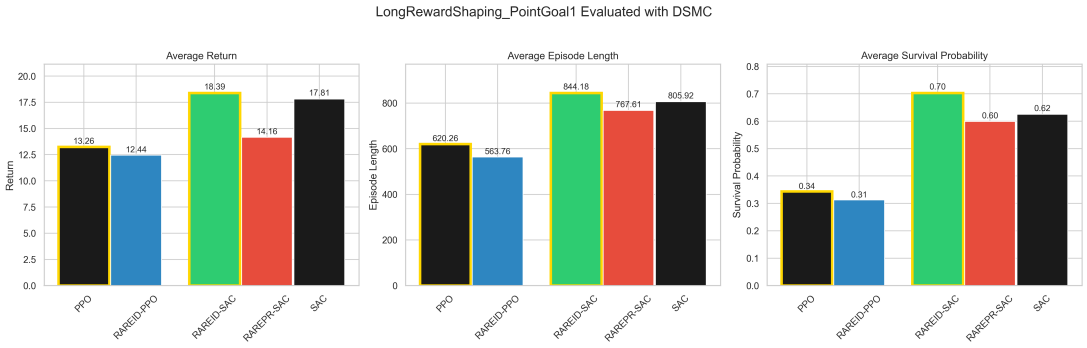


(a) Evaluation mean reward, mean episode length and survival probability for car agent with shaped rewards on difficulty level 1 evaluated with DSMC.

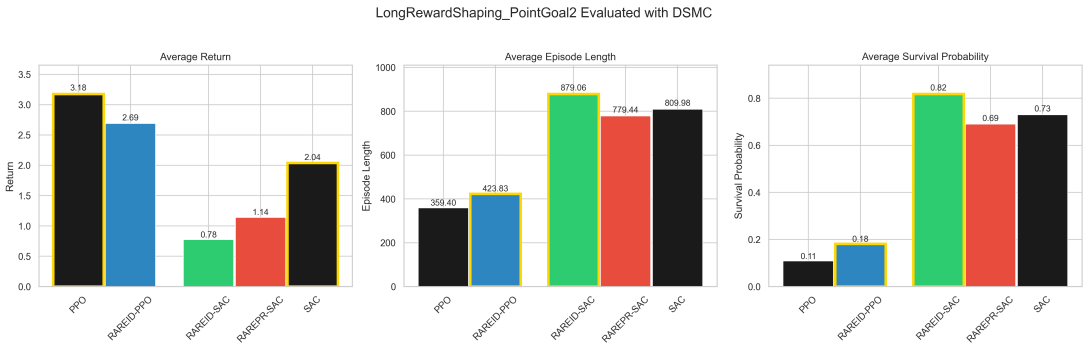


(b) Evaluation mean reward, mean episode length and survival probability for car agent with shaped rewards on difficulty level 2 evaluated with DSMC.

Figure 6.21.: Evaluation results for the car agent in the goal task with shaped rewards on difficulty levels 1 and 2 evaluated with DSMC.



(a) Evaluation mean reward, mean episode length and survival probability for point agent with shaped rewards on difficulty level 1 evaluated with DSMC.



(b) Evaluation mean reward, mean episode length and survival probability for point agent with shaped rewards on difficulty level 2 evaluated with DSMC.

Figure 6.22.: Evaluation results for the point agent in the goal task with shaped rewards on difficulty levels 1 and 2 evaluated with DSMC.

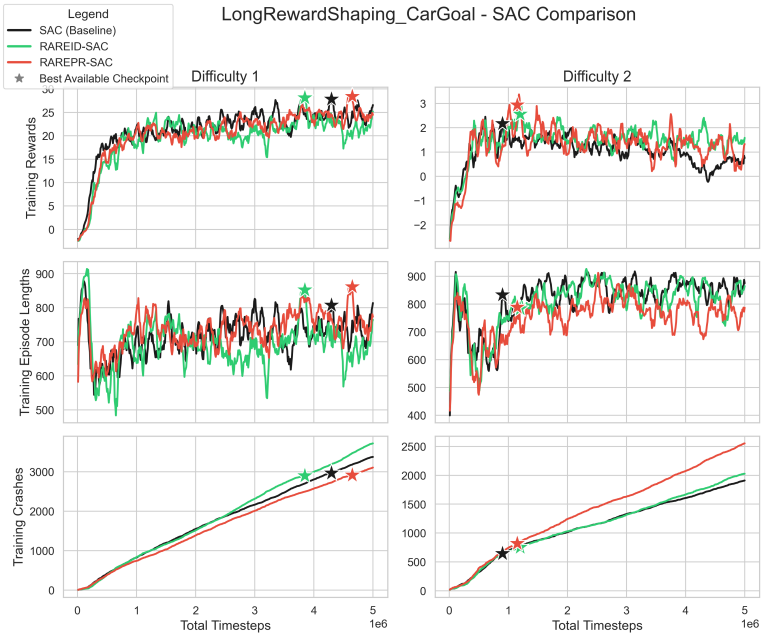


Figure 6.23.: Training mean reward, mean episode length and cumulative crashes for the car agent in the goal task with SAC variants and shaped rewards on all difficulty levels.

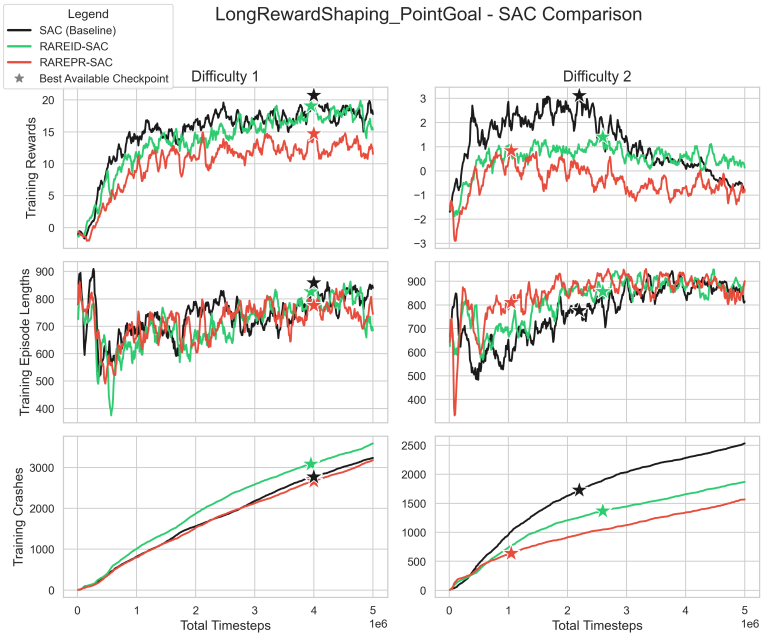


Figure 6.24.: Training mean reward, mean episode length and cumulative crashes for the point agent in the goal task with SAC variants and shaped rewards on all difficulty levels.

In summary the results are mixed. While in the point agent the rewards are consistently less in the RARE policies than in the SAC policy, the other metrics do not confirm our initial hypothesis that RARE would perform worse during training. In particular it stands out that all three algorithms achieve very low rewards on difficulty 2.

SAC Baseline - Evaluation Results

Evaluating the best checkpoints for the SAC algorithms reveals distinct performance differences.

For the car agent on difficulty level 1 (Figure 6.21a), RAREID-SAC achieves the highest average return, average episode length and average survival probability, clearly outperforming the baseline. But RAREPR-SAC performs worst in all three metrics in this scenario. Nonetheless, the results are still in a an acceptable range. On difficulty level 2 (Figure 6.21b), RAREID-SAC again stands out, achieving the highest performance in the safety metrics and achieving a comparable reward to the baseline. RAREPR outperforms RAREID and the baseline with respect to the reward metric, and matches the baseline's performance in the safety metrics.

For the point agent on difficulty level 1 (Figure 6.22a), RAREID-SAC again performs best across all metrics. RAREPR-SAC, however, again falls behind the performance of the baseline, in particular in the reward metric. Although, episode length and survival probability are very close to the baseline, the reward has a larger, but still acceptable gap. On difficulty level 2 (Figure 6.22b), RAREID performs best in the safety metrics while RAREPR achieves results close to the baseline. In the return metrics, however, RAREID performs worst, followed by RAREPR, which both fall behind the SAC baseline. We want to point out that the scaling of our bar chart can suggest a larger difference in the reward than there actually is, the difference of around 1 is not very large.

While these results might suggest that RARE improves the safety at the price of reward, it is also important to emphasize that the rewards for all agents and algorithms on difficulty level 2 are very low. This means, that the agent could just remain still in the majority of complex hazard layouts and increase it's safety performance without much loss of reward. Nonetheless, the results suggest a safety benefit for RAREID-SAC over the baseline. RAREPR-SAC does not show such a benefit, as it underperforms in all but one experiment.

In conclusion, the evaluation results under the shaped reward structure favor RAREID-SAC over both the baseline SAC and RAREPR-SAC across both agent types and difficulty levels in these single-run experiments. RAREID-SAC consistently achieves the best results in nearly all evaluated metrics, suggesting that the combination of SAC, the shaped reward's direct penalty, and RAREID's exploration mechanism can lead to superior policies, at least based on the best checkpoint found during these

specific training runs. RAREPR-SAC, however, did not show benefits compared to the baseline.

6.2.3. Sparse Reward Structure

RARE was tested in the original paper [9] in the Ractrack [28] and MiniGrid [29] environments. Both of these environments do not only utilize a discrete action space, but also a sparse reward structure. In this section we examine how RARE performs in a continuous domain with sparse rewards. For this we set the distance reward $R_t^{distance}$ to 0. While we trained models with goal reward R_{goal} set to 1, 5 and 10, we only discuss $R_{goal} = 5$ here. The reason for this is, that with a reward of 1 both difficulty 2 experiments achieve on average a negative reward for all algorithms. This indicates that the policies failed to learn the desired behavior. Consequently, high safety metrics are deceiving as they reflect the inaction of the agent. With $R_{goal} = 5$ and $R_{goal} = 10$ however, all environments show similar signs of learning. As the results are similar and $R_{goal} = 5$ achieves slightly better values for the safety metrics, we only discuss this one in detail, and share the relevant graphs for $R_{goal} = 1$ and $R_{goal} = 10$ in Appendix B for completeness. As before, we only examine difficulty levels 1 and 2, which include hazards.

PPO Baseline - Training Results

We begin by analyzing the training performance of PPO and RAREID-PPO under the sparse reward structure. As before be present the grouped training curves by difficulty in Figure 6.25 and Figure 6.26

For the car agent on difficulty level 1, RAREID-PPO initially shows similar curves in all metrics compared to the baseline but quickly diverges. Then PPO outperforms RARE consistently.

On difficulty level 2 for the car agent, RAREID consistently outperforms the baseline in the reward metric, but achieves significantly less average episode lengths and accumulates more crashes. Notably the baseline generates a reward close to 0 during the whole training. We can therefore assume that the training did not converge to a sensible policy.

Turning to the point agent on difficulty level 1, both algorithms develop very similar over the training.

On difficulty level 2 for the point agent, RAREID-PPO again exhibits better performance in the reward metrics. Episode lengths are similar for both algorithms, while RAREID crashes more often overall.

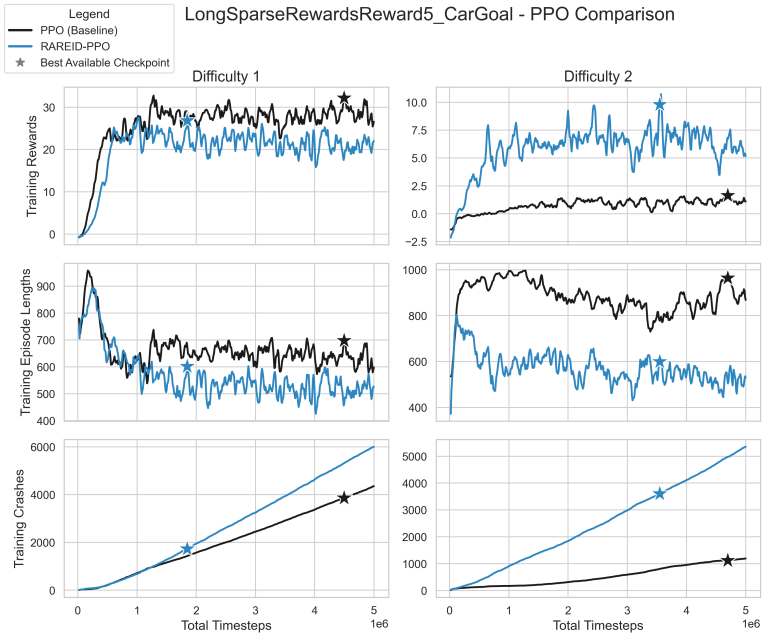


Figure 6.25.: Training mean reward, mean episode length and cumulative crashes for the car agent in the goal task with PPO variants and sparse rewards on all difficulty levels.

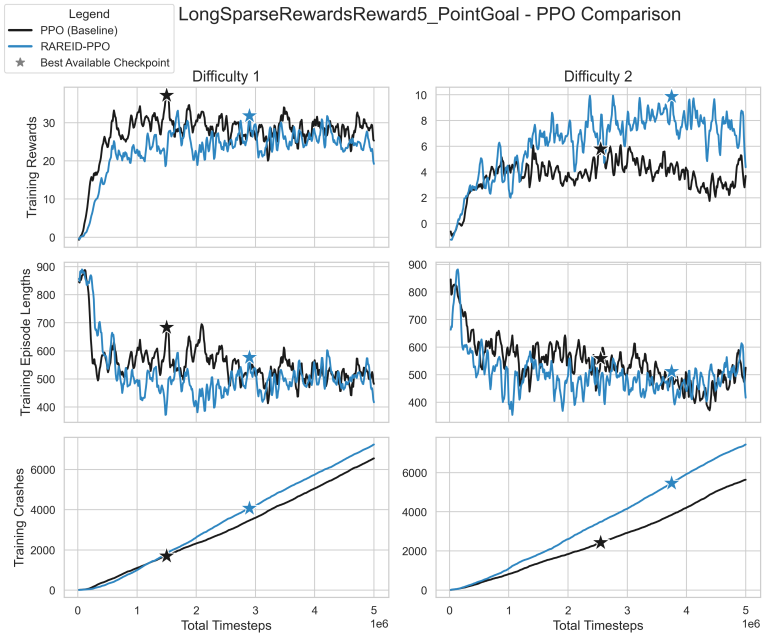
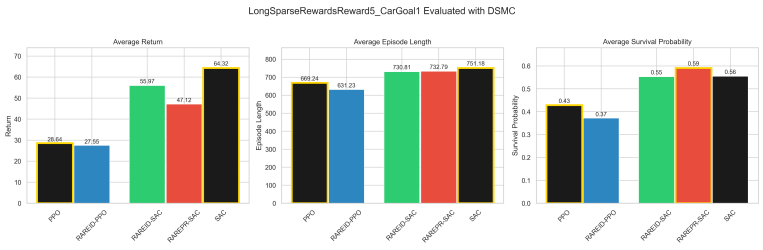
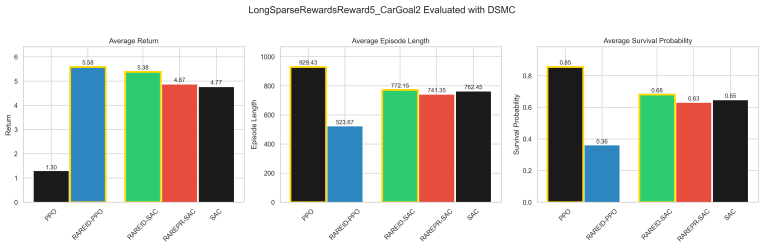


Figure 6.26.: Training mean reward, mean episode length and cumulative crashes for the point agent in the goal task with PPO variants and sparse rewards on all difficulty levels.



(a) Evaluation mean reward, mean episode length and survival probability for car agent with sparse rewards on difficulty level 1 evaluated with DSMC.



(b) Evaluation mean reward, mean episode length and survival probability for car agent with sparse rewards on difficulty level 2 evaluated with DSMC.

Figure 6.27.: Evaluation results for the car agent in the goal task with sparse rewards on difficulty levels 1 and 2 evaluated with DSMC.

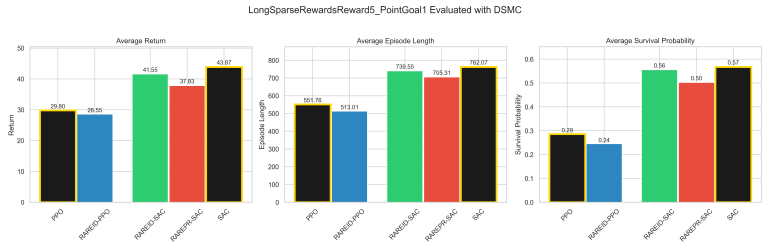
In summary, under the sparse reward setting, the training data does not support our training time hypothesis. While RAREID does consistently accumulate more crashes and either lower or equal episode lengths, it does also achieve a greater average return. This showcases RARE's exploration benefits in sparse reward structures, as it is able to learn a policy on difficulty level 2 for the car agent where the baseline failed.

PPO Baseline - Evaluation Results

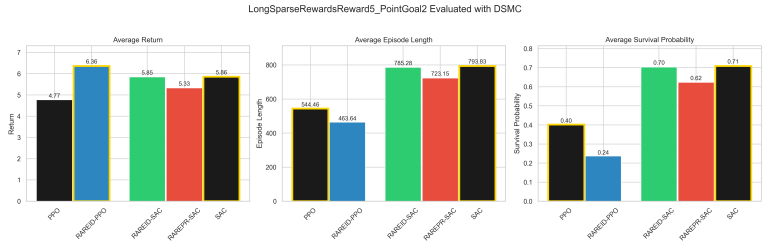
We now evaluate the DSMC results of the best model checkpoints observed during training.

For the car agent on difficulty level 1 (Figure 6.27a), the evaluation results favor the baseline PPO. In terms of reward both algorithms perform equally well, while the safety metrics are slightly better under the PPO algorithm.

On difficulty level 2 for the car agent (Figure 6.27b), the results are mixed. RAREID-PPO achieves a significantly higher average return (5.58 vs 1.30). However, the baseline demonstrates superior safety, with much longer average episode lengths (929.43 vs



(a) Evaluation mean reward, mean episode length and survival probability for point agent with sparse rewards on difficulty level 1 evaluated with DSMC.



(b) Evaluation mean reward, mean episode length and survival probability for point agent with sparse rewards on difficulty level 2 evaluated with DSMC.

Figure 6.28.: Evaluation results for the point agent in the goal task with sparse rewards on difficulty levels 1 and 2 evaluated with DSMC.

523.67) and a much higher survival probability (0.85 vs 0.36).

For the point agent on difficulty level 1 (**??**), PPO again outperforms RAREID-PPO across all metrics. In particular it achieved noticeably stronger safety metrics.

On difficulty level 2 for the point agent (**??**), similar to CarGoal2, the results are split. RAREID-PPO achieves a higher average return (6.36 vs 4.77), while PPO shows better safety performance with longer average episode lengths and higher survival probability.

In conclusion, the evaluation results for PPO in the sparse reward setting are mixed. On difficulty level 1 with car and with point agent both RAREID-PPO and PPO learned a similar performing policy. However, for both agents the baseline performed slightly better. In contrast, on difficulty 2 for the car agent the baseline seems unable to learn a policy that effectively navigates the environment. The baseline performs better on the safety metrics, but in combination with the very low average reward it is sufficiently probable that these better performance originate from a lack of action in the environment. RAREID-PPO on the other hand is able to learn a policy, whose average return suggests that consistently one goal is reached (as the average return is above R_{goal}). This showcases RARE's improved exploration capabilities.

SAC Baseline - Training Results

Next, we analyze the training performance of SAC, RAREID-SAC, and RAREPR-SAC under the sparse reward structure. The plots can be seen in Figure 6.29 and Figure 6.30

For the car agent on difficulty level 1, the SAC baseline generally performs slightly above RAREID in rewards, and clearly above RAREPR. The RARE variants perform similar to the baseline in regards to episode length, while they accumulate less crashes during training overall.

On difficulty level 2 for the car agent, both RARE variants perform similar to each other in the reward and outmatch the baseline consistently. For episode length all three algorithms develop similarly. RAREID and the baseline crash approximately equally often throughout training, while RAREPR crashes to a lesser degree than both.

Turning to the point agent on difficulty level 1, all three algorithms share very similar curves for rewards and episode lengths, while the RARE variants accumulate slightly less crashes overall.

On difficulty level 2 for the point agent, RAREID-SAC and RAREPR-SAC follow a similar trend in the reward, and outperform the baseline for the majority of timesteps. In terms of episode length no algorithm clearly excels over another. Further RAREPR crashes less often while RAREID and baseline behave equally well.

Figure 6.29.: Training mean reward, mean episode length and cumulative crashes for the car agent in the goal task with SAC variants and sparse rewards on all difficulty levels.

Figure 6.30.: Training mean reward, mean episode length and cumulative crashes for the point agent in the goal task with SAC variants and sparse rewards on all difficulty levels.

In summary, the baseline performs slightly better or comparable to the RARE variants on the first difficulty level, while the RARE variants crash less often. On the second difficulty level this behavior is inverted and both RARE variants show an improvement to the baseline. In both difficulty levels and with both agents the episode length develops similar for each algorithm. Further, RAREPR consistently crashes less often than the baseline, in contrast to RAREID, which sometimes exceeds the baseline in this metrics. Our hypothesis of worse training performance for RARE therefore does not hold.

SAC Baseline - Evaluation Results

Finally, we analyze the evaluation data for the SAC group.

For the car agent on difficulty level 1 (Figure 6.27a), the SAC baseline outperforms both RARE variants significantly in the reward metric, where the baseline achieves a reward of 64.32 against RAREID's 55.97 and RAREPR's 47.12. In terms of episode length both RARE variants perform equally well and fall slightly behind the baseline. The average survival probability is highest for RAREPR, but very similar for all three algorithms.

On difficulty level 2 for the car agent (Figure 6.27b), the RAREID-SAC algorithm performs best in all three metrics. For the average reward it achieves 5.38 against the similar performing RAREPR (4.87) and SAC (4.77) algorithms. RAREPR and the baseline also perform similar in the two safety metrics, where RAREID leads with a small gap.

For the point agent on difficulty level 1 (Figure 6.28a), the SAC baseline outperforms both RARE variants in each metric. Hereby RAREID's performance are still very similar to the baseline, while RAREPR shows a bigger gap.

The same pattern repeats for level 2 of the point agent (Figure 6.28b). While the baseline outperforms both RARE algorithms, the performance of RAREID-SAC is very close. They achieve nearly identical average returns (SAC: 5.86, RAREID: 5.85), episode lengths (SAC: 793.83, RAREID: 785.28) and survival probabilty (SAC: 0.71, RAREID: 0.70). In contrast, RAREPR-SAC lacks behind both but still achieves acceptable rewards (5.33), episode lengths (723.15) and survival probability (0.62).

In conclusion, the evaluation results for the SAC group in the sparse reward setting are promising. Especially RAREID seems to either match or improve the performance of the baseline in each of the 4 settings, in particular for the safety relevant metrics. RAREPR on the other hand seems to slightly decrease performance overall, as in each setting it either matches or decreases the performance compared to the baseline. However, RAREPR also shows improvement for the survival probability on difficulty level 1 of the car agent. As this is a single-seeded experiment it could still be that RAREPR shows similar improvements under different random initializations.

7. Discussion and Conclusion

In this thesis we lifted the RARE algorithm [9] to continuous state and action spaces. We achieved this by implementing RARE on top of the StableBaselines3 package, which provides us with various well-tested deep reinforcement learning algorithms that can be extended through the use of *callbacks*. In particular we implemented RAREID on top of the PPO algorithm [14], and both RAREID and RAREPR on top of the SAC algorithm [15, 16] using these callbacks. This also showcases the flexibility of RARE as an extension to existing algorithms.

We tested this extended RARE version for continuous state and action spaces on the safety gymnasium benchmark [18], which we modified slightly to enable state restorations. We used two tasks, which each contain three difficulty levels and various reward structures to evaluate our algorithms against the SB3 baseline.

For each configuration and algorithm we recorded the average reward, average episode length and accumulated crashes during training. We further used deep statistical model checking [33] to evaluate the best model checkpoints with respect to the rewards observed during training. This gave use high-confidence mean performance for the reward, episode length and survival probability of each algorithm.

In the case of the circle task we did so with 5 seeds per experiment, which allows us to make more generalizable claims about the algorithms average performance. Due to time constraints, we trained only one seed for the goal task. This does not enable us to make claims about the average performance, but is still usable to showcase RARE's performance in the training and evaluation data.

In this chapter we summarize the results we obtained by these experiments and discuss them. Afterwards we point out limitations and future research directions. Ultimately we conclude the thesis by answering our research hypothesis.

7.1. Discussion

Our results for the circle task show that during evaluation both RAREID and RAREPR either match the performance of the baselines, or outperform them. These observations are particularly apparent under the original reward structure with the car agent. Here, the SAC-based RARE variants clearly outperform the baseline in each metric, while RAREID-PPO variant either improves the average or the variance of each metric.

With the point agent this changes slightly. Here the benefit of RARE is less pronounced, while still clearly present on difficulties 0 and 1. Difficulty 2 presents us with an outlier where RARE performs visibly worse than the baseline. We assume, as we report the averages over 5 seeds and observe a large standard deviation, that one of our seeds underperforms and thereby shifts the mean. A larger experiment could confirm this hypothesis.

The improvements of the RARE variants upon the baselines are also visible under the shaped reward structure. Although, generally the benefit is less strong as under the original reward structure. We claim that this arises from the improved performance of the baselines, compared to their performance under the original reward structure. While RARE utilizes the baseline's update rule, its effectiveness depends on the baseline's ability to learn from the data distribution RARE provides. When the baseline already performs near-optimally, the additional benefit from RARE's data selection may be less pronounced.

In particular, RAREID-PPO also improves the safety on difficulty level 2 with the point agent. It seems like the more direct feedback helps the algorithms to more reliable converge to good policies. Therefore, besides one outlier, the results for the circle task suggest that RARE-enhanced algorithms on average achieve better results than the baselines. In particular, RARE seems to greatly improve performance in the absence of direct negative feedback signals.

Our results for the goal task are less clear. As we only experiment with one seed, it is not possible to make general statements about RARE's performance in these experiments. We can however identify that either RAREID and RAREPR result in a policy in each of our experiments that matches the baseline and in some experiments (e.g. shaped rewards with the car agent) improve upon the baseline. However, we can also observe multiple instances where the RARE variant vastly underperforms compared to the baseline (e.g. RAREID-SAC and RAREPR-SAC under the original rewards with the point agent on difficulty 2).

Just as in the circle task, these improvements of RARE upon the baselines are less strong under the shaped reward structure than under the original reward structure. We again interpret this as an improvement in the baseline, that cannot be further improved by RARE and therefore results in more similar results. Discrepancies, such as those in the rewards for the SAC-based algorithms on the second difficulty level of the point agent under the shaped reward structure could arise from the statistical inaccuracies because we only have one available seed.

Another observation that was also made in the original paper [9] is that RARE can aid policy convergences due to its improvements in exploration capabilities. Our data also suggests that in the case of the sparse reward structure for the car agent. However, due to the lack of differently seeded experiments we cannot decisively claim RARE as the reason for this improvement.

Our results generally indicate that the RARE framework is capable of providing aid to the baseline algorithm in the continuous goal environment. The extend of this benefit remains unclear as we lack the data to properly support claims about the average performance of RARE agents and their baselines in the goal environment with the different reward structures.

Further our initial hypothesis about the training performance of RARE does not hold. We assumed that, because of the state restorations which place the agent in situations where it is performing suboptimal, we would encounter more crashes, shorter episodes and less reward for RARE agent during training. While this behavior appears sometimes, generally it is not the case. We propose that the reason for this lies in the environment structure. If the structure is relatively simple, as in the circle task, we can imagine that the policy quickly learn how to behave in critical states. Therefore, after this initial learning is consolidated the policy performs better again, which would show in the plots as a brief dip in performance, followed by an increase. This quick learning can also explain why the RARE policies often accumulate less crashes during training, as they learn earlier how to avoid them.

Concerning the difference between RAREID and RAREPR, we could not find clear differences between both variants' performance in the evaluation data. While RAREPR-SAC performs better with the car agent under the original reward structure of the circle task, RAREID-SAC performs better with the point agent. Under the shaped reward structure of the circle task, RAREPR achieves better values in three of the four experiments. In the goal task, under the original reward structure, both variants again perform very similar. However, RAREID obtains a small edge in the second difficulty for the car agent. Under the shaped and sparse reward structure of the goal task, RAREID achieves the higher performance of the two variants. We thus assume both variants can achieve improvements compared to the baseline, with no clearly better variant.

7.2. Limitations and Future Work

The research conducted within this thesis serves as an initial exploration of the RARE framework in continuous state and action space domains. While our results identified potential of RARE in these domains, the full scope and limitations of this potential require further exploration. The aspects which we find pose an interesting continuation of this thesis's research are outlined in this section.

On the technical side, a refactoring of the existing code into a more modular and decoupled approach would greatly benefit the ease of extensions and ensure reliable results. Both callbacks and vectorized environment wrappers for RAREID and RAREPR share a lot of common functionality like the handling of the archives, sampling and restoration of states or the deployment of evaluation stages. Refactoring this into a common interface would benefit the extendability of RARE, and would also ease the

integration of additional SB3 algorithms.

During the course of this thesis, several limitations and obstacles were identified. A primary challenge is that safety-relevant environments do not necessarily offer a state restoration method [18, 30], which restricts RARE's direct applicability in its current form. Furthermore, the strategy of mapping continuous observations to discrete archive cells based on x-y coordinates loses more information in the SG benchmark than in Racetrack or MiniGrid which were used in the original RARE paper [9]. This discretization might merge distinct observation states, especially with the random object placements in the Goal task. This potentially hinders RARE's ability to identify critical states. Future work should therefore revisit the original paper's suggestions [9] of implementing a goal-conditioned policy to circumvent the need for the restoration functionality and exploring abstract, latent space representations (e.g., via autoencoders) for the archive to enable more general and robust state mapping.

Methodologically, the adaptation of the value heuristic for the SAC algorithm required an approximation based on the action-value function and deterministic action sampling, as detailed in Chapter 5. This approximation might not accurately capture the temporal difference error, which potentially impacts the state relevance calculations and the effectiveness of RARE variants using this heuristic with SAC. This could also be the reason that some of our SAC-based RARE variants performed under the baseline's performance. Therefore investigating the experiments with the novelty heuristic represents a valuable direction.

Furthermore, RARE's strength lies in providing statistical guarantees via DSMC. In this thesis we only explored these statistical guarantees with respect to the return. While these are informative to some degree about the safety performance, using the costs or survival probability would allow RARE to prioritize state restorations based on potential safety improvements. If costs are used this would also enable comparability with CMDP-based methods, like CSC [8] or CPO [27]. This connects to a broader limitation in our safety evaluation, which uses the average episode length and survival probability. While correlated with safety, the algorithm can also maximize these metrics through inaction. An agent achieving high survival probability by remaining passive does not demonstrate effective safe behavior. Future analyses should incorporate more direct safety metrics.

Additionally, the presented experiments utilized default hyperparameters for the baseline algorithms and the RARE framework. Performance is possibly sensitive to these settings, and a systematic hyperparameter optimization study could yield further insights and potentially greater improvements from RARE.

Finally, while this thesis finds potential in RARE's application in continuous state and action spaces, it does so within the robotics-focused SG benchmark. It would be beneficial to establishing general applicability and robustness of the approach, if RARE were also evaluated on a larger variety of benchmarks that cover different domains, e.g.

finance or autonomous driving.

By exploring these avenues, future research can enhance the practicality and reliability of RARE in continuous domains.

7.3. Conclusion

This thesis investigated the research question: *Can RARE improve evaluation safety in continuous state and action spaces?* Based on the implementation and evaluation of RAREID and RAREPR variants for PPO and SAC algorithms within modified Safety Gymnasium environments, our findings suggest: *yes, RARE can improve evaluation safety in continuous state and action spaces.*

The results from the seeded experiments in the circle task demonstrate that RARE possesses the potential to improve evaluation safety metrics in continuous domains under certain conditions. We observed instances, particularly with the original reward structure and the more complex car agent, where RARE variants led to significant improvements in average survival probability and episode length, or notably reduced the variance of these metrics across different training runs compared to the baseline algorithms. This reduced variance indicates more reliable average performance of RARE, therefore also improving the safety of these tasks. However, the effectiveness was not universal. The benefits were less pronounced for the simpler point agent, and one specific scenario (Point agent, difficulty 2) showed degraded performance. Furthermore, the introduction of shaped rewards improved baseline performance, consequently reducing the relative safety advantage offered by RARE in those specific settings.

The experiments in the more complex Goal Task, while limited by the use of a single seed per configuration, provide preliminary indications that RARE can contribute to safer policies, but definitive conclusions about average performance cannot be drawn. In several evaluations, RARE variants matched or occasionally exceeded the baseline's safety metrics. Additionally, aligning with findings from the original RARE paper, our results in the sparse reward setting exemplifies RARE's ability to improve exploration, potentially leading to policies that can solve tasks the baseline struggles with.

Comparing RAREID and RAREPR did not reveal a consistently superior variant across all tested scenarios.

In summary, while not a guaranteed improvement for all scenarios, the evidence suggests RARE can be a valuable tool for enhancing evaluation safety in continuous control. Its ability to improve average safety metrics and reduce performance variance across runs was demonstrated in different settings. However, its effectiveness is clearly influenced by the specific agent dynamics, task complexity, baseline algorithm, and reward structure. The promising but inconclusive results in the goal task underscore the need for further

research to fully understand the conditions under which RARE delivers safety benefits in continuous state and action spaces.

Bibliography

[1] Badr Ben Elallid et al. "A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving". In: *Journal of King Saud University - Computer and Information Sciences* 34.9 (Oct. 2022), pp. 7366–7390. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2022.03.013. URL: https://www.sciencedirect.com/science/article/pii/S1319157822000970.

[2] Chen Tang et al. *Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes*. arXiv:2408.03539 [cs]. Sept. 2024. DOI: 10.48550/arXiv.2408.03539. URL: http://arxiv.org/abs/2408.03539.

[3] OpenAI et al. *Learning Dexterous In-Hand Manipulation*. arXiv:1808.00177 [cs]. Jan. 2019. DOI: 10.48550/arXiv.1808.00177. URL: http://arxiv.org/abs/1808.00177.

[4] Azalia Mirhoseini et al. "A graph placement methodology for fast chip design". en. In: *Nature* 594.7862 (June 2021). Publisher: Nature Publishing Group, pp. 207–212. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03544-w. URL: https://www.nature.com/articles/s41586-021-03544-w.

[5] Ian Fox et al. *Deep Reinforcement Learning for Closed-Loop Blood Glucose Control*. arXiv:2009.09051 [cs]. Sept. 2020. DOI: 10.48550/arXiv.2009.09051. URL: http://arxiv.org/abs/2009.09051.

[6] Bettina Könighofer et al. "Shield Synthesis for Reinforcement Learning". en. In: *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*. Ed. by Tiziana Margaria and Bernhard Steffen. Cham: Springer International Publishing, 2020, pp. 290–306. ISBN: 978-3-030-61362-4. DOI: 10.1007/978-3-030-61362-4_16.

[7] Greg Anderson, Swarat Chaudhuri, and Isil Dillig. *Guiding Safe Exploration with Weakest Preconditions*. arXiv:2209.14148 [cs]. Feb. 2023. DOI: 10.48550/arXiv.2209.14148. URL: http://arxiv.org/abs/2209.14148.

[8] Homanga Bharadhwaj et al. *Conservative Safety Critics for Exploration*. arXiv:2010.14497 [cs, stat]. Apr. 2021. DOI: 10.48550/arXiv.2010.14497. URL: http://arxiv.org/abs/2010.14497.

[9] Timo P. Gros et al. "Safe Reinforcement Learning Through Regret and State Restorations in Evaluation Stages". en. In: *Principles of Verification: Cycling the Probabilistic Landscape : Essays Dedicated to Joost-Pieter Katoen on the Occasion of His 60th Birthday, Part III*. Ed. by Nils Jansen et al. Cham: Springer Nature Switzerland, 2025, pp. 18–38. ISBN: 978-3-031-75778-5. DOI: 10.1007/978-3-031-75778-5_2. URL: https://doi.org/10.1007/978-3-031-75778-5_2.

[10] Benjamin Eysenbach et al. *Leave no Trace: Learning to Reset for Safe and Autonomous Reinforcement Learning*. arXiv:1711.06782 [cs]. Nov. 2017. DOI: 10.48550/arXiv.1711.06782. URL: http://arxiv.org/abs/1711.06782.

[11] Adrien Ecoffet et al. *Go-Explore: a New Approach for Hard-Exploration Problems*. arXiv:1901.10995 [cs]. Feb. 2021. DOI: 10.48550/arXiv.1901.10995. URL: http://arxiv.org/abs/1901.10995.

[12] Adrien Ecoffet et al. "First return, then explore". en. In: *Nature* 590.7847 (Feb. 2021). Publisher: Nature Publishing Group, pp. 580–586. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03157-9. URL: https://www.nature.com/articles/s41586-020-03157-9.

[13] Timo P. Gros et al. "DSMC Evaluation Stages: Fostering Robust and Safe Behavior in Deep Reinforcement Learning". en. In: *Quantitative Evaluation of Systems*. Ed. by Alessandro Abate and Andrea Marin. Cham: Springer International Publishing, 2021, pp. 197–216. ISBN: 978-3-030-85172-9. DOI: 10.1007/978-3-030-85172-9_11.

[14] John Schulman et al. *Proximal Policy Optimization Algorithms*. arXiv:1707.06347 [cs]. Aug. 2017. DOI: 10.48550/arXiv.1707.06347. URL: http://arxiv.org/abs/1707.06347.

[15] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. arXiv:1801.01290 [cs]. Aug. 2018. DOI: 10.48550/arXiv.1801.01290. URL: http://arxiv.org/abs/1801.01290.

[16] Tuomas Haarnoja et al. *Soft Actor-Critic Algorithms and Applications*. arXiv:1812.05905 [cs]. Jan. 2019. DOI: 10.48550/arXiv.1812.05905. URL: http://arxiv.org/abs/1812.05905.

[17] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[18] Jiaming Ji et al. "Safety Gymnasium: A Unified Safe Reinforcement Learning Benchmark". In: *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2023. URL: https://openreview.net/forum?id=WZmlxIuIGR.

[19] Yafei Ou and Mahdi Tavakoli. "Towards Safe and Efficient Reinforcement Learning for Surgical Robots Using Real-Time Human Supervision and Demonstration". In: *2023 International Symposium on Medical Robotics (ISMR)*. ISSN: 2771-9049. Apr. 2023, pp. 1–7. DOI: 10.1109/ISMR57123.2023.10130214. URL: https://ieeexplore.ieee.org/document/10130214.

[20] *Starting on the Right Foot with Reinforcement Learning*. Boston Dynamics. URL: https://bostondynamics.com/blog/starting-on-the-right-foot-with-reinforcement-learning/ (visited on 03/31/2025).

[21] *Superior Robot Mobility: Where AI Meets the Real World*. ANYbotics. URL: https://www.anybotics.com/news/superior-robot-mobility-where-ai-meets-the-real-world/ (visited on 03/31/2025).

[22] *This is How Outokumpu Wants to Make Work Safer*. ANYbotics. URL: https://www.anybotics.com/news/this-is-how-outokumpu-wants-to-make-work-safer/ (visited on 03/31/2025).

[23] *Academia & Education*. Boston Dynamics. URL: https://bostondynamics.com/industry/academia-education/ (visited on 03/31/2025).

[24] Long Ouyang et al. *Training language models to follow instructions with human feedback*. arXiv:2203.02155 [cs]. Mar. 2022. DOI: 10.48550/arXiv.2203.02155. URL: http://arxiv.org/abs/2203.02155.

[25] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. 2019. arXiv: 1509.02971 [cs.LG]. URL: https://arxiv.org/abs/1509.02971.

[26] Javier García, Fern, and O Fernández. "A Comprehensive Survey on Safe Reinforcement Learning". In: *Journal of Machine Learning Research* 16.42 (2015), pp. 1437–1480. ISSN: 1533-7928. URL: http://jmlr.org/papers/v16/garcia15a.html.

[27] Joshua Achiam et al. *Constrained Policy Optimization*. arXiv:1705.10528 [cs]. May 2017. DOI: 10.48550/arXiv.1705.10528. URL: http://arxiv.org/abs/1705.10528.

[28] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, second edition: An Introduction*. Englisch. 2nd ed. Cambridge, Massachusetts: Bradford Books, Nov. 2018. ISBN: 978-0-262-03924-6.

[29] Maxime Chevalier-Boisvert et al. "Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks". In: *CoRR* abs/2306.13831 (2023).

[30] Alex Ray, Joshua Achiam, and Dario Amodei. "Benchmarking Safe Exploration in Deep Reinforcement Learning". en. In: ().

[31] Eitan Altman. *Constrained Markov Decision Processes: Stochastic Modeling*. en. 1st ed. Boca Raton: Routledge, Dec. 2021. ISBN: 978-1-315-14022-3. DOI: 10.1201/9781315140223. URL: https://www.taylorfrancis.com/books/9781315140223.

[32] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping". In: *Proceedings of the Sixteenth International Conference on Machine Learning*. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., June 1999, pp. 278–287. ISBN: 978-1-55860-612-8.

[33] Timo P. Gros et al. "Deep Statistical Model Checking". en. In: *Formal Techniques for Distributed Objects, Components, and Systems*. Ed. by Alexey Gotsman and Ana Sokolova. Cham: Springer International Publishing, 2020, pp. 96–114. ISBN: 978-3-030-50086-3. DOI: 10.1007/978-3-030-50086-3_6.

[34] Yuri Burda et al. *Exploration by Random Network Distillation*. 2018. arXiv: 1810.12894 [cs.LG]. URL: https://arxiv.org/abs/1810.12894.

[35] John Schulman et al. *Trust Region Policy Optimization*. arXiv:1502.05477 [cs]. Apr. 2017. DOI: 10.48550/arXiv.1502.05477. URL: http://arxiv.org/abs/1502.05477.

[36] John Schulman et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. arXiv:1506.02438 [cs]. Oct. 2018. DOI: 10.48550/arXiv.1506.02438. URL: http://arxiv.org/abs/1506.02438.

Appendices

A. Hyperparameter Overview

For reference we provide the full hyperparameter list as a table here.

Table A.1.: Complete Hyperparameter List.

Parameter	Value / Setting
RL Algorithm (PPO/SAC)	
Algorithm Hyperparameters	Default values from Stable-Baselines3 (SB3) library [17].
Environment Setup	
initial_x	0
initial_y	0
n_envs	5 (parallel environments during training)
General RARE Settings	
resolution	10 (State discretization resolution)
archive_size	25
max_coordinate	2.0 for Circle task (all difficulties) and Goal task (difficulties 0, 1). 3.0 for Goal task (difficulty 2).
relevance_strategy	Value Heuristic
reduction_strategy	Cluster
RARE Evaluation Stage (DSMC)	
es_alpha	1
es_kappa	0.05
es_minimal_prio	0.01
es_epsilon	1.0 for Circle task. 0.5 for Goal task.
es_initial_runs	10 (Minimal simulations per state)
psi_min	0.2 (Minimum survival probability threshold)
psi_max	0.8 (Maximum survival probability threshold)
deterministic	False (Stochastic action selection during episode collection)
Reward / Task Specific Settings	
negative_reward	Used for reward interpolation in RARE. -2.5 for Circle task (Original & Shaped) and Goal task (Sparse). -5 for Goal task (Original & Shaped).

Continued on next page

Table A.1 – continued from previous page

Parameter	Value / Setting
positive_reward	Used for reward interpolation in RARE. 30 for Circle task (Original & Shaped). 40 for Goal task (Original & Shaped). $5 \times R_{goal}$ for Goal task (Sparse), where R_{goal} is 1, 5, or 10.
reward_shaping	Boolean flag controlling penalty addition. False for experiments using the 'Original' reward structure. True for experiments using 'Shaped' or 'Sparse' reward structures.
reward_shaping_penalty	-2.5 (Applied when `reward_shaping` is True).
sparse_reward	Boolean flag enabling sparse reward calculation. False for experiments using 'Original' or 'Shaped' reward structures. True for experiments using 'Sparse' reward structures.
sparse_goal_reward	Base reward value when reaching a goal in sparse settings. 1, 5, or 10 for the respective 'Sparse' reward experiments. N/A otherwise.
Training Duration	
total_timesteps	1,000,000 for Circle task (Original reward). 5,000,000 for Circle task (Shaped reward) and all Goal task experiments.
num_eval_stages	Number of RARE evaluation stages during training. 20 (for 1M step runs, evaluating every 50,000 steps). 100 (for 5M step runs, evaluating every 50,000 steps).

B. Sparse Reward Plots

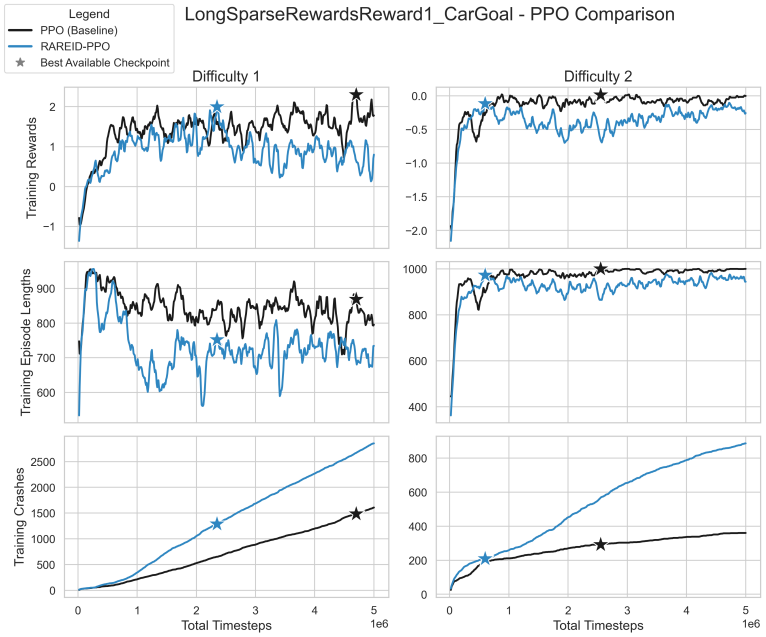


Figure B.1.: Training mean reward, mean episode length and cumulative crashes for the PPO variants with the car agent in the goal task under the sparse reward structure with goal reward 1 on all difficulty levels.

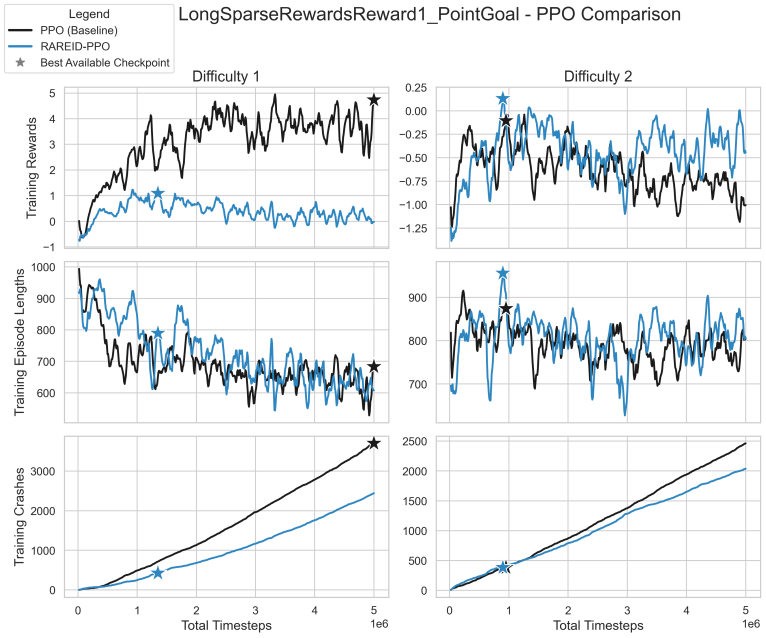


Figure B.2.: Training mean reward, mean episode length and cumulative crashes for the PPO variants with the car agent in the goal task under the sparse reward structure with goal reward 1 on all difficulty levels

Figure B.3.: Training mean reward, mean episode length and cumulative crashes for the SAC variants with the car agent in the goal task under the sparse reward structure with goal reward 1 on all difficulty levels.

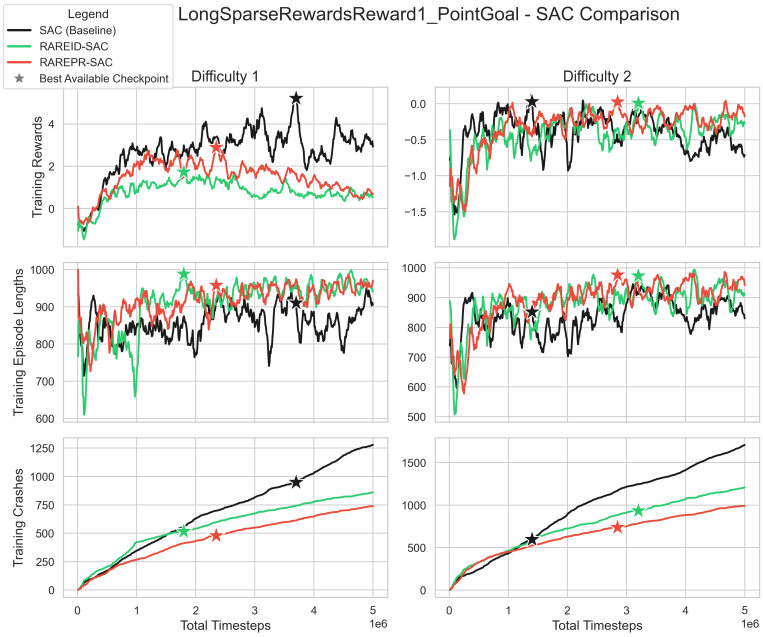
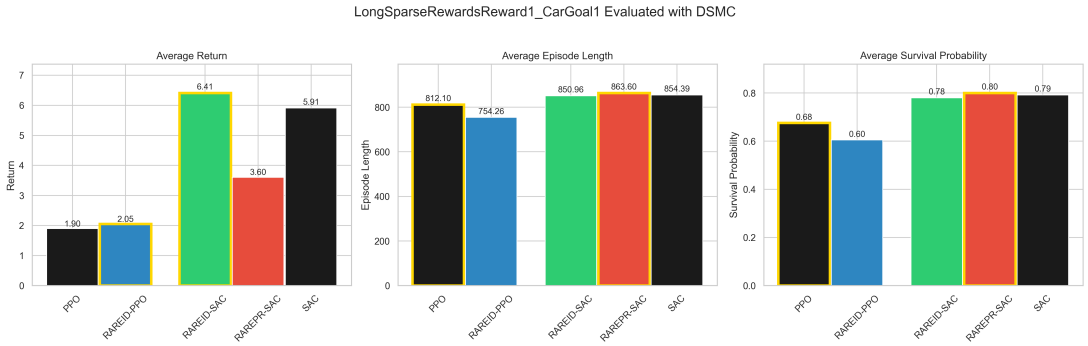
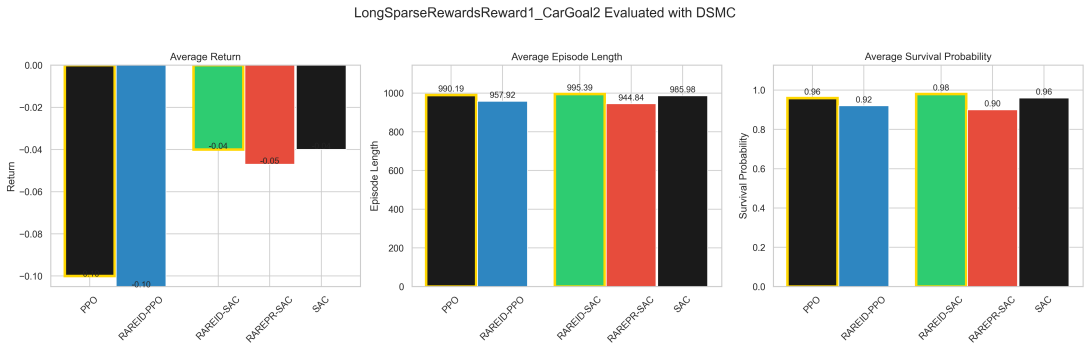


Figure B.4.: Training mean reward, mean episode length and cumulative crashes for the SAC variants with the car agent in the goal task under the sparse reward structure with goal reward 1 on all difficulty levels

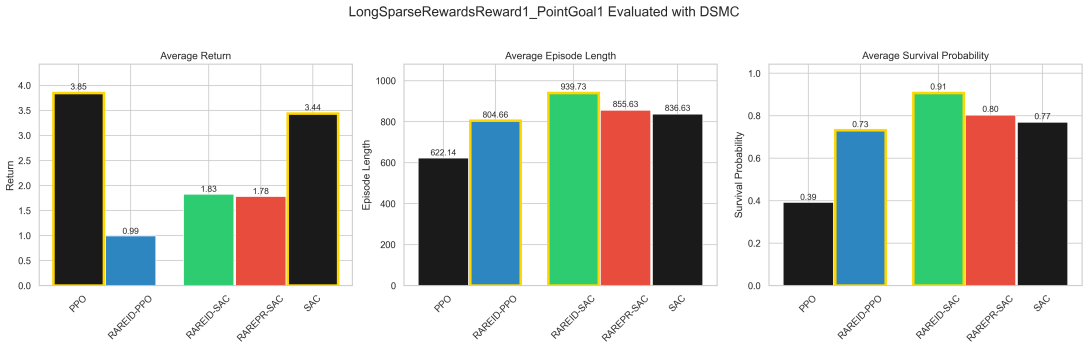


(a) Evaluation mean reward, mean episode length and survival probability for car agent under sparse rewards on difficulty level 1 evaluated with DSMC.

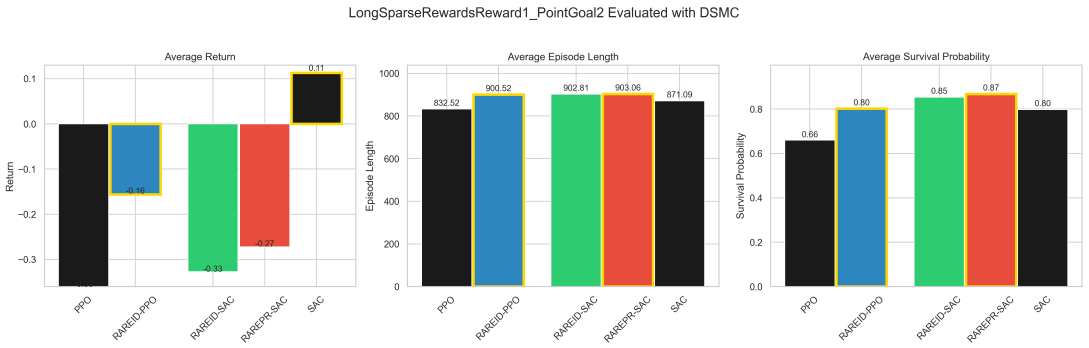


(b) Evaluation mean reward, mean episode length and survival probability for car agent under sparse rewards on difficulty level 2 evaluated with DSMC.

Figure B.5.: Evaluation results for the car agent in the goal task under sparse rewards with goal reward 1 on difficulty levels 1 and 2 evaluated with DSMC.



(a) Evaluation mean reward, mean episode length and survival probability for point agent under sparse rewards on difficulty level 1 evaluated with DSMC.



(b) Evaluation mean reward, mean episode length and survival probability for point agent under sparse rewards on difficulty level 2 evaluated with DSMC.

Figure B.6.: Evaluation results for the point agent in the goal task under sparse rewards with goal reward 1 on difficulty levels 1 and 2 evaluated with DSMC.

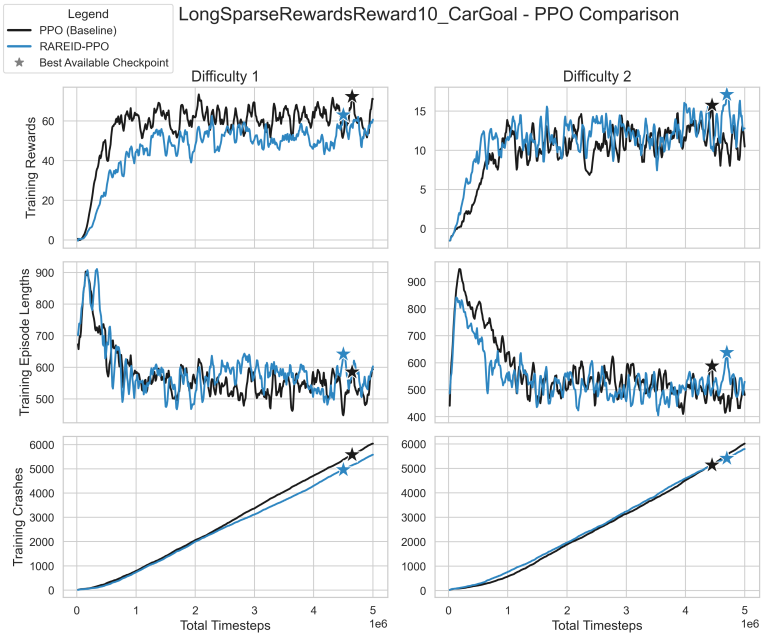


Figure B.7.: Training mean reward, mean episode length and cumulative crashes for the PPO variants with the car agent in the goal task under the sparse reward structure with goal reward 10 on all difficulty levels.

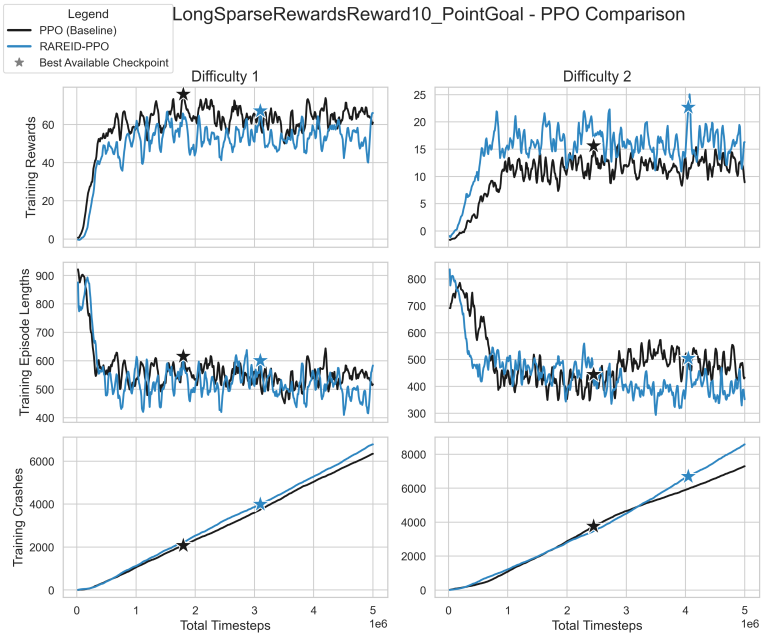


Figure B.8.: Training mean reward, mean episode length and cumulative crashes for the PPO variants with the car agent in the goal task under the sparse reward structure with goal reward 10 on all difficulty levels

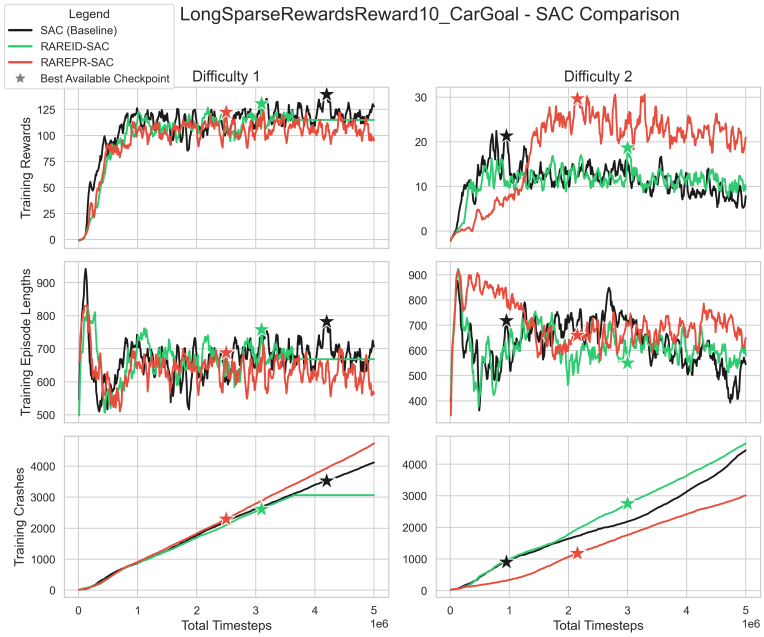
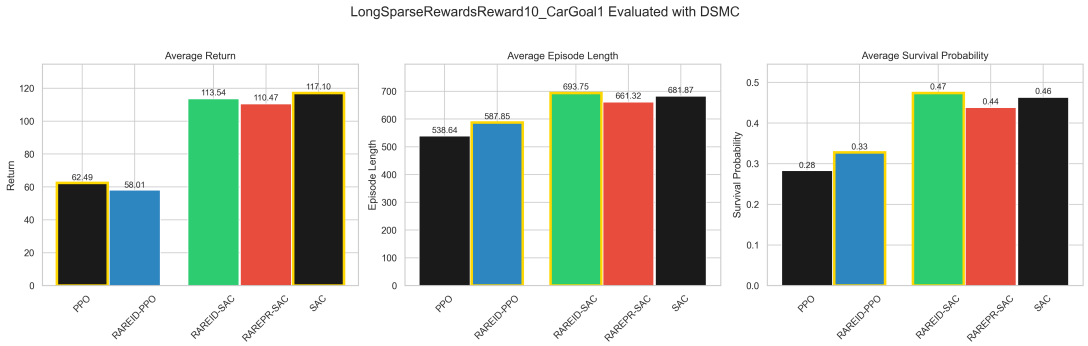
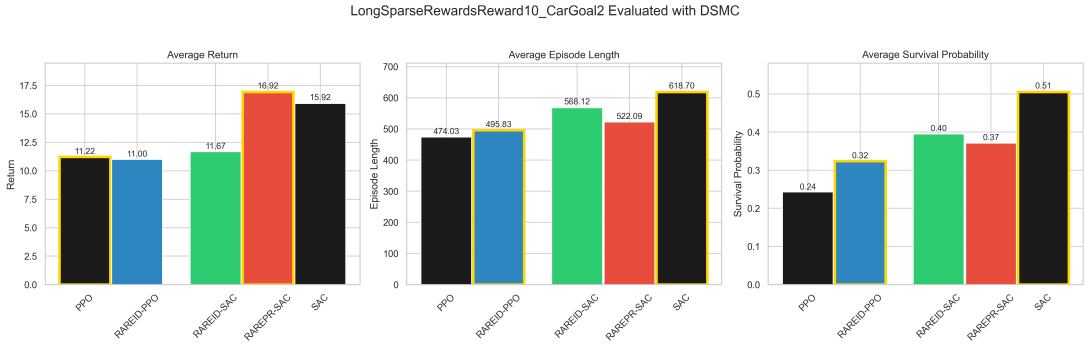


Figure B.9.: Training mean reward, mean episode length and cumulative crashes for the SAC variants with the car agent in the goal task under the sparse reward structure with goal reward 10 on all difficulty levels.

Figure B.10.: Training mean reward, mean episode length and cumulative crashes for the SAC variants with the car agent in the goal task under the sparse reward structure with goal reward 10 on all difficulty levels

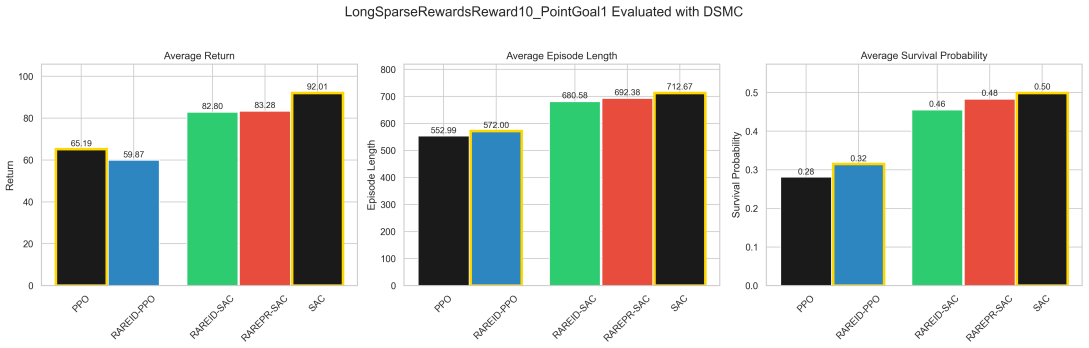


(a) Evaluation mean reward, mean episode length and survival probability for car agent under sparse rewards on difficulty level 1 evaluated with DSMC.

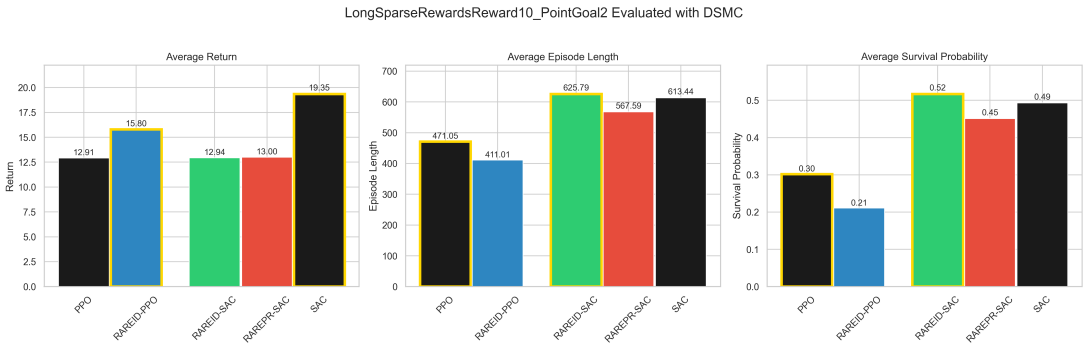


(b) Evaluation mean reward, mean episode length and survival probability for car agent under sparse rewards on difficulty level 2 evaluated with DSMC.

Figure B.11.: Evaluation results for the car agent in the goal task under sparse rewards with goal reward 10 on difficulty levels 1 and 2 evaluated with DSMC.



(a) Evaluation mean reward, mean episode length and survival probability for point agent under sparse rewards on difficulty level 1 evaluated with DSMC.



(b) Evaluation mean reward, mean episode length and survival probability for point agent under sparse rewards on difficulty level 2 evaluated with DSMC.

Figure B.12.: Evaluation results for the point agent in the goal task under sparse rewards with goal reward 10 on difficulty levels 1 and 2 evaluated with DSMC.

C. AI Usage

For this thesis the usage of AI was explicitly permitted. Because of this, parts of the code were implemented with the help of generative AI:

In particular the code which creates the plots for both the training curves and evaluation bar charts was mostly done with AI. For this different Large Language Models, namely Claude 3.7, Deepseek R1 and Gemini 2.5 were used. The code was then checked for correctness and used in the creation of the plots.

While ChatGPT was also used to help with formulations during writing, no part of the write-up is AI generated.