Making Reinforcement Learning Safer via Curriculum Learning

Kelvin Toonen and Thiago D. Simão
Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands

Keywords: Safe Reinforcement Learning, Curriculum Learning, Constrained Reinforcement Learning.

Abstract: The growth of deep reinforcement learning gives rise to safety concerns about applications using reinforcement learning. Therefore, it is crucial to investigate the safety aspect in this field, especially in the domain of robotics where agents can break surrounding objects or themselves. Curriculum learning has the potential to help with creating safer agents, because it helps the agent with learning faster and it allows for the agent to learn in safer and more controlled environments leading up to the target environment. More specifically, we change the environment only slightly to make it easier to transfer knowledge from one environment to the next, while still influencing the exploration process of the agent. This project combines curriculum learning with constrained reinforcement learning, a specific form of incorporating safety, to create a framework that allows agents to learn safely, even during training. This framework is also extended to include automation of the curriculum.

1 INTRODUCTION

Deep reinforcement learning is quite prevalent in, for example, the robotics domain and works by having an agent, the model, learn how to behave in the environment by exploring and earning a reward when it performs an action that we deem to be good. Here we want to optimize the parameters of the agent in such a way that the cumulative reward is maximized, which is the objective of the agent. Ultimately, in many practical applications, we want to do this without having unwanted side effects or using unsafe exploration, which are more likely to occur in complex agents and environments (Amodei et al., 2016).

Plenty of research has been done on safety in deep reinforcement learning (Kadota et al., 2006; Simão et al., 2021; Yang et al., 2023). Many of these methods still require a lot of training on top of the large amount needed for regular reinforcement learning. As this increases the number of environment interactions, the agent has more opportunities to make mistakes.

For this research, we look at a method of creating safe agents in a faster way, namely through curriculum learning. Curriculum learning is based on the concept that, similar to humans, it is much easier for an agent to learn a task by starting from easy examples and progressively making the task more difficult until the target task, the task that we want the agent to perform, is reached (Narvekar et al., 2020).

Previous attempts at using curriculum learning for safe reinforcement learning have focused on a strict definition of safety, where a state is safe as long as the agent can return to an initial state (Eysenbach et al., 2018; Turchetta et al., 2020). This definition is limiting as many more forms of unsafe scenarios exist, where some small costs may be tolerated. An aspect that is often overlooked by similar discussions, is that part, or the whole, of training is exempt from considerations of safety. As in any real setting, mistakes during training are also very important, we want to take the whole training process into account when discussing safety. Furthermore, these approaches require the training of several additional agents to train the primary agent.

For this research, we investigate the combination of curriculum learning and safe reinforcement learning. We create a curriculum of tasks with slightly different environments that lead up to a target task where we want to stay within certain safety constraints as much as possible throughout training. The fact that we create a curriculum through small changes in the environment gives us the freedom to create safer initial environments to train on, while teaching the agent valuable knowledge that it can use in the more unsafe, safety-critical environments. This is first done for a static curriculum that is defined before training of any agents. Then this is extended to an adaptive curriculum that adjusts based on the performance of each individual agent. The creation of these curricula does not require any pre-training or other agents to

Toonen, K. and Simão, T. D.
Making Reinforcement Learning Safer via Curriculum Learning.
DOI: 10.5220/0013388100003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 3*, pages 1465-1472
ISBN: 978-989-758-737-5; ISSN: 2184-433X

help train the primary agent, giving it an edge in this regard to state of the art.

Contribution. We propose a framework using curriculum learning that reduces the safety regret of RL agents with minimal loss in task performance. The sequencing for this curriculum is adaptive, meaning that it allows the agents to go to more difficult tasks only when they are considered ready.

2 RELATED WORK

Constrained RL. In safe RL there are two common approaches to incorporating safety: ones that alter the optimization criterion (Basu et al., 2008; Tamar et al., 2013), or ones that alter the exploration process (Tang et al., 2010; Simão et al., 2021; Yang et al., 2023). Constrained RL falls into the former category and has sprung forth many different ways of incorporating safety through constraints. Many approaches exist that use Lagrangian relaxation to include the constraints in the objective directly (Ray et al., 2019; Zhang et al., 2020; Yang et al., 2022) or use a similar approach through penalty functions (Zhang et al., 2022). Others use trust regions together with approximations of the objective and constraints (Achiam et al., 2017), or stop the episode when the agent has exceeded the cost limit (Sun et al., 2021).

Curriculum Learning in RL. Curriculum learning is much harder to categorize in a similar way, due to the vast amount of techniques used. Search algorithms are used to find a good ordering of the tasks by training agents on different curricula (Foglino et al., 2019a; Foglino et al., 2019b). In a similar fashion, a curriculum can be created iteratively by starting on the target task and then creating prior tasks that teach the agent what it was unable to do in the current task (Narvekar et al., 2016). GANs can be used to create a generator that generates the tasks and a discriminator that decides whether they are of appropriate difficulty (Florensa et al., 2018). Self-paced learning uses a distribution over possible tasks and adjusts the distribution based on the performance of the agent, moving to more difficult tasks when the agent does well (Klink et al., 2019; Klink et al., 2021; Eimer et al., 2021).

Curriculum Learning in Safe RL. There is not much research that combines curriculum learning in safe RL. One of these few performs curriculum induction by training two agents, one that learns the task and another that "resets" the former agent by moving it back to an initial state (Eysenbach et al., 2018). The algorithm induces a curriculum for the resetting agent only, as the better both agents become, the later the early abort will occur. Therefore, it learns to recover from more distant states as both agents learn. A similar research extends this concept by replacing the resetting agent with a teacher, who has access to a collection of resetting agents and learns when to use which of them (Turchetta et al., 2020). To do this, the teacher must be trained on several students such that it can adaptively choose the next resetting agent for any student. A major limitation of these approaches is their definition of safety. They define any unrecoverable state as unsafe, which is a very restricting definition. Therefore, we want to use a more general definition of safety, namely that of satisfying some constraints. This is an extension of their definition, as we can add the constraint that we cannot go into any unrecoverable states.

3 BACKGROUND

This section introduces constrained RL and curriculum learning. Finally, it formalizes the problem of designing a curriculum to improve the safety of constrained RL.

3.1 Constrained RL

Constrained RL is a form of safe RL where safety is defined as staying within certain constraints, such as requiring the expected return to be high enough (Kadota et al., 2006). Adding such constraints to a Markov decision process (MDP) results in a constrained MDP that describes the process with those constraints (Altman, 1999). The goal of constrained RL is to train the agent to stay within these constraints. This is often done by penalizing the violation of the constraints (Smith et al., 1997; Altman, 1998). In this paper, we will focus on the constraint that limits the costs that the agent is allowed to incur. Note that costs in this context are different from negative rewards, as they have semantics that cannot easily be translated to rewards, such as energy consumption. Intuitively, they represent actions that the agent can only execute a limited number of times.

Definition 3.1 (Constrained MDP)**.** We define a constrained MDP as the tuple $\langle S; A; p; r; c; \alpha \rangle$ with S the state space, A the action space, $p : S \times A \times S \to [0,1]$ the transition function, $r : S \times A \times S \to \mathbb{R}$ the reward function, $c : S \times A \times S \to \mathbb{R}$ the cost function and α the cost limit.

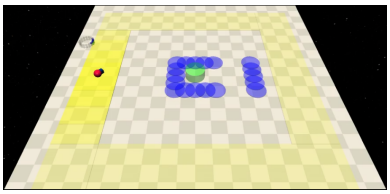


Figure 1: The target task from our experiments. Red is the agent, green is the goal, blue are the hazards and yellow are the possible starting locations, where a higher opacity relates to a higher probability of starting there.

An example environment can be found in Figure 1, where the agent needs to reach the goal. However, going over the hazards results in costs, as it, for example, may be a difficult-to-traverse area that results in a lot of energy consumption. Therefore a constraint on these costs would allow the agent to reach the goal without running out of energy.

Similarly to the return that is defined as the cumulative discounted reward $R = \sum_{t=0}^{\infty} \gamma^t r_t$, where t is the time step, r_t represents the reward, and $\gamma \in [0,1]$ is the discount factor that allows the influence of future rewards to be controlled (Puterman, 1994), we define the notion of cumulative discounted cost $C = \sum_{t=0}^{\infty} \gamma^t c_t$, where c_t is the cost at time step t. To optimize, the agents need an objective and in this case, the constrained criterion is:

$$\max_{\pi \in \Pi} \mathbb{E}_\pi(R) \text{ subject to } \mathbb{E}_\pi(C) \leq \alpha. \tag{1}$$

While there are multiple ways to constrain the cost, we choose to bound the expected cumulative cost because it is the simplest and most widely used (Wachi et al., 2024). Similarly to the return, this is approximated through the cost value function $V_C^\pi = \mathbb{E}_\pi(C \mid s_0 = s)$,, which is the expected cumulative discounted cost following policy π starting from the starting states s_0.

3.2 Curriculum Learning

The idea of curriculum learning is that it is easier to learn a complex task by iteratively building up to that task than to start on that task immediately. A curriculum is a sequence of tasks, which in this context are constrained MDPs, such that the order accelerates or improves learning. Curriculum learning is then finding a curriculum such that the order of tasks is optimized (Narvekar et al., 2020).

Curriculum learning has three main elements: task generation, task sequencing, and transfer learning (Narvekar et al., 2020). Here we focus on task sequencing, where we allow only changes in the reward, cost and transition functions, and in the starting or goal state distributions between different tasks in the curriculum.

Definition 3.2 (Curriculum)**.** A curriculum is defined as a directed acyclic graph $\langle \mathcal{V}, \mathcal{E}, \mathcal{T}, g \rangle$, where $\mathcal{V}$ is the set of vertices, $\mathcal{E}$ is the set of directed edges $\{(v_i, v_{i+1}) \mid (v_i, v_{i+1}) \in \mathcal{V} \times \mathcal{V}\}$, $\mathcal{T}$ is the set of tasks $m_i = \langle S; A; p_i; r_i; c_i; \alpha \rangle$, where a task is thus a constrained MDP, and $g : \mathcal{V} \to \mathcal{T}$ is a function that associates each vertex with a task.

In the case of a linear curriculum, before we can train on the task of node v_{i+1}, we need to train on the task of node v_i. There also is a single sink node v_t that contains the target task.

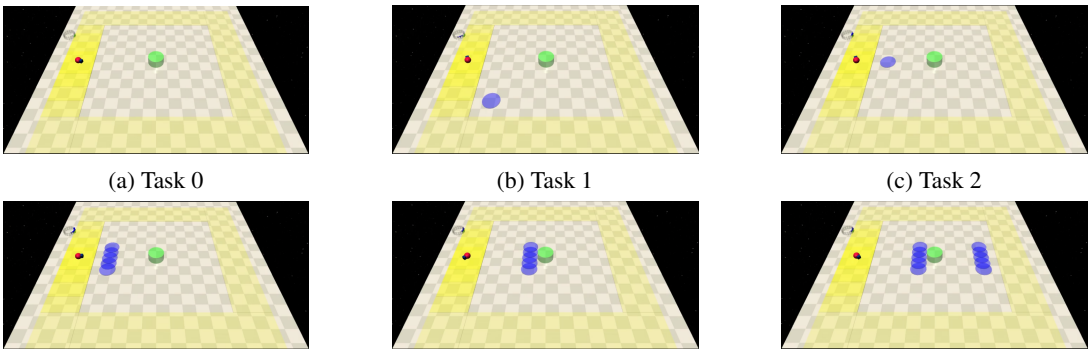
Many different types of curriculum learning techniques exist, but we put an emphasis on their different levels of automation. We distinguish between four levels of automation that each build upon the previous level. The first level is a fully hand-made curriculum, where the tasks are created and ordered by hand. The second level is automated task sequencing, where the order of tasks is calculated beforehand by an algorithm and where these tasks are still created by humans. The third level is adaptive task sequencing, where it is adaptively decided for each agent individually when a task change should occur during training through an algorithm. Finally, the fourth level is automated task generation, where also the creation of tasks is automated such that each element of the curriculum generation is automated.

4 PROBLEM STATEMENT

Our goal is to find whether curriculum learning allows for safer learning for the agents. To measure safety within a constrained MDP, we use an adapted definition of constraint violation regret from (Efroni et al., 2020). This is useful as it ignores the cost if it is below the allowed limit, which means that this metric works better than simply taking the total cost incurred (Müller et al., 2024). The constraint violation regret, which we will shorten to regret, is defined as:

$$\text{Reg}(K, c, \alpha) = \sum_{k=1}^{K} [V_c^{\pi_k}(s_0) - \alpha]_+, \tag{2}$$

where K is the number of episodes, π_k is the policy at episode k, s_0 is the starting state and $[x]_+ = \max(x, 0)$. We use regret to compare performance with respect to safety, as it does not suffice to look at the difference in constraint violation. This is the case as in constrained RL it does not matter how much the constraint is violated, only whether it is violated. Furthermore, we put an emphasis on the safety during training, thus we need a metric that can summarize the performance over several episodes.



(a) Task 0 (b) Task 1 (c) Task 2

(d) Task 3 (e) Task 4 (f) Task 5

Figure 2: Curriculum tasks used for most experiments. Red is the agent, green is the goal, blue are the hazards and yellow are the possible starting locations, where a higher opacity relates to a higher probability of starting there.

Using the definitions from the previous sections, the goal of our research is the following. We want to build a curriculum $\langle \mathcal{V}, \mathcal{E}, \mathcal{T}, g \rangle$ that helps an agent solve constrained MDP $m_t \in \mathcal{T}$ according to Criterion 1, associated with vertex v_t, while minimizing the regret $\text{Reg}(K, c_i, \alpha)$ at each vertex.

5 CURRICULUM LEARNING FOR CONSTRAINED RL

We describe the methods that we use to create an automated curriculum learning algorithm for constrained RL.

5.1 Hand-Made Curriculum

As a first step a hand-made curriculum is created, which will be expanded to contain some automation. To create a curriculum $\langle \mathcal{V}, \mathcal{E}, \mathcal{T}, g \rangle$, we follow four guidelines (Peng et al., 2018). We *isolate complexity* by creating tasks that try to teach the agent one new skill, such as not going in a straight line to the goal if there is a hazard in the way. This is done by *selecting the simplest environments to introduce one complexity at a time*, meaning that we create the simplest versions of the tasks that teach the agent a new skill. These tasks are chosen such that they are *most similar to the target environment*, such as by introducing more and more parts of the target environment into the tasks. This is also an example of *introducing complexity by building on previous tasks*, which in general can be done by requiring skills learned from previous tasks to complete the current task. In this way, a curriculum can be created with an implied ordering.

The resulting tasks can be found in Figure 2, where each task is slightly more difficult than the tasks before. These tasks all share the same state space $\mathcal{S}$ and action space $\mathcal{A}$, so it therefore falls in our definition of only having small changes between the tasks. Now the set of tasks $\mathcal{T}$ is clear, we need to create g to map vertices $\mathcal{V}$ to each of these tasks. This is simple as we can just map the ith vertex to the ith task:

$$g(v_i) = m_i, \forall i \in [0,6], \tag{3}$$

where $v_i \in \mathcal{V}$, $m_i \in \mathcal{T}$ and $|\mathcal{V}| = |\mathcal{T}| = 7$. The only remaining part of the curriculum is $\mathcal{E}$, which determines the relationship between tasks. As we have created a curriculum with a linear ordering, we will create the edges as:

$$e_i = (v_i, v_{i+1}), \forall i \in [0,5], \tag{4}$$

with $e_i \in \mathcal{E}$ and $|\mathcal{E}| = 6$.

Schedule. Although we have defined a curriculum, in practice we still need to define when to change tasks. There are too many options to try all of them, but we will follow some general heuristics. The first one is that later tasks, which are also the harder tasks as they require the knowledge of previous ones, need progressively more training time. The second is that we do not want to train until convergence for each intermediate task, because there might be a risk of overfitting on an easy task, resulting in possibly unsafe behavior in future tasks. More importantly, it is faster to not train until convergence (Narvekar and Stone, 2019). As for this type of curriculum, it needs to be decided beforehand when the task changes occur, we need to look at when the average agent converges and stop the task some episodes before that. After some small experiments, we decided to change tasks at epochs 10, 20, 40, 100, 300 and 700, where there is thus a shift from task 0 to 1 at epoch 10, from task 1 to 2 at epoch 20, etc. In our case, an epoch is 1000 steps and may consist of multiple episodes.

Algorithm 1: Learning algorithm with an adaptive curriculum for the basic instantiation.

Input: Number of training epochs n
Input: Task distribution $\vec{p}_c$
Input: Cost constraint α
Input: Cost threshold factor β
Input: Threshold of successful epochs needed κ
1: $k \leftarrow 0$ ▷ *The number of successful epochs*
2: **for** i = 0, 1, ..., n **do**
3: Sample task $m_i \sim \vec{p}_c$
4: `task_completed`, `cost` $\leftarrow$ Train on task m_i
5: **if** `task_completed` **and** `cost` $\leq \beta \cdot \alpha$ **then**
6: $k \leftarrow k+1$
7: **if** $k \geq \kappa$ **then**
8: UPDATEDISTRIBUTION($\vec{p}_c$) ▷ *This shifts the Dirac distribution to the next task*
9: $k \leftarrow 0$ ▷ *Reset counter of successful epochs*

5.2 Adaptive Curriculum

To automate part of the curriculum, we make the sequencing adaptive, which means that each curriculum is tailored to the specific agent in terms of task changes. For our approach, we take inspiration from two other approaches by generalizing them and combining them. One of these strategies maintains a probability distribution over the tasks and shifts the distribution to harder tasks on a successful episode or to easier tasks on an unsuccessful episode (Wu and Tian, 2017). The other strategy allows the agent to move on to the next task in the sequence only when the change in Q-values between updates is small enough, indicating that the agent has converged on this task (Asada et al., 1996). These two methods can be combined by using a probability distribution over the tasks and finding a good measure to decide when and how the distribution over the tasks should shift.

In this research, we take a very basic approach that uses a Dirac distribution $\vec{p}_c$ over the tasks and where we only allow the distribution to shift to the next task or stay at the current task. Furthermore, we introduce hyperparameters κ and β, where κ is the amount of epochs where the agents needs to have completed an episode with an expected cost of at most $\beta \cdot \alpha$. Here β is based on the idea that we can either be lenient with the cost limit, larger than 1, to allow for reaching the target task quicker or that we can be strict with the cost limit, smaller than 1, forcing the agent to already be very safe on the intermediate tasks. The expected cost is used instead of the actual cost, as this is what the agent is optimizing for and we want to decrease the influence of lucky starting positions, by not using the exact costs made during the epochs.

Pseudocode for this algorithm can be found in Algorithm 1. At the start of each epoch, we sample a task from the distribution $\vec{p}_c$ (Line 3), which can be any distribution, but in this case it is a Dirac distribution with probability 1 on task 0. After training on this task (Line 4), we look at the cost and whether an episode has been completed, and check if this resulted in a successful epoch (Lines 5-6). The condition in the if-statement can be changed depending on the definition of a successful episode. After this if-statement, we check whether there were enough successful epochs on this distribution (Line 7). If this is the case we update the distribution to be a Dirac distribution over the next task and reset the number of successful epochs (Line 8). The condition in the if-statement and the exact functionality of updating the distribution can be changed depending on the strategy for when to change the distribution.

6 EMPIRICAL EVALUATION

We first set up experiments for the static hand-made curriculum and then for the adaptive curriculum.

For the experiments, we train agents using several different safe RL algorithms with a cost limit of 5, which has as a consequence that the agent cannot go through the center of a hazard or through the overlapping area of two adjacent hazards without going over this limit. For most algorithms, we train a baseline version without using the curriculum and a curriculum version, and later adaptive curriculum version, trained with their respective curricula. The only exceptions are PPO (Schulman et al., 2017) and CPO (Achiam et al., 2017), which are trained with only a baseline version. Here PPO is a general RL algorithm that gives us insights mainly about the optimal return when not considering constraints and CPO is a very popular safe RL algorithm that we consider less promising for use with a curriculum than the other algorithms.

Although our experiments primarily focus on safety during training, we also evaluate the agents on the target task throughout training, for 5 episodes every 10 epochs.

6.1 Static Curriculum

Comparing Safe RL Algorithms. The agents used in this experiment consist of a curriculum version and a baseline version, which does not use a curriculum, of the algorithms CRPO (Xu et al., 2021), CUP (Yang et al., 2022), FOCOPS (Zhang et al., 2020), PCPO (Yang et al., 2020), PPOEarlyTerminated (Sun et al., 2021) and PPOLag (Ray et al., 2019). For PPOLag we have performed some hyper-

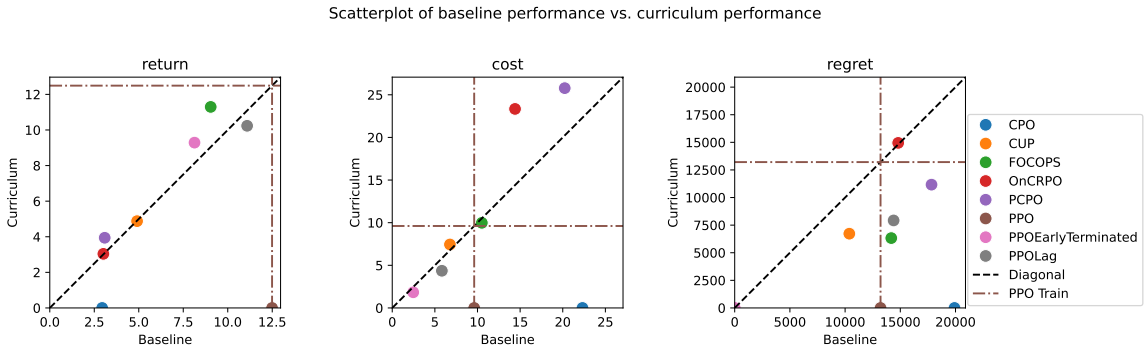


Figure 3: Plots showing how the algorithms performed with and without the curriculum, with the target task being task 4. The return and cost are the average of the metric in the final epoch of each repetition and the regret is the average of each repetition. Since PPO and CPO have only been used for the baseline, their respective curriculum values are considered 0.

parameter tuning to find proper Lagrangian parameters for agents without a curriculum, which were also used for the curriculum version. We included two more agents with only a baseline version, namely PPO and CPO. The agents are trained with task 4 as the target task, as the goal of this experiment is to find if there are safe RL algorithms that benefit more from using a curriculum. Since task 4 is a relatively easy target task, we expect the agents to converge and thus give proper insights into what a curriculum changes in terms of their performance. Both types of agents are trained for 1000 epochs, which means that the curriculum agent trains for 900 epochs on task 4.

Figure 3 shows the results during training of the experiments. Most agents trained with a curriculum have a lower regret compared to the baselines, while the return is about as good. Most of the agents trained with a curriculum also end up with a lower regret than the PPO agent, while for the baseline agents only the CUP and PPOEarlyTerminated versions manage to do so. The end costs are comparable between the two versions, but there are two cases where the curriculum version has a higher cost. However, these costs are extreme for both versions, as an expected cost of 15 is already three times the cost limit. With all of these insights, the best-performing algorithms are PPOLag, FOCOPS, CUP and PPOEarlyTerminated. These four are used for the other experiments. Note that PPOEarlyTerminated cannot have a cost above the cost limit, as the episode ends when the agent would go above the limit.

6.2 Adaptive Curriculum

Comparing to Baseline and Static Curriculum. The agents used in this experiment consist of an adaptive curriculum version, a static curriculum version and a baseline version, which does not use a curriculum, of the algorithms CUP, FOCOPS, PPOLag and PPOEarlyTerminated. We included two more agents with only a baseline version, namely PPO and CPO. The agents are trained on each task in the curriculum as the target task, except for task 0, resulting in six subexperiments that each have a different task as a target task. This means that first task 1 is used as the target task, where the baseline versions train only on task 1, while both curriculum versions consider the whole curriculum to only contain tasks 0 and 1, i.e. the curriculum up to and including this target task. Then similarly for task 2 and all tasks up to and including the actual target task. This is done to investigate the difference in performance of the three versions with different levels of difficulty for the target task and different lengths of the curriculum.

Figure 4 shows the results of these experiments for the PPOLag agent on the last three tasks. The adaptive curriculum version has a lower regret than the other two versions on all tasks. This is due to the decrease in the initial cost spike that the baseline experiences and, to a lesser extent, the static curriculum. The return is slightly lower for both curriculum versions, compared to the baseline. However, the success rate, which indicates the percentage of evaluation episodes that reached the goal with a cost below the cost limit, shows that the versions are much closer.

Adaptive Curriculum Task Progression. Due to the implementation of the adaptive curriculum, the agent is not guaranteed to reach the target task, as it may never consider itself ready to do so. Figure 5 shows the task progression of individual agents per algorithm, where it can be seen that most of the PPOLag and PPOEarlyTerminated agents did eventually reach the target task, but for CUP and FOCOPS agents this was not the case. Regardless, patterns can be seen in the task progression where there are many agents that

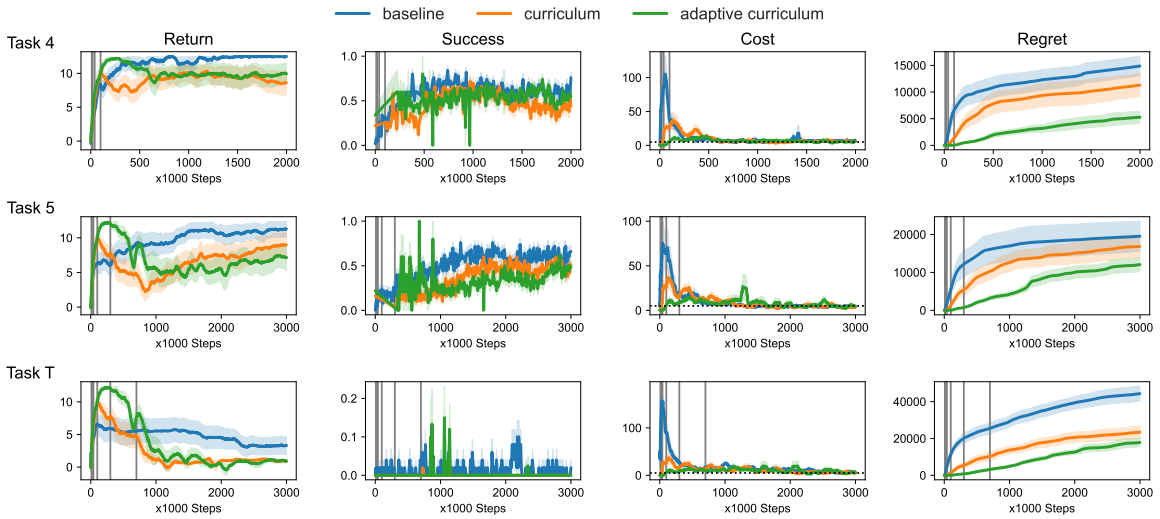


Figure 4: Training performance curves of the PPOLag agents, comparing baseline, curriculum and adaptive curriculum progression. The shaded area indicates the standard error. The vertical grey lines indicate the epochs at which a task change occurs for the curriculum agents. The horizontal dashed line in the cost plots represents the cost limit.

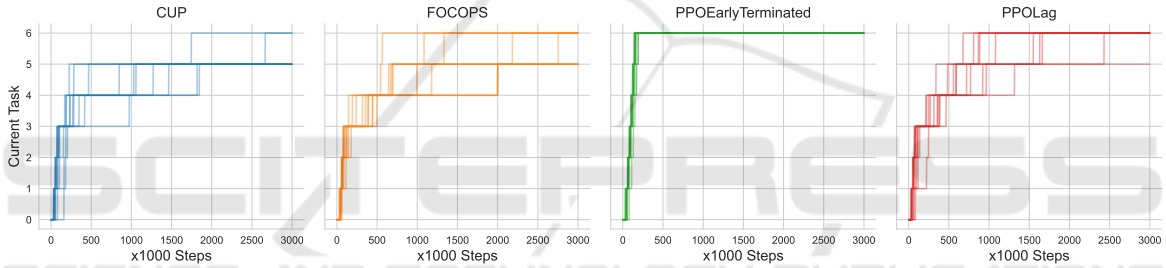


Figure 5: Task progression of the different safe RL algorithms used with an adaptive curriculum.

switch to the next task at roughly the same time. This means that it is possible to use such results to create a better static curriculum, which can guarantee that agents reach the target task.

7 CONCLUSIONS

In this work, we investigated the impact of a curriculum on the safety regret of a safe RL agent. We proposed an adaptive curriculum that asserts the competency of each agent and allows them to move on to more difficult tasks in the curriculum when they have shown to be safe enough on the current task.

Through empirical evaluation, we concluded that using a static curriculum reduces safety regret during training, with minimal performance loss. Our version of an adaptive curriculum improved on our static curriculum in the same way. This is due to the reduction in the cost spike near the start of training, which is a consequence of first training the agents on easier tasks to understand the basics of the environment.

Future Work. Our approach requires tasks to be designed beforehand. Therefore, a logical next step would be to automate task generation. However, it is not clear how to automatically design new tasks where the agent will remain safe. Furthermore, the tasks that we created are focused on changing the position of hazards. Therefore, investigating curricula that change the starting positions or the cost limit throughout training could lead to new insights.

REFERENCES

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. In *ICML*, pages 22–31. PMLR.

Altman, E. (1998). Constrained Markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Math. Methods Oper. Res.*, 48(3):387–417.

Altman, E. (1999). *Constrained Markov Decision Processes*, volume 7. Routledge.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., and Mané, D. (2016). Concrete problems in AI safety. *CoRR*, abs/1606.06565.

Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K. (1996). Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Mach. Learn.*, 23(2-3):279–303.

Basu, A., Bhattacharyya, T., and Borkar, V. S. (2008). A learning algorithm for risk-sensitive cost. *Math. Oper. Res.*, 33(4):880–898.

Efroni, Y., Mannor, S., and Pirotta, M. (2020). Exploration-exploitation in constrained MDPs. *arXiv preprint arXiv:2003.02189*.

Eimer, T., Biedenkapp, A., Hutter, F., and Lindauer, M. (2021). Self-paced context evaluation for contextual reinforcement learning. In *ICML*, pages 2948–2958. PMLR.

Eysenbach, B., Gu, S., Ibarz, J., and Levine, S. (2018). Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *ICLR (Poster)*. OpenReview.net.

Florensa, C., Held, D., Geng, X., and Abbeel, P. (2018). Automatic goal generation for reinforcement learning agents. In *ICML*, pages 1514–1523. PMLR.

Foglino, F., Christakou, C. C., and Leonetti, M. (2019a). An optimization framework for task sequencing in curriculum learning. In *ICDL-EPIROB*, pages 207–214. IEEE.

Foglino, F., Leonetti, M., Sagratella, S., and Seccia, R. (2019b). A gray-box approach for curriculum learning. In *WCGO*, pages 720–729. Springer.

Kadota, Y., Kurano, M., and Yasuda, M. (2006). Discounted Markov decision processes with utility constraints. *Comput. Math. Appl.*, 51(2):279–284.

Klink, P., Abdulsamad, H., Belousov, B., D'Eramo, C., Peters, J., and Pajarinen, J. (2021). A probabilistic interpretation of self-paced learning with applications to reinforcement learning. *J. Mach. Learn. Res.*, 22:182:1–182:52.

Klink, P., Abdulsamad, H., Belousov, B., and Peters, J. (2019). Self-paced contextual reinforcement learning. In *CoRL*, pages 513–529. PMLR.

Müller, A., Alatur, P., Cevher, V., Ramponi, G., and He, N. (2024). Truly no-regret learning in constrained MDPs. In *ICML*. OpenReview.net.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *J. Mach. Learn. Res.*, 21:181:1–181:50.

Narvekar, S., Sinapov, J., Leonetti, M., and Stone, P. (2016). Source task creation for curriculum learning. In *AAMAS*, pages 566–574. ACM.

Narvekar, S. and Stone, P. (2019). Learning curriculum policies for reinforcement learning. In *AAMAS*, pages 25–33. International Foundation for Autonomous Agents and Multiagent Systems.

Peng, B., MacGlashan, J., Loftin, R. T., Littman, M. L., Roberts, D. L., and Taylor, M. E. (2018). Curriculum design for machine learners in sequential decision tasks. *IEEE Trans. Emerg. Top. Comput. Intell.*, 2(4):268–277.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley.

Ray, A., Achiam, J., and Amodei, D. (2019). Benchmarking safe exploration in deep reinforcement learning.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Simão, T. D., Jansen, N., and Spaan, M. T. J. (2021). Alwayssafe: Reinforcement learning without safety constraint violations during training. In *AAMAS*, pages 1226–1235. ACM.

Smith, A. E., Coit, D. W., Baeck, T., Fogel, D., and Michalewicz, Z. (1997). Penalty functions. *Handbook of evolutionary computation*, 97(1).

Sun, H., Xu, Z., Fang, M., Peng, Z., Guo, J., Dai, B., and Zhou, B. (2021). Safe exploration by solving early terminated MDP. *CoRR*, abs/2107.04200.

Tamar, A., Xu, H., and Mannor, S. (2013). Scaling up robust mdps by reinforcement learning. *CoRR*, abs/1306.6189.

Tang, J., Singh, A., Goehausen, N., and Abbeel, P. (2010). Parameterized maneuver learning for autonomous helicopter flight. In *ICRA*, pages 1142–1148. IEEE.

Turchetta, M., Kolobov, A., Shah, S., Krause, A., and Agarwal, A. (2020). Safe reinforcement learning via curriculum induction. In *NeurIPS*.

Wachi, A., Shen, X., and Sui, Y. (2024). A survey of constraint formulations in safe reinforcement learning. In *IJCAI*, pages 8262–8271. ijcai.org.

Wu, Y. and Tian, Y. (2017). Training agent for first-person shooter game with actor-critic curriculum learning. In *ICLR (Poster)*. OpenReview.net.

Xu, T., Liang, Y., and Lan, G. (2021). CRPO: A new approach for safe reinforcement learning with convergence guarantee. In *ICML*, pages 11480–11491. PMLR.

Yang, L., Ji, J., Dai, J., Zhang, L., Zhou, B., Li, P., Yang, Y., and Pan, G. (2022). Constrained update projection approach to safe policy optimization. In *NeurIPS*.

Yang, Q., Simão, T. D., Jansen, N., Tindemans, S. H., and Spaan, M. T. J. (2023). Reinforcement learning by guided safe exploration. In *ECAI*, pages 2858–2865. IOS Press.

Yang, T., Rosca, J., Narasimhan, K., and Ramadge, P. J. (2020). Projection-based constrained policy optimization. In *ICLR*. OpenReview.net.

Zhang, L., Shen, L., Yang, L., Chen, S., Wang, X., Yuan, B., and Tao, D. (2022). Penalized proximal policy optimization for safe reinforcement learning. In *IJCAI*, pages 3744–3750. ijcai.org.

Zhang, Y., Vuong, Q., and Ross, K. W. (2020). First order constrained optimization in policy space. In *NeurIPS*.