AlwaysSafe: Reinforcement Learning without Safety Constraint Violations during Training

Thiago D. Simão
Delft University of Technology
The Netherlands
t.diassimao@tudelft.nl

Nils Jansen
Radboud University
Nijmegen, The Netherlands
n.jansen@science.ru.nl

Matthijs T. J. Spaan
Delft University of Technology
The Netherlands
m.t.j.spaan@tudelft.nl

ABSTRACT

Deploying reinforcement learning (RL) involves major concerns around safety. Engineering a reward signal that allows the agent to maximize its performance while remaining safe is not trivial. Safe RL studies how to mitigate such problems. For instance, we can decouple safety from reward using constrained Markov decision processes (CMDPs), where an independent signal models the safety aspects. In this setting, an RL agent can autonomously find tradeoffs between performance and safety. Unfortunately, most RL agents designed for CMDPs only guarantee safety after the learning phase, which might prevent their direct deployment. In this work, we investigate settings where a concise abstract model of the safety aspects is given, a reasonable assumption since a thorough understanding of safety-related matters is a prerequisite for deploying RL in typical applications. Factored CMDPs provide such compact models when a small subset of features describe the dynamics relevant for the safety constraints. We propose an RL algorithm that uses this abstract model to learn policies for CMDPs safely, that is without violating the constraints. During the training process, this algorithm can seamlessly switch from a conservative policy to a greedy policy without violating the safety constraints. We prove that this algorithm is safe under the given assumptions. Empirically, we show that even if safety and reward signals are contradictory, this algorithm always operates safely and, when they are aligned, this approach also improves the agent's performance.

KEYWORDS

Reinforcement Learning; CMDP; Safe Reinforcement Learning

ACM Reference Format:
Thiago D. Simão, Nils Jansen, and Matthijs T. J. Spaan. 2021. AlwaysSafe: Reinforcement Learning without Safety Constraint Violations during Training. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 10 pages.

1 INTRODUCTION

Despite all the astonishing successes in Reinforcement Learning [RL; 45], safe exploration is still a major concern preventing its deployment in real-world tasks [5]. This problem has motivated the study of constrained RL to ensure safety [16]. In this framework, an agent interacts with an environment modeled as a Constrained Markov Decision Process [CMDP; 4] without knowledge about the transition, reward, and cost functions. In safe RL, the cost function is used as a proxy to distinguish between safe and unsafe behaviors [19]. Therefore, the agent must find a policy with maximum expected reward among the safe (feasible) policies, namely those with expected cost smaller than a safety threshold.

We would like to distinguish between two constrained RL settings. The first is the *common* setting, where the agent trains in an assumed perfect simulator and only cares about constraint violations later, when deployed in the real environment. In this case, safe exploration is not a major issue, since the agent is free to explore during the learning period. The second, which is the focus of this work, is what we may call the *true* setting, where the agent interacts directly with the environment and is not allowed to violate the constraints while learning.

Following the *optimism in the face of uncertainty* framework to trade off exploration and exploitation, Efroni et al. [17] proposed different algorithms for constrained RL with bounded regret in terms of performance and in terms of constraint violation. However, these algorithms may still violate the constraints, since they encourage the agent to explore unknown parts of the environment, making them unsuitable for the true RL setting. We aim to develop RL algorithms that can learn without violating the constraints, that is, with no regret in terms of constraint violation.

We observe that often most of the state description is only relevant to the reward signal and does not influence the safety of the agent. In this setting, it can be easy for an expert to define the dynamics relevant for safety. Consider for instance imposing a limit on the consecutive movements of a robot arm to avoid overheating or indicating unsafe areas such as stairs on a mobile robot's map, in these situations the target location is not relevant for safety. Such constraints may be represented in a compact model and are a prerequisite for deploying RL in practice. Without such knowledge, typical RL algorithms would need to perform random exploration, which is not an option in safety-critical applications. Hence, we assume that this compact model is known and is represented by an abstract CMDP $\bar{\mathcal{M}}$. This assumption allows us to safely trade off exploration and exploitation, so the agent has an incentive to explore, but **always** within a set of **safe** policies. Figure 1 shows a CMDP where this kind of abstraction can be found. In this example, we observe that the variable y does not influence the cost function. We will use this problem as a running example henceforth.

This work has a novel perspective on the use of abstractions. The literature usually focuses on building a policy for $\bar{\mathcal{M}}$ that will later be executed in the ground CMDP $\mathcal{M}$. Our approach, however, focuses on computing a policy in the ground CMDP $\mathcal{M}$ and uses the abstract model $\bar{\mathcal{M}}$ to guarantee safety, which decouples the safety concerns from the reward signal.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

Our contribution is four-fold: (*i*) we study the kind of abstraction sufficient to concisely describe and distill safety dynamics. Using factored MDPs [8], (*ii*) we devise an example of such abstract model. Assuming such model is given, (*iii*) we propose a safe algorithm that learns an optimal policy for the CMDP without violating the constraints. Finally, (*iv*) we show that this algorithm is always safe and has no regrets in terms of constraint violation.

The empirical analysis showcases the capabilities of the proposed always safe algorithm: (*i*) as expected it respects the constraints during training; (*ii*) it eventually achieves optimal performance; and (*iii*) when the cost function is aligned with the reward it reduces the performance regret. Code: https://github.com/AlgTUDelft/AlwaysSafe.

2 BACKGROUND

We start by reviewing the MDP, the constrained MDP and the factored MDP formalisms. Next, we discuss the constrained RL problem and an algorithm to solve it.

2.1 Constrained MDPs

A Markov Decision Process [MDP; 36] models the interaction between an agent and its environment. Let $\Pi(\mathbb{Y})$ be the set of probability distributions over the finite set $\mathbb{Y}$. We consider an MDP with a finite state space $\mathbb{S}$, a finite action space $\mathbb{A}$, a transition function $P : \mathbb{S} \times \mathbb{A} \to \Pi(\mathbb{S})$ that represents the conditional probability distribution $P(s'|s, a)$ of moving to a successor state $s' \in \mathbb{S}$ after executing action $a \in \mathbb{A}$ in state $s \in \mathbb{S}$, a bounded reward function $R : \mathbb{S} \times \mathbb{A} \to [0, 1]$, and an initial state distribution $\mu \in \Pi(\mathbb{S})$. We focus on finite-horizon problems, where the agent interacts H times with the environment.

A Constrained Markov Decision Process [CMDP; 4] has the same elements as an MDP plus a bounded cost function $C : \mathbb{S}\times\mathbb{A} \to [0, 1]$ and an upper bound on the expected cost $\hat{c} \in [0, H]$. We consider a policy with expected cost larger than $\hat{c}$ unsafe. Although we present CMDPs with a single cost function in favor of clarity, our results can easily be extended to problems with multiple cost functions.

A factored CMDP [8] can compactly represent a CMDP using a dynamic Bayesian network [14]. It uses a set of *state variables* $X = \{X_1, \dots, X_{|X|}\}$ to represent the state space, where each variable assumes a value x_i from its domain $dom(X_i)$. Assuming the outcome of each variable X_i is conditionally independent of the outcome of the other variables given its parents $\mathrm{Pa}_a(X_i) \subseteq X$ (the set of variables on which the value of X_i depends when executing action a), the transition function is represented by the product of a set of conditional probability distributions

$$P(s' \mid s, a) = \prod_{X_i \in X} P(s'[X_i] \mid s[\mathrm{Pa}_a(X_i)], a),$$

where $s[X_i] \in dom(X_i)$ denotes the value of the variable $X_i \in X$ (or the set of variables $\Delta \subseteq X$) in state $s \in \mathbb{S}$. Let $\mathbb{N}_n = \{1, \dots, n\}$, $\forall n \in \mathbb{N}$. We can also succinctly represent the reward and cost functions by the sum of m and n local functions, respectively:

$$R(s, a) = \sum_{i \in \mathbb{N}_m} R_i(s[\Delta_i^R], a) \text{ and } C(s, a) = \sum_{i \in \mathbb{N}_n} C_i(s[\Delta_i^C], a),$$

where R_i (C_i) is the i-th local reward (cost) function that only depends on the subset of variables Δ_i^R (Δ_i^C) $\subseteq X$.

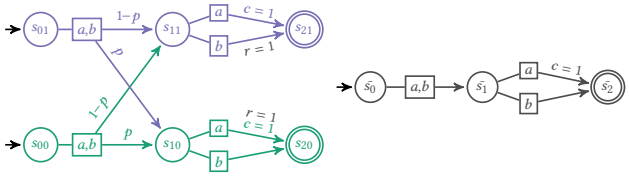

Figure 1: A factored CMDP with 2 features (x, y) and 6 states (left) and the corresponding abstract CMDP built with a model-cost-irrelevant abstraction that ignores the feature y (right). Costs and rewards with value 0 are omitted as well as the probability of deterministic transitions.

A policy $\pi : \mathbb{S}\times\mathbb{N}_H \to \Pi(\mathbb{A})$ defines the behavior of the agent. It induces a state-action occupancy $y_t(s, a) = \mu_t(s)\pi(a \mid s, t)$, where $\mu_t(s)$ is the occupancy of the state s at time step t:

$$\mu_t(s) = \begin{cases} \mu(s) & \text{if } t = 1, \\ \sum_{s^\circ, a^\circ \in \mathbb{S}\times\mathbb{A}} y_{t-1}(s^\circ, a^\circ) P(s \mid s^\circ, a^\circ) & \text{otherwise.} \end{cases}$$

An optimal policy π^* for a CMDP maximizes the expected accumulated reward while the expected accumulated cost is below the upper bound $\hat{c}$:

$$\max_\pi V_R^\pi(\mu) = \sum_{s \in \mathbb{S}} \mu(s) V_R^\pi(s, 1) = \mathbb{E}_\pi \left[\sum_{t \in \mathbb{N}_H} R_t \mid \mu \right]$$

$$\text{s. t. } V_C^\pi(\mu) = \sum_{s \in \mathbb{S}} \mu(s) V_C^\pi(s, 1) = \mathbb{E}_\pi \left[\sum_{t \in \mathbb{N}_H} C_t \mid \mu \right] \le \hat{c},$$

where R_t and C_t are random variables indicating the reward and cost the agent receives at time step t, respectively. The expected cost of following a policy π starting from state s at time step t can be computed according to its occupancy measure y as follows:

$$V_C^\pi(s, t) = \sum_{s,a,k \in \mathbb{S}\times\mathbb{A}\times\{t,\cdots,H\}} y_k(s, a) C(s, a).$$

The expected value $V_R^\pi(s, t)$ is defined similarly replacing the cost function C by the reward function R.

Example 2.1 (The optimal policy). Consider the CMDP from Figure 1. In state s_{11} action b has no cost and gives a reward of 1, so the optimal policy always assigns $\pi(b \mid s_{11}) = 1$, which maximizes the reward and does not incur any cost. This way, all the cost would come from the state s_{10}: so we have $V_C^\pi(\mu) = p\pi(a \mid s_{10})$. Since only action a gives a reward in state s_{10}, the problem reduces to $\max \pi(a \mid s_{10})$ s. t. $p\pi(a \mid s_{10}) \le \hat{c}$. For $p = 0$ the solution is trivial and $\pi(a \mid s_{10})$ might assume any value, while for $p > 0$ we can reformulate the constraint, obtaining $\pi(a \mid s_{10}) \le \frac{\hat{c}}{p}$. Since our objective is to maximize $\pi(a \mid s_{10})$, we find that the optimal policy for this problem is $\pi^*(a \mid s_{10}) = \frac{\hat{c}}{p}$.

The following Linear Program (LP) solves a CMDP:

$$\max \sum_{s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H} y_t(s, a) R(s, a) \text{ s. t. C1–C5.} \qquad \text{(LP1)}$$

$$\sum_{s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H} y_t(s, a) C(s, a) \le \hat{c}. \qquad \text{(C1)}$$

$$y_t(s,a) = \sum_{s' \in \mathbb{S}} x_t(s,a,s') \ : \ \forall s,a,t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H. \quad \text{(C2)}$$

$$\sum_{s^\circ, a^\circ \in \mathbb{S}\times\mathbb{A}} x_{t-1}(s^\circ, a^\circ, s) = \sum_{a \in \mathbb{A}} y_t(s,a) : \forall s, t \in \mathbb{S}\times\mathbb{N}_H \backslash \{1\}. \quad \text{(C3)}$$

$$\sum_{a \in \mathbb{A}} y_1(s,a) = \mu(s) : \forall s \in \mathbb{S}. \quad \text{(C4)}$$

$$x_t(s,a,s') = P(s' \mid s,a) y_t(s,a) : \forall s,a,s', t \in \mathbb{S}\times\mathbb{A}\times\mathbb{S}\times\mathbb{N}_H. \quad \text{(C5)}$$

In LP1, C1 bounds the expected cost, C2 shows the linear relation between y and x, C3 controls the inflow and outflow of each state at each time step, C4 is the initial state distribution and C5 ensures the flow respects the transition function. A solution for LP1 induces an optimal stochastic policy

$$\pi(a \mid s, t) = \frac{y_t(s,a)}{\sum_{a' \in \mathbb{A}} y_t(s,a')} : \forall s,a,t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H. \quad (1)$$

2.2 Constrained RL

In Reinforcement Learning [RL; 45] the agent does not have access to the description of the underlying MDP, so it must find a balance between *exploration*, to learn about the environment potentially increasing its performance, and *exploitation*, taking advantage of its current knowledge to collect reward. The difference between the value of the policy executed and the value of the optimal policy, called *regret*, is one measure of the efficiency of RL algorithms. Intuitively, an agent with bounded regret can make a good trade-off between exploration and exploitation.

To evaluate the efficiency of constrained RL algorithms we may consider two types of regret: performance regret and constraint violation regret [17]. The performance regret is similar to the one in traditional episodic RL settings [6, 26, 32]:

$$\text{Reg}(K, R) = \sum_{k \in [K]} \left[V_R^{\pi^*}(\mu) - V_R^{\pi_k}(\mu) \right]_+,$$

where $[x]_+ = \max\{x, 0\}$ and K is the number of episodes. Note that this definition ignores values larger than the value of the optimal policy (this is only possible if the constraint is violated). The constraint violation regret is the cumulative cost violation:

$$\text{Reg}(K, C) = \sum_{k \in [K]} \left[V_C^{\pi_k}(\mu) - \hat{c} \right]_+.$$

In this case we note that there is only regret when the expected cost is larger the given bound, so a policy has no regret for having an expected cost lower or equal to the bound. In the next section, we describe an algorithm for constrained RL with bounded regret.

2.3 Solving CMDPs with Optimism

OptCMDP [17, Algorithm 1 with $\bar{\mathcal{M}} = \emptyset$] is an extension of the UCRL2 algorithm [26] to the CMDP setting. This algorithm requires no knowledge about the components of the CMDP and bounds the performance regret with respect to the optimal policy as well as the constraint violation regret.

Intuitively, at the beginning of each episode, OptCMDP defines a set of CMDPs Ξ that contains the true CMDP $\mathcal{M}$ with high probability $1 - \delta$. It computes an optimistic policy, assuming it can also choose the best CMDP in Ξ:

$$\underset{P', R', C' \in \Xi, \pi}{\arg\max} \quad V_{R'}^{\pi_k}(\mu) \quad \text{s.t. } V_{C'}^{\pi_k}(\mu) \le \hat{c}. \quad (2)$$

Then the algorithm executes the computed policy for one episode, collecting data to update Ξ. Over time the set Ξ shrinks and the computed policy approaches the optimal policy.

Algorithm 1 OptCMDP / AbsOptCMDP / AlwaysSafe

Input: $\delta \in (0,1)$: confidence level;
Input: $\bar{\mathcal{M}} \in \{\emptyset, \langle \bar{\mathbb{S}}, \mathbb{A}, \bar{P}, \bar{R}, \bar{\mu}, \bar{C}, \hat{c} \rangle\}$
1: **for** $k \in [1, \cdots, K]$ **do**
2: Update the empirical model.
3: **if** $\bar{\mathcal{M}} = \emptyset$ **then**
4: Compute π_k with LP2.
5: **else**
6: Compute π_k using $\bar{\mathcal{M}}$ and (5), (6), (7) or Algorithm 2.
7: Execute policy π_k for one episode.

Efroni et al. [17] show that, under an optimistic perspective, we can simplify (2) by choosing the upper bound of the reward function and the lower bound of the cost function. Therefore, using $\hat{f}$ to denote the maximum likelihood estimate of the function f, we only consider a set of transition functions when defining Ξ:

$$\begin{aligned}\Xi = \{P', R', C' : \\ P' \in [\hat{P}(s' \mid s,a) - e_\delta^P(s,a,s'), \hat{P}(s' \mid s,a) + e_\delta^P(s,a,s')], \\ R' = \hat{R}(s,a) + e_\delta^R(s,a), \\ C' = \hat{C}(s,a) - e_\delta^C(s,a), \forall s,a,s' \in \mathbb{S}, \mathbb{A}, \mathbb{S}\},\end{aligned}$$

where the confidence intervals e_δ^P, e_δ^R and e_δ^C are based on the Bernstein inequality (for the transition function) and Hoeffding inequality (for the cost and reward function) [25]. We refer to [17, Equation 20] for a detailed derivation of the confidence intervals such that $P(\mathcal{M} \in \Xi) \ge 1 - \delta$. Now we can solve (2) with the following optimistic LP [17, 24, 39]:

$$\begin{aligned}&\max \sum_{s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H} y_t(s,a)(\hat{R}(s,a) + e_\delta^R(s,a)) \\ &\text{s.t. C2–C4 (LP1), C6–C8.}\end{aligned} \quad \text{(LP2)}$$

$$\sum_{s,a,t \in \mathbb{S}\times\mathbb{A}\times\mathbb{N}_H} y_t(s,a)(\hat{C}(s,a) - e_\delta^C(s,a)) \le \hat{c}. \quad \text{(C6)}$$

$$x_t(s,a,s') \le (\hat{P}(s'|s,a) + e_\delta^P(s,a,s')) y_t(s,a) \quad \forall (s,a,s',t) \in \mathbb{S}\times\mathbb{A}\times\mathbb{S}\times\mathbb{N}_H. \quad \text{(C7)}$$

$$x_t(s,a,s') \ge (\hat{P}(s'|s,a) - e_\delta^P(s,a,s')) y_t(s,a) \quad \forall (s,a,s',t) \in \mathbb{S}\times\mathbb{A}\times\mathbb{S}\times\mathbb{N}_H. \quad \text{(C8)}$$

Compared to LP1, LP2 replaces the equality C5 by two inequalities, which ensure that the chosen transition function is close to the true transition function with high probability. Besides an optimistic policy, computed by (1), a solution for this LP also gives us the optimistic transition function picked for problem (2):

$$P_t'(s' \mid s,a) = \frac{x_t(s,a,s')}{y_t(s,a)}, \forall t \in \mathbb{N}_H.$$

Over the episodes, as the agent collects more experiences, the estimate $\hat{P}$ improves as the confidence interval e^P_δ narrows. This way, P' approaches P and the policy computed by LP2 gets closer to an optimal one.

Even though the OptCMDP algorithm has a bounded constraint violation regret, in safety-critical applications even a small regret would not be acceptable, so this algorithm could not be directly deployed to real-world tasks. In the next section, we present a model that allow us to compactly represent a CMDP. Later, we will use this model to define a compact abstraction of the dynamics that are relevant for the safety constraints and use it to build a constrained RL algorithm with no constraint violation regret.

3 ABSTRACTION FOR EXPECTED COST

Before describing our method, let us consider again the robot with an arm that can overheat. In this example, we could keep track of how often the motor was activated in the last hour and constraint the RL agent's policies to avoid excessive consecutive movements. This is a simple example of an abstraction that lets the robot act safely. In this section, we will formalize an abstract version of the problem that captures the knowledge required to ensure safety in the constrained RL setting. As an example, we explore factored CMDPs where the cost function is independent of some variables, that is, only a subset of the variables are relevant for the constraints.

Following the definitions by Li et al. [31], we denote a state abstraction function by $\phi : \mathbb{S} \to \bar{\mathbb{S}}$, where $\bar{\mathbb{S}}$ is the finite abstract state space. The inverse of ϕ is denoted by $\phi^{-1} : \bar{\mathbb{S}} \to 2^{\mathbb{S}}$, that is $\phi^{-1}(\bar{s})$ is the set of ground states whose abstract state is $\bar{s}$ according to ϕ. Given a CMDP $\mathcal{M} = \langle \mathbb{S}, \mathbb{A}, P, R, \mu, C, \hat{c} \rangle$ and an abstraction function ϕ, the respective abstract CMDP is $\bar{\mathcal{M}}_\phi = \langle \bar{\mathbb{S}}, \mathbb{A}, \bar{P}, \bar{R}, \bar{\mu}, \bar{C}, \hat{c} \rangle$, where

$$\begin{aligned}
&\bar{P}(\bar{s}' \mid \bar{s}, a) = \textstyle\sum_{s \in \phi^{-1}(\bar{s})} \sum_{s' \in \phi^{-1}(\bar{s}')} w(s) P(s' \mid s, a),\\
&\bar{R}(\bar{s}, a) = \textstyle\sum_{s \in \phi^{-1}(\bar{s})} w(s) R(s, a),\\
&\bar{C}(\bar{s}, a) = \textstyle\sum_{s \in \phi^{-1}(\bar{s})} w(s) C(s, a),\\
&\bar{\mu}(\bar{s}) = \textstyle\sum_{s \in \phi^{-1}(\bar{s})} \mu(s)
\end{aligned}$$

and $w(s)$ indicates the contribution of each state $s \in \phi^{-1}(\bar{s})$ to the abstract state $\bar{s}$, with the constraint that $\sum_{s \in \phi^{-1}(\bar{s})} w(s) = 1$.

3.1 Cost-model Irrelevance

Li et al. [31, Definition 3] use the above formalism to define different types of abstractions. For instance, Q^π_R-irrelevant abstractions preserve the Q^π_R function. This may be useful when solving an MDP, as we could compute Q^π_R over the abstract state space, which can speed up the convergence of an MDP solver [21] or an RL agent [49]. Following this idea, we define an abstraction related to the model necessary to compute the expected cost V^π_C.

Definition 3.1. Given an CMDP $\mathcal{M} = \langle \mathbb{S}, \mathbb{A}, P, R, \mu, C, \hat{c} \rangle$, we say that an abstraction function ϕ is *cost-model-irrelevant* when

$$\begin{aligned}
&\phi(s_1) = \phi(s_2) \Rightarrow\\
&\quad \sum_{s' \in \phi^{-1}(\bar{s})} P(s' \mid s_1, a) = \sum_{s' \in \phi^{-1}(\bar{s})} P(s' \mid s_2, a) \text{ and}\\
&\quad C(s_1, a) = C(s_2, a) \quad \forall a, s_1, s_2, \bar{s} \in \mathbb{A} \times \mathbb{S} \times \mathbb{S} \times \bar{\mathbb{S}}. \qquad (3)
\end{aligned}$$

Definition 3.1 is similar to model-irrelevance [31] considering the cost function instead of the reward function. It says that if a cost-model-irrelevant abstraction maps two states to the same abstract state then the cost of executing action $a \in \mathbb{A}$ and the distribution over the next abstract state is the same in both states.

We would like to use prior knowledge of a model of the abstract CMDP to guarantee that a policy for the ground CMDP will not violate the cost constraints. So, while most literature on abstraction for RL is interested in deploying the policy computed in the abstract CMDP to the ground CMDP, we use the abstract CMDP to test the safety of a policy defined in the ground CMDP.

3.2 A Cost-model-irrelevant Abstraction

In factored CMDPs, we can define a cost-model-irrelevant abstraction by considering only the subset of state variables that influence the cost function. Given a set of variables $\Delta \subseteq X$, we define their parents as $\mathrm{Pa}(\Delta) = \bigcup_{X_i, a \in \Delta \times \mathbb{A}} \mathrm{Pa}_a(X_i)$ and their ancestors as:

$$\mathrm{Anc}(\Delta) = \begin{cases} \Delta & \text{if } \mathrm{Pa}(\Delta) \subseteq \Delta, \\ \mathrm{Anc}(\mathrm{Pa}(\Delta) \cup \Delta) & \text{otherwise.} \end{cases}$$

Intuitively, the set $\mathrm{Anc}(\Delta)$ might influence Δ over multiple time steps, while the set $\mathrm{Pa}(\Delta)$ are only variables that have an immediate influence on Δ. Let $\mathrm{Pa}(C) = \cup_{i \in \mathbb{N}_n} \Delta^C_i$ be the set of variable that directly influences the cost function, we define a cost-model-irrelevant abstraction ϕ_C based on their ancestors $\mathrm{Anc}(\mathrm{Pa}(C))$:

$$\phi_C(s[X]) = s[\mathrm{Anc}(C)]. \qquad (4)$$

Our running example (Figure 1) shows an instance of such abstraction. The efficiency of this abstraction, related to the size reduction from the original CMDP to the abstract CMDP, corresponds to the size of the set of ancestors of the cost function: $|\mathrm{Anc}(C)|$. For instance, if $\mathrm{Anc}(C) = X$, this abstraction would be the identity function and the abstract CMDP would be the same as the original. If, however, $\mathrm{Anc}(C) = \emptyset$, the abstract CMDP would contain a single state since the cost function is independent of the state variables.

Theorem 3.2. *ϕ_C is a* cost-model-irrelevant *abstraction.*

Proof. See supplementary material [41]. □

While ϕ_C is a convenient and natural *cost-model-irrelevant* abstraction, it is not necessarily the most compact. We refer to Givan et al. [20] for a discussion on how to find more compact abstract models in factored MDPs.

3.3 Planning with the Abstract CMDP

Given a cost-model-irrelevant abstraction, we extend LP1 to take this knowledge in consideration, by adding variables z that represent the occupancy of each pair of abstract state and action for each time step. Our goal is to decouple the expected cost from the full transition function, to ensure that the policy computed still respects the cost constraints.

$$\begin{aligned}
&\max \sum_{s,a,t \in \bar{\mathbb{S}} \times \mathbb{A} \times \mathbb{N}_H} y_t(s, a) R(s, a) \qquad \text{(LP3)}\\
&\text{s. t. } \texttt{C2–C5 (LP1), C9–C11.}
\end{aligned}$$

$$\sum_{\bar{s},a,t \in \bar{\mathbb{S}} \times \mathbb{A} \times \mathbb{N}_H} z_t(\bar{s},a)\bar{C}(\bar{s},a) \leq \hat{c}. \tag{C9}$$

$$z_t(\bar{s},a) = \sum_{s \in \phi^{-1}(\bar{s})} y_t(s,a) \quad \forall \bar{s},a,t \in \bar{\mathbb{S}} \times \mathbb{A} \times \mathbb{N}_H. \tag{C10}$$

$$\sum_{a \in \mathbb{A}} z_t(\bar{s},a) = \sum_{\bar{s}^\circ,a^\circ \in \bar{\mathbb{S}} \times \mathbb{A}} \bar{P}(\bar{s} \mid \bar{s}^\circ, a^\circ) z_{t-1}(\bar{s}^\circ, a^\circ)$$
$$\forall \bar{s}, t \in \bar{\mathbb{S}} \times \mathbb{N}_H \setminus \{1\}. \tag{C11}$$

In LP3, constraint C10 helps us to connect the flow from the ground CMDP and the abstract CMDP, by ensuring that the flow leaving an abstract state is the sum of the flow that leaves the respective ground states. Constraint C9 replaces constraint C1, notice that it uses the abstract cost function and the expected cost is computed according to the occupancy of the abstract CMDP. Finally, constraint C11 ensures that the flow of the abstract CMDP respects the abstract transition function. Although this last constraint is redundant since this flow is already specified in the ground CMDP, it will be important for our method later when we do not have access to the underlying transition function.

Since LP3 keeps variables for the ground CMDP and the abstract CMDP, the policy it computes in the ground CMDP might be different in states that were merged. In other words, the policy induced by the abstract variables z is different from the policy induced by the ground variables x and y.

In the next section, we show how to use this formulation to devise an RL algorithm compliant with the safety constraints.

4 ALWAYS SAFE

Now we consider the setting where the agent has access to the abstract CMDP generated from a cost-model irrelevance abstraction, but does not have access to the full transition function or the reward function. We propose an algorithm that learns the optimal policy of the underlying CMDP without incurring any constraint violation regret using an optimistic approach.

4.1 The Linear Program

The idea is to combine the abstract CMDP created with a cost-model-irrelevant abstraction (Section 3.1) with an optimistic approach for exploration (Section 2.3). LP4 puts all the pieces together:

$$\max \sum_{s,a,t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H} y_t(s,a)(\hat{R}(s,a) + e^R_\delta(s,a)) \tag{LP4}$$
$$\text{s. t. C2–C4 (LP1), C7–C8 (LP2), C9–C11 (LP3).}$$

The main difference between LP4 and LP2 is the use of the extra variables z that represent the flow in the abstract CMDP. Therefore we replace C6 that constrains the expected cost on the ground CMDP by C9 that constrains the expected cost in the abstract CMDP. Constraints C9–C11 control the flow in the abstract CMDP.

We can compile two basic policies using a solution for LP4. An *abstract policy* π_A using z and a *ground policy* π_G using y:

$$\pi_A(a \mid s,t) = \frac{z_t(\phi(s),a)}{\sum_{a' \in \mathbb{A}} z_t(\phi(s),a')} : \forall s,a,t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H, \tag{5}$$

$$\pi_G(a \mid s,t) = \frac{y_t(s,a)}{\sum_{a' \in \mathbb{A}} y_t(s,a')} : \forall s,a,t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H. \tag{6}$$

Next we analyze the properties of these two policies.

Theorem 4.1. *Given an uncertainty set Ξ that contains the underlying CMDP $\mathcal{M}$, a cost-model-irrelevant abstraction ϕ and the respective abstract CMDP $\bar{\mathcal{M}}_\phi = \langle \bar{\mathbb{S}}, \mathbb{A}, \bar{P}, \bar{R}, \bar{\mu}, \bar{C}, \hat{c} \rangle$, policy π_A computed according to* LP4 *and* (5) *does not violate the constraints when applied in $\mathcal{M}$: $V_C^{\pi_A}(\mu) \leq \hat{c}$.*

Proof sketch. Li et al. [31] show that a model-irrelevant abstraction preserves the expected value. In the same way, a cost-model-irrelevant abstraction preserves the expected cost. So the policy computed in the abstract state has the same expected cost in the ground state, $V_C^{\pi_A}(\mu) = V_{\bar{C}}^{\pi_A}(\mu)$. From the constraint C9 in LP4, we have $V_{\bar{C}}^{\pi_A}(\mu) \leq \hat{c}$. Therefore, $V_C^{\pi_A}(\mu) \leq \hat{c}$. □

Theorem 4.1 essentially shows that the policy π_A is *safe*, independent of the uncertainty over the transition function. However, this policy is not expressive enough to describe an optimal policy for the underlying CMDP, since its domain might ignore variables that influence the reward function (see Example 4.2).

Example 4.2 (π_A might be sub-optimal). Considering the CMDP from Figure 1 again, we may notice that a policy defined in the abstract MDP would assign at most probability $\hat{c}$ to action a in the abstract state $\bar{s_1}$, while a policy defined on the ground state can distinguish between s_{10} and s_{11}, as we saw in Example 2.1.

We conjecture that an algorithm following policy π_G in each episode has a bounded performance regret, inherited from the OptCMDP algorithm, since it makes the same assumptions. However, it can still exhibit some safety violation stemming from the unknown transition function (see Example 4.3).

Example 4.3 (π_G might be unsafe). Let us consider the CMDP from Figure 1 from an optimistic perspective. We may assume that our estimate of transition function is perfect, $\hat{p} = p$, but we still have some uncertainty about it, represented by $e(p)$. This way, we know $p \in [\hat{p} - e(p), \hat{p} + e(p)]$. In the optimistic CMDP, we are also picking the parameters of the transition function. This means, the agent chooses the value of p, which in this case would be the lower bound $p' = \hat{p} - e(p)$, since it minimizes the chance of reaching state s_{10}. Following the same reasoning as in Example 2.1, we find a greedy policy $\pi_G(a \mid s_{10}) = \frac{\hat{c}}{p - e(p)}$ which is unsafe, since it is larger than the maximum value we found in Example 2.1 ($\pi^*(a \mid s_{10}) = \frac{\hat{c}}{p}$).

4.2 Policies

In this section, we study how to switch between π_A and π_G, to find an RL algorithm that has no constraint regret and can still find an optimal policy for the underlying CMDP.

4.2.1 Policy π_T. To devise an algorithm that can eventually find an optimal policy for the underlying CMDP, we propose to use the ground policy based on a test:

$$\pi_T = \begin{cases} \pi_G & \text{if } \max_{P' \in \Xi} V_C^{\pi_G}(\mu) \leq \hat{c} \\ \pi_A & \text{otherwise.} \end{cases} \tag{7}$$

Algorithm 2 Dynamic Constraint Tightening (π_α)

Input: Ξ: uncertainty set
Input: $\bar{\mathcal{M}}$: abstract model
Input: α: learning rate
1: $\beta \leftarrow 1$
2: **repeat**
3: $y, z, status \leftarrow$ solve LP4 with $\beta\hat{c}$
4: **if** *status* is infeasible **then**
5: $\pi_\alpha \leftarrow \pi_A$ ▷ (5) based on z
6: **return** π_α
7: $\pi_\alpha \leftarrow \pi_G$ ▷ (6) based on y
8: $maxV \leftarrow \max_{P',\cdot,\cdot \in \Xi} V_C^{\pi_\alpha}(\mu)$ ▷ LP5 based on π_α
9: $\beta \leftarrow \beta - \alpha \frac{\max\{maxV - \hat{c}, 0\}}{\hat{c}}$
10: **until** $maxV \leq \hat{c}$
11: **return** π_α

To test if we can deploy π_G, we compute the maximum expected cost within the uncertainty set, fixing the policy π_G:

$$\max_{x,y,z} \sum_{\bar{s},a,t \in \bar{\mathbb{S}} \times \mathbb{A} \times \mathbb{N}_H} z_t(\bar{s}, a)\bar{C}(\bar{s}, a) \quad \text{(LP5)}$$
$$\text{s. t. C2–C4 (LP1), C7–C8 (LP2), C10 (LP3), C12.}$$

$$y_t(s, a) = \pi_G(a \mid s, t) \sum_{a' \in \mathbb{A}} y_t(s, a') : \forall s, a, t \in \mathbb{S} \times \mathbb{A} \times \mathbb{N}_H. \quad \text{(C12)}$$

In this LP, the constraint C12 ensures policy π_G is applied in every ground state. The value of $\pi_G(a \mid s, t)$ are constants computed according to the solution of LP4, for every state, action and time step. Intuitively, LP5 chooses the transition function in the uncertainty set Ξ with the highest expected cost.

Although this approach could be more efficient, for instance testing if $\exists P' \in \Xi : V_C^{\pi_G}(\mu) > \hat{c}$, we opt to compute the maximum expected cost, because it give us an indication of how much the constraint might be violated. This can help us find a more conservative policy in the ground CMDP, as we describe next.

4.2.2 Policy π_α. The previous solution may never choose the ground policy π_G, in particular when the optimal policy has an expected cost close to the bound $\hat{c}$. In this case, even a small confidence interval could put the maximum expected cost above the given cost bound. Inspired by de Nijs et al. [13], we propose to compute a policy that is more conservative such that it passes the test from (7)[1].

Algorithm 2 describes one way to compute such a policy. The algorithm initializes the coefficient β with value 1. Then, in each iteration, the algorithm solves LP4 using an adjusted bound $\beta\hat{c}$. If the algorithm did not meet any of its stopping criteria, it lowers the coefficient β and repeats. The algorithm can terminate in two ways: (*i*) by finding a policy that respects the constraints in all CMDPs in the uncertainty set; or (*ii*) by setting a cost bound too low, such that none of the CMDPs can satisfy the constraints.

Figure 2 demonstrates a successful search for a safe ground policy with Algorithm 2. Each plot shows the distribution of expected cost (according to the CMDPs in Ξ) for policies computed with a certain

[1]Another option would be to change the test in (7) allowing a small error ($\hat{c} + \epsilon$). This would give us an approximately safe algorithm.

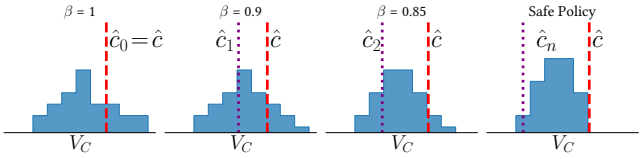

Figure 2: Search for a ground policy that respects the constraints in all CMDPs from the uncertainty set Ξ. The x-axis indicates the expected cost and the y-axis the frequency we can find a CMDP in Ξ for which the policy computed has that expected cost.

bound $\hat{c}_i$. The first three plots shows how the cost bound $\hat{c}_i$ changes over the iterations and the last plot shows one of the stopping conditions of the algorithm, when the policy computed according to $\hat{c}_n$ respect the original constraints in all CMDPs in Ξ.

4.3 Theory

We would like to show that the AlwaysSafe algorithm equipped with a cost-model-irrelevant abstract CMDP $\bar{\mathcal{M}}$ and one of the safe policies π_A, π_T, or π_α has no constraint violations with high probability, as stated in Theorem 4.4. In summary, the algorithm AlwaysSafe relies on the fact that the underlying CMDP $\mathcal{M}$ is a member of Ξ with high probability, so it can test if the proposed ground policy is safe for all CMDPs in Ξ, and when this cannot be guaranteed, it executes π_A which is guaranteed to be safe to collect more data.

Theorem 4.4. *Given an abstract CMDP built according to a cost-model irrelevance abstraction and a fixed $\delta \in (0, 1)$, the algorithm AlwaysSafe equipped with policies π_A, π_T or π_α has no constraint violation regret with probability $1 - \delta$.*

Proof sketch. Theorem 4.1 is enough to show that AlwaysSafe with π_A will not violate the constraints. By definition the transition of the true CMDP belongs to the uncertainty set Ξ with probability $1 - \delta$. Since the expected cost of the policies π_T and π_α is less or equal to $\hat{c}$ in all CMDPs in Ξ, these policies are safe with probability $1 - \delta$. Details in the supplementary material [41]. □

5 EMPIRICAL RESULTS

For the empirical analysis, we selected three environments that showcase different features of the AlwaysSafe algorithm. While we provide more details about each environment in the supplementary material [41], here we present a summary of each one.

i). The **simple** CMDP was adapted from a problem proposed by Zheng and Ratliff [55], it has 3 states ($\mathbb{S} = \mathbb{N}_3$) and 2 actions: *stay* in the current state, which does not incur any cost or reward; or *move* to the next state, which incurs a cost of 1 and a reward equal to the current state index. We set $\hat{c} = 3$, $H = 6$, $K = 100$, and we ignore the state since the cost depends only on the action. In this way, the cost-model-irrelevance abstraction maps all states to a single state. Similar to Example 4.2, in this environment the reward depends on the ground state, so the optimal policy cannot be computed in the abstract state space. Therefore, this environment serves to

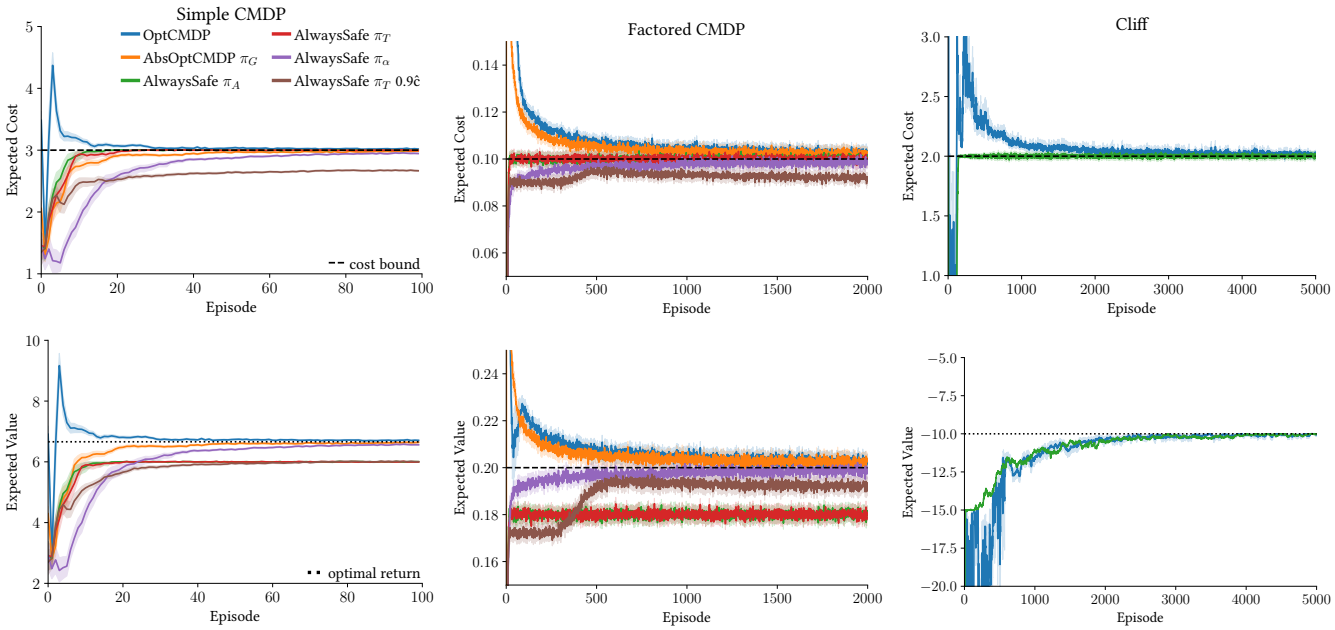

Figure 3: Results for the simple CMDP (left), factored CMDP (middle), and cliff environment (right), all averaged over 100 runs with a 95% confidence interval. The first row shows the expected cost of the policy executed in each episode while the second row shows the expected value of the policy (these values are estimated with 1000 simulations). A dashed line depicts the bound on the expected cost $\hat{c}$ and a dotted line depicts the optimal expected value.

check whether the algorithms based on the safety abstraction can compute an optimal policy.

ii). The **factored** CMDP from Figure 1 with $p=0.9$. We set $\hat{c}=0.1$, $H=2$, $K=5000$. We use the state abstraction that ignores the state variable y (Figure 1 right). This is a particularly challenging environment from a safety perspective, because an optimistic algorithm may underestimate the value of p, as discussed in Example 4.3, leading to an unsafe behavior.

iii). The **cliff walking**, a 4×6 grid-world where the agent must get to a goal position without falling off a cliff [45, Example 6.6]. We used the augmented version with a cost for walking close to the cliff [30]. We set $\hat{c}=2$, $H=15$, $K=5000$, and we do not ignore any variables. In this example, the cost function depends on both variables, so we need to use an identity function to define the abstraction. We set $\hat{c}=2$ and thereby ensure that an optimal policy is stochastic, as it needs to randomize between two longer paths.

Since the reward and cost functions in these environments are not in the interval $[0,1]$, we normalize the confidence intervals according to their spans. We also make them tighter to handle the large magnitude difference in the rewards of the cliff environment (details in the supplementary material [41]). Finally, we follow a doubling epoch schedule [28], where a new policy is computed only when one of the state-action counters doubles.

We evaluate the AlwaysSafe algorithm equipped with different policies from Section 4 (π_A, π_T and π_α with $\alpha=0.5$), plus an instance using π_T with an adjusted cost bound $0.9\hat{c}$. For the cliff environment, we only consider the algorithm AlwaysSafe π_A since $\mathbb{S}=\bar{\mathbb{S}}$ which implies $\pi_G=\pi_A$, making all the algorithms virtually the same. We use the algorithms OptCMDP and AbsOptCMDP π_G (Algorithm 1 equipped with a cost-model-irrelevant abstract CMDP $\bar{\mathcal{M}}$ and policy π_G) as baselines.

5.1 Analysis

We compare the algorithms in terms of constraint violations and performance regret. Figure 3 shows the expected cost (top row) and expected value (bottom row) of the policy in each episode.

We start the analysis with the simple CMDP (left column). We can observe that OptCMDP obtains policies with an expected value larger than the optimal constrained policy (bottom left), however, to do so it has to violate the constraints (top left). Although the algorithm AbsOptCMDP π_G has no safety guarantees, in this domain it converges to the optimal policy without violating the constraints.

As expected, all instances of the AlwaysSafe algorithm respect the cost constraint. However, only π_α converges to the optimal policy, while the others converged to a sub-optimal policy, indicating that the ground policy did not pass the safety test and the abstract policy was used. We notice that in the first episodes, AlwaysSafe π_α uses a conservative policy, which shows that while the confidence interval was loose, the final $\hat{c}$ was low enough to make the ground policy safe in the whole uncertainty set.

The experiments with the factored CMDP (middle column) show that AbsOptCMDP π_G is not safe. We can also see that AlwaysSafe π_T with $0.9\hat{c}$ changes from policy π_A to policy π_G after $\sim$ 500 episodes

but still does not reach the optimal performance, while the algorithm AlwaysSafe π_T always executes π_A. Only AlwaysSafe π_α safely reaches the optimal performance.

We conclude our analysis with the cliff environment (right column). We observe, to no surprise, that the AlwaysSafe π_A algorithm is able to always execute policies with expected cost lower than the given bound. The OptCMDP algorithm, on the other hand, violates the constraints for hundreds of episodes. Analyzing the expected value of the policies executed (bottom right), we notice that, in the cliff environment, the constraints on the expected cost are actually beneficial for the AlwaysSafe π_A algorithm, that accumulates a smaller regret in terms of performance as well.

5.2 Discussion

Following Ray et al. [37], we may conclude that the algorithm AlwaysSafe equipped with the safe policies π_A, π_T, π_α or π_T with $0.9\hat{c}$ dominates OptCMDP and AbsOptCMDP π_G since the former does not violate the constraints, while the latter does. Then AlwaysSafe π_α dominates AlwaysSafe π_A, AlwaysSafe π_T and AlwaysSafe π_T with $0.9\hat{c}$ since in general it achieves higher performance. Nevertheless, these results come with the requirement that the abstract model relevant for the safety is known. We believe that in cases where this model is only partially known, these algorithms would still be useful to make the most of the knowledge available.

In general, we notice that, on the one hand, the algorithms OptCMDP and AbsOptCMDP π_G eventually approach the safety bound, but might use unsafe policies during this process, which is a clear consequence of their optimism with respect to the transition and cost function. On the other hand, the AlwaysSafe algorithm equipped with safe policies sacrifices performance to ensure safety. However, when the cost function is well aligned with the reward function it can also have a smaller performance regret.

6 RELATED WORK

There are two popular directions in safe reinforcement learning [19]: (*i*) shaping the optimization criterion towards risk sensitivity [11] and (*ii*) changing the exploration process by, for instance, assuming the existence of a safe policy [55]. Our work is in the intersection of these directions since we employ external knowledge to modify the exploration process using a different optimization criterion.

Zheng and Ratliff [55] adapt the UCRL2 algorithm to CMDPs and assume that the full transition model of the MDP is known and the agent has access to a safe (baseline) policy. In contrast, we only require an abstraction of the transition model that is relevant for the cost function. Recently, HasanzadeZonuzy et al. [24] showed that the sample complexity of learning in CMDPs increases only logarithmically in comparison to unconstrained problems. However, their probably-approximately-correct (PAC) scheme does not provide any safety guarantees during the learning phase.

Furthermore, RL algorithms that provide guarantees of not violating the constraints during the learning phase include methods that model the environment dynamics using Gaussian processes [7, 48, 50, 51], design Lyapunov functions to guarantee the global constraints [12] or use analogies [38]. In general, these methods assume an initial safe policy to begin exploring, allowing the agents to slowly expand the set of known safe policies/states.

For problems with high-dimensional input spaces, different policy search algorithms have been proposed, that provide certain (though not hard) guarantees of not violating the constraints [2, 47, 53] or find safe policies only at the end of the training process [37]. In this setting, the safety constraint has also been generalized to consider the tail of the distribution of accumulated expected costs, instead of the mean [52].

Factored MDPs have been explored in different RL settings, developing algorithms with near-optimal regret bounds in factored MDPs without constraints [34] and with constraints [10], sample efficient algorithms [9, 15, 44] and off-policy policy evaluation [22], In the safety literature, factored MDPs have been used to reduce the sample complexity of batch RL algorithms with safety guarantees with respect to the performance of a baseline policy [42, 43] and to allow an agent to query a supervisor about the features of the factored MDP to avoid side effects [54].

Reachability constraints enforce policies to avoid catastrophic states. Fatemi et al. [18] avoid such states with high probability and Taleghan and Dietterich [46] look into deterministic policies that are easier to perceive than the usual randomized policies. Similarly, works from the formal methods community use reachability constraints and their extension, temporal logic constraints, to argue about safety during exploration using prior knowledge about the transition model [3, 23, 27, 29]. Finally, a control-theoretic simplex architecture has been employed to switch between safe and high-performance controllers [35].

7 CONCLUSIONS

This work considers settings where safety-relevant dynamics are given. We proposed the AlwaysSafe algorithm, that can be optimistic with respect to the reward, while ensuring safety at all times.

In particular, we used an abstract version of the safety-relevant dynamics to compute an abstract policy that is always safe and a ground policy that can achieve high performance. We showed how to switch between these two policies to find an algorithm that is safe and eventually converges to the optimal policy. This method not only enforces the agent to always act safely, but can also prune under-performing actions, improving the training efficiency when the cost function is aligned with the reward function.

Future work includes: finding new methods to devise the abstractions of the safety dynamics, for instance using core states [40]; investigating how the AlwaysSafe copes with an approximation of the abstract CMDP [1]; and developing new algorithms that can aggregate states online [33] without violating the constraints.

In summary, the proposed algorithm is always safe during the learning process, eventually reaches the optimal policy; and decouples exploration from safety issues in RL.

ACKNOWLEDGMENTS

Thanks to Canmanie Ponnambalam for fruitful discussions on the use of abstractions. This research is funded by the Netherlands Organisation for Scientific Research (NWO), as part of the Energy System Integration: planning, operations, and societal embedding program and the grants NWO OCENW.KLEIN.187: “Provably Correct Policies for Uncertain Partially Observable Markov Decision Processes” and NWA.1160.18.238: “PrimaVera”.

REFERENCES

[1] David Abel, D. Ellis Hershkowitz, and Michael L. Littman. 2016. Near Optimal Behavior via Approximate State Abstraction. In *Proceedings of the 33nd International Conference on Machine Learning*. JMLR.org, New York City, NY, USA, 2915–2923.

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, Sydney, NSW, Australia, 22–31.

[3] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, New Orleans, Louisiana, USA, 2669–2678.

[4] Eitan Altman. 1999. *Constrained Markov decision processes*. Vol. 7. CRC Press, Boca Raton, Florida, USA.

[5] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. 2016. Concrete Problems in AI Safety. arXiv:1606.06565

[6] Peter Auer and Ronald Ortner. 2006. Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning. In *Advances in Neural Information Processing Systems 19*. MIT Press, Vancouver, British Columbia, Canada, 49–56.

[7] Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. 2017. Safe Model-based Reinforcement Learning with Stability Guarantees. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., Long Beach, CA, USA, 908–918.

[8] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. 1995. Exploiting Structure in Policy Construction. In *Proc. Int. Joint Conf. on Artificial Intelligence*. Morgan Kaufmann, Montréal, Québec, Canada, 1104–1113.

[9] Doran Chakraborty and Peter Stone. 2011. Structure Learning in Ergodic Factored MDPs without Knowledge of the Transition Function's In-Degree. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, Madison, WI, USA, 737–744.

[10] Xiaoyu Chen, Jiachen Hu, Lihong Li, and Liwei Wang. 2021. Efficient Reinforcement Learning in Factored MDPs with Application to Constrained RL. In *International Conference on Learning Representations*. OpenReview.net, online, 10 pages.

[11] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. 2017. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *J. Mach. Learn. Res.* 18 (2017), 167:1–167:51.

[12] Yinlam Chow, Ofir Nachum, Edgar A. Duéñez-Guzmán, and Mohammad Ghavamzadeh. 2018. A Lyapunov-based Approach to Safe Reinforcement Learning. In *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., Montréal, Canada, 8103–8112.

[13] Frits de Nijs, Erwin Walraven, Mathijs Michiel de Weerdt, and Matthijs T. J. Spaan. 2017. Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI Press, San Francisco, California, USA, 3562–3568.

[14] Thomas Dean and Keiji Kanazawa. 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5, 3 (1989), 142–150.

[15] Carlos Diuk, Lihong Li, and Bethany R. Leffler. 2009. The Adaptive k-meteorologists Problem and Its Application to Structure Learning and Feature Selection in Reinforcement Learning. In *Proceedings of the 26th International Conference on Machine Learning*. ACM, Montreal, Quebec, Canada, 249–256.

[16] Gabriel Dulac-Arnold, Daniel J. Mankowitz, and Todd Hester. 2019. Challenges of Real-World Reinforcement Learning. arXiv:1904.12901 ICML Workshop on Reinforcement Learning for Real Life.

[17] Yonathan Efroni, Shie Mannor, and Matteo Pirotta. 2020. Exploration-Exploitation in Constrained MDPs. arXiv:2003.02189 ICML Workshop on Theoretical Foundations of Reinforcement Learning.

[18] Mehdi Fatemi, Shikhar Sharma, Harm Van Seijen, and Samira Ebrahimi Kahou. 2019. Dead-ends and Secure Exploration in Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, Long Beach, California, USA, 1873–1881.

[19] Javier García and Fernando Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* 16 (2015), 1437–1480.

[20] Robert Givan, Thomas L. Dean, and Matthew Greig. 2003. Equivalence notions and model minimization in Markov decision processes. *Artif. Intell.* 147, 1-2 (2003), 163–223.

[21] Nakul Gopalan, Marie desJardins, Michael L. Littman, James MacGlashan, Shawn Squire, Stefanie Tellex, John Winder, and Lawson L. S. Wong. 2017. Planning with Abstract Markov Decision Processes. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*. AAAI Press, Pittsburgh, Pennsylvania, USA, 480–488.

[22] Assaf Hallak, François Schnitzler, Timothy Arthur Mann, and Shie Mannor. 2015. Off-policy Model-based Learning under Unknown Factored Dynamics. In *Proceedings of the 32nd International Conference on Machine Learning*. JMLR.org, Lille, France, 711–719.

[23] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2020. Cautious Reinforcement Learning with Logical Constraints. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. IFAAMAS, Auckland, New Zealand, 483–491.

[24] Aria HasanzadeZonuzy, Dileep M. Kalathil, and Srinivas Shakkottai. 2021. Learning with Safety Constraints: Sample Complexity of Reinforcement Learning for Constrained MDPs. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, online.

[25] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. Amer. Statist. Assoc.* 58, 301 (1963), 13–30.

[26] Thomas Jaksch, Ronald Ortner, and Peter Auer. 2010. Near-optimal Regret Bounds for Reinforcement Learning. *J. Mach. Learn. Res.* 11 (2010), 1563–1600.

[27] Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. 2020. Safe Reinforcement Learning Using Probabilistic Shields (Invited Paper). In *CONCUR (LIPIcs, Vol. 171)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Vienna, Austria, 3:1–3:16.

[28] Chi Jin, Tiancheng Jin, Haipeng Luo, Suvrit Sra, and Tiancheng Yu. 2020. Learning Adversarial MDPs with Bandit Feedback and Unknown Transition. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Vienna, Austria, 1–10.

[29] Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. 2016. Safety-Constrained Reinforcement Learning for MDPs. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, Eindhoven, The Netherlands, 130–146.

[30] Jongmin Lee, Youngsoo Jang, Pascal Poupart, and Kee-Eung Kim. 2017. Constrained Bayesian Reinforcement Learning via Approximate Linear Programming. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. ijcai.org, Melbourne, Australia, 2088–2095.

[31] Lihong Li, Thomas J. Walsh, and Michael L. Littman. 2006. Towards a Unified Theory of State Abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics*. ISAIM, Fort Lauderdale, Florida, USA, 10 pages.

[32] Grigory Neustroev and Mathijs M. de Weerdt. 2020. Generalized Optimistic Q-Learning with Provable Efficiency. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, Auckland, New Zealand, 913–921.

[33] Ronald Ortner. 2013. Adaptive aggregation for reinforcement learning in average reward Markov decision processes. *Annals OR* 208, 1 (2013), 321–336.

[34] Ian Osband and Benjamin Van Roy. 2014. Near-optimal Reinforcement Learning in Factored MDPs. In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., Montreal, Quebec, Canada, 604–612.

[35] Dung T. Phan, Radu Grosu, Nils Jansen, Nicola Paoletti, Scott A. Smolka, and Scott D. Stoller. 2020. Neural Simplex Architecture. In *NFM (Lecture Notes in Computer Science, Vol. 12229)*. Springer, Switzerland, 97–114.

[36] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). John Wiley & Sons, Inc., New York, NY, USA.

[37] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning. https://github.com/openai/safety-gym

[38] Melrose Roderick, Vaishnavh Nagarajan, and J. Zico Kolter. 2021. Provably Safe PAC-MDP Exploration Using Analogies. In *International Conference on Artificial Intelligence and Statistics*. PMLR, online. arXiv:2007.03574

[39] Aviv Rosenberg and Yishay Mansour. 2019. Online Convex Optimization in Adversarial Markov Decision Processes. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, Long Beach, California, USA, 5478–5486.

[40] Roshan Shariff and Csaba Szepesvári. 2020. Efficient Planning in Large MDPs with Weak Linear Function Approximation. In *Advances in Neural Information Processing Systems 33*. Curran Associates, Inc., Vancouver, British Columbia, Canada, 12 pages.

[41] Thiago D. Simão, Nils Jansen, and Matthijs T. J. Spaan. 2021. AlwaysSafe: Reinforcement Learning without Safety Constraint Violations during Training — Supplementary Material. https://research.tudelft.nl/en/publications/alwayssafe-reinforcement-learning-without-safety-constraint-viola. , 4 pages.

[42] Thiago D. Simão and Matthijs T. J. Spaan. 2019. Safe Policy Improvement with Baseline Bootstrapping in Factored Environments. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press, Honolulu, Hawaii, USA, 4967–4974.

[43] Thiago D. Simão and Matthijs T. J. Spaan. 2019. Structure Learning for Safe Policy Improvement. In *Proc. Int. Joint Conf. on Artificial Intelligence*. ijcai.org, Macao, China, 3453–3459.

[44] Alexander L. Strehl, Carlos Diuk, and Michael L. Littman. 2007. Efficient Structure Learning in Factored-State MDPs. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, Vancouver, British Columbia, Canada, 645–650.

[45] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (2 ed.). MIT press, Cambridge, MA, USA.

[46] Majid Alkaee Taleghan and Thomas G. Dietterich. 2018. Efficient Exploration for Constrained MDPs. In *2018 AAAI Spring Symposia*. AAAI Press, Palo Alto, California, USA, 313–319.

[47] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. 2019. Reward Constrained Policy Optimization. In *International Conference on Learning Representations*. OpenReview.net, New Orleans, LA, USA, 11 pages.

[48] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. 2019. Safe Exploration for Interactive Machine Learning. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., Vancouver, BC, Canada, 2887–2897.

[49] Harm van Seijen, Shimon Whiteson, and Leon J. H. M. Kester. 2014. Efficient Abstraction Selection in Reinforcement Learning. *Comput. Intell.* 30, 4 (2014), 657–699.

[50] Akifumi Wachi and Yanan Sui. 2020. Safe Reinforcement Learning in Constrained Markov Decision Processes. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Vienna, Austria, 9797–9806.

[51] Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. 2018. Safe Exploration and Optimization of Constrained MDPs Using Gaussian Processes. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, New Orleans, Louisiana, USA, 6548–6556.

[52] Qisong Yang, Thiago D. Simão, Simon H. Tindemans, and Matthijs T. J. Spaan. 2021. WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, online.

[53] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. 2020. Projection-Based Constrained Policy Optimization. In *8th International Conference on Learning Representations*. OpenReview.net, Addis Ababa, Ethiopia, 1–11.

[54] Shun Zhang, Edmund H. Durfee, and Satinder P. Singh. 2018. Minimax-Regret Querying on Side Effects for Safe Optimality in Factored Markov Decision Processes. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. ijcai.org, Stockholm, Sweden, 4867–4873.

[55] Liyuan Zheng and Lillian Ratliff. 2020. Constrained Upper Confidence Reinforcement Learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. PMLR, online, 620–629.