No More Hand-Tuning Rewards: Masked Constrained Policy Optimization for Safe Reinforcement Learning

Stef Van Havermaet
IDLab, Ghent University - imec
stef.vanhavermaet@ugent.be

Yara Khaluf
IDLab, Ghent University - imec
yara.khaluf@ugent.be

Pieter Simoens
IDLab, Ghent University - imec
pieter.simoens@ugent.be

ABSTRACT

In safe Reinforcement Learning (RL), the agent attempts to find policies which maximize the expectation of accumulated rewards and guarantee its safety to remain above a given threshold. Hence, it is straightforward to formalize safe RL problems by both a reward function and a safety constraint. We define safety as the probability of survival in environments where taking risky actions could lead to early termination of the task. Although the optimization problem is already constrained by a safety threshold, reward signals related to unsafe terminal states influence the original maximization objective of the task. Selecting the appropriate value of these signals is often a time consuming and challenging reward engineering task, which requires expert knowledge of the domain.

This paper presents a safe RL algorithm, called Masked Constrained Policy Optimization (MCPO), in which the learning process is constrained by safety and excludes the unsafe reward signals. We develop MCPO as an extension of gradient-based policy search methods, in which the updates of the policy and the expected reward models are masked. Our method benefits from having a high probability of satisfying the given constraints for every policy in the learning process. We validate the proposed algorithm in two continuous tasks. Our findings prove the proposed algorithm is able to neglect unsafe reward signals, and thereby resolving the desired safety-performance trade-off without having the need for hand-tuning rewards.

KEYWORDS

Safe Reinforcement Learning; Constrained Policy Optimization; Reward Engineering

ACM Reference Format:
Stef Van Havermaet, Yara Khaluf, and Pieter Simoens. 2021. No More Hand-Tuning Rewards: Masked Constrained Policy Optimization for Safe Reinforcement Learning. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

In reinforcement learning (RL), agents in an unknown environment learn how to act in order to maximize the expected long-term return on the basis of a real valued reward signal. A particular way of acting is referred to as a policy that is formally defined as a mapping from environmental observations to the probabilities of selecting actions. In several real-world tasks, some actions can be harmful to the agent itself or to others in the same environment. In such cases, the learning process should consider both the rewards and the potential risk in the environment. Safe RL methods aim to maximize the expectation of accumulated rewards, while respecting safety constraints during the learning and deployment process [9]. These methods can be categorized into two approaches: (i) modifying the exploration strategy, and (ii) modifying the optimization criterion.

Exploration strategies describe the balance between exploring the environment to collect more information, and choosing actions that tend to be most rewarding, given the current information. Modification of the exploration strategy to avoid risky situations can be achieved through the incorporation of (i) external knowledge, or (ii) a risk metric. An example of the first approach is provided in [1], where the agent that is involved in a helicopter flying task is provided with a set of demonstrations from an instructor. Using these demonstrations, the agent learns a model of the helicopter dynamics and derives a safe policy in an off-line manner. Instructors can provide advice to assist agents' exploration also during the online learning processes [8]. Surely, these techniques may suffer from biases that are introduced by the instructor. The second approach encourages the agent to explore certain regions of the environment by introducing a safety metric as an exploration bonus [11]. Learning safe policies by this approach is not always reliable, considering the safety metric is often not correctly approximated in the early steps of the learning process [9].

Modification of the optimization criterion, i.e. the expected value of accumulated rewards, can be categorized into three methods. In the first, the policy is optimized over its worst case return [10]. Hence, this method tends to produce overly conservative policies, and lacks the ability of managing the trade-off between safety and performance. The second method redefines the maximization objective as a weighted formulation of both the reward signals and the potential risk observed in the environment. Despite improving transparency to the safety-performance trade-off, this method requires task-specific expert knowledge to find the appropriate weights of the objective function in order to satisfy given safety constraints [14, 15, 22]. In addition, the policy model might become trapped in plateaus (i.e. saddle points or local minima) during learning, due to the implicit interference between the two separate goals in the same objective function [18]. The last method avoids these issues by incorporating safety via constraints. The standard formulation for these RL techniques is the constrained Markov Decision Process (CMDP) framework [3], where the learned policies must satisfy constraints on expectations of auxiliary costs. In case actions are continuous and possibly high-dimensional, state-of-the-art approaches are based on policy search methods [5]. Heuristic algorithms for policy search in CMDPs have been proposed [23]. Furthermore, approaches based on primal-dual methods [4, 17] have been shown to converge to constraint-satisfying policies. However,

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.

these policy search methods do not guarantee constraint satisfaction throughout the learning process. Constrained Policy Optimization (CPO) is a trust region policy search method for CMDPs, which provides such guarantees by limiting the influence of approximation errors in the learning process [2]. This is achieved by restricting the size of the change applied to the policy at each iteration.

In this paper, we study safe RL in a continuous CMDP where the policy safety constraints are defined by a lower-bound on the probability to remain in a predefined set of safe states. We follow [12] in defining an unsafe state as a terminal state that is not part of the agent's objective to complete its task. For example, in a navigation task of a difficult terrain, the unsafe states correspond to an unmanned ground vehicle being jammed. We showcase how the reward signals associated to unsafe states influence the degree of safety of the optimal policy, although the optimization criterion is already constrained by a safety threshold. Therefore, the unsafe reward signals require to be hand-tuned, which is a highly demanding task only suitable to domain-specific experts. The optimization criterion is thus preferably unmodified by the unsafe rewards. We propose a novel approach, named *Masked Constrained Policy Optimization* (MCPO), which excludes the unsafe reward signals from the maximization objective. This is achieved by using the method of *masking*, which zeroes certain values based on a predicate being true or false. The contribution of this paper is (i) an illustration of how the optimization criterion of a CMDP is redundantly modified by unsafe reward signals, and (ii) a proposed algorithm to free RL task designers from the unnecessary time-consuming and challenging task of hand-tuning unsafe reward signals

The rest of the paper is organized as follows: section 2 formalizes the problem of selecting values of unsafe reward signals in a constrained optimization problem, and section 3 illustrates this by a toy example. Afterwards, MCPO is presented in section 4 as a safe policy learning method, based on the concepts of importance sampling, trust region policy optimization and constrained policy optimization. A comparison between the performance of CPO and MCPO in two continuous navigation tasks is then provided in section 5. Finally, concluding remarks are presented in section 6.

2 PROBLEM FORMULATION

2.1 Preliminaries

We formalize the interaction between the agent and the environment by a Markov Decision Process (MDP), defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the transition probability function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$ is the reward function with $\mathcal{R} \subset \mathbb{R}$. At each time step t, the agent perceives a state $s_t \in \mathcal{S}$ and selects an action $a_t \sim \pi(\cdot \mid S_t)$ where $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$ is the stochastic policy. The agent transitions into the next state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ and receives a reward $r_{t+1} = R(s_t, a_t, s_{t+1})$. A trajectory $\tau \doteq \{s_0, a_0, r_1, s_1, a_1, r_2, ...\}$ is the time sequence of states, actions and rewards during one episode, which ends when a terminal state is reached.

In a RL control task, we aim to find a policy π which maximizes the objective function $J(\pi)$ of the rewards accumulated over a trajectory generated by π. The objective function is often defined as the expectation of the discounted cumulative rewards, i.e. $J(\pi) \doteq \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_{t+1}\right]$ with $T \geq 0$ as the time of termination and $\gamma \in [0,1)$ as the discount factor. T is random variable which defines the duration of an episode. The discount factor essentially determines how much rewards in the distant future, relative to those in the immediate future, are of interest to the agent.

In order to express the difference in performance between two policies, let us define the return as $G_t \doteq \sum_{k=0}^{T-t} \gamma^k r_{t+1+k}$. The value function of a state s under a policy π is then defined as $V^\pi(s) \doteq \mathbb{E}_{\tau \sim \pi}[G_t \mid s_t = s]$, the action-value function as $Q^\pi(s,a) \doteq \mathbb{E}_{\tau \sim \pi}[G_t \mid s_t = s, a_t = a]$, and the advantage function, which expresses the advantage of taking action a compared to the average action taken by policy π in state s, as $A^\pi(s,a) \doteq Q^\pi(s,a) - V^\pi(s)$. The action-value function can be written in terms of V^π as $Q^\pi(s,a) = \mathbb{E}_{s' \sim P}[R(s,a,s') + \gamma V^\pi(s')]$. From this follows

$$A^\pi(s,a) = \mathbb{E}_{s' \sim P}\left[R(s,a,s') + \gamma V^\pi(s') - V^\pi(s)\right] \tag{1}$$

and thereby

$$J(\pi_{k+1}) = J(\pi_k) + \mathbb{E}_{\tau \sim \pi_{k+1}}\left[\sum_{t=0}^{T} \gamma^t A^{\pi_k}(s_t, a_t)\right] \tag{2}$$

with π_{k+1} and π_k as two arbitrary policies. For a formal proof of Equation (2), see [13].

2.2 Safe Policies

In this paper, we study RL tasks in a *risk*-embedded MDP, where we define risk as the probability of ending an episode in an *unsafe* state [12]. Formally, let $s^\triangle$ denote the unsafe terminal state which the agent is required to avoid with a certain probability. Consider, for example, a self-driving vehicle learning to remain in the center of its lane [24]. Some actions may lead to damage itself. Such an episode does not end by reaching the *goal* state, but rather having the agent nonfunctional due to the agent entering a state $s^\triangle$. Consequently, we refer to a MDP as risk-embedded if and only if $\exists\, s, a : p(s,a,s^\triangle) > 0$, and define the level of safety from following a policy as follows:

$$\textit{A policy } \pi \textit{ is } \delta\textit{-safe} \iff \mathbb{E}_{\tau \sim \pi}\left[\Pr(s^\triangle \in \tau)\right] \leq 1 - \delta \tag{3}$$

A common approach to learn δ-safe policies is to redefine the maximization objective as a weighted combination of the accumulated reward and the notion of safety:

$$J_\omega(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t \left(r_{t+1} + \omega \Pr(s^\triangle \in \tau)\right)\right] \tag{4}$$

with ω as a parameter that determines the trade-off between the task rewards and the potential risk. However, selecting the value of ω is a highly challenging task, even for domain experts, since the relation between ω and δ is far from straightforward.

As an alternative approach to learning δ-safe policies, the risk-embedded MDP is augmented to a constrained Markov Decision Process (CMDP) by defining an auxiliary cost function $C : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to C$ with $C \subset \mathbb{R}$. Specifically, the cost function is defined as $C(s,a,s') \doteq \mathbb{1}\left(s' = s^\triangle\right)$ with $\mathbb{1}$ as the indicator function such that $\mathbb{1}\left(s' = s^\triangle\right) = 1$ if $s' = s^\triangle$ holds and zero otherwise. The value, action-value, and advantage functions for the costs are analogous to those of the rewards, and respectively denoted by V_C^π, Q_C^π, and A_C^π. Similar to the objective of sampled rewards, $J_C(\pi)$ denotes

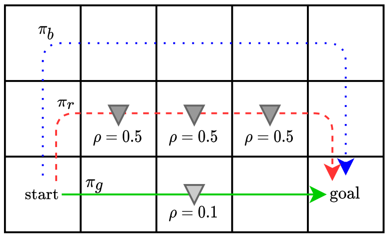


Figure 1: A risk-embedded navigation environment with exemplary trajectories shown.

the expectation of the discounted cumulative costs, i.e. $J_C(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma_c^t c_{t+1}\right]$ with $c_{t+1} = C(s_t, a_t, s_{t+1})$. An optimal δ-safe policy π^* would then satisfy the following constrained optimization problem:

$$\pi^* = \arg\max_{\pi} J(\pi) \quad \text{s.t. } J_C(\pi) \leq 1 - \delta \tag{5}$$

Although the safety costs are excluded from the maximization objective in CMDPs (5), a similar issue to that of Equation (4) occurs for learning in risk-embedded MDPs. For the purpose of embedding risk into a MDP, the unsafe terminal state $s^\triangle$ was introduced. Consequently, the original task reward function should be extended with a mapping $R^\triangle$ from unsafe transition tuples $\langle s, a, s^\triangle\rangle$ to rewards. However, the implementation of $R^\triangle$ is equivalent to the selection of the parameter ω in Equation (4), and thus it suffers from the same drawbacks previously mentioned. In fact, the mapping $R^\triangle$ influences the balance between safety and task completion, and when incorrectly defined, the agent learns suboptimal policies. We illustrate this in the following section.

3 PROBLEM ILLUSTRATION

Figure 1 shows the environment of a risk-embedded navigation task in which the goal for the agent is to learn the shortest δ-safe path between two given positions. Each tile in the grid represents a state. The agent's start and goal states are respectively placed in the bottom left and right of the grid. The agent is capable of moving to a neighboring tile by one of its four actions: up, down, left, right. Tiles with a gray triangle $\triangledown$ represent states with a probability $\rho > 0$ of transitioning to $s^\triangle$ for any action. Transitions from other states are deterministic. An episode is terminated upon reaching either the goal or an unsafe state. As a navigation task, the agent should find the shortest path to the goal state. Therefore, the original reward function is defined as $\forall s, s' \in \mathcal{S}^\triangledown, \forall a \in \mathcal{A} : r(s, a, s') = -1$ with $\mathcal{S}^\triangledown = \mathcal{S} \setminus s^\triangle$. We consider three trajectories $\{\tau_g, \tau_r, \tau_b\}$ (green solid, red dashed, blue dotted) sampled from different δ-safe deterministic policies $\{\pi_g, \pi_r, \pi_b\}$ respectively, for $\delta = 0.1$. The ordering by risk of the policies is $J_C(\pi_b) < J_C(\pi_g) < J_C(\pi_r) \leq 1 - \delta$. Since all three policies are δ-safe, the agent should find the optimal policy $\pi^* = \pi_g$ in the RL problem as defined in Equation (5).

In what follows, we demonstrate how the value of the reward signal given to the agent upon entering the unsafe state may lead to the learning of suboptimal policies. Let $r^\triangle$ denote this signal, i.e. $R(s, a, s^\triangle) = r^\triangle$ for all s, a. Table 1 shows the task performance of the three policies for different values of $r^\triangle$. A naive approach is to provide the agent with a reward of zero upon termination due to an encountered risk, as followed in [6]. In this case, the agent would learn to behave according to policy π_r. Since the original reward function is strictly negative, the agent will seek to obtain $r^\triangle = 0$ as much as allowed, and will consequently maximize the safety constraint objective $J_C(\pi)$ towards its upper bound. Another common approach is to set $r^\triangle$ to large negative values, essentially modifying the optimization criterion as risk-averse [10]. A well-known example of this approach in RL literature, is the *Cliff Walking* task [21]. For instance, let $r^\triangle = -100$, in this case, the agent would find the overly conservative policy π_b as a solution to Equation (5). Finally, the optimal δ-safe path π_g is found for $r^\triangle = -10$: which was carefully selected based on the expected rewards and costs obtained by each policy. Hence, in order to select a value for $r^\triangle$ that enables finding the optimal δ-safe policy, the task designer has to know such policy in advance. In the next section, we propose a novel approach to learn optimal δ-safe policies, while avoiding the selection process of such reward signals.

Table 1: Performance of the respective policies shown in Figure 1 for different settings of the unsafe reward signal $r^\triangle$.

$r^\triangle$	$J(\pi_g)$	$J(\pi_r)$	$J(\pi_b)$
0	-3.8	**-3**	-8
-100	-13.8	-90.5	**-8**
-10	**-4.8**	-11.75	-8

4 SAFE POLICY LEARNING

Policy search methods rely on finding a parameterization θ^* of the policy that maximizes the expected return:

$$\theta^* \doteq \arg\max_{\theta} J(\pi_\theta)$$

Gradient-based methods achieve this by following policies with the steepest increase in rewards, based on the policy gradient $\mathbf{g}$.

4.1 Importance Sampling

As the policy is changed at every iteration, new samples are collected and samples by older policies would not be reusable. To overcome this sampling inefficiency, the maximization objective can be rewritten to use samples from an old policy π_{old}. Importance sampling proposes the following surrogate objective, derived from from Equation (2), to appropriately weight old samples [16]:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi_{\text{old}}}\left[\sum_{t=0}^{T}\gamma^t \frac{\pi(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)} A^{\pi_{\text{old}}}(s_t, a_t)\right] \tag{6}$$

since $\max_\pi J(\pi) = \max_\pi J(\pi) - J(\pi_{\text{old}})$, and the probability ratio between π and π_{old} alters the advantages appropriately.

4.2 Trust Region Policy Optimization

Optimizing policies by taking updates of the form $\theta_{k+1} = \theta_k + \alpha\mathbf{g}$ may suffer from performance collapse when the learning rate α allows for update steps that are too large [5]. Trust Region Policy Optimization (TRPO) addresses this issue by maximizing $J(\pi)$ over a local neighborhood of the most recent policy π_{θ_k} [19]:

$$\begin{aligned}&\max_{\theta} J(\pi_\theta)\\ &\quad \text{s.t.}\ \mathbb{E}_{\tau\sim\pi_{\theta_k}}\left[D_{\mathrm{KL}}\left(\pi_\theta(s), \pi_{\theta_k}(s)\right)\right] \le \kappa\end{aligned} \tag{7}$$

where $\pi_{\text{old}} = \pi_{\theta_k}$ in Equation (6), D_{KL} is the Kullback-Leibler (KL) divergence to measure how different π_θ is from π_{θ_k}, and $\kappa > 0$ is a step size parameter which controls how much the policy is allowed to change per iteration. The set of policies which satisfy the constraint is called the trust region. Although the trust region is determined by approximation, TRPO generally provides monotonic improvements of the policy throughout the learning process [19].

4.3 Constrained Policy Optimization

In order to restrict the optimization of π_θ with respect to the safety probabilities defined in (3), the constrained optimization problems of (5) and (7) are combined as follows

$$\begin{aligned}&\max_{\theta} J(\pi_\theta)\\ &\quad\text{s.t.}\begin{cases} J_C(\pi_\theta) \le 1-\delta\\ \mathbb{E}_{\tau\sim\pi_{\theta_k}}\left[D_{\mathrm{KL}}\left(\pi_\theta(s), \pi_{\theta_k}(s)\right)\right] \le \kappa\end{cases}\end{aligned}$$

which can be rewritten in terms of expectations based on Equations (2) and (6) as:

$$\begin{aligned}&\max_{\theta} \mathbb{E}_{\tau\sim\pi_{\theta_k}}\left[\sum_{t=0}^{T}\gamma^t \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_k}(a_t \mid s_t)} A^{\pi_{\theta_k}}(s_t, a_t)\right]\\ &\quad\text{s.t.}\begin{cases} J_C(\pi_{\theta_k}) + \mathbb{E}_{\tau\sim\pi_{\theta_k}}\left[\sum_{t=0}^{T}\gamma^t \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A_C^{\pi_{\theta_k}}(s_t,a_t)\right] \le 1-\delta\\ \mathbb{E}_{\tau\sim\pi_{\theta_k}}\left[D_{\mathrm{KL}}\left(\pi_\theta(s), \pi_{\theta_k}(s)\right)\right] \le \kappa\end{cases}\end{aligned} \tag{8}$$

Constrained Policy Optimization (CPO) solves (8) at each iteration in the learning process, where expectations are estimated with samples [2]. For high-dimensional policy parameter spaces (e.g. the weights within neural networks), CPO relies on approximation of (8), with sufficiently small step sizes κ, in order to remain computationally efficient. The objective function $J(\pi_\theta)$ and the safety cost function $J_C(\pi_\theta)$ are approximated through linearization around π_{θ_k}. The average KL-divergence is approximated by the second order expansion at $\theta = \theta_k$, for which the KL-divergence and its gradient are both zero. The local approximation to (8) is then:

$$\begin{aligned}&\max_{\theta} \mathbf{g}^\top(\theta-\theta_k)\\ &\quad\text{s.t.}\begin{cases} J_C(\pi_{\theta_k}) + \mathbf{g}_c^\top(\theta-\theta_k) \le 1-\delta\\ \frac{1}{2}(\theta-\theta_k)^\top \mathbf{H}(\theta-\theta_k) \le \kappa\end{cases}\end{aligned} \tag{9}$$

where $\mathbf{g}$ is the gradient of the objective, $\mathbf{g}_c$ is the gradient of the safety cost, and $\mathbf{H}$ is the Hessian of the expected KL-divergence. In case the constrained optimization problem is feasible, it can be efficiently solved using duality, since the problem is convex as $\mathbf{H}$ is always positive semi-definite (and CPO assumes it to be positive-definite in the following). Letting λ and ν denote the Lagrange multipliers, the dual to (9) can be expressed as

$$\max_{\substack{\lambda\ge 0\\ \nu\ge 0}} \frac{-1}{2\lambda}\left(\mathbf{g}^\top\mathbf{H}^{-1}\mathbf{g} - 2\nu\mathbf{g}^\top\mathbf{H}^{-1}\mathbf{g}_c + \nu^2\mathbf{g}_c^\top\mathbf{H}^{-1}\mathbf{g}_c\right) + \nu d - \frac{\lambda\kappa}{2} \tag{10}$$

where $d \doteq J_C(\pi_{\theta_k}) - (1-\delta)$. The policy update is then defined as

$$\theta = \theta_k + \frac{1}{\lambda^*}\mathbf{H}^{-1}(\mathbf{g} - \mathbf{g}_c\nu^*) \tag{11}$$

with λ^* and ν^* as a solution to the dual defined in Equation (10). For proof of (11), see [2]. However, due to approximation errors or bad initialization of the policy parameters, CPO may be unable to find a feasible solution to the dual problem. In such cases, the policy update of (11) is replaced by the following update

$$\theta = \theta_k - \sqrt{\frac{2\kappa}{\mathbf{g}_c^\top\mathbf{H}^{-1}\mathbf{g}_c}}\mathbf{H}^{-1}\mathbf{g}_c \tag{12}$$

in which no variable associated to the maximization objective is present, as the update is solely focused on decreasing the constraint value in order to produce a feasible policy. Finally, CPO applies backtracking line search to the policy proposal found by (11) or (12) to overcome the approximation errors in satisfaction of sample estimates of the constraints.

4.4 Masked Constrained Policy Optimization

By applying CPO, the agent is restricted to learning δ-safe policies through the safety constraint. The maximization objective, however, also involves the reward signals obtained upon entering unsafe states. As shown in section 3, these unsafe rewards redundantly define the level of safety of the optimal policy, and need to be carefully hand-tuned. Hence, we propose a novel approach to exclude these signals from the maximization objective by using boolean *masking*—i.e. to set a value to zero based on a predicate being true or false. Masked Constrained Policy Optimization (MCPO) defines a mask on the reward advantage function. First, we will describe our method for advantages estimated by a Generalized Advantage Estimator (GAE), which has been shown to provide efficient learning of highly challenging continuous control tasks [20]. Later in this section, other advantage estimators are discussed.

4.4.1 *Generalized Advantage Estimator.*

Let $\hat{A}^\pi$ and $\hat{V}^\pi$ denote the estimators of the advantage and value functions respectively. The temporal difference (TD) residual of $\hat{V}^\pi$ is defined as $\Delta_t^\pi \doteq r_{t+1} + \gamma\hat{V}^\pi(s_{t+1}) - \hat{V}^\pi(s_t)$ [21]. Note that Δ_t^π can be considered as an estimate of the advantage of action a_t by Equation (1). The GAE-Λ is then defined as a discounted sum of the TD residuals accumulated along a trajectory:

$$\hat{A}^\pi(s_t, a_t) \doteq \sum_{k=0}^{T-t}(\gamma\Lambda)^k\Delta_{t+k}^\pi \tag{13}$$

with $0 < \Lambda < 1$ as a parameter that controls the compromise between bias and variance.

4.4.2 *Masking the GAE-Λ.*

The maximization objective of the constrained optimization problem as defined in Equation (8), solely relies on the expected advantages as feedback from the MDP. Therefore, our approach relies on masking the advantage estimator, which is achieved by two steps. First, the TD residuals based on an unsafe reward are set to zero. For this purpose, Equation (13) is redefined as follows:

$$\hat{A}^\pi(s_t, a_t) \doteq \sum_{k=0}^{T-t}(\gamma\Lambda)^k\mathbb{1}(s_{t+k+1} \neq s^\triangle)\Delta_{t+k}^\pi \tag{14}$$

Algorithm 1: Masked Constrained Policy Optimization

```
Input: arbitrary initial policy parameters $\theta_0$
for each iteration $k = 0, 1, 2, \ldots$ do
    Sample a set of trajectories $\mathcal{T} = \{\tau\} \sim \pi_k = \pi(\theta_k)$
    Compute estimate of g based on $\hat{A}^{\pi_k}$ by (14) and (15) from $\mathcal{T}$
    Compute estimates of $\mathbf{g}_c$, $\mathbf{H}$, and $d$ from $\mathcal{T}$
    if (9) is feasible then
        Solve (10) for $\lambda^*_k$ and $\nu^*_k$
        Compute $\theta'$ by (11)
    else
        Compute $\theta'$ by (12)
    end
    Compute $\theta_{k+1}$ by backtracking line search with $\theta'$ as starting position and satisfying the constraints in (8) based on $\mathcal{T}$.
    Update $\hat{R}^{\pi}_i$ based on prediction errors (16) from $\mathcal{T}$.
    Update $V^{\pi}_c$ to discounted cumulative costs from $\mathcal{T}$.
end
```

Secondly, the unsafe reward should be excluded from the expectation of accumulated future rewards in $\hat{V}^{\pi}$. This is not as straightforward as substituting the reward to zero, as illustrated in section 3. Let us define the expected reward at time step $t + k$, given the agent was in state s at time t and selected action $a_t \sim \pi(s_t)$, as $R^{\pi}_k(s) \doteq \mathbb{E}_{a_{t+k}\sim\pi}\left[r_{t+k+1} \mid s_t = s\right]$, and denote its estimator by $\hat{R}^{\pi}_k$. The estimated value function can then be rewritten as the discounted sum of reward estimates:

$$\hat{V}^{\pi}(s) = \sum_{k=0}^{T'} \gamma^k \hat{R}^{\pi}_k(s) \tag{15}$$

with T' as the upper bound of the time horizon, i.e. $T \in [0, T']$, which may be infinite. As T is a random variable, the rewards in the set $\{r_{t+k+1} \mid T - t < k \leq T' - t\}$ are equal to zero. We now describe how the loss function of each reward estimator is masked. Let $E^{\pi}_k(s)$ denote the prediction error between the estimate value $\hat{R}^{\pi}_k(s)$ and a reward sample r_{t+k+1} from a trajectory $\tau \sim \pi$ where $s_t = s$ holds for t. The errors are discarded (i.e. set to zero) when unsafe state-actions have caused termination without task completion. The masked prediction error is defined as follows:

$$E^{\pi}_k(s) \doteq \mathbb{1}\left(\forall\, l \leq k : s_{t+l+1} \neq s^{\triangle} \mid s_t = s\right)\left(\hat{R}^{\pi}_k(s) - r_{t+k+1}\right) \tag{16}$$

The parameters of the reward estimators are then optimized by minimizing a loss function based on Equation (16), such as the mean square error $\frac{1}{N}\sum_i^N \frac{1}{T_i}\sum_t^{T_i}\left[E^{\pi}_k(s_t)\right]^2$ with N as the number of sampled trajectories. The pseudocode of our proposed algorithm is given in Algorithm 1.

4.4.3 Practical Implementation. Learning all reward estimators $\{\hat{R}^{\pi}_k\}_{0:T'}$ would often be infeasible as T' is large in practice. For this purpose, we limit the estimation of singular rewards for n time steps, and bootstrap from the value of the state $n + 1$ steps later as an estimation for the discounted cumulative sum of the remaining rewards [21]. The value function estimator in Equation (15) can be rewritten as follows:

$$\hat{V}^{\pi}(s) = \sum_{k=0}^{n-1}\left[\gamma^k \hat{R}^{\pi}_k(s)\right] + \mathbb{E}_{a_{t+n}\sim\pi}\left[\hat{V}^{\pi}(s_{t+n+1}) \mid s_t = s\right] \tag{17}$$

with $n \geq 1$. Let $\hat{R}^{\pi}_n$ denote the estimator of the bootstrap term.

While each reward is defined to be estimated by distinct sets of parameters, using a shared set of parameters would serve as a more practical approach. For instance, a single neural network with n output nodes, each estimating a reward at a different time step, would share the parameters of all hidden layers.

Furthermore, we note that it may be impractical for the implementation of MCPO to rely on identification of the unsafe state. A more efficient approach is to compute (14) and (16) based on the safety costs of the CMDP, given that $\mathbb{1}(s_{t+1} = s^{\triangle}) \equiv \mathbb{1}(c_{t+1} = 1)$.

4.4.4 Other Advantage Estimators. We now discuss the extension of our proposed method to the following estimators of $A^{\pi}(s_t, a_t)$ (other than GAE), which do not introduce bias in Equation (6) to estimate the objective [20]:

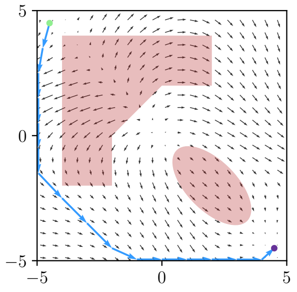
- $\hat{Q}^{\pi}(s_t, a_t)$, which can be written as the discounted sum of reward estimates, equivalent to Equation (15). The prediction errors of reward estimators are masked as described beforehand.
- $\hat{Q}^{\pi}(s_t, a_t) - \hat{V}^{\pi}(s_t)$, where masking is applied to the prediction errors of the two sets of reward estimators (one defined by input of state-action, and the other only by state).
- G_t sampled with π, which should be dismissed if the episode terminated in an unsafe state. Evidently, many samples are likely to be disregarded when there is high risk in the environment. We therefore expect this method to exhibit lower sampling efficiency compared to other masked estimators.

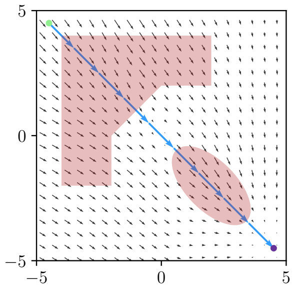
4.4.5 Limitations. A state s can be expressed as highly unsafe for a policy π when the probability of succumbing to a risk, starting from s and following π thereafter, is close to one. It could be that some reward estimators are rarely updated when the likelihood of unsafe states under π is high, due to the masking of the prediction error. Let $\{\hat{R}^{\pi}_h\}$ denote the set of such estimators, then the agent would consistently terminate as a result of risk before h time steps have passed in a single episode. This can be expressed by:

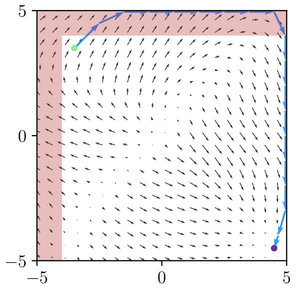
$$\mathbb{E}_{\tau\sim\pi}\left[\Pr(\exists\, l \leq h : s_{t+l+1} = s^{\triangle} \mid s_t = s)\right] = 1 - \varepsilon \Rightarrow \mathbb{E}\left[E^{\pi}_h(s)\right] \approx 0$$

with ε as a very small number. However, for l to exist in the equation formulated above, the trajectories would have to be sampled from a policy π that is δ'-safe with δ' very close to zero. In case the agent is learning δ-safe policies with $\delta > \delta'$, a bad policy update must have occurred due to approximation errors. The computation of new policy parameters will not rely on estimates of the advantages, and therefore the lack of updates of some reward estimators would have no effect on the new policy update. On the other hand, if the agent is learning to satisfy safety by a threshold of $\delta \leq \delta'$, the constraint defined by the CMDP in (5) is satisfied for any policy. The optimization problem can be solved by standard policy search methods, and thus we argue that it would be no longer a matter of safe policy learning.

4.4.6 Performance. MCPO aims to optimize the performance $J^{\triangledown}$ defined as the expected return of only the trajectories that were not


(a) CPO in LakeWorld


(b) MCPO in LakeWorld


(c) CPO in CoastWorld

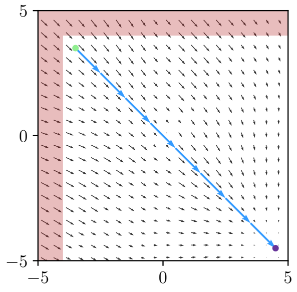
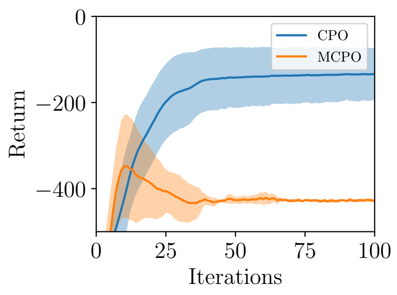
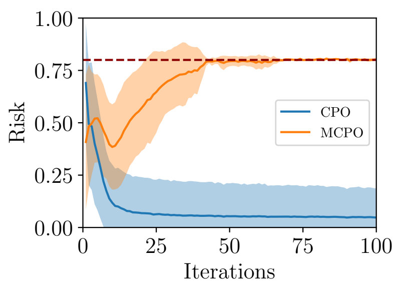

(d) MCPO in CoastWorld

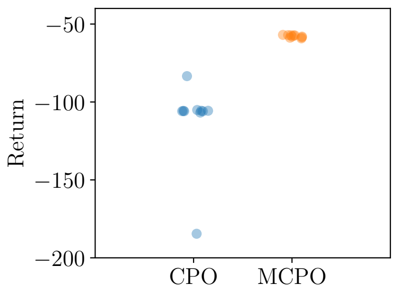
Figure 2: Navigation policies learned after 100 iterations with a random starting position in each episode. A trajectory from top left (-4.5, 4.5) to bottom right (4.5, 4.5) is sampled from the learned Gaussian policy in a deterministic manner (i.e. by its mean vector)



(a) Returns



(b) Risk



(c) Returns of completed trajectories

Figure 3: CPO and MCPO in LakeWorld with an unsafe reward signal of -500.

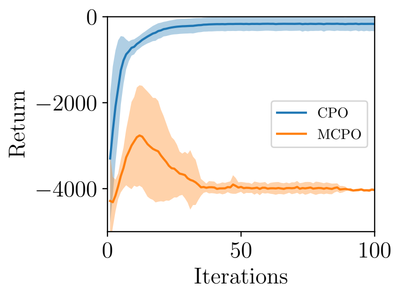
terminated by accessing the unsafe state. The relative difference in performance between CPO and MCPO therefore depends on the values of the safety threshold parameter δ and the unsafe reward signal $r^{\triangle}$. For instance, if $r^{\triangle} = 0$, then CPO could only achieve δJ^{∇} of MCPO's performance in the worst-case where $r^{\triangle}$ deviates significantly from its appropriate value. Even for high values of δ (i.e. the agent is learning to be risk-averse), the loss may be intolerable due to the critical nature of the performance measure in the specific application. In the following section, we showcase this performance difference through means of simulation experiments for different cases of $\langle \delta, r^{\triangle} \rangle$.

5 EXPERIMENTS

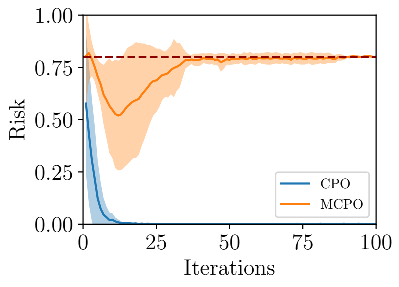
We design a set of experiments to assess the capability of MCPO to learn the optimal δ-safe policy in CMPDs with arbitrary unsafe reward signals. As part of the analysis, MCPO is compared to CPO with regard to task performance and safety.

In all experiments, policies are modeled as a multivariate Gaussian distribution. The mean vectors of the actions are optimized as output nodes of neural networks with two hidden layers of (64,32) nodes and `tanh` activation functions. The covariance matrices are part of the policy parameters to be optimized, although they are not a function of inputs. For CPO and MCPO, advantages are estimated by GAE-Λ as defined in Equations (13) and (14) respectively. Furthermore, both algorithms estimate the constraint advantages by GAE-Λ (13). For MCPO, the $n = 32$ reward estimates (with bootstrapping) are modeled as n output nodes of a neural network that has the same hidden layer architecture as the policy networks. The other value functions have equal hidden layer architecture with one output node. The constraint value function networks have a `sigmoid` output activation function. We set the discount factor as $\gamma = 0.995$, the GAE parameter as $\Lambda = 0.95$, and the KL-divergence step size as $\kappa = 0.01$. Per iteration, each learning process consists of 100 iterations with $5 \cdot 10^4$ time steps. Results are collected from 10 seeds. Our implementation is based on the open source toolkit `garage` [7].

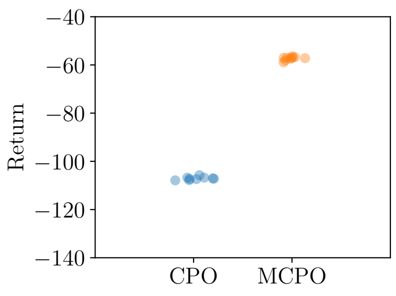
We consider experiments of learning δ-safe policies for lower values of δ, where the performance difference between CPO and MCPO is most noticeable. In addition, we want to observe whether the learning process of MCPO would suffer from a high frequency of masking prediction errors as discussed in section 4. We consider two continuous control tasks that are easy to interpret: (i) a navigation task, and (ii) a foraging task.



(a) Returns

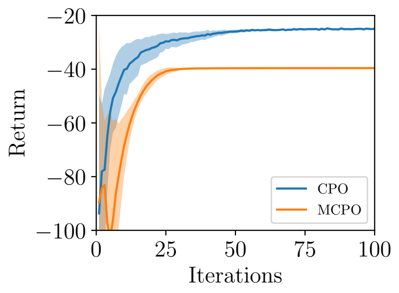


(b) Risk

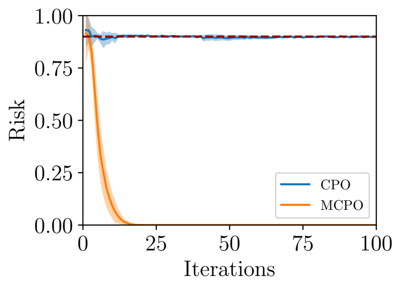


(c) Returns of completed trajectories

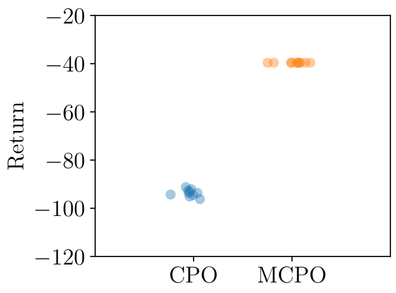
Figure 4: CPO and MCPO in LakeWorld with an unsafe reward signal of -5000.



(a) Returns



(b) Risk



(c) Returns of completed trajectories

Figure 5: CPO and MCPO in CoastWorld.

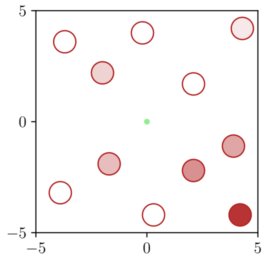
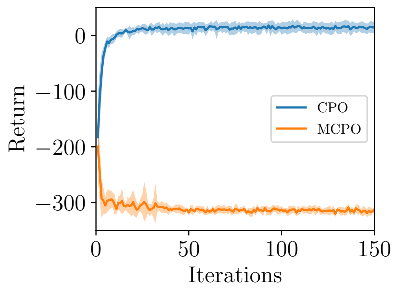


Figure 6: The environment of the foraging task at the beginning of an episode.

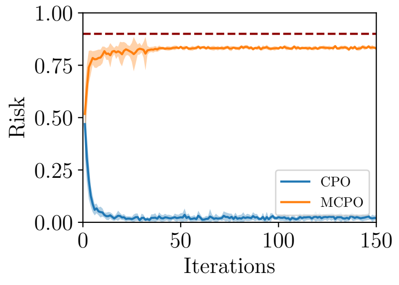
5.1 Navigation Task

In this task, the agent's goal is to find the shortest path between starting and goal positions that is δ-safe, in an environment filled with unsafe areas. Formally, the risk-embedded CMDP defining this problem is composed of the state space $\mathcal{S} = [-5, 5] \times [-5, 5]$, and the action space $\mathcal{A} = [-1, 1] \times [-1, 1]$. The state is the (x, y) position of the agent, and the action is the agent's velocity $(\dot{x}, \dot{y})$. The agent is motivated to move to a goal position by defining the reward function as the negative distance to that goal: $R(s, a, s') = -\|s' - s_{\text{GOAL}}\|$. The goal is located at $s_{\text{GOAL}} = (4.5, -4.5)$. Transitions from state s by action a are deterministic to $s' = (x + \dot{x}, y + \dot{y})$, unless the agent travels over an unsafe area. Let ρ denote the constant risk rate, and consider $D(s, a)$ as the distance on unsafe areas that the agent would cross from s with a. The probability of transitioning to an unsafe terminal state $s^{\triangle}$ is then defined as $P(s^{\triangle} \mid s, a) = 1 - (1 - \rho)^{D(s,a)}$. The reward function is extended with transitions to $s^{\triangle}$ by $R(s, a, s^{\triangle}) = r^{\triangle}$, where we refer to the constant $r^{\triangle}$ as the unsafe reward. We consider two different layouts of the unsafe areas, with their own configuration of parameters $\langle \delta, \rho, r^{\triangle} \rangle$. For convenience, we name the layout of Figures 2(a) and 2(b) as *LakeWorld*, and that of Figures 2(c) and 2(d) as *CoastWorld*.

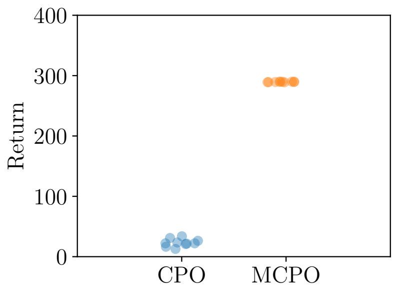
5.1.1 LakeWorld. Figures 2(a) and 2(b) illustrate the navigation vector flows learned with a random starting position in each episode. Trajectories shown from $s_{\text{START}} = (-4.5, 4.5)$ to s_{GOAL} were sampled from the learned policies. Figure 3 shows the results from learning processes with a deterministic starting position (i.e. s_{START}) in each episode. The agents learned δ-safe policies with $\delta = 0.2$, $\rho = 0.25$ and $r^{\triangle} = -500$. As the unsafe reward is lower than any other reward received at a transition, CPO learns to mostly avoid unsafe areas. Consequently, Figures 3(a) and 3(b) show that CPO achieves higher returns and lower probability of entering the unsafe state than MCPO. Note that δ is illustrated as the horizontal dashed line in Figure 3(b), which shows that both algorithms are δ-safe. Nevertheless, as the sample trajectories indicate, CPO actually learns a suboptimal policy of the original task objective (i.e. finding the shortest path). To illustrate this, we set ρ to zero and



(a) Returns



(b) Risk



(c) Returns of completed trajectories

Figure 7: CPO and MCPO in the foraging task

sample 10^3 trajectories for the learned policies of each seed. The average returns for these trajectories, which all resulted in a task completion, are shown in Figure 3(c). MCPO achieves higher task performance than CPO, while meeting the constraints.

To further investigate the impact of the unsafe reward signal, we opt for an even larger negative value: $r^{\triangle} = -5000$. The same metrics are again considered and shown in Figure 4. CPO has become entirely risk-averse, at the cost of lowering its task performance. On the other hand, MCPO achieves equal performance to the case of $r^{\triangle} = -500$. This further indicates that MCPO successfully neglects the unsafe reward. A comparison of Figures 3(b) and 4(b) shows how the value of $r^{\triangle}$ determines the probability of taking risks. Decreasing (increasing) the unsafe reward lowers (raises) the upper-bound on the risk probability of the policies learned by CPO. Thus, for CPO to find the policy which optimizes task performance—i.e. the policy shown in Figure 2(b)—, this upper-bound should be equal to the risk probability of that optimal policy. This is achieved by hand-tuning the unsafe reward signal.

5.1.2 CoastWorld. For this CMDP, we set the safety-threshold as $\delta = 0.1$, the constant risk rate as $\rho = 0.33$, and the unsafe reward as $r^{\triangle} = 0$. Navigation vector flows and sampled trajectories are shown in Figures 2(c) and 2(d) for CPO and MCPO, respectively. In this case, not only does CPO learn a suboptimal policy, but also one that is substantially higher than MCPO in risk. Since the unsafe reward is higher than any other safe reward, CPO learns to maximize risk (see Figures 5(a) and 5(b)). The average return, in completed tasks, shows again that MCPO finds a shorter path than CPO. We hope to showcase with this example that the naive approach of selecting the unsafe reward as zero [6] should be carefully considered.

5.2 Foraging Task

In this task, the goal for the agent is to collect the 6 highest rewards out of the 11 placed as patches in a 2D space. Figure 6 shows the initialization of the environment, where the states and actions are defined identical to those of the navigation task in 5.1. Upon entering a patch, the agent either receives a positive reward or terminates its episode due to a risk for which it is given a reward $r^{\triangle} = -500$. A visited patch is immediately removed from the environment for the remainder of that episode. There are five patches which have no risk and provide a reward of 10, and five patches $\{\text{patch}_i\}_{0:4}$ that are designed with reward $r_i = 100 - 20i$ and risk $\rho_i = 0.5 - 0.1i$. In addition, a patch with reward 10 and risk 0.92 is included.

Figure 7 shows the results of CPO and MCPO learning for 150 iterations, to find an optimal δ-safe policy where $\delta = 0.1$. CPO learns to maximize the sum of rewards involving the unsafe reward signal (see Figure 7(a)), while MCPO aims to only optimize the return of trajectories which have successfully completed the task (see Figure 7(c)). Consequently, MCPO has learned to navigate towards higher rewards than CPO, while both algorithms rely on the same safety constraint.

6 CONCLUSIONS

In this paper, we have studied the problem of hand-tuning reward signals in safe reinforcement learning tasks, where safety is defined based on the probability of early termination due to unsafe transitions of the MDP. More specifically, we illustrated how rewards related to unsafe transitions influence the optimization criterion. For learning policies that optimize task performance and simultaneously guarantee that the probability of succumbing to risk remains under a given threshold, the unsafe rewards are required to be carefully selected with expert knowledge. We, therefore, proposed a novel algorithm called Masked Constrained Policy Optimization (MCPO), which avoids the burden on task designers to engineer unsafe reward signals. MCPO neglects these signals by masking the advantage estimator, which is used to compute the maximization criterion in the iterative policy improvement algorithm. We derive our algorithm based on GAE, and additionally discuss a practical implementation, other advantage estimators, possible limitations, and the performance difference to CPO. Numerical results for continuous tasks show that MCPO outperforms CPO in the average of accumulated rewards, for the episodes that were not terminated due to risk, when unsafe reward signals are not properly hand-tuned. Our proposed algorithm would be particularly promising for tasks whose objective can be achieved only by taking a certain number of unsafe actions, considering such tasks are preferred to be performed by replaceable artificial agents instead of humans.

ACKNOWLEDGMENTS

The authors would like to acknowledge Johannes Nauta, who aided in the visualization of the experiment results.

REFERENCES

[1] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. 2010. Autonomous Helicopter Aerobatics through Apprenticeship Learning. *Int. J. Rob. Res.* 29, 13 (Nov. 2010), 1608–1639.

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization.

[3] Eitan Altman. 1999. *Constrained Markov Decision Processes.* Taylor & Francis.

[4] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. 2017. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *J. Mach. Learn. Res.* 18, 1 (Jan. 2017), 6070–6120.

[5] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. 2016. Benchmarking Deep Reinforcement Learning for Continuous Control *(Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 1329–1338.

[6] William Fedus, Carles Gelada, Yoshua Bengio, Marc G. Bellemare, and Hugo Larochelle. 2019. Hyperbolic Discounting and Learning over Multiple Horizons. arXiv:1902.06865 [stat.ML]

[7] The garage contributors. 2019. Garage: A toolkit for reproducible reinforcement learning research.

[8] Javier García and Fernando Fernández. 2012. Safe Exploration of State and Action Spaces in Reinforcement Learning. 45, 1 (Sept. 2012), 515–564.

[9] Javier García and Fernando Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* 16, 42 (2015), 1437–1480.

[10] Chris Gaskett. 2003. Reinforcement learning under circumstances beyond its control. (2003). https://researchonline.jcu.edu.au/632/

[11] Clement Gehring and Doina Precup. 2013. Smart Exploration in Reinforcement Learning Using Absolute Temporal Difference Errors. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems* (St. Paul, MN, USA) *(AAMAS '13)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1037–1044.

[12] Peter Geibel and Fritz Wysotzki. 2005. Risk-Sensitive Reinforcement Learning Applied to Control under Constraints. *J. Artif. Int. Res.* 24, 1 (July 2005), 81–108.

[13] Sham Kakade and John Langford. 2002. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML '02)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 267–274.

[14] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. 2017. AI Safety Gridworlds.

[15] Horia Mania, Aurelia Guy, and Benjamin Recht. 2018. Simple random search provides a competitive approach to reinforcement learning.

[16] Alberto Maria Metelli, Matteo Papini, Nico Montali, and Marcello Restelli. 2020. Importance Sampling Techniques for Policy Optimization. *Journal of Machine Learning Research* 21, 141 (2020), 1–75. http://jmlr.org/papers/v21/20-124.html

[17] Santiago Paternain, Miguel Calvo-Fullana, Luiz F. O. Chamon, and Alejandro Ribeiro. 2019. Safe Policies for Reinforcement Learning via Primal-Dual Methods. arXiv:1911.09101 [eess.SY]

[18] Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. 2019. Ray Interference: a Source of Plateaus in Deep Reinforcement Learning.

[19] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust Region Policy Optimization. In *International conference on machine learning*. 1889–1897.

[20] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[21] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA.

[22] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. 2018. Reward Constrained Policy Optimization.

[23] Eiji Uchibe and Kenji Doya. 2009. Constrained Reinforcement Learning from Intrinsic and Extrinsic Rewards. In *Theory and Novel Applications of Machine Learning*, Meng Joo Er and Yi Zhou (Eds.). IntechOpen, Rijeka, Chapter 11.

[24] Lu Wen, Jingliang Duan, Shengbo Eben Li, Shaobing Xu, and H. Peng. 2020. Safe Reinforcement Learning for Autonomous Vehicles through Parallel Constrained Policy Optimization. *ArXiv* abs/2003.01303 (2020).