ROSARL: REWARD-ONLY SAFE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

An important problem in reinforcement learning is designing agents that learn to solve tasks safely in an environment. A common solution is to define either a penalty in the reward function or a cost to be minimised when reaching unsafe states. However, designing reward or cost functions is non-trivial and can increase with the complexity of the problem. To address this, we investigate the concept of a *Minmax penalty*, the smallest penalty for unsafe states that leads to safe optimal policies, regardless of task rewards. We derive an upper and lower bound on this penalty by considering both environment *diameter* and *solvability*. Additionally, we propose a simple algorithm for agents to estimate this penalty while learning task policies. Our experiments demonstrate the effectiveness of this approach in enabling agents to learn safe policies in high-dimensional continuous control environments.

1 INTRODUCTION

Reinforcement learning (RL) has recently achieved success across a variety of domains, such as video games (Shao et al., 2019), robotics (Kalashnikov et al., 2018; Kahn et al., 2018) and autonomous driving (Kiran et al., 2021). However, if we hope to deploy RL in the real world, agents must be capable of completing tasks while avoiding unsafe or costly behaviour. For example, a navigating robot must avoid colliding with objects and actors around it, while simultaneously learning to solve the required task. Figure 1 shows an example.

Many approaches in RL deal with this problem by allocating arbitrary penalties to unsafe states when hand-crafting the reward function. However, the problem of specifying a reward function for desirable, safe behaviour is notoriously difficult (Amodei et al., 2016). *Importantly, penalties that are too small may result in unsafe behaviour, while penalties that are too large may result in increased learning times.* Furthermore, these rewards must be specified by an expert for each new task an agent faces. If our aim is to design truly autonomous, general agents, it is then simply impractical to require that a human designer specify penalties to guarantee optimal but safe behaviours for every task.

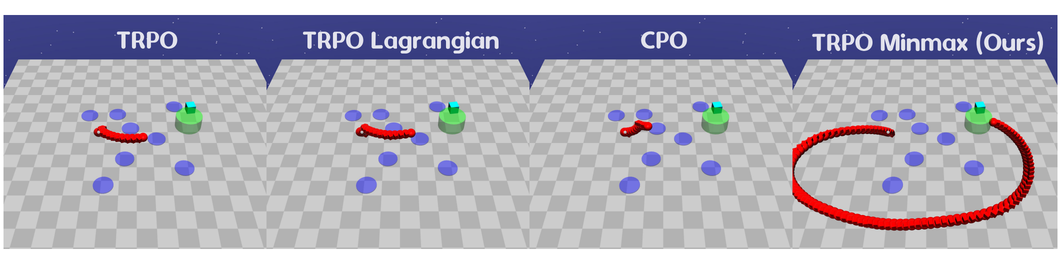


Figure 1: Sample trajectories of representative prior works—TRPO (Schulman et al., 2015) (left-most), TRPO-Lagrangian (Ray et al., 2019) (middle-left), CPO (Achiam et al., 2017) (middle-right)—compared to ours (right-most) in the Safety Gym domain (Ray et al., 2019). For each, a point mass agent learns to reach a goal location (green cylinder) while avoiding unsafe regions (blue circles). The cyan block is a randomly placed movable obstacle. Our approach learns safer policies than the baselines, and works by simply changing the rewards received for entering unsafe regions to a learned penalty (keeping the rewards received for all other transitions unchanged).

When safety is an explicit goal, a common approach is to constrain policy learning according to some threshold on cumulative cost (Schulman et al., 2015; Ray et al., 2019; Achiam et al., 2017). While effective, these approaches require the design of a cost function whose specification can be as challenging as designing a reward function. Additionally, these methods may still result in unacceptably frequent constraint violations in practice, due to the large cost threshold typically used.

Rather than attempting to both maximise a reward function and minimise a cost function, which requires specifying both rewards and costs and a new learning objective, we should simply aim to have a better reward function—since we then do not have to specify yet another scalar signal nor change the learning objective. This approach is consistent with the *reward hypothesis* (Sutton & Barto, 2018) which states: " *All of what we mean by goals and purposes can be well thought of as maximisation of the expected value of the cumulative sum of a received scalar signal (reward).* " Therefore, the question we examine in this work is how to determine the *Minmax penalty*—the smallest penalty assigned to unsafe states such that the probability of reaching safe goals is maximised by an optimal policy. Rather than requiring an expert's input, we show that this penalty can be bounded by taking into account the *diameter* and *solvability* of an environment, and a practical estimate of it can be learned by an agent using its current value estimates. We make the following contributions:

(i) **Bounding the Minmax penalty**: We provide analytical upper and lower bounds on the Minmax penalty for unsafe transitions, and prove that using the upper bound results in policies that minimise the probability of reaching unsafe transitions (Theorem 2).

(ii) **Learning safety bounds**: We show that these bounds can be accurately estimated using policy evaluation (Sutton & Barto, 2018) (Theorem 1). Additionally, we show that estimating the Minmax penalty or bounds is NP-hard since it requires solving a longest path problem (Theorem 3).

(iii) **Learning safe policies**: Building on our theoretical analysis, we present a practical, model-free algorithm that allows agents to learn a sufficient penalty for unsafe transitions while simultaneously learning task policies (Algorithm 1). Since this approach only modifies the reward received for unsafe transitions, it is easily integrated into any existing RL pipeline that uses value-based methods.

2 BACKGROUND

We consider the typical RL setting where the task faced by an agent is modelled by a Markov Decision Process (MDP). An MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0\ 1]$ is the transition probability function, and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [R_{\text{MIN}}\ R_{\text{MAX}}]$ is the reward function. Our focus is on undiscounted MDPs that model stochastic shortest path problems (Bertsekas & Tsitsiklis, 1991) in which an agent must reach some goals in the non-empty set of absorbing states $\mathcal{G} \subset \mathcal{S}$. The set of non-absorbing states $\mathcal{S} \setminus \mathcal{G}$ are referred to as *internal states*. We will also refer to the tuple $\langle \mathcal{S}, \mathcal{A}, P \rangle$ as the environment, and the MDP $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$ as a task to be solved. The agent is then associated with a *policy* $\pi : \mathcal{S} \to \mathcal{A}$ which it uses to take actions in the environment. The quality of a policy is usually defined by its *value function* $V^\pi(s) = \mathbb{E}^\pi[\sum_{t=0}^{\infty} R(s_t, a_t, s_{t+1})]$, which specifies the expected return under that policy starting from state s.

Standard RL: The standard goal of an agent is to learn an optimal policy π^* that maximises the value function $V^{\pi^*}(s) = \max_\pi V^\pi(s)$ for all $s \in \mathcal{S}$. Since tasks are undiscounted, π^* is guaranteed to exist by assuming that the value function of *improper policies* is unbounded from below—where *proper policies* are those that are guaranteed to reach an absorbing state (Van Niekerk et al., 2019). Since there always exists a deterministic π^* (Sutton & Barto, 1998), and π^* is proper, we will focus our attention on the set of all deterministic proper policies Π.

Safe RL: This setting is typically modelled in prior works by a constrained Markov Decision Process (CMDP) $\langle \mathcal{S}, \mathcal{A}, P, R, K, l \rangle$, which augments an MDP with a cost function $K : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ and a cost threshold $l \in \mathbb{R}$ (Altman, 1999). Here, a given policy π can also be characterised by its cost value function $V_K^\pi(s) = \mathbb{E}^\pi[\sum_{t=0}^{\infty} K(s_t, a_t, s_{t+1})]$, and the policy is *feasible* if $V_K^\pi(s) \leq l$ for all $s \in \mathcal{S}$. Where $\widehat{\Pi}$ is the set of all feasible policies, the goal of an agent here is now to learn an optimal safe policy $\widehat{\pi}^*$ that maximises the value function $V^{\widehat{\pi}^*}(s) = \max_{\widehat{\pi} \in \widehat{\Pi}} V^{\widehat{\pi}}(s)$ for all $s \in \mathcal{S}$ (Ray et al., 2019). To ensure that $\widehat{\pi}^*$ exists and is well defined, $\widehat{\Pi}$ must not be empty, which means that K and l must be chosen carefully such that there exists a policy π that satisfies $V_K^\pi(s) \leq l$ for all $s \in \mathcal{S}$.

ROSARL (Ours): In contrast to most prior works, in this work we are interested in learning safe policies without the need to specify cost functions and cost thresholds. In particular, we are interested in learning policies that can maximise rewards while avoiding unsafe transitions, where any unsafe transition immediately leads to termination in a set of unsafe absorbing states $\mathcal{G}^! \subset \mathcal{G}$. Since some environments may have no policy that avoids unsafe transitions with probability 1, we formally define a safe policy as a proper policy that minimises the probability of unsafe transitions (Definition 1). Hence, where $\widehat{\Pi}$ is the set of all safe policies, the goal of an agent in this work is to learn an optimal safe policy $\widehat{\pi}^*$ that maximises the value function $V^{\widehat{\pi}^*}(s) = \max_{\widehat{\pi}\in\widehat{\Pi}} V^{\widehat{\pi}}(s)$ for all $s \in \mathcal{S}$.

Definition 1 *Consider an environment $\langle \mathcal{S}, \mathcal{A}, P\rangle$ with unsafe states $\mathcal{G}^! \subset \mathcal{G}$. Where s_T is the final state of a trajectory starting from state s, let $P_s^\pi(s_T \in \mathcal{G}^!)$ be the probability of reaching $\mathcal{G}^!$ from s under a proper policy $\pi \in \Pi$. Then π is called safe if $\pi \in \arg\min_{\pi'\in\Pi} P_s^{\pi'}(s_T \in \mathcal{G}^!)$ for all $s \in \mathcal{S}$.*

3 Avoiding Unsafe Absorbing States

Given an environment, we aim to bound the smallest penalty (hence the largest reward) to use for unsafe transitions to guarantee optimal safe policies. We define this penalty as the Minmax penalty R_{Minmax}, which is the largest reward for unsafe transitions that lead to optimal safe policies:

Definition 2 *Consider an environment $\langle \mathcal{S}, \mathcal{A}, P\rangle$ where task rewards $R(s, a, s')$ are bounded by $[R_{MIN}\ R_{MAX}]$ for all $s' \notin \mathcal{G}^!$. Let π^* be an optimal policy for one such task $\langle \mathcal{S}, \mathcal{A}, P, R\rangle$. We define the Minmax penalty of this environment as the scalar $R_{Minmax} \in \mathbb{R}$ that satisfies the following:*

(i) If $R(s, a, s') < R_{Minmax}$ for all $s' \in \mathcal{G}^!$, then π^* is safe for all R;

(ii) If $R(s, a, s') > R_{Minmax}$ for some $s' \in \mathcal{G}^!$ reachable from $\mathcal{S} \setminus \mathcal{G}$, then there exists an R s.t. π^* is unsafe.

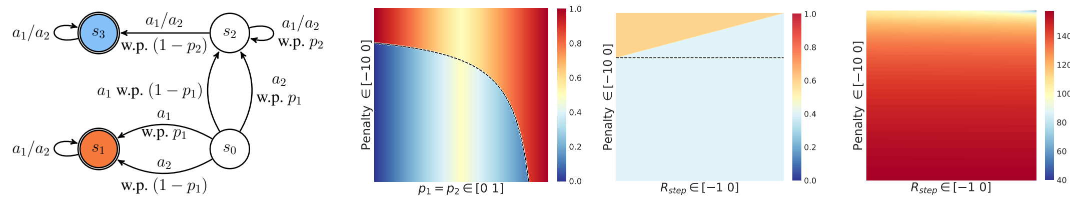
Hence, the Minmax penalty represents the boundary where on one side *no* reward function has an optimal policy that is unsafe, and on the other *there exist* a reward function with an optimal policy that is unsafe. Interestingly, when $R(s, a, s') = R_{\text{Minmax}}$, there may exist optimal safe and unsafe policies simultaneously—hence no RL algorithm with such rewards can be guaranteed to converge to optimal safe policies. We next demonstrate this using the Chain-walk running example.

3.1 A Motivating Example: The Chain-Walk Environment

To illustrate the difficulty in designing reward functions for safe behaviour, consider the simple *chain-walk* environment in Figure 2a. It consists of four states s_0, s_1, s_2, s_3 where $\mathcal{G} = \{s_1, s_3\}$ and $\mathcal{G}^! = \{s_1\}$. The agent has two actions a_1, a_2, the initial state is s_0, and the diagram denotes the transition probabilities. Task rewards for safe transitions are bounded by $[R_{\text{MIN}}\ R_{\text{MAX}}] = [-1\ 0]$. The absorbing transitions have a reward of 0 while all other transitions have a reward of $R_{step} = -1$, and the agent must reach the goal state s_3, but not the unsafe state s_1. Hence, the question here is what penalty to give for transitions from s_0 into s_1 such that the optimal policies are safe. Figures 2b-2d exemplify how too large penalties result in longer convergence times, while too small ones result in unsafe policies, demonstrating the need to find the Minmax penalty.

Since the transitions per action can be stochastic, controlled by $p_1, p_2 \in [0\ 1]$, and s_3 is further from the start state s_0 than s_1, the agent may not always be able to avoid s_1. Consider for example the deterministic case when $p_1 = p_2 = 0$. For any penalty less than -2 for transitions into s_1, the optimal policy in s_0 is to always pick a_2 which always reaches s_1. For a sufficiently high penalty for reaching s_1 (any penalty higher than -2), the optimal policy in s_0 is to always pick action a_1, which always reaches s_3. Interestingly, if the penalty is exactly -2, then both action a_1 (safe transition to s_2) and action a_2 (unsafe transition to s_1) are optimal—hence an RL algorithm here will not necessarily converge to the optimal safe action a_1. Additionally, for $p_1 = p_2 = 0.4$ (Figure 2c), a higher penalty is required for a_1 to stay optimal in state s_0.

To capture this relationship between the stochasticity of an environment and the required penalty to obtain safe policies, we introduce a notion of *solvability*, which measures the ability of an agent to reach safe goals. Additionally, observe that as p_2 increases, the probability that the agent can



(a) Chain-walk (b) Failure rates with $R_{step} = -1$ (c) Failure rates with p_1=p_2=0.4 (d) Total timesteps with p_1=p_2=0.4

Figure 2: The effect of different choices of penalty for unsafe transitions (s_0 to s_1) on optimal policies in the chain-walk environment. (a) The transition probabilities of the chain-walk environment (where $p_1, p_2 \in [0\ 1]$); (b) The failure rate for each penalty in $[-10\ 0]$ and each transition probabilities ($p_1 = p_2 \in [0\ 1]$), with a task reward of $R_{step} = -1$; (c) The failure rate for each penalty in $[-10\ 0]$ and each task reward in $[-1\ 0]$, with transition probabilities given by $p_1 = p_2 = 0.4$; (d) The total timesteps needed to learn optimal policies to convergence (using value iteration (Sutton & Barto, 1998)) for each penalty in $[-10\ 0]$ and each task reward in $[-1\ 0]$, with transition probabilities given by $p_1 = p_2 = 0.4$. The black dashed lines in (b) and (c) show the Minmax penalty.

transition from s_2 to s_3 decreases—thereby increasing the number of timesteps spent to reach the goal. Therefore, the penalty for s_1 must also consider the environment's *diameter* to ensure an optimal policy will not simply reach s_1 to avoid self-transitions in s_2.

3.2 On the Diameter and Solvability of Environments

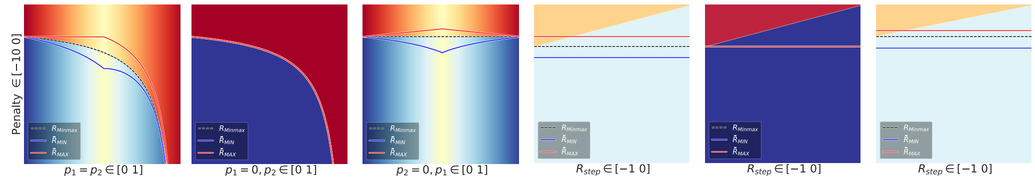
Clearly, the size of the penalty that needs to be given for unsafe states depends on the *size* of the environment. We define this size as the *diameter* of the environment, which is the highest expected timesteps to reach an absorbing state from an internal state when following a proper policy:

Definition 3 *Define the diameter of an environment as* $D \coloneqq \max\limits_{s \in \mathcal{S} \setminus \mathcal{G}} \max\limits_{\pi \in \Pi} \mathbb{E}\left[T(s_T \in \mathcal{G} | \pi)\right]$, *where* $T(s_T \in \mathcal{G}|\pi)$ *is the timesteps taken to reach* $\mathcal{G}$ *from* s *when following a proper policy* π.

This definition of diameter is similar to the one used in Auer et al. (2008), except that here we are maximising over deterministic proper policies instead of minimising over all deterministic policies. Given this diameter, a possible natural choice for the reward for unsafe states is to give a penalty that is as large as receiving the smallest task reward for the longest path to safe goal states: $\bar{R}_{\text{MAX}} \coloneqq R_{\text{MIN}} D'$, where D' is the diameter for safe policies $D' \coloneqq \max\limits_{s \in \mathcal{S} \setminus \mathcal{G}} \max\limits_{\pi \in \Pi} \mathbb{E}\left[T(s_T \in \mathcal{G} \setminus \mathcal{G}^! | \pi)\right]$. However, while $\bar{R}_{\text{MAX}}$ aims to make reaching unsafe states worse than reaching safe goals, it does not consider the solvability of an environment, nor the possibility that an unsafe policy receives R_{MAX} everywhere in its trajectory. We can formally define the solvability of an environment as follows:

Definition 4 *Define the degree of solvability as* $C \coloneqq \min\limits_{s \in S \setminus \mathcal{G}} \min\limits_{\substack{\pi \in \Pi \\ P^\pi_s(s_T \notin \mathcal{G}^!) \neq 0}} P^\pi_s(s_T \notin \mathcal{G}^!)$.

C measures the degree of solvability of the environment by simply taking the smallest non-zero probability of reaching safe goal states by following a proper policy. For example, if the dynamics are deterministic, then any deterministic policy π will either reach a safe goal or not. That is, $P^\pi_s(s_T \notin \mathcal{G}^!)$ will either be 0 or 1. Since we require $P^\pi_s(s_T \notin \mathcal{G}^!) \neq 0$, it must be that $C = 1$. Consider, for example, the chain-walk environment with different choices for p. Since actions in s_2 do not affect the transition probability, there are only 2 relevant deterministic policies $\pi_1(s) = a_1$ and $\pi_2(s) = a_2$. This gives $P^{\pi_1}_{s_1}(s_T \notin \mathcal{G}^!) = (1 - p_1)\mathbb{1}(p_2 = 1)$ and $P^{\pi_2}_{s_1}(s_T \notin \mathcal{G}^!) = p_1\mathbb{1}(p_2 = 1)$. Here, $C = 1$ when $p_1 = p_2 = 0$ because the task is deterministic and s_3 is reachable. C then tends to 0.5 as p_1 and p_2 gets closer to 0.5, making the environment uniformly random. Finally, the environment is not solvable when $p = 1$ since s_3 is unreachable from s_2. Hence we can also think of $C = 0$ as the *limit* of C when safe goals are unreachable. Interestingly, this means that in deterministic environments our definition of solvability is similar to *reachability* in temporal-logic tasks—where there may or may not exist a policy that satisfies a task specification (Tasse et al., 2022).



(a) $R_{step} = -1$ (b) $R_{step} = -1$ (c) $R_{step} = -1$ (d) p_1=p_2=0.4 (e) p_1=0, p_2=0.4 (f) p_1=0.4, p_2=0

Figure 3: Failure rates of optimal policies in the chain-walk environment. We show the effect of stochasticity (p_1 and p_2) and task rewards (R_{step}) on the bounds ($\bar{R}_{\text{MIN}}$ and $\bar{R}_{\text{MAX}}$) of the Minmax penalty (R_{Minmax}). The solvability and diameter for the bounds are estimated using Algorithm 2.

Given the diameter and solvability of an environment, we can now define a choice for the Minmax penalty that takes into account both D, C, and R_{MAX}: $\bar{R}_{\text{MIN}} := (R_{\text{MIN}} - R_{\text{MAX}})\frac{D}{C}$. This choice of penalty says that since stochastic shortest path tasks require an agent to learn to achieve desired terminal states, if the agent enters an unsafe terminal state, it should receive the largest penalty possible by a proper policy. We now investigate the effect of these penalties on the failure rate of optimal policies.

3.3 On the Failure Rate of Optimal Policies

We begin by proposing a simple model-based algorithm for estimating the diameter and solvability, from which the penalties are then obtained. We describe the method here and present the pseudo-code in **Algorithm 2** in Appendix B. Here, the diameter is estimated as follows: (i) For each deterministic policy π, estimate its expected timesteps $T(s_T \in \mathcal{G})$ (or $T(s_T \in \mathcal{G} \setminus \mathcal{G}^!)$ for D') by using policy evaluation (Sutton & Barto, 2018) with rewards of 1 at all internal states; (ii) Then, calculate D using the equation in Definition 3. Similarly, the solvability is estimated by estimating the reach probability $P_s^\pi(s_T \notin \mathcal{G}^!)$ of each deterministic policy π using rewards of 1 for transitions into safe goal states and zero otherwise. This approach converges via the convergence of policy evaluation (**Theorem 1**).

Theorem 1 (Estimation) *Algorithm 2 converges to D and C for any given solvable environment.*

Figure 3 shows the result of applying this algorithm in the chain-walk MDP. Here, R_{Minmax} is compared to accounting for D only ($\bar{R}_{\text{MAX}}$) and accounting for both C and D ($\bar{R}_{\text{MIN}}$). Interestingly, we can observe $\bar{R}_{\text{MIN}} \leq R_{\text{Minmax}}$ and $\bar{R}_{\text{MAX}} \geq R_{\text{Minmax}}$ consistently, highlighting how considering the diameter only is insufficient to guarantee optimal safe policies. It also indicates that these penalties may bound R_{Minmax} in general. We show in **Theorem 2** that this is indeed the case.

Theorem 2 (Safety Bounds) *Consider a solvable environment where task rewards are bounded by $[R_{MIN}\ R_{MAX}]$ for all $s' \notin \mathcal{G}^!$. Then $\bar{R}_{MIN} \leq R_{Minmax} \leq \bar{R}_{MAX}$.*

Theorem 2 says that for any MDP whose rewards for unsafe transitions are bounded above by $\bar{R}_{\text{MIN}}$, the optimal policy both minimises the probability of reaching unsafe states and maximises the probability of reaching safe goal states. Hence, any penalty $\bar{R}_{\text{MIN}} - \epsilon$, where $\epsilon > 0$ can be arbitrarily small, will guarantee optimal safe policies. Similarly, the theorem shows that any reward higher than $\bar{R}_{\text{MAX}}$ may have optimal policies that do not minimise the probability of reaching unsafe states. These can be observed in Figure 3. The figure demonstrates why considering both the diameter and solvability of an MDP is necessary to guarantee safe policies, because the diameter alone does not always minimise the failure rate.

4 Practical Algorithm for Learning Safe Policies

While the Minmax penalty of an MDP can be accurately estimated using policy evaluation (Algorithm 2), it requires knowledge of the environment dynamics (or an estimate of it). These are difficult quantities to estimate from an agent's experience, which is further complicated by the need to also learn the true optimal policy for the estimated Minmax penalty. Hence, obtaining an accurate estimate of the Minmax penalty is impractical in model-free and function approximation settings where the state and action spaces are large. In fact, it is NP-hard since it depends on the diameter, which requires solving a longest-path problem.

Theorem 3 (Complexity) *Estimating the Minmax penalty R_{Minmax} accurately is NP-hard.*

Given the above challenges, we require a practical method for learning the Minmax penalty. Ideally, this method should require no knowledge of the environment dynamics and should easily integrate with existing RL approaches. To achieve this, we first note that $(R_{\text{MIN}} - R_{\text{MAX}})\frac{D}{C} = (DR_{\text{MIN}} - DR_{\text{MAX}})\frac{1}{C} = (V_{\text{MIN}} - V_{\text{MAX}})\frac{1}{C}$, where V_{MIN} and V_{MAX} are the value function bounds. Hence, a practical estimate of the Minmax penalty can be efficiently learned by estimating the value gap $V_{\text{MIN}} - V_{\text{MAX}}$ using observations of the reward and the agent's estimate of the value function. **Algorithm 1** shows the full pseudo-code. The agent here receives a reward r_t after each environment interaction and updates its estimate of the reward bounds $R_{\text{MIN}} \leftarrow \min(R_{\text{MIN}}, r_t)$ and $R_{\text{MAX}} \leftarrow \max(R_{\text{MAX}}, r_t)$, the value bounds $V_{\text{MIN}} \leftarrow \min(V_{\text{MIN}}, R_{\text{MIN}}, V(s_t))$ and $V_{\text{MAX}} \leftarrow \max(V_{\text{MAX}}, R_{\text{MAX}}, V(s_t))$, and the Minmax penalty $\bar{R}_{\text{MIN}} \leftarrow V_{\text{MIN}} - V_{\text{MAX}}$, where $V(s_t)$ is the learned value function at time step t. We note how the solvability C is also not explicitly considered in this estimate of $\bar{R}_{\text{MIN}}$, since it is also expensive to estimate. Instead, given that the main purpose of C is to make $\bar{R}_{\text{MIN}}$ more negative the more stochastic the environment is, we notice that this is already achieved in practice by the reward and value estimates. Since R_{MIN} is estimated using $R_{\text{MIN}} \leftarrow \min(R_{\text{MIN}}, r_t)$, then every time the agent enters an unsafe state, we have that: $r_t \leftarrow \bar{R}_{\text{MIN}}$, $R_{\text{MIN}} \leftarrow \bar{R}_{\text{MIN}}$, and then $\bar{R}_{\text{MIN}} \leftarrow \bar{R}_{\text{MIN}} - V_{\text{MAX}}$. This means that when the estimated V_{MAX} is greater than zero, the penalty estimate $\bar{R}_{\text{MIN}}$ become more negative every time the agent enters an unsafe state.

Finally, whenever an agent encounters an unsafe state, the reward can be replaced by $\bar{R}_{\text{MIN}}$ to disincentivise unsafe behaviour. Since V_{MAX} is estimated using $V_{\text{MAX}} \leftarrow \max(V_{\text{MAX}}, R_{\text{MAX}}, V(s_t))$, it leads to an optimistic estimation of $\bar{R}_{\text{MIN}}$. Hence, we observe no need to add an $\epsilon > 0$ to $\bar{R}_{\text{MIN}}$.

Algorithm 1: RL while learning Minmax penalty

```
Input      : RL algorithm A, max timesteps T
Initialise : R_MIN = 0, R_MAX = 0, V_MIN = R_MIN, V_MAX = R_MAX, π and V as per A
  for t in T do
    observe a state s_t, take an action a_t using π as per A, and observe s_{t+1}, r_t
    R_MIN, R_MAX ← min(R_MIN, r_t), max(R_MAX, r_t)
    V_MIN, V_MAX ← min(V_MIN, R_MIN, V(s_t)), max(V_MAX, R_MAX, V(s_t))
    R̄_MIN ← V_MIN − V_MAX
    r_t ← R̄_MIN if s_{t+1} ∈ G^! else r_t
    update π and V with (s_t, a_t, s_{t+1}, r_t) as per A
  end for
```

5 Experiments

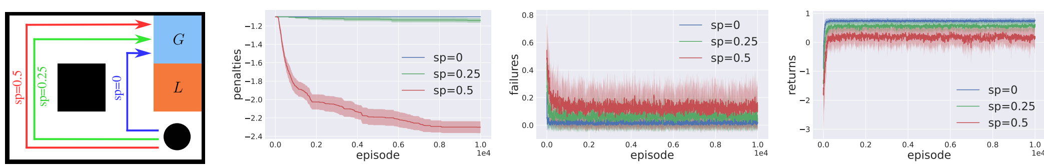
While the theoretical Minmax penalty is guaranteed to lead to optimal safe policies, it is unclear whether this also holds for the practical estimate proposed in Section 4. Hence, this section aims to investigate three main natural questions regarding the proposed practical algorithm (see the Appendix for more experiments): (i) How does Algorithm 1 behave when the theoretical assumptions are satisfied? (ii) How does Algorithm 1 behave when the theoretical assumptions are *not* satisfied? (iii) How does Algorithm 1 compare to prior approaches towards Safe RL? For each result, we report the mean (solid line) and one standard deviation around it (shaded region).

5.1 Behaviour when theory holds

For this experiment, we consider the Russell & Norvig (2016) gridworld described below. It satisfies the setting we assumed in Section 2 since it is a stochastic shortest path with finite states and actions.

Domain (Lava Gridworld) This is a gridworld with 11 positions ($|\mathcal{S}| = 11$) and 4 cardinal actions ($|\mathcal{A}| = 4$). The agent here must reach a goal location G while avoiding a lava location L (hence $\mathcal{G} = \{L, G\}$ and $\mathcal{G}^! = \{L\}$). A wall is also present in the environment and, while not unsafe, must be navigated around. The environment has a *slip probability (sp)*, so that with probability sp the agent's action is overridden with a random action. The agent receives $R_{\text{MAX}} = +1$ reward for reaching the goal, as well as $R_{step} = -0.1$ reward at each timestep to incentivise taking the shortest path to the goal. To test our approach, we modify Q-learning (Watkins, 1989) with ϵ-greedy exploration such that the agent updates its estimate of the Minmax penalty as learning progresses and uses it as the reward whenever the lava state is reached, following the procedure outlined in Section 4. The action-value function is initialised to 0 for all states and actions, $\epsilon = 0.1$ and the learning rate $\alpha = 0.1$.

Setup and Results We examine the performance of our modified Q-learning approach across three values of the slip probability of the LAVA GRIDWORLD. A slip probability of 0 represents a fully deterministic environment, while a slip probability of 0.5 represents a more stochastic environment. Results are plotted in Figure 4. In the case of the fully deterministic environment, the Minmax penalty bound obtained via Algorithm 2 is $\bar{R}_{\text{MIN}} = -9.9$, since $C = 1$ and $D = 9$. However, the agent is able to learn a relatively smaller penalty (-1.1 in Figure 4b) to consistently minimise failure rate and maximise returns (Figures 4c and 4d). The resulting optimal policy then chooses the shorter path that passes near the lava location ($sp = 0$ in Figure 4a). As the stochasticity of the environment increases, a larger penalty is learned to incentivise longer, safer policies ($sp = 0.25$ and $sp = 0.5$ in Figure 4a). We can, therefore, conclude that while there is a gap between the true Minmax penalty and the one learned via Algorithm 1, this algorithm can still learn optimal safe policies when the theoretical setting holds.



(a) Trajectories (b) Learned penalty (c) Failure rate (↓) (d) Average returns (↑)

Figure 4: Effect of increase in the slip probability (sp) of the LAVA GRIDWORLD on the learned Minmax penalty and corresponding failure rate and returns. The black circle in (a) represents the agent. The results are averaged over 20 random seeds—shaded regions represent one standard deviation.

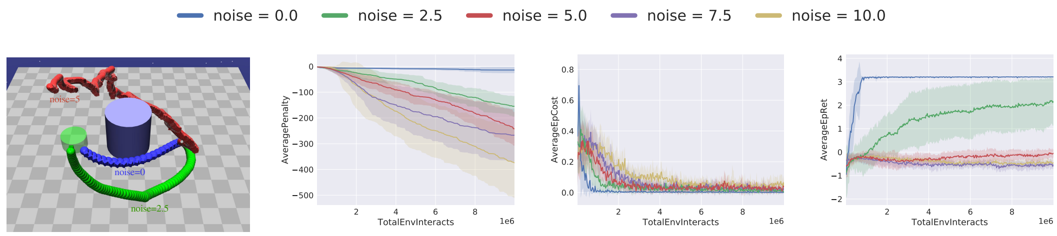
5.2 BEHAVIOUR WHEN THEORY DOES NOT HOLD

For this experiment, we consider the Safety Gym (Ray et al., 2019) domain described below. It does not satisfy the setting we assumed in Section 2 since it is continuous and not a shortest path task[1].

Domain (Safety Gym PILLAR) This is a custom Safety Gym domain in which the simple point robot must navigate to a goal location around a large pillar (hence $\mathcal{G} = \{ , \}$ and $\mathcal{G}^! = \{ \}$). All details of the environment are the same as in Ray et al. (2019) except when stated otherwise. Just as in Ray et al. (2019), the agent uses *pseudo-lidar* to observe the distance to objects around it ($|\mathcal{S}| = \mathbb{R}^{60}$), and the action space is continuous over two actuators controlling the direction and forward velocity ($|\mathcal{A}| = [-1, 1]^2$). This direction and forward velocity can be noisy, determined by a $noise$ scalar as follows: $a_{new} = a + (noise)a_{noise}$ where a_{new} is the new direction and forward velocity, $a \in \mathcal{A}$ is the agent's action, and $a_{noise} \in \mathcal{A}$ is a uniformly sampled random vector. The goal, pillar, and agent locations remain unchanged for all episodes. Each episode terminates once the agent reaches the goal or collides with the pillar (with a reward of -1). Otherwise, episodes terminate after 1000 timesteps. To test our approach in this setting, we modify TRPO (Schulman et al., 2015) (denoted TRPO-Minmax) to use the estimate of the Minmax penalty as described in Algorithm 1.

Setup and Results We examine the performance of TRPO-Minmax for five levels of $noise$ in the PILLAR environment, similarly to the experiments in Section 5.1. Results are plotted in Figure 5. We observe similar results to Section 5.1, where the agent uses its learned Minmax penalty (Figure 5b) to successfully learn safe policies (Figure 5c) while solving the task (Figure 5d), using safer paths for more noisy dynamics (Figure 5a). Interestingly, it also correctly prioritises low failure rates when the dynamics are too noisy to safely reach the goal ($noise \geq 5$). We can, therefore, conclude that Algorithm 1 can learn safe policies even in discounted high-dimensional continuous-control domains requiring function approximation.

[1]The PILLAR domain does not satisfy the formal shortest shortest path setting we assume since: it is discounted and policies that do not reach $\mathcal{G}$ are not guaranteed to have value functions that are unbounded from below (due to the default dense rewards in Safety Gym which positively rewards moving towards the goal).



(a) Trajectories (b) Learned penalty (c) Failure rate (↓) (d) Average returns (↑)

Figure 5: Performance of TRPO-Minmax in the PILLAR environment with varying noise levels. Each training run is over 10 million steps and the results are averaged over 10 random seeds—shaded regions represent one standard deviation.

5.3 COMPARISON TO REPRESENTATIVE BASELINES

For this experiment, we consider representative baselines in the Safety Gym PILLAR domain.

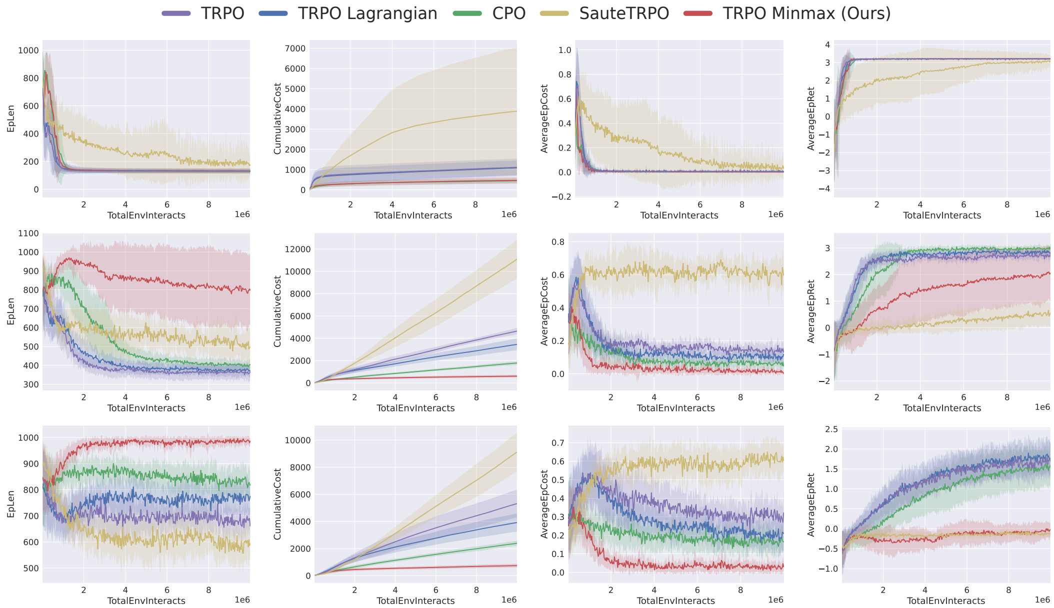
Baselines As a baseline representative of typical RL approaches, we use Trust Region Policy Optimisation (TRPO) (Schulman et al., 2015). To represent constraint-based approaches, we compare against Constrained Policy Optimisation (CPO) (Achiam et al., 2017), TRPO with Lagrangian constraints (TRPO-Lagrangian) (Ray et al., 2019), and Sauté RL with TRPO (Sauté-TRPO) (Sootla et al., 2022). All baselines except Sauté-TRPO use the implementations provided by Ray et al. (2019), and form a set of widely used baselines in safety domains (Zhang et al., 2020; Sootla et al., 2022; Yang et al., 2023). Sauté-TRPO uses the implementation provided by Sootla et al. (2022). As in Ray et al. (2019), all approaches use feed-forward MLPs, value networks of size (256,256), and $tanh$ activation functions. The cost threshold for the constrained algorithms is set to 0, the best we found. The experiments are run over 10 million episodes and averaged over 10 runs.

Setup and Results We compare the performance of TRPO-Minmax to that of the baselines for different levels of noise in the PILLAR domain. Figure 6 shows the results. We observe that in the deterministic case $noise = 0$, all the algorithms achieve similar performance (except Sauté-TRPO), successfully maximising returns (Figure 6d **top**) while minimising the failure rates (Figure 6c **top**). However, for the stochastic cases $noise > 0$, we can observe that all the baselines except Sauté-TRPO achieve significantly high returns (Figure 6d) at the expense of a rapidly increasing cumulative cost (Figure 6b). These results are also consistent with the benchmarks of Ray et al. (2019) where the cumulative cost of TRPO is greater than that of TRPO-Lagrangian, which is greater than that of CPO. Interestingly, Sauté-TRPO is the worst-performing of all the baselines. It successfully maximises returns while minimising cost only for the deterministic environment ($noise = 0$), but completely fails for the stochastic ones ($noise > 0$). Finally, by examining the episode length (Figure 6a) and failure rates (Figure 6c) for all the baselines in the stochastic cases, we can conclude that they have all learned risky policies that maximise rewards over short trajectories that are highly likely to result in collisions.

In contrast, the results obtained show that TRPO-Minmax successfully solves the tasks while minimising cost for both deterministic and stochastic environments, when the noise levels are not too high ($noise \in [0, 2.5]$). When the noise level is too high ($noise = 5$), TRPO-Minmax consistently prioritises maintaining low failure rates over maximising returns. In addition, we can observe from the episode lengths that TRPO-Minmax chooses the shortest path to the goal when there is no noise, but chooses longer paths as the noise increases. This demonstrates its ability to trade off between rewards maximisation and safety, with a strong bias towards safety—in contrast to the baselines which seem strongly biased towards reward maximisation. This can also be seen from evaluating the learned policies, as shown in Table 1 in the Appendix.

6 RELATED WORK

Guiding agents toward desirable behaviors has been explored through reward shaping, which augments reward functions to improve learning efficiency but requires that the optimal policy is



(a) Episode length (↑) (b) Cummulative cost (↓) (c) Failure rate (↓) (d) Average returns (↑)

Figure 6: Performance comparison in the PILLAR environment with varying noise. **(top)** $noise = 0$, **(middle)** $noise = 2.5$, and **(bottom)** $noise = 5$. Since the environment is noisy, higher episode lengths are better (↑) because that means choosing safer longer paths (except for $noise = 0$). The results are averaged over 10 random seeds and the shaded regions represent one standard deviation.

unaltered (Ng et al., 1999; Devidze et al., 2021). This is undesirable in safe RL where the optimal policy may be unsafe according to some safety constraints. More popularly, works in constrained RL usually impose safety constraints to limit cost violations while maximizing rewards (Ray et al., 2019; Sootla et al., 2022). In contrast, our work optimizes terminal state rewards to minimize undesirable behaviors directly. Finally, other works like Shielding complements these approaches by using model or human interventions to prevent unsafe actions (Dalal et al., 2018; Wagener et al., 2021; Tennenholtz et al., 2022). As shielding typically modifies transition dynamics rather than reward functions, it aligns naturally with our reward-focused framework. See Appendix C for an expended related works.

7 Discussion and Future Work

This paper investigates a new approach towards safe RL by asking the question: *Is a scalar reward enough to solve tasks safely?* To answer this question, we bound the Minmax penalty, which takes into account the diameter and solvability of an environment in order to minimise the probability of encountering unsafe states. We prove that the penalty does indeed minimise this probability, and present a method that uses an agent's value estimates to learn an estimate of the penalty. Our results in tabular and high-dimensional continuous settings have demonstrated that, by encoding the safe behaviour directly in the reward function via the Minmax penalty, agents are able to solve tasks while prioritising safety, learning safer policies than popular constraint-based approaches. Our method is also easy to incorporate with any off-the-shelf RL algorithms that maintain value estimates, requiring no changes to the algorithms themselves. By autonomously learning the penalty, our method also alleviates the need for a human designer to manually tweak rewards or cost functions to elicit safe behaviour.

Finally, while we show that scalar rewards are indeed enough for safe RL, the current analysis is only applicable to unsafe terminal states—which only covers tasks that can be naturally represented by stochastic-shortest path MDPs. Given that other popular RL settings like discounted MDPs can be converted to stochastic shortest path MDPs (Bertsekas, 1987; Sutton & Barto, 1998), a promising future direction could be to find the dual of our results for other theoretically equivalent settings. In conclusion, we see this reward-only approach as a promising direction towards truly autonomous agents capable of independently learning to solve tasks safely.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31. PMLR, 2017.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Eitan Altman. *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

Dimitri P Bertsekas. *Dynamic Programming: Determinist. and Stochast. Models*. Prentice-Hall, 1987.

Dimitri P Bertsekas and John N Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A Lyapunov-based approach to safe reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Rati Devidze, Goran Radanovic, Parameswaran Kamalaruban, and Adish Singla. Explicable reward design for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34: 20118–20131, 2021.

Aria HasanzadeZonuzy, Archana Bura, Dileep Kalathil, and Srinivas Shakkottai. Learning with safety constraints: Sample complexity of reinforcement learning for constrained MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7667–7674, 2021.

Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *Journal of Machine Learning Research*, 25(285):1–6, 2024. URL http://jmlr.org/papers/v25/23-0681.html.

Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5129–5136. IEEE, 2018.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673. PMLR, 2018.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

Zachary C Lipton, Kamyar Azizzadenesheli, Abhishek Kumar, Lihong Li, Jianfeng Gao, and Li Deng. Combating reinforcement learning's Sisyphean curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*, 2016.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pp. 278–287, 1999.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019.

Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897. PMLR, 2015.

Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*, 2019.

Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from? In *Proceedings of the Annual Conference of the Cognitive Science Society*, pp. 2601–2606. Cognitive Science Society, 2009.

Aivar Sootla, Alexander I Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H Mguni, Jun Wang, and Haitham Ammar. Sauté RL: Almost surely safe reinforcement learning using state augmentation. In *International Conference on Machine Learning*, pp. 20423–20443. PMLR, 2022.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID Lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

Richard Sutton and Andrew Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Geraud Nangue Tasse, Devon Jarvis, Steven James, and Benjamin Rosman. Skill machines: Temporal logic skill composition in reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2022.

Guy Tennenholtz, Nadav Merlis, Lior Shani, Shie Mannor, Uri Shalit, Gal Chechik, Assaf Hallak, and Gal Dalal. Reinforcement learning with a terminator. *Advances in Neural Information Processing Systems*, 35:35696–35709, 2022.

Benjamin Van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. In *International Conference on Machine Learning*, pp. 6401–6409. PMLR, 2019.

Nolan C Wagener, Byron Boots, and Ching-An Cheng. Safe reinforcement learning using advantage-based intervention. In *International Conference on Machine Learning*, pp. 10630–10640. PMLR, 2021.

C. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

Yujie Yang, Yuxuan Jiang, Yichen Liu, Jianyu Chen, and Shengbo Eben Li. Model-free safe reinforcement learning through neural barrier certificate. *IEEE Robotics and Automation Letters*, 8 (3):1295–1302, 2023.

Yiming Zhang, Quan Vuong, and Keith Ross. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 33:15338–15349, 2020.

A Proofs of Theoretical Results

Theorem 1 (Estimation) *Algorithm 2 converges to D and C for any given solvable environment.*

Proof This follows from the convergence guarantee of policy evaluation (Sutton & Barto, 1998). ■

Theorem 2 (Safety Bounds) *Consider a solvable environment where task rewards are bounded by $[R_{MIN}\ R_{MAX}]$ for all $s' \notin \mathcal{G}^!$. Then $\bar{R}_{MIN} \leq R_{Minmax} \leq \bar{R}_{MAX}$.*

Proof Let π^* be an optimal policy for an arbitrary task $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$ in the environment. Given the definition of the Minmax penalty (Definition 2), we need to show the following:

(i) If $R(s, a, s') < \bar{R}_{\text{MIN}}$ for all $s' \in \mathcal{G}^!$, then π^* is safe for all R; and

(ii) If $R(s, a, s') > \bar{R}_{\text{MAX}}$ for some $s' \in \mathcal{G}^!$ reachable from $\mathcal{S} \setminus \mathcal{G}$, then there exists an R s.t. π^* is unsafe.

(i) Since π^* is optimal, it is also proper and hence must reach $\mathcal{G}$.

Assume π^* is unsafe. Then there exists another proper policy π that is safe, such that

$$P_s^{\pi}(s_T \in \mathcal{G}^!) < P_s^{\pi^*}(s_T \in \mathcal{G}^!) \quad \text{for some } s \in \mathcal{S}.$$

Then,

$$
\begin{aligned}
& V^{\pi^*}(s) \geq V^{\pi}(s) \\
\implies & \mathbb{E}_s^{\pi^*}\left[\sum_{t=0}^{\infty} R(s_t, a_t, s_{t+1})\right] \geq \mathbb{E}_s^{\pi}\left[\sum_{t=0}^{\infty} R(s_t, a_t, s_{t+1})\right] \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1} + R(s_T, a_T, s_{T+1})\right] \geq \mathbb{E}_s^{\pi}\left[G^{T-1} + R(s_T, a_T, s_{T+1})\right], \\
& \text{where } G^{T-1} = \sum_{t=0}^{T-1} R(s_t, a_t, s_{t+1}) \text{ and } T \text{ is a random variable denoting when } s_{T+1} \in \mathcal{G}. \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(P_s^{\pi^*}(s_T \notin \mathcal{G}^!) R(s_T, a_T, s_{T+1}) + P_s^{\pi^*}(s_T \in \mathcal{G}^!) \bar{R}_{\text{unsafe}}(s_T, a_T, s_{T+1})\right) \\
& \geq \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \notin \mathcal{G}^!) R(s_T, a_T, s_{T+1}) + P_s^{\pi}(s_T \in \mathcal{G}^!) \bar{R}_{\text{unsafe}}(s_T, a_T, s_{T+1})\right), \\
& \text{where } \bar{R}_{\text{unsafe}} \text{ denotes the rewards for transitions into } \mathcal{G}^! \text{ and } a_T = \pi^*(s_T). \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(P_s^{\pi^*}(s_T \notin \mathcal{G}^!) R(s_T, a_T, s_{T+1}) + \bar{R}_{\text{unsafe}}(s_T, a_T, s_{T+1})\right) \\
& \geq \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \notin \mathcal{G}^!) R(s_T, a_T, s_{T+1}) + P_s^{\pi}(s_T \in \mathcal{G}^!) \bar{R}_{\text{unsafe}}(s_T, a_T, s_{T+1})\right), \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(1 - P_s^{\pi}(s_T \in \mathcal{G}^!)\right) \bar{R}_{\text{unsafe}}(s_T, a_T, s_{T+1}) \\
& \geq \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \notin \mathcal{G}^!) - P_s^{\pi^*}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}) \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(1 - P_s^{\pi}(s_T \in \mathcal{G}^!)\right) \bar{R}_{\text{MIN}} \\
& > \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \notin \mathcal{G}^!) - P_s^{\pi^*}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}), \\
& \text{since } \bar{R}_{\text{unsafe}}(s_T, a_T, s_{T+1}) < \bar{R}_{\text{MIN}}. \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(1 - P_s^{\pi}(s_T \in \mathcal{G}^!)\right) (R_{\text{MIN}} - R_{\text{MAX}}) \frac{D}{C} \\
& > \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \notin \mathcal{G}^!) - P_s^{\pi^*}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}) \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + (R_{\text{MIN}} - R_{\text{MAX}}) D \\
& > \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \notin \mathcal{G}^!) - P_s^{\pi^*}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}), \text{ using definition of } C. \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] - R_{\text{MAX}} D
\end{aligned}
$$

$$
\begin{aligned}
&> \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \notin \mathcal{G}^!) - P_s^{\pi^*}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}) - R_{\text{MIN}} D \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] - R_{\text{MAX}} D > 0, \\
&\text{since } \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \notin \mathcal{G}^!) - P_s^{\pi^*}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}) \geq R_{\text{MIN}} D \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] > R_{\text{MAX}} D.
\end{aligned}
$$

But this is a contradiction since the expected return of following an optimal policy up to a terminal state without the reward for entering the terminal state must be less than receiving R_{MAX} for every step of the longest possible trajectory to $\mathcal{G}$. Hence we must have $\pi^* \in \arg\min_{\pi} P_s^{\pi}(s_T \in \mathcal{G}^!)$.

(ii) Assume π^* is safe. Then, $P_s^{\pi^*}(s_T \notin \mathcal{G}^!) \geq P_s^{\pi'}(s_T \notin \mathcal{G}^!)$ for all $s \in \mathcal{S}$, $\pi' \in \Pi$.

Let π be the policy that maximises the probability of reaching $s' \in \mathcal{G}^!$ from some state $s \in \mathcal{G}$. Then, similarly to (i), we have

$$
\begin{aligned}
& V^{\pi^*}(s) \geq V^{\pi}(s) \\
\implies & \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(P_s^{\pi^*}(s_T \in \mathcal{G}^!) - P_s^{\pi}(s_T \in \mathcal{G}^!)\right) \bar{R}_{\text{unsafe}}(s_T, a_T, s_{T+1}) \\
&\geq \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \notin \mathcal{G}^!) - P_s^{\pi^*}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}) \\
\implies & \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \in \mathcal{G}^!) - P_s^{\pi^*}(s_T \in \mathcal{G}^!)\right) \bar{R}_{\text{unsafe}}(s_T, a_T, s_{T+1}) \\
&\leq \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(P_s^{\pi^*}(s_T \notin \mathcal{G}^!) - P_s^{\pi}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}) \\
\implies & \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \in \mathcal{G}^!) - P_s^{\pi^*}(s_T \in \mathcal{G}^!)\right) \bar{R}_{\text{MAX}} \\
&< \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(P_s^{\pi^*}(s_T \notin \mathcal{G}^!) - P_s^{\pi}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}), \text{ since } \bar{R}_{\text{unsafe}} > \bar{R}_{\text{MAX}}. \\
\implies & \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + \left(P_s^{\pi}(s_T \in \mathcal{G}^!) - P_s^{\pi^*}(s_T \in \mathcal{G}^!)\right) R_{\text{MIN}} D' \\
&< \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(P_s^{\pi^*}(s_T \notin \mathcal{G}^!) - P_s^{\pi}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}), \text{ by definition of } \bar{R}_{\text{MAX}}. \\
\implies & \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + R_{\text{MIN}} D' \\
&< \mathbb{E}_s^{\pi^*}\left[G^{T-1}\right] + \left(P_s^{\pi^*}(s_T \notin \mathcal{G}^!) - P_s^{\pi}(s_T \notin \mathcal{G}^!)\right) R(s_T, a_T, s_{T+1}) \\
\implies & \mathbb{E}_s^{\pi}\left[G^{T-1}\right] + R_{\text{MIN}} D' < 0
\end{aligned}
$$

But this is a contradiction when R is such that the agent receives a reward of $R_{\text{MAX}} \geq |R_{\text{MIN}}| D'$ at least once in its trajectory when following π and zero everywhere else.

■

Theorem 3 (Complexity) *Estimating the Minmax penalty R_{Minmax} accurately is NP-hard.*

Proof This follows from the NP-hardness of longest-path problems. Since the Minmax penalty is bounded by $\bar{R}_{\text{MIN}}$ and $\bar{R}_{\text{MAX}}$, both are defined by the diameter, which is in turn defined as the expected total timesteps of the longest path. ■

B Algorithms

Algorithm 2: Estimating the Diameter and Solvability

Input : $\langle \mathcal{S}, \mathcal{A}, P \rangle$, $R_D(s') \coloneqq \mathbb{1}(s' \notin \mathcal{G})$, $R_C(s,a,s') \coloneqq \mathbb{1}(s \notin \mathcal{G}$ and $s' \in \mathcal{G} \setminus \mathcal{G}^!)$
Initialise : Diameter $D = 0$, Solvability $C = 1$, Value functions $V_D^\pi(s) = 0$, $V_C^\pi(s) = 0$, Error $\Delta = 1$

```
for π ∈ Π do
    /* Policy evaluation for D */
    while Δ > 0 do
        Δ ← 0
        for s ∈ S do
            v′ ← Σ_{s′} P(s′|s, π(s))(R_D(s′) + V_D^π(s′))
            Δ = max{Δ, |V_D^π(s) − v′|}
            V_D^π(s) ← v′
        end for
    end while
    for s ∈ S do
        D = max{D, V_D^π(s)}
    end for
end for

for π ∈ Π do
    /* Policy evaluation for C */
    while Δ > 0 do
        Δ ← 0
        for s ∈ S do
            v′ ← Σ_{s′} P(s′|s, π(s))(R_C(s, π(s), s′) + V_C^π(s′))
            Δ = max{Δ, |V_C^π(s) − v′|}
            V_C^π(s) ← v′
        end for
    end while
    for s ∈ S do
        C = min{C, V_C^π(s)} if V_C^π(s) ≠ 0 else C
    end for
end for
```

C Extended Related Work

C.1 Reward Shaping

The problem of designing reward functions to produce desired policies in RL settings is well-studied (Singh et al., 2009). Particular focus has been placed on the practice of *reward shaping*, in which an initial reward function provided by an MDP is augmented in order to improve the rate at which an agent learns the same optimal policy (Ng et al., 1999; Devidze et al., 2021). While sacrificing some optimality, other approaches like Lipton et al. (2016) propose shaping rewards using an idea of intrinsic fear. Here, the agent trains a supervised fear model representing the probability of reaching unsafe states in a fixed horizon, scales said probabilities by a fear factor, and then subtracts the scaled probabilities from Q-learning targets.

These approaches differ from ours in that they seek to find reward functions that improve convergence while preserving the optimality from an initial reward function. In contrast, we seek to determine the optimal rewards for terminal states in order to minimise undesirable behaviours irrespective of the original reward function and optimal policy.

C.2 Constrained RL

Disincentivising or preventing undesirable behaviours is core to the field of safe RL. A popular approach is to define constraints on the behaviour of an agent using CMDPs, tasking the agent with limiting the accumulation of costs associated with violating safety constraints while simultaneously maximising reward (Altman, 1999; Achiam et al., 2017; Chow et al., 2018; Ray et al., 2019; HasanzadeZonuzy et al., 2021). Widely used examples of these approaches include constrained policy optimisation (CPO) (Achiam et al., 2017), which augments TRPO (Schulman et al., 2015) with constraints to satisfy a constrained MDP, and TRPO-Lagrangian (Ray et al., 2019), which combines Lagrangian methods with TRPO. Another example is Sauté RL (Sootla et al., 2022), which incorporates the cost function into the rewards and augments the state with the remaining ”cost budget” spent by violating safety constraints. Other constraint-based approaches include Projection-based CPO (Yang et al., 2020), which projects a TRPO policy onto a space defined by constraints, and PID Lagrangian methods (Stooke et al., 2020), which augment Lagrangian methods with PID control.

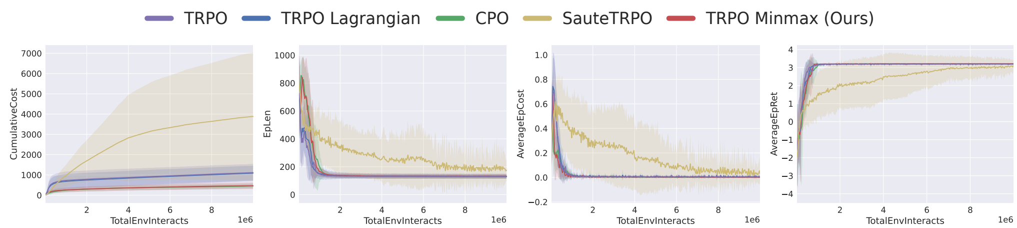
In deterministic environments with a cost threshold of 0, the set of safe policies for these approaches are the same as ours. However, in stochastic environments, these approaches require the correct choice of inequality constraints to even be well defined. If the cost threshold is not carefully chosen, there may exist no policy that satisfies the CMDP constraints, implying there would exist no optimal safe policy to converge to. For example, in the Lava Gridworld or the Pillar domains with $noise > 0$, a cost threshold of 0 can never be satisfied by any policy for all states, making these approaches theoretically ill-defined in these environments with that cost threshold. That said, we found in practice that a cost threshold of 0 gave them the best performance in the safety-gym experiments (compared to 1 and the default of 25). In contrast, we showed the existence of a Minmax penalty irrespective of the stochasticity of the environment. Additionally, while these approaches in general theoretically define or learn safety parameters—like Lagrange coefficients—for each reward function even when the cost function and cost threshold remain unchanged, our minmax penalty approach is theoretically defined and learned for all reward functions.

C.3 Shielding

Finally, another important line of work involves relying on interventions from a model (Dalal et al., 2018; Wagener et al., 2021) or human (Tennenholtz et al., 2022) to prevent unsafe actions from being considered by the agent (shielding the agent) or prevent the environment from executing those unsafe actions by correcting them (shielding the environment). Other approaches here also look at using temporal logics to define or enforce safety constraints on the actions considered or selected by the agent (Alshiekh et al., 2018).

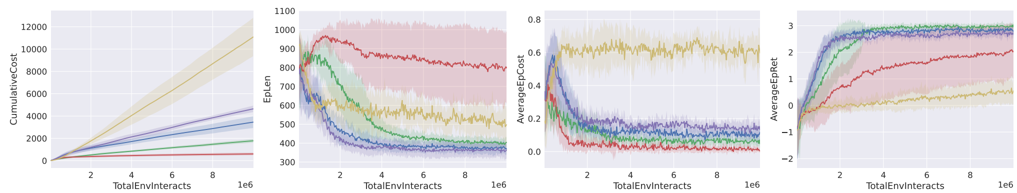
These approaches fit seamlessly into our proposed reward-only framework since they are primarily about modifications on the transition dynamics and not the reward function—for example, unsafe actions here can simply lead to unsafe goal states.

D Safety-Gym Pillar Training and Testing results with Ray et al. (2019) baselines



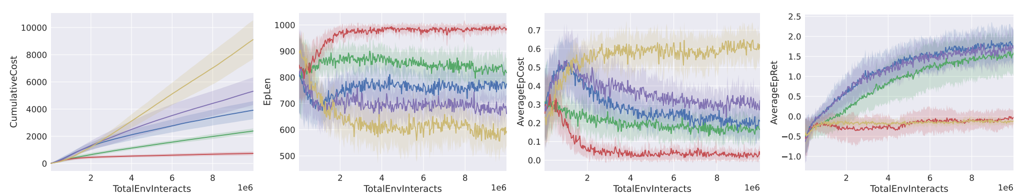
(a) Cumulative cost (↓) (b) Episode length (↓) (c) Failure rate (↓) (d) Average returns (↑)

Figure 7: Training curves in the Pillar environment with $noise = 0$.



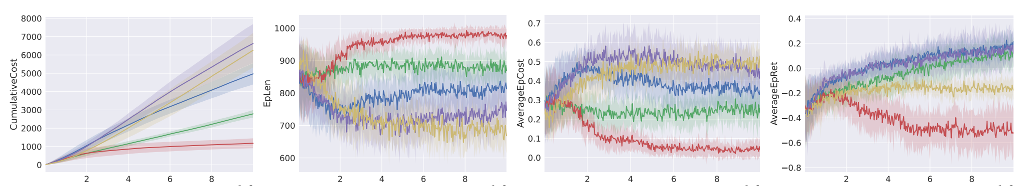
(a) Cumulative cost (↓) (b) Episode length (↑) (c) Failure rate (↓) (d) Average returns (↑)

Figure 8: Training curves in the Pillar environment with $noise = 2.5$.



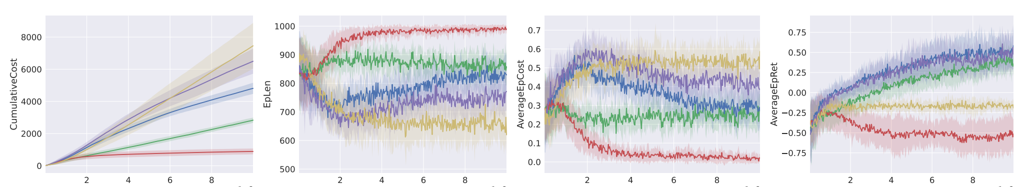
(a) Cumulative cost (↓) (b) Episode length (↑) (c) Failure rate (↓) (d) Average returns (↑)

Figure 9: Training curves in the Pillar environment with $noise = 5$.



(a) Cumulative cost (↓) (b) Episode length (↑) (c) Failure rate (↓) (d) Average returns (↑)

Figure 10: Training curves in the Pillar environment with $noise = 7.5$.

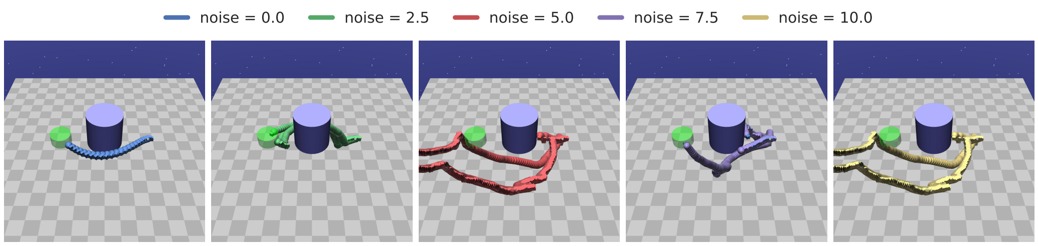


(a) Cumulative cost (↓) (b) Episode length (↑) (c) Failure rate (↓) (d) Average returns (↑)

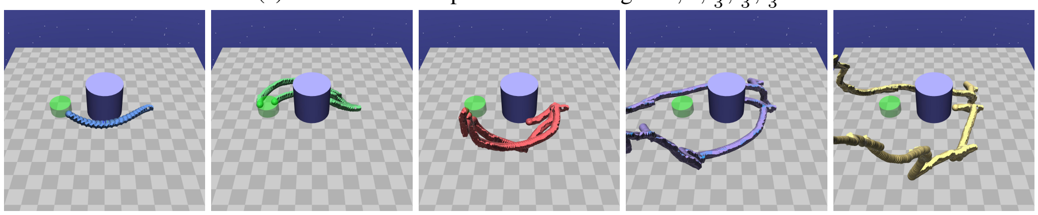
Figure 11: Training curves in the Pillar environment with $noise = 10$.

Noise	Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps ↓
0.0	TRPO	**0.00** ± **0.00**	**1.00** ± **0.00**	**3.21** ± **0.00**	130.30 ± 14.94
	TRPO-Lagrangian	**0.00** ± **0.01**	**1.00** ± **0.01**	3.20 ± 0.02	132.16 ± 14.43
	CPO	**0.00** ± **0.00**	**1.00** ± **0.00**	**3.21** ± **0.01**	**128.06** ± **14.40**
	Sauté-TRPO	0.04 ± 0.19	0.95 ± 0.21	3.09 ± 0.55	176.51 ± 117.93
	TRPO-Minmax	**0.00** ± **0.00**	**1.00** ± **0.00**	**3.21** ± **0.01**	131.53 ± 15.15
					Total Steps ↑
2.5	TRPO	0.18 ± 0.03	0.82 ± 0.03	2.58 ± 0.12	351.33 ± 40.17
	TRPO-Lagrangian	0.13 ± 0.03	0.86 ± 0.02	2.73 ± 0.09	364.41 ± 32.24
	CPO	0.08 ± 0.03	**0.92** ± **0.03**	**2.91** ± **0.10**	393.36 ± 29.50
	Sauté-TRPO	0.62 ± 0.49	0.16 ± 0.37	0.59 ± 1.27	484.24 ± 340.57
	TRPO-Minmax	**0.02** ± **0.02**	0.47 ± 0.38	2.00 ± 1.02	**799.41** ± **181.46**
5.0	TRPO	0.32 ± 0.07	**0.41** ± **0.16**	1.66 ± 0.43	665.62 ± 38.34
	TRPO-Lagrangian	0.20 ± 0.07	0.39 ± 0.16	**1.78** ± **0.47**	760.66 ± 43.54
	CPO	0.18 ± 0.04	0.27 ± 0.21	1.53 ± 0.54	807.28 ± 51.38
	Sauté-TRPO	0.62 ± 0.49	0.01 ± 0.07	-0.09 ± 0.54	594.09 ± 363.81
	TRPO-Minmax	**0.05** ± **0.03**	0.00 ± 0.00	-0.00 ± 0.19	**975.59** ± **17.81**
7.5	TRPO	0.43 ± 0.06	**0.02** ± **0.03**	0.45 ± 0.21	726.97 ± 31.42
	TRPO-Lagrangian	0.30 ± 0.06	0.01 ± 0.01	**0.55** ± **0.18**	806.91 ± 41.44
	CPO	0.28 ± 0.04	0.00 ± 0.01	0.38 ± 0.13	830.78 ± 25.03
	Sauté-TRPO	0.54 ± 0.50	0.00 ± 0.03	-0.15 ± 0.48	650.94 ± 364.90
	TRPO-Minmax	**0.02** ± **0.02**	0.00 ± 0.00	-0.46 ± 0.20	**989.69** ± **7.78**
10.0	TRPO	0.46 ± 0.08	**0.00** ± **0.00**	0.13 ± 0.11	725.03 ± 49.64
	TRPO-Lagrangian	0.36 ± 0.09	**0.00** ± **0.00**	**0.17** ± **0.09**	789.52 ± 42.68
	CPO	0.27 ± 0.06	**0.00** ± **0.00**	0.10 ± 0.10	859.58 ± 30.94
	Sauté-TRPO	0.46 ± 0.50	**0.00** ± **0.00**	-0.18 ± 0.48	701.60 ± 355.32
	TRPO-Minmax	**0.07** ± **0.05**	**0.00** ± **0.00**	-0.48 ± 0.20	**960.96** ± **28.39**

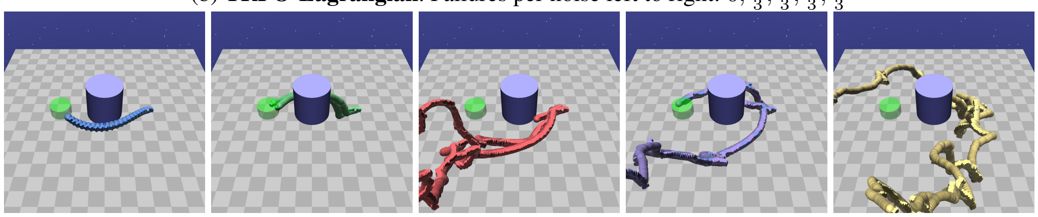
Table 1: Evaluation of trained models with Ray et al. (2019) baselines in the PILLAR environment with varying noise levels. For each algorithm in each noise level, we train using 10 random seeds for 10 million steps and evaluate the learned policies over 100 random seeds, for a total of 1000 evaluation episodes. We report the mean and standard errors of various performance metrics, **bolding** the ones with the best mean. Figures 7-11 shows the training curves. Here, higher episode lengths are better for $noise > 0$ because that means the policy is taking longer safer paths. We observe that only TPRO-Minmax prioritises minimising the probability of unsafe transitions, consistently achieving the lowest cost while trading off the rewards. It achieves the same highest success rate as the baselines only in the deterministic case, since the pure maximisation of rewards here doesn't come at the cost of higher unsafe transitions. It also does not completely ignore the rewards when the noise is not too large ($noise = 2.5$). We can also observe from the training curves of $noise = 2.5$ (Figure 8) that TPRO-Minmax has not converged in its rewards performance and is still increasing.



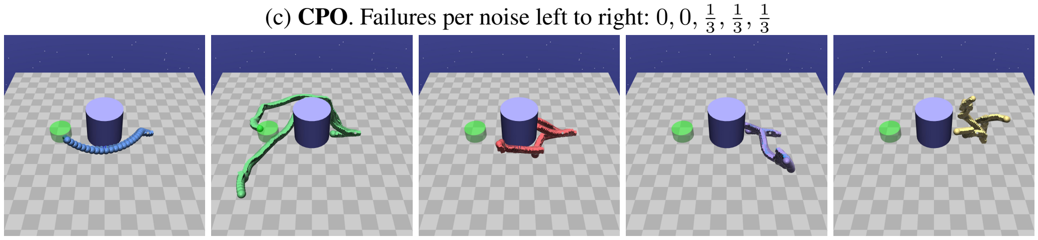
(a) **TRPO**. Failures per noise left to right: $0, 0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}$



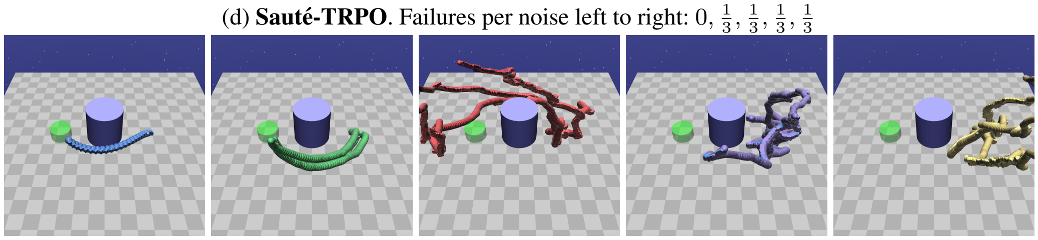
(b) **TRPO-Lagrangian**. Failures per noise left to right: $0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}$



(c) **CPO**. Failures per noise left to right: $0, 0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}$



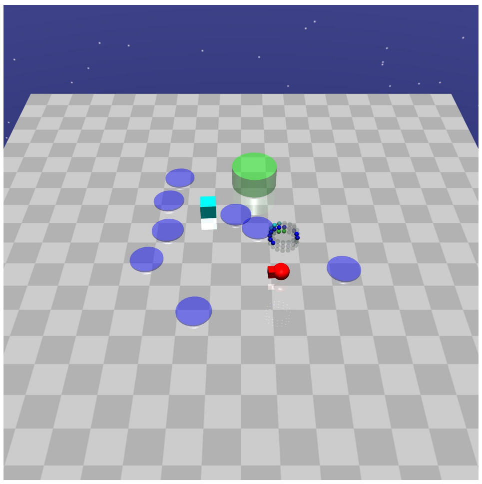
(d) **Sauté-TRPO**. Failures per noise left to right: $0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}$



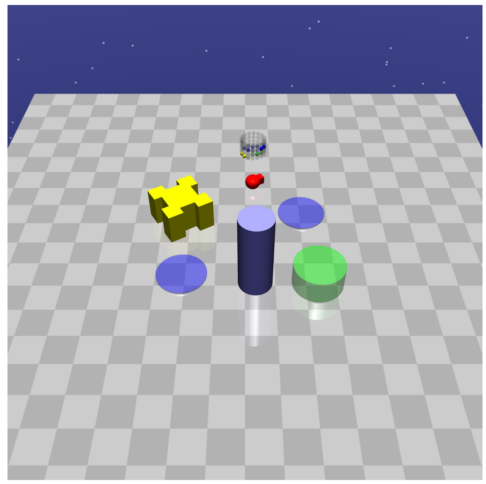
(e) **TRPO-Minmax**. Failures per noise left to right: $0, 0, 0, \frac{1}{3}, 0$

Figure 12: Sample trajectories of policies learned by each baseline and our **TRPO-Minmax** approach in the Safety Gym PILLAR environment with varying noise levels. To sample the trajectories for each noise level, we use the same three environment random seeds across all the algorithms. We can observe that $noise \geq 5$ is too noisy to learn safe policies, at least after 10 million training steps.

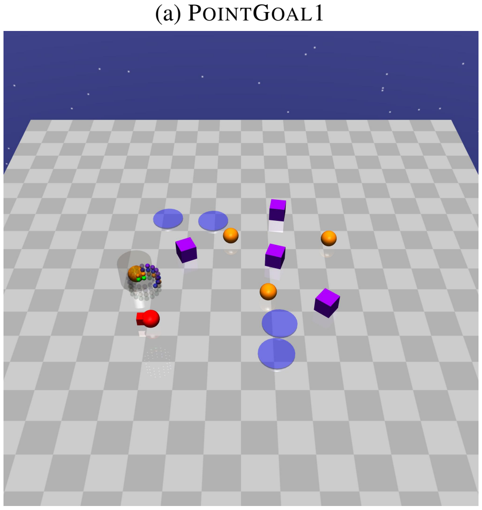
E ABLATIONS IN SAFETY-GYM DEFAULT ENVIRONMENTS WITH RAY ET AL. (2019) BASELINES



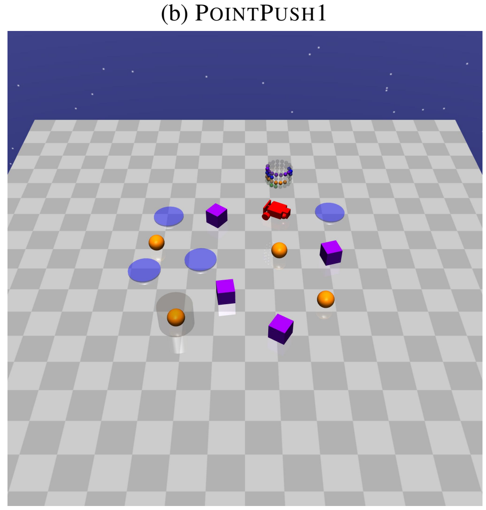
(a) POINTGOAL1



(b) POINTPUSH1

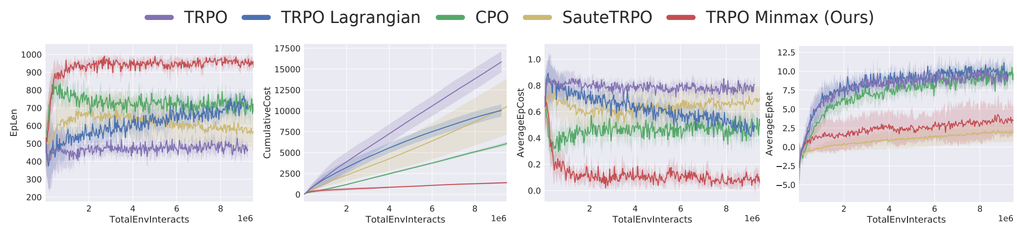


(c) POINTBUTTON1



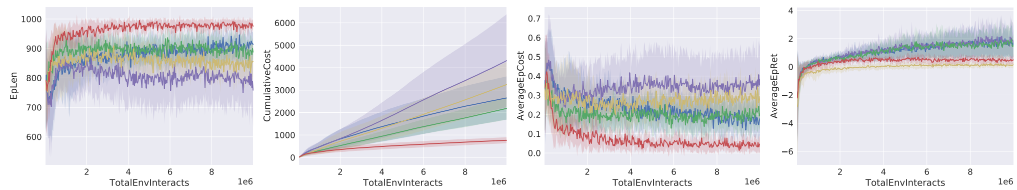
(d) CARBUTTON1

Figure 13: Sample default task's from OpenAI's Safety Gym environments (Ray et al., 2019). We use these to investigate the effect of termination in complex, high-dimensional, continuous control tasks. In all of the default tasks, $\mathcal{G} = \emptyset$ by default. (a) Here, a simple robot must navigate to a goal location across a 2D plane while avoiding several hazards. The agent's sensors, actions, and rewards are identical to the PILLAR domain. Unlike the PILLAR domain, the goal location is randomly reset when the agent reaches it, but does not terminate the episode. (b) This task is similar to POINTGOAL1, but with the addition of a pillar obstacle and a large box the agent must push to the goal location to receive the goal reward. (c-d) These tasks are also similar to POINTGOAL1, but with the more complex car robot for CARBUTTON1 and the addition of: (i) *Gremlins*, which are dynamic obstacles that move around the environment and must be avoided; and (ii) Buttons, where the agent must reach the goal button with a cylinder to receive the goal reward.



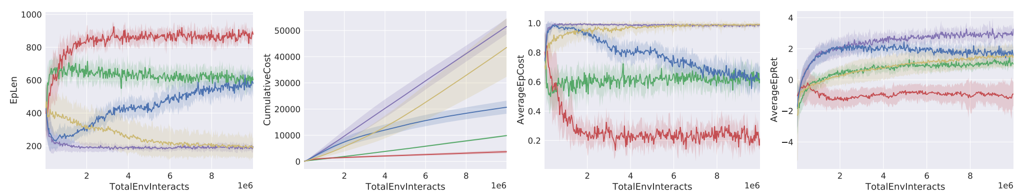
(a) Episode length (↑) (b) Cumulative cost (↓) (c) Failure rate (↓) (d) Average returns (↑)

Figure 14: Comparison with baselines in POINTGOAL1, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \{\}$. Here, higher episode lengths are better because episodes only terminate when the agent reaches $\mathcal{G}^!$ or after 1000 timesteps. Similar to Figure 6, all the baselines except Sauté-RL achieve significantly high returns at the expense of a rapidly increasing cumulative cost. By comparison, TRPO-Minmax dramatically reduces the failure rate while still being able to solve the task, as observed by average returns achieved as well as the trajectories observed. However, returns are lower since TRPO-Minmax learns safer longer paths to the goals (see sample trajectories in Figure 18).



(a) Episode length (↑) (b) Cumulative cost (↓) (c) Failure rate (↓) (d) Average returns (↑)

Figure 15: Comparison with baselines in POINTPUSH1, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \{,\}$. Here, higher episode lengths are better because episodes only terminate when the agent reaches $\mathcal{G}^!$ or after 1000 timesteps. Similar to Figure 6, the baselines achieve higher returns at the expense of a rapidly increasing cumulative cost while TRPO-Minmax consistently prioritises maintaining low failure rates by sacrificing rewards.

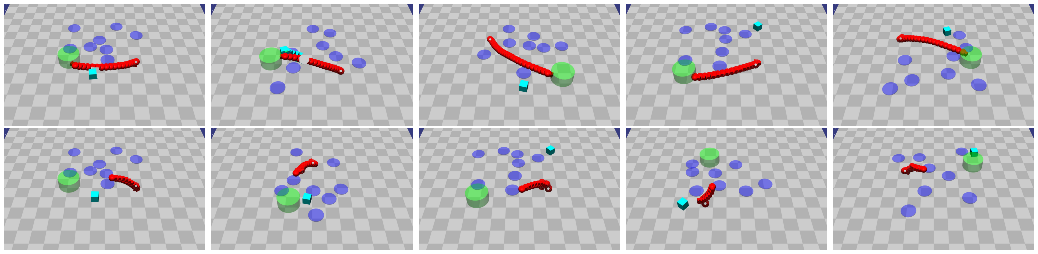


(a) Episode length (↑) (b) Cumulative cost (↓) (c) Failure rate (↓) (d) Average returns (↑)

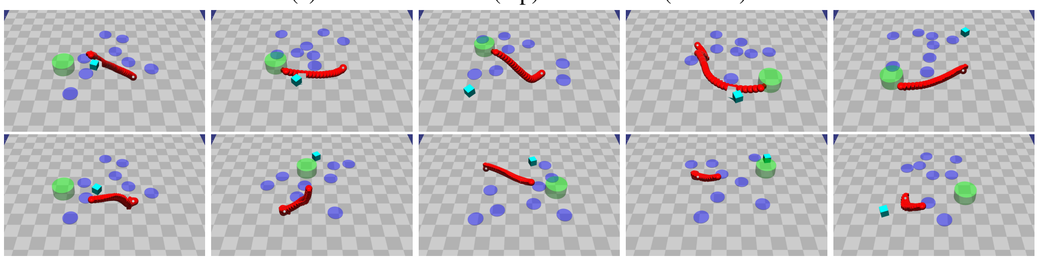
Figure 16: Comparison with baselines in POINTBUTTON1, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \{,,\}$. Here, higher episode lengths are better since epsiodes only terminate when the agent reaches $\mathcal{G}^!$ or after 1000 timesteps. Similar to Figure 6, the baselines achieve significantly high returns at the expense of a rapidly increasing cumulative cost while TRPO-Minmax consistently prioritises maintaining low failure rates.

(a) Episode length (↑) (b) Cumulative cost (↓) (c) Failure rate (↓) (d) Average returns (↑)

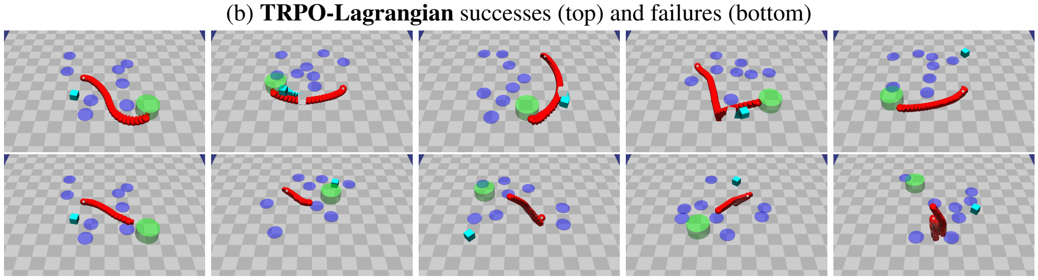
Figure 17: Comparison with baselines in CARBUTTON1, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \{,,\})$. Here, higher episode lengths are better since epsiodes only terminate when the agent reaches $\mathcal{G}^!$ or after 1000 timesteps. Similar to Figure 6, the baselines achieve significantly high returns at the expense of a rapidly increasing cumulative cost while TRPO-Minmax consistently prioritises maintaining low failure rates.



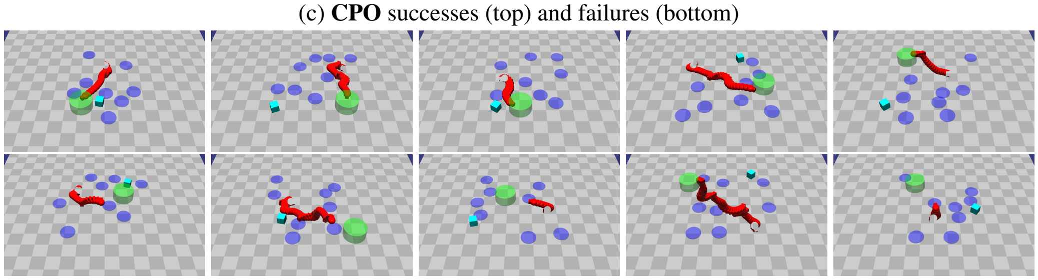
(a) **TRPO** successes (top) and failures (bottom)



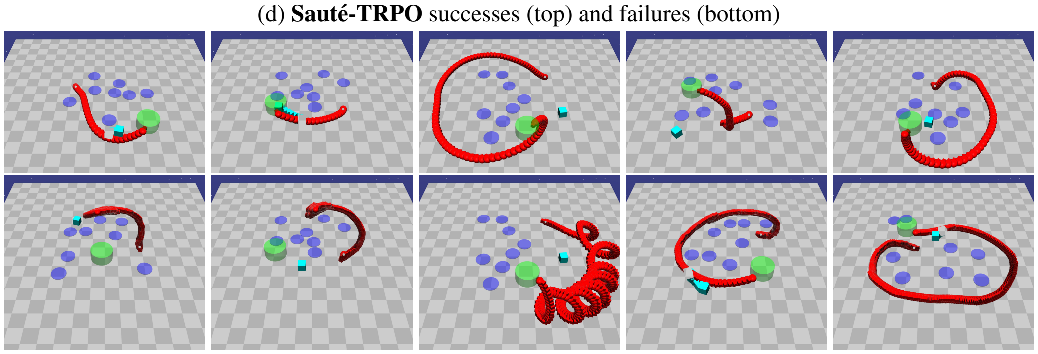
(b) **TRPO-Lagrangian** successes (top) and failures (bottom)



(c) **CPO** successes (top) and failures (bottom)

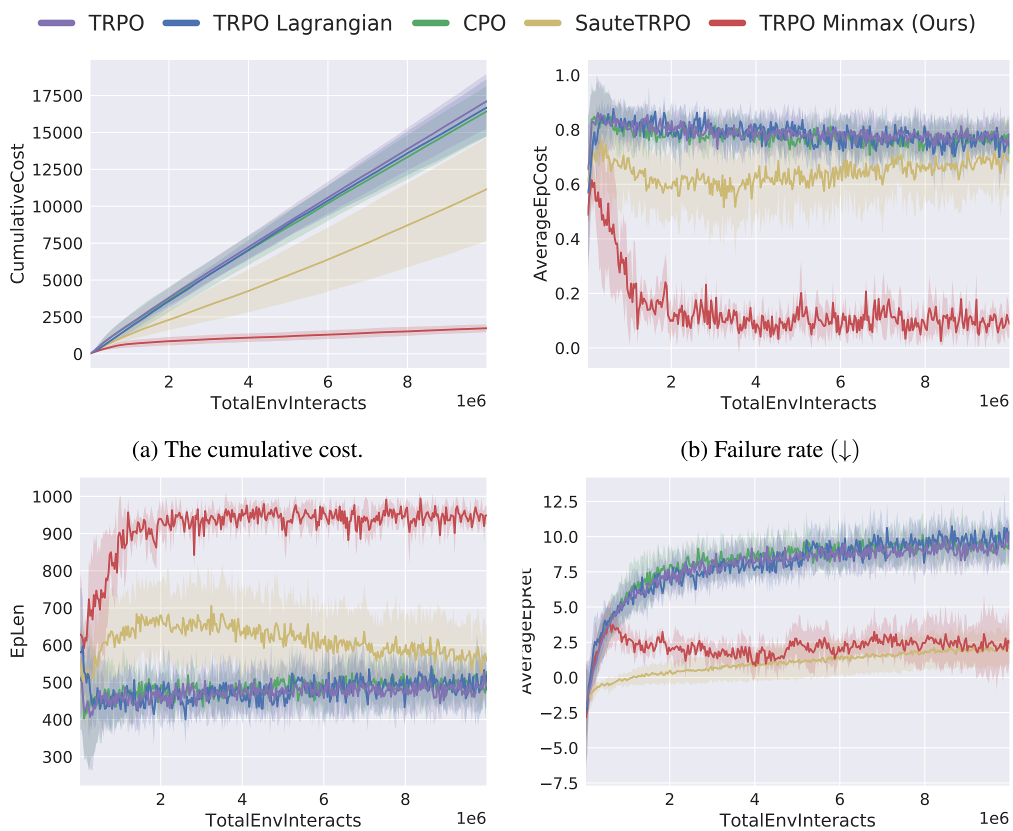


(d) **Sauté-TRPO** successes (top) and failures (bottom)



(e) **TRPO-Minmax** successes (top) and failures (bottom)

Figure 18: Sample trajectories of policies learned by each baseline and our Minmax approach in the Safety Gym PointGoal1 domain, in the experiments of Figure 14. Trajectories that hit hazards or take more than 1000 timesteps to reach the goal location are considered failures.



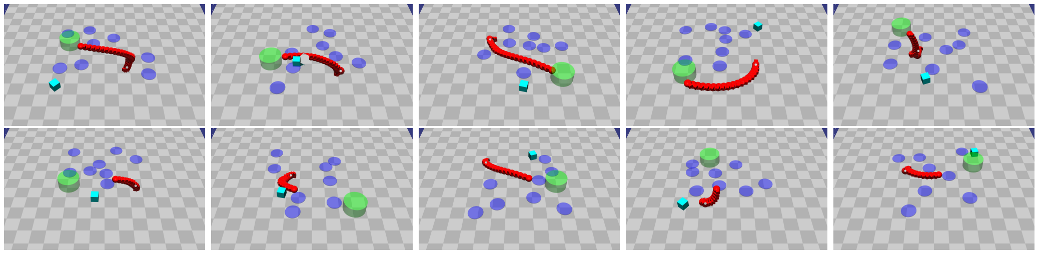
(a) The cumulative cost.

(b) Failure rate (↓)

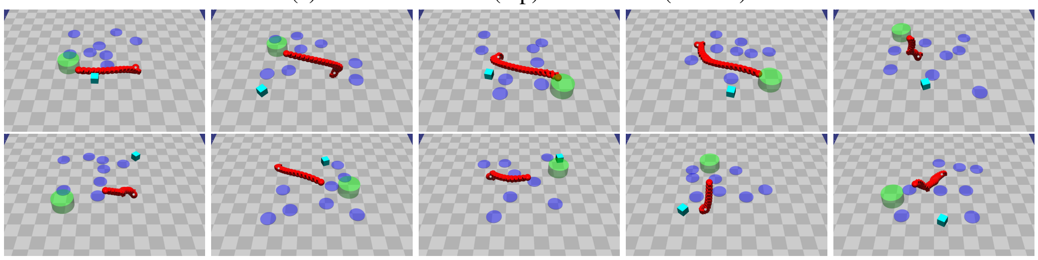
(c) Average episode length

(d) Average returns (↑)

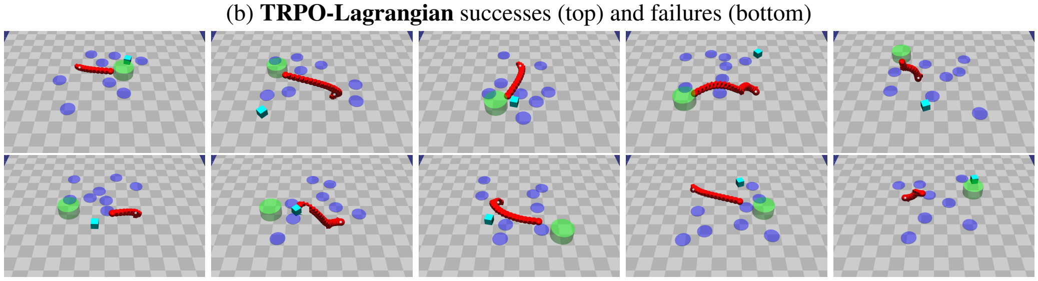
Figure 19: Comparison with baselines in POINTGOAL1, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \{\bigcirc\}$. Here, higher episode lengths are better since episodes only terminate when the agent reaches a hazard or after 1000 timesteps. This experiment is similar to Figure 14, but instead of a cost threshold of 0, it uses a cost threshold of 25 for the baselines (as in Ray et al. (2019)) to check its effect on the performance of the baselines when episodes immediately terminate at unsafe states. We can observe drastically worse failure rates and cumulative costs for the baselines compared to their performance in Figure 14. Similar results where obtained when using a cost threshold of 1. These show how sensitive such approaches are to the cost threshold, while a reward only approach like TRPO-Minmax does not depend on such hyperparameters.



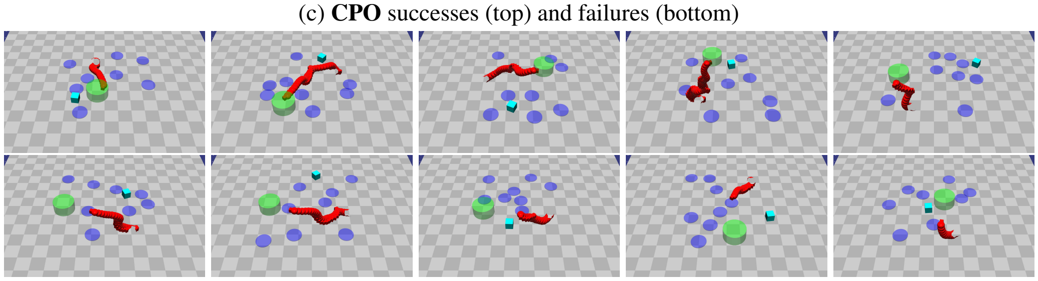
(a) **TRPO** successes (top) and failures (bottom)



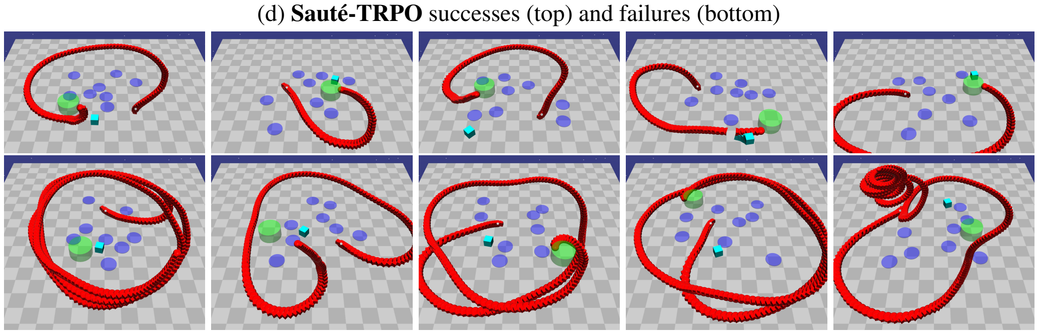
(b) **TRPO-Lagrangian** successes (top) and failures (bottom)



(c) **CPO** successes (top) and failures (bottom)

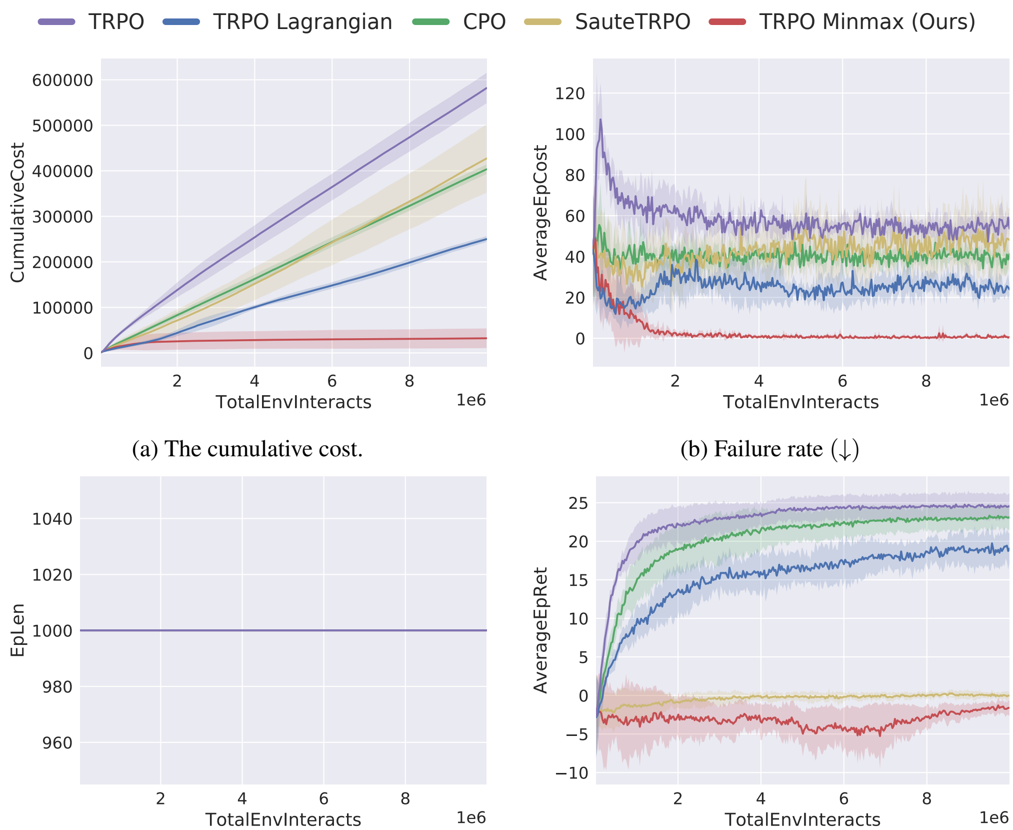


(d) **Sauté-TRPO** successes (top) and failures (bottom)



(e) **TRPO-Minmax** successes (top) and failures (bottom)

Figure 20: Sample trajectories of policies learned by each baseline and our Minmax approach in the Safety Gym POINTGOAL1 domain, in the experiments of Figure 19. Trajectories that hit hazards or take more than 1000 timesteps to reach the goal location are considered failures.



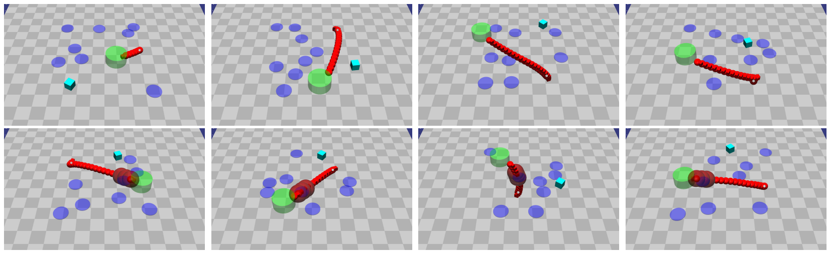
(a) The cumulative cost.

(b) Failure rate (↓)

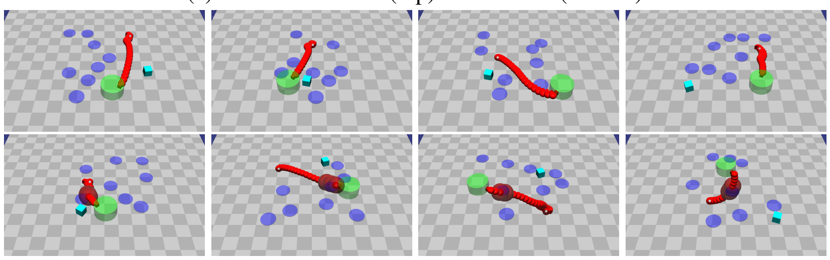
(c) Average episode length

(d) Average returns (↑)

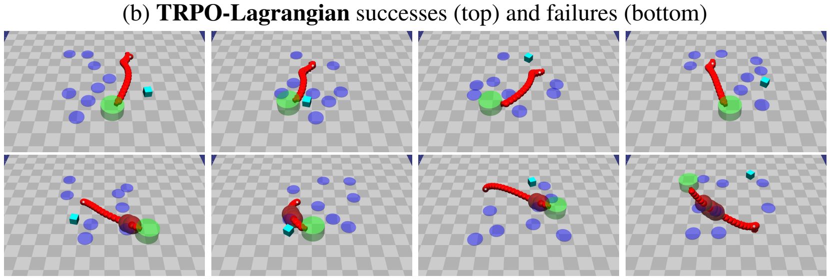
Figure 21: Comparison with baselines in the original Safety Gym POINTGOAL1 environment. Here, episodes do not terminate when a hazard is hit ($\mathcal{G} = \mathcal{G}^! = \emptyset$). Hence every episode only terminates after 1000 steps. We set the cost threshold for the baselines to 25 as in Ray et al. (2019). For TRPO-Minmax, we replace the reward with the Minmax penalty every time the agent is in an unsafe state (that is every time the cost is greater than zero), as in previous experiments and as per Algorithm 1. While TRPO-Minmax still beats the baselines in safe exploration (a-b), unlike the previous results with termination (Figure 19), it struggles to maximise rewards while avoiding unsafe states (d).



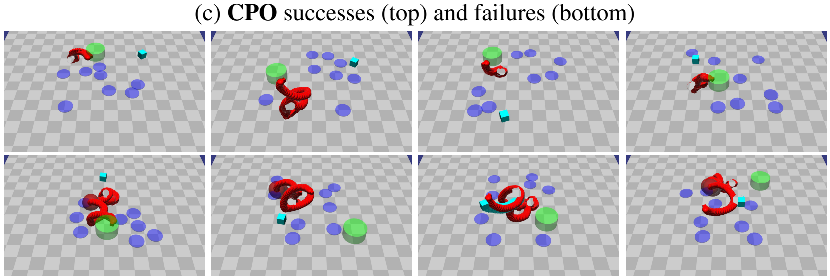
(a) **TRPO** successes (top) and failures (bottom)



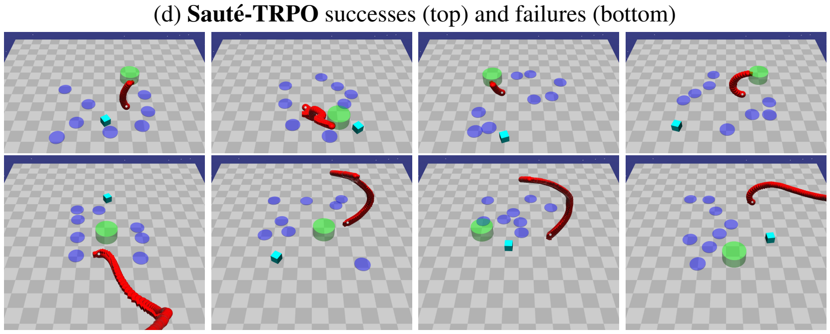
(b) **TRPO-Lagrangian** successes (top) and failures (bottom)



(c) **CPO** successes (top) and failures (bottom)



(d) **Sauté-TRPO** successes (top) and failures (bottom)



(e) **TRPO-Minmax** successes (top) and failures (bottom)

Figure 22: Sample trajectories of policies learned by each baseline and our Minmax approach in the Safety Gym POINTGOAL1 domain, in the experiments of Figure 21. Trajectories that hit hazards (the hits are highlighted by the red spheres) or take more than 1000 timesteps to reach the goal location are considered failures.

F SAFETY-GYM PILLAR TRAINING AND TESTING RESULTS WITH JI ET AL. (2024) OMNISAFE BASELINES

Noise	Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps ↓
0.0	TRPO-Minmax (Ours)	**0.00 ± 0.00**	**1.00 ± 0.00**	**3.21 ± 0.00**	**136.60 ± 12.32**
	TRPO-Lagrangian	**0.00 ± 0.00**	**1.00 ± 0.00**	**3.21 ± 0.00**	137.52 ± 14.50
	Sauté-TRPO	**0.00 ± 0.00**	0.99 ± 0.02	3.20 ± 0.03	142.88 ± 12.37
	TRPO	**0.00 ± 0.00**	**1.00 ± 0.00**	**3.21 ± 0.00**	138.92 ± 14.47
	CPO*	0.10 ± 0.29	0.18 ± 0.36	-0.51 ± 3.42	818.85 ± 289.68
	P3O	0.10 ± 0.30	0.83 ± 0.35	2.74 ± 1.01	205.50 ± 220.31
					Total Steps ↑
1.5	TRPO-Minmax (Ours)	**0.06 ± 0.02**	**0.94 ± 0.02**	**3.01 ± 0.08**	262.19 ± 28.06
	TRPO-Lagrangian	0.09 ± 0.04	0.91 ± 0.04	2.90 ± 0.12	255.55 ± 26.62
	Sauté-TRPO	0.11 ± 0.04	0.89 ± 0.04	2.81 ± 0.14	232.26 ± 10.55
	TRPO	0.13 ± 0.08	0.87 ± 0.08	2.74 ± 0.29	262.91 ± 32.70
	CPO*	0.08 ± 0.12	0.00 ± 0.00	-0.44 ± 0.45	952.51 ± 74.45
	P3O	0.11 ± 0.13	0.76 ± 0.33	2.43 ± 1.04	**391.09 ± 221.08**
2.5	TRPO-Minmax (Ours)	**0.14 ± 0.05**	**0.80 ± 0.11**	**2.61 ± 0.27**	503.49 ± 98.67
	TRPO-Lagrangian	0.20 ± 0.05	0.72 ± 0.24	2.38 ± 0.46	461.89 ± 132.78
	Sauté-TRPO	0.19 ± 0.09	0.76 ± 0.24	2.45 ± 0.54	435.18 ± 104.06
	TRPO	0.28 ± 0.10	0.63 ± 0.22	2.05 ± 0.52	446.21 ± 143.94
	CPO*	0.09 ± 0.10	0.00 ± 0.01	-0.50 ± 0.40	962.74 ± 41.12
	P3O	0.17 ± 0.07	0.71 ± 0.16	2.42 ± 0.34	**552.42 ± 139.28**

Table 2: Evaluation of trained models with Ji et al. (2024) OmniSafe baselines in the PILLAR environment with varying noise levels. For valid comparison, TRPO-Minmax here is implemented by using Algorithm 1 with OmniSafe's implementation of TRPO. For each algorithm in each noise level, we train using 10 random seeds for 10 million steps and evaluate the learned policies over 100 random seeds, for a total of 1000 evaluation episodes. We report the mean and standard errors of various performance metrics, **bolding** the ones with the best mean. Figures 23-28 shows the training curves, including other noise levels for only TRPO-Minmax, TRPO-Lagrangian, and P3O. Here, higher episode lengths are better because that means the policy is taking longer safer paths. We observe CPO in general struggles to learn to solve the tasks irrespective of noise level, even in the simplest case with $noise = 0$. We suspect this could be due to an implemention issue with Omnisafe's codebase, since Ray et al. (2019) codebase did not have this issue. Hence we exclude CPO from our analysis (denoted by a *) since its results are not consistent with those of Ray et al. (2019) and Achiam et al. (2017). All the other results are consistent with Ji et al. (2024). Given that, we observe that only TPRO-Minmax prioritises minimising the probability of unsafe transitions, consistently achieving the lowest cost while trading off the rewards. Interestingly, by using Algorithm 1 with OmniSafe's implementation of TRPO, TPRO-Minmax achieves the lowest cost, highest success rate, and highest returns across all noise levels.

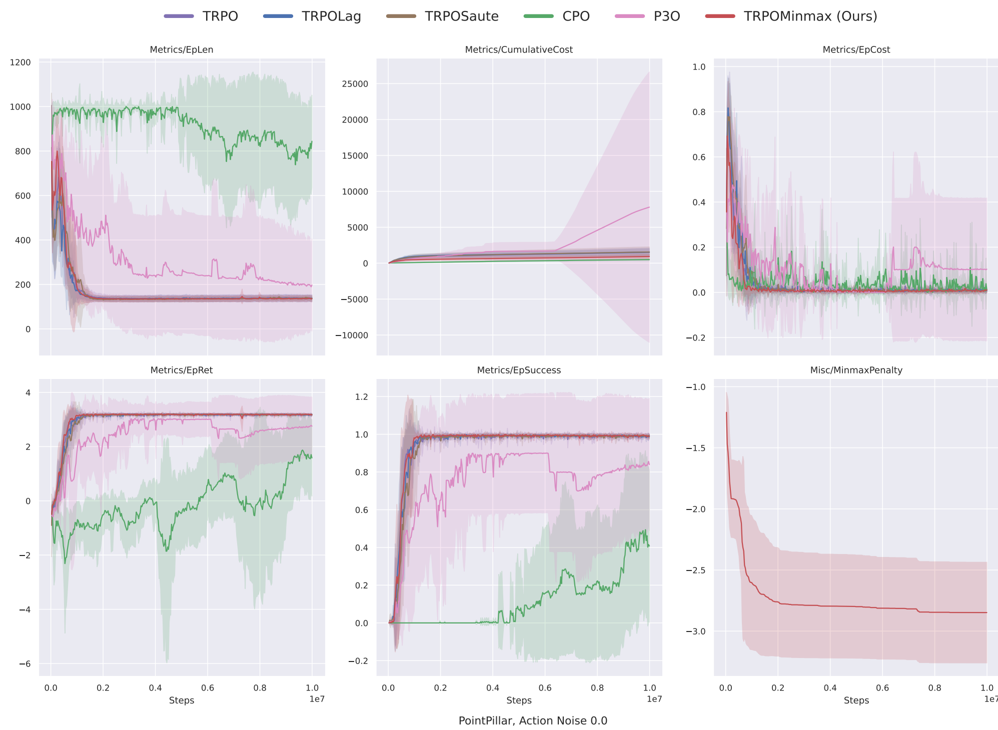


Figure 23: Training curves using OmniSafe in the PILLAR environment with $noise = 0$

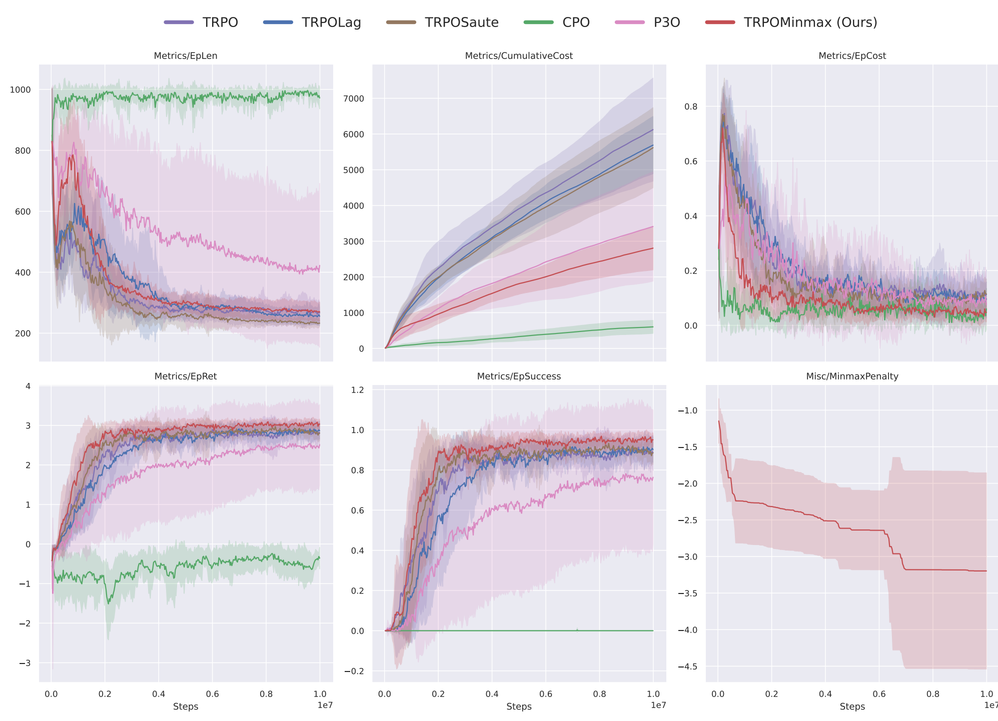


Figure 24: Training curves using OmniSafe in the PILLAR environment with $noise = 1.5$

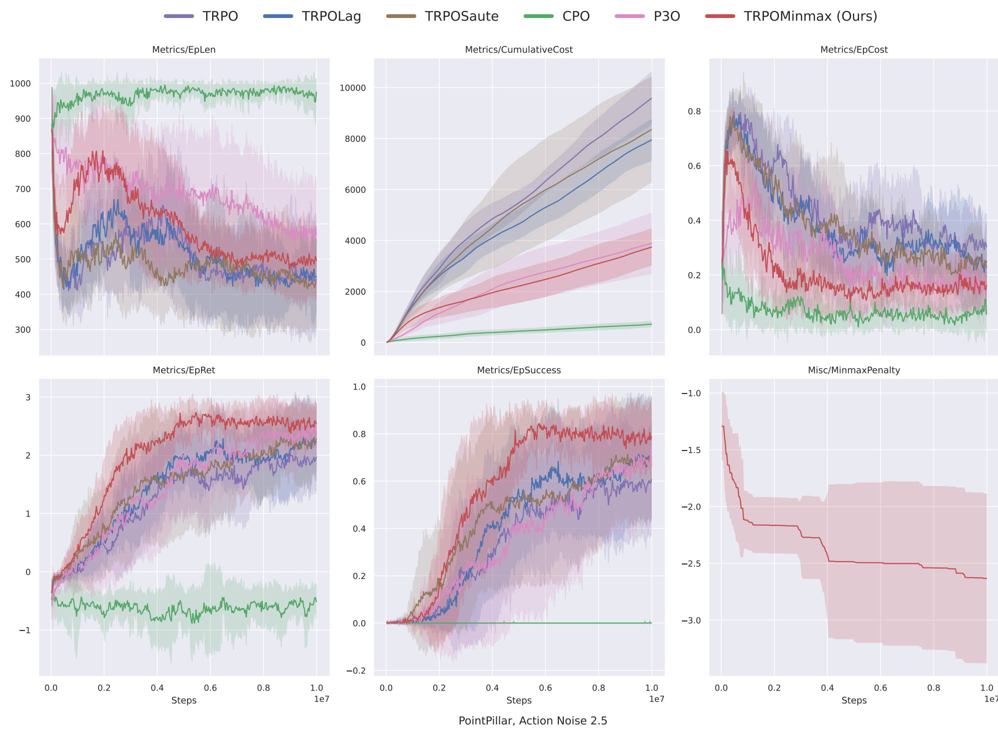


Figure 25: Training curves using OmniSafe in the PILLAR environment with $noise = 2.5$

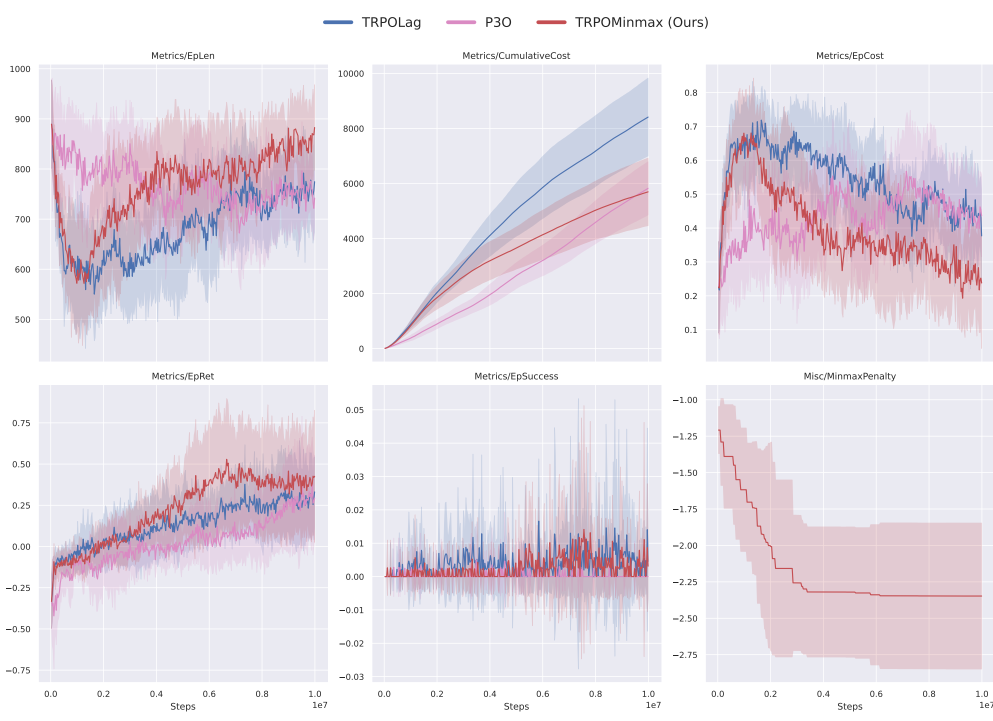


Figure 26: Training curves using OmniSafe in the PILLAR environment with $noise = 5$

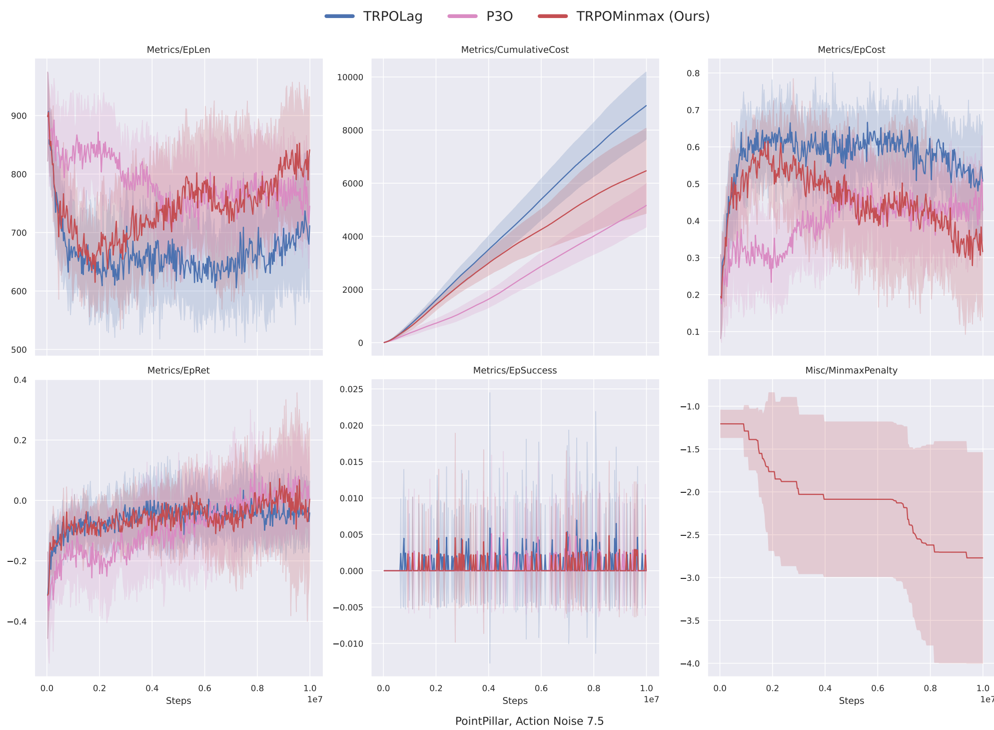


Figure 27: Training curves using OmniSafe in the PILLAR environment with $noise = 7.5$

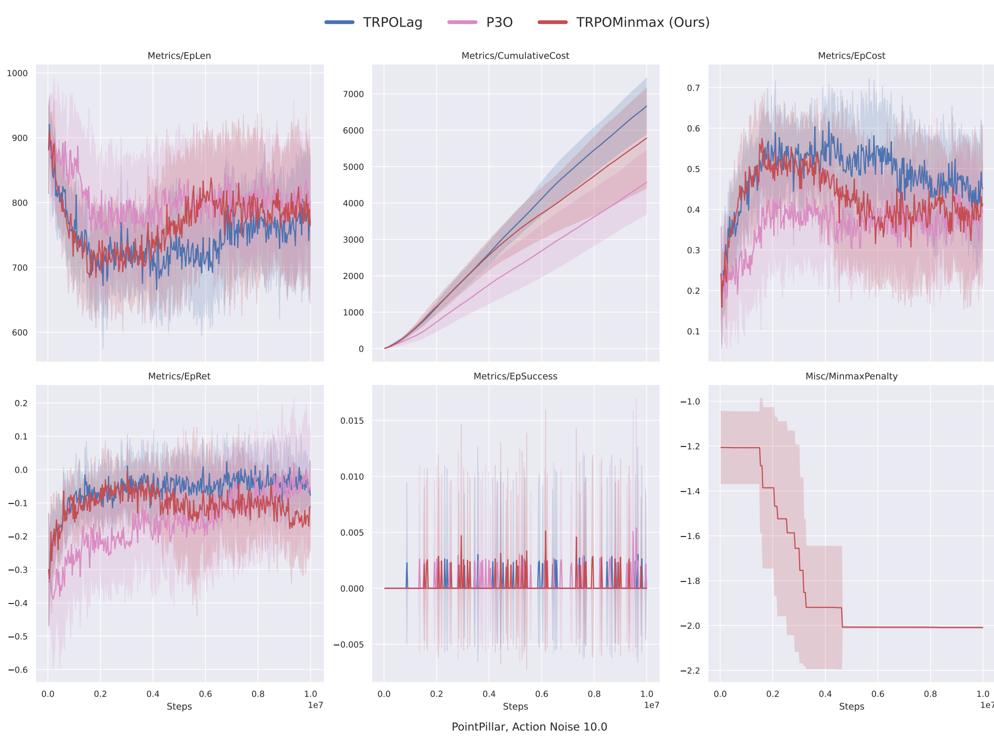


Figure 28: Training curves using OmniSafe in the PILLAR environment with $noise = 10$

G Ablations in Safety-Gymnasium default environments with Ji et al. (2024) OmniSafe baselines

Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps
TRPO-Minmax (Ours)	**0.04** $\pm$ **0.03**	0.50 $\pm$ 0.17	0.84 $\pm$ 0.45	532.08 $\pm$ 148.18
PPO-Minmax (Ours)	**0.04** $\pm$ **0.02**	0.84 $\pm$ 0.06	1.64 $\pm$ 0.18	253.69 $\pm$ 49.72
TRPO-Lagrangian	0.09 $\pm$ 0.03	0.86 $\pm$ 0.03	1.76 $\pm$ 0.08	119.18 $\pm$ 19.82
Sauté-TRPO	0.12 $\pm$ 0.03	0.87 $\pm$ 0.03	1.77 $\pm$ 0.09	77.97 $\pm$ 10.33
TRPO	0.10 $\pm$ 0.02	0.90 $\pm$ 0.02	1.84 $\pm$ 0.04	73.86 $\pm$ 4.37
CPO*	0.04 $\pm$ 0.04	0.06 $\pm$ 0.02	-0.48 $\pm$ 0.51	940.59 $\pm$ 23.63
P3O	0.08 $\pm$ 0.02	**0.91** $\pm$ **0.02**	**1.86** $\pm$ **0.06**	101.98 $\pm$ 13.03

Table 3: Evaluation of trained models with Ji et al. (2024) OmniSafe baselines in Safety-Gymnasium POINTGOAL1, modified to terminate in $\mathcal{G} = \{\;,\;\}$ where $\mathcal{G}^! = \{\;\}$. Episodes terminate when the agent reaches $\mathcal{G}$ or after 1000 timesteps, but due to the large number of hazards, shorter or longer timesteps are better depending on the random positions of hazards. Similarly to Table 2, we exclude CPO from our analysis (denoted by a *) since its results are not consistent with those of Ray et al. (2019) and Achiam et al. (2017). Given that, we observe that our approach consistently achieves the lowest cost while trading off the rewards.

Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps ↑
TRPO-Minmax (Ours)	**0.08** $\pm$ **0.05**	0.45 $\pm$ 0.16	1.87 $\pm$ 1.60	**950.31** $\pm$ **31.45**
PPO-Minmax (Ours)	0.13 $\pm$ 0.05	**0.63** $\pm$ **0.13**	4.45 $\pm$ 2.45	927.75 $\pm$ 28.89
TRPO-Lagrangian	0.62 $\pm$ 0.07	0.34 $\pm$ 0.06	10.17 $\pm$ 0.77	607.44 $\pm$ 56.15
Sauté-TRPO	0.79 $\pm$ 0.03	0.21 $\pm$ 0.03	**11.01** $\pm$ **0.55**	493.01 $\pm$ 24.51
TRPO	0.78 $\pm$ 0.05	0.22 $\pm$ 0.05	10.68 $\pm$ 0.74	483.42 $\pm$ 33.07
CPO*	0.02 $\pm$ 0.02	0.15 $\pm$ 0.06	-0.03 $\pm$ 0.23	988.25 $\pm$ 8.46
P3O	0.56 $\pm$ 0.07	0.42 $\pm$ 0.07	10.63 $\pm$ 0.74	667.08 $\pm$ 49.14

Table 4: Evaluation of trained models with Ji et al. (2024) OmniSafe baselines in Safety-Gymnasium POINTGOAL1, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \{\;\}$. Here, higher episode lengths are better because episodes terminate only when the agent reaches $\mathcal{G}^!$ or after 1000 timesteps. Similarly to Table 2, we exclude CPO from our analysis (denoted by a *) since its results are not consistent with those of Ray et al. (2019) and Achiam et al. (2017). Given that, we observe that despite the absence of terminal safe goals, our approach still prioritises minimising the probability of unsafe transitions, consistently achieving the lowest cost while trading off the rewards.

Algorithm	Costs ↓	Success Rate ↑	Returns ↑	Total Steps
TRPO-Minmax (Ours)	4.11 $\pm$ 4.34	0.10 $\pm$ 0.04	-2.21 $\pm$ 1.52	1000.00 $\pm$ 0.00
PPO-Minmax (Ours)	**3.38** $\pm$ **3.08**	0.13 $\pm$ 0.05	-3.18 $\pm$ 2.71	1000.00 $\pm$ 0.00
TRPO-Lagrangian	18.18 $\pm$ 5.03	**0.48** $\pm$ **0.05**	9.24 $\pm$ 2.21	1000.00 $\pm$ 0.00
Sauté-TRPO	4.49 $\pm$ 3.12	0.17 $\pm$ 0.12	0.03 $\pm$ 0.63	1000.00 $\pm$ 0.00
TRPO	52.90 $\pm$ 3.27	0.07 $\pm$ 0.02	**27.16** $\pm$ **0.07**	1000.00 $\pm$ 0.00
CPO*	5.26 $\pm$ 7.90	0.10 $\pm$ 0.05	-1.34 $\pm$ 0.52	1000.00 $\pm$ 0.00
P3O	30.72 $\pm$ 56.92	0.05 $\pm$ 0.03	-1.18 $\pm$ 0.79	1000.00 $\pm$ 0.00

Table 5: Evaluation of trained models with Ji et al. (2024) OmniSafe baselines the Safety-Gymnasium POINTGOAL1, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \emptyset$. Here, every episode terminates only after 1000 timesteps. Similarly to Table 2, we exclude CPO from our analysis (denoted by a *) since its results are not consistent with those of Ray et al. (2019) and Achiam et al. (2017). Given that, we observe that despite no termination in the environment, our approach still achieves the lowest cost.

Algorithm	**Costs** $\downarrow$	**Success Rate** $\downarrow$	**Returns** $\uparrow$	**Total Steps**
TRPO-Minmax (Ours)	**0.08** $\pm$ **0.03**	0.01 $\pm$ 0.01	0.47 $\pm$ 0.11	940.58 $\pm$ 20.33
PPO-Minmax (Ours)	0.12 $\pm$ 0.07	**0.09** $\pm$ **0.14**	**1.12** $\pm$ **1.30**	927.77 $\pm$ 31.18
TRPO-Lagrangian	0.12 $\pm$ 0.05	0.03 $\pm$ 0.03	0.62 $\pm$ 0.21	914.53 $\pm$ 29.49
Sauté-TRPO	0.13 $\pm$ 0.05	0.08 $\pm$ 0.14	0.92 $\pm$ 0.73	905.51 $\pm$ 33.87
TRPO	0.14 $\pm$ 0.06	0.05 $\pm$ 0.06	0.72 $\pm$ 0.37	903.23 $\pm$ 37.82
CPO*	0.02 $\pm$ 0.02	0.01 $\pm$ 0.01	0.11 $\pm$ 0.12	989.71 $\pm$ 8.27
P3O	0.13 $\pm$ 0.04	0.06 $\pm$ 0.05	0.76 $\pm$ 0.33	921.17 $\pm$ 26.78

Table 6: Evaluation of trained models with Ji et al. (2024) OmniSafe baselines in Safety-Gymnasium POINTPUSH1, modified to terminate in $\mathcal{G} = \{\ ,\ ,\ \}$ where $\mathcal{G}^! = \{\ ,\ \}$. Episodes terminate when the agent reaches $\mathcal{G}$ or after 1000 timesteps, but due to the large object the agent needs to push to the goal while avoiding both hazards and the pillar, shorter or longer timesteps are better depending on the random positions of the hazards and pillar. Similarly to Table 2, we exclude CPO from our analysis (denoted by a *) since its results are not consistent with those of Ray et al. (2019) and Achiam et al. (2017). Given that, we observe that our approach consistently achieves the lowest cost while obtaining the highest success rate and rewards.

Algorithm	**Costs** $\downarrow$	**Success Rate** $\uparrow$	**Returns** $\uparrow$	**Total Steps** $\uparrow$
TRPO-Minmax (Ours)	**0.09** $\pm$ **0.03**	0.05 $\pm$ 0.04	0.53 $\pm$ 0.15	**905.26** $\pm$ **20.12**
PPO-Minmax (Ours)	0.10 $\pm$ 0.02	0.03 $\pm$ 0.02	0.52 $\pm$ 0.07	914.64 $\pm$ 16.92
TRPO-Lagrangian	0.11 $\pm$ 0.03	0.13 $\pm$ 0.18	0.83 $\pm$ 0.49	844.21 $\pm$ 110.39
Sauté-TRPO	0.12 $\pm$ 0.05	0.10 $\pm$ 0.12	0.68 $\pm$ 0.30	838.98 $\pm$ 106.42
TRPO	0.15 $\pm$ 0.07	**0.16** $\pm$ **0.21**	**0.86** $\pm$ **0.56**	795.70 $\pm$ 157.18
CPO*	0.02 $\pm$ 0.01	0.01 $\pm$ 0.01	0.16 $\pm$ 0.25	983.25 $\pm$ 11.54
P3O	0.13 $\pm$ 0.06	0.11 $\pm$ 0.12	0.78 $\pm$ 0.36	859.75 $\pm$ 74.76

Table 7: Evaluation of trained models with Ji et al. (2024) OmniSafe baselines in Safety-Gymnasium POINTPUSH1, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \{\ ,\ \}$. Here, higher episode lengths are better because episodes terminate only when the agent reaches $\mathcal{G}^!$ or after 1000 timesteps. Similarly to Table 2, we exclude CPO from our analysis (denoted by a *) since its results are not consistent with those of Ray et al. (2019) and Achiam et al. (2017). Given that, we observe that despite the absence of terminal safe goals, our approach still prioritises minimising the probability of unsafe transitions, consistently achieving the lowest cost while trading off the rewards.

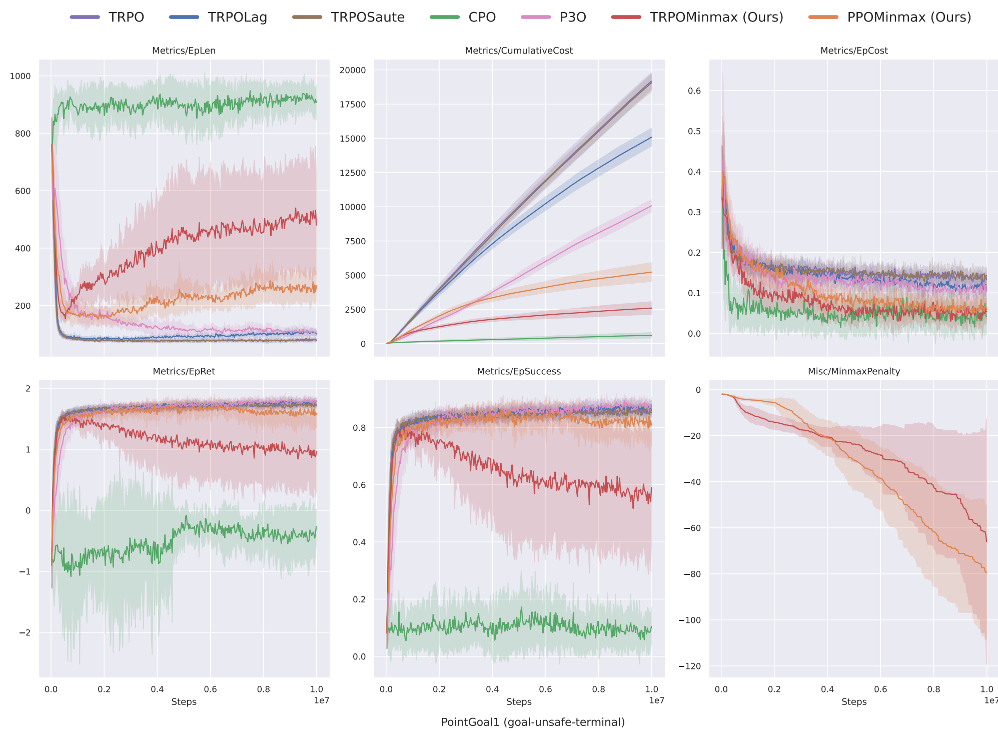


Figure 29: Training curves of models trained with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium POINTGOAL1 environment, modified to terminate in $\mathcal{G} = \{$, $\}$ where $\mathcal{G}^! = \{$ $\}$.

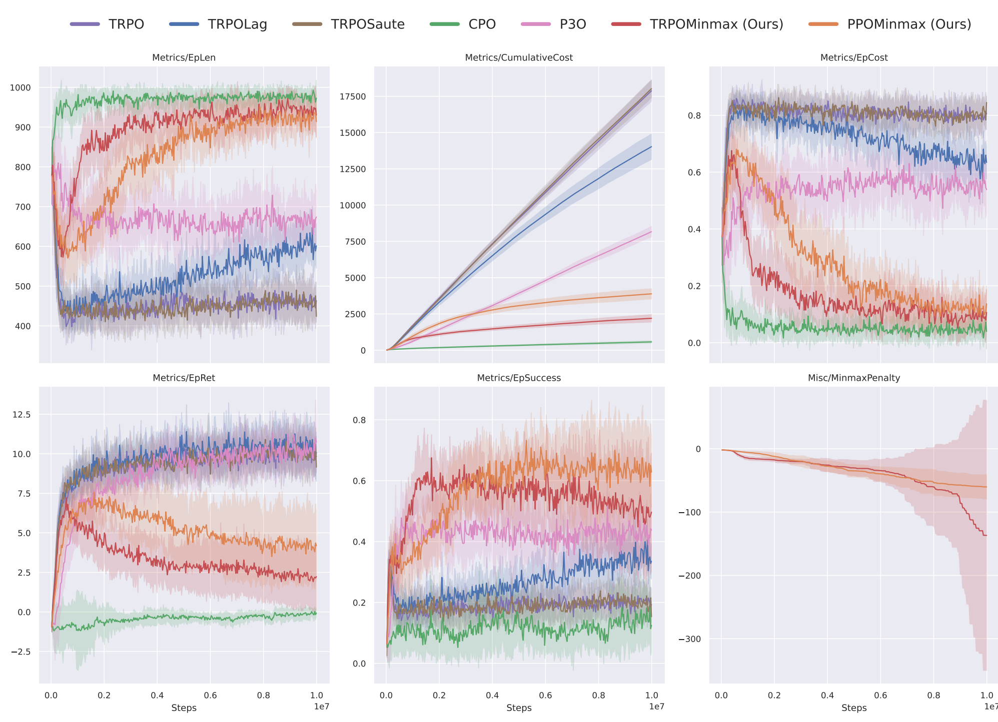


Figure 30: Training curves of models trained with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium POINTGOAL1 environment, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \{$ $\}$.

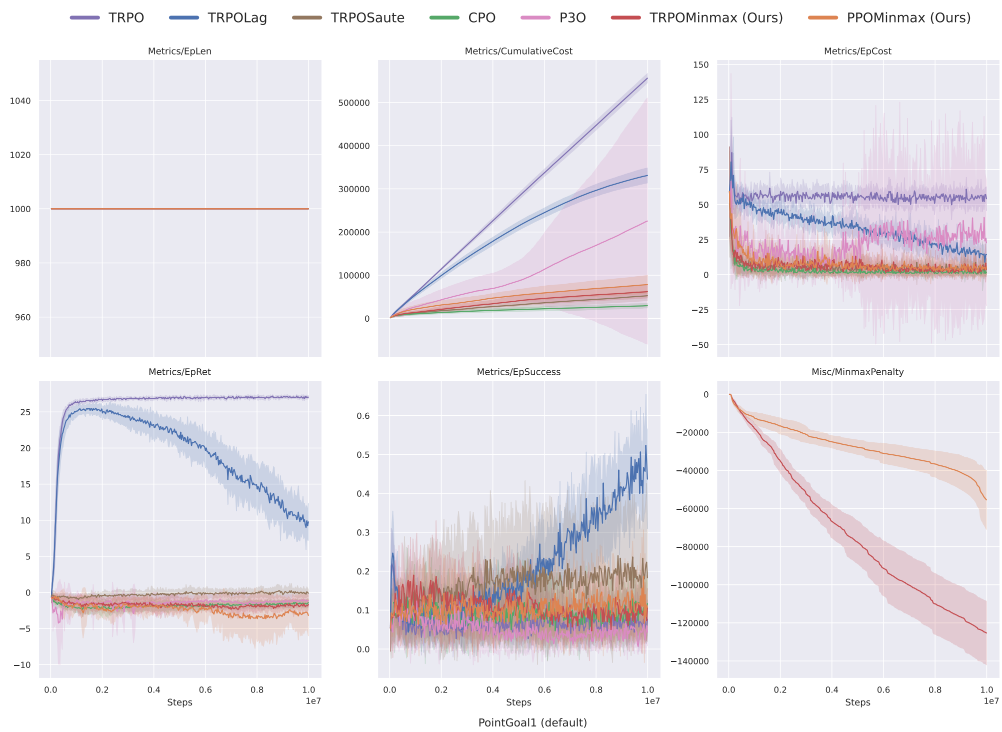


Figure 31: Training curves for trained models with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium PointGoal1 environment, modified to terminate in $\mathcal{G} = \mathcal{G}^! = \emptyset$.

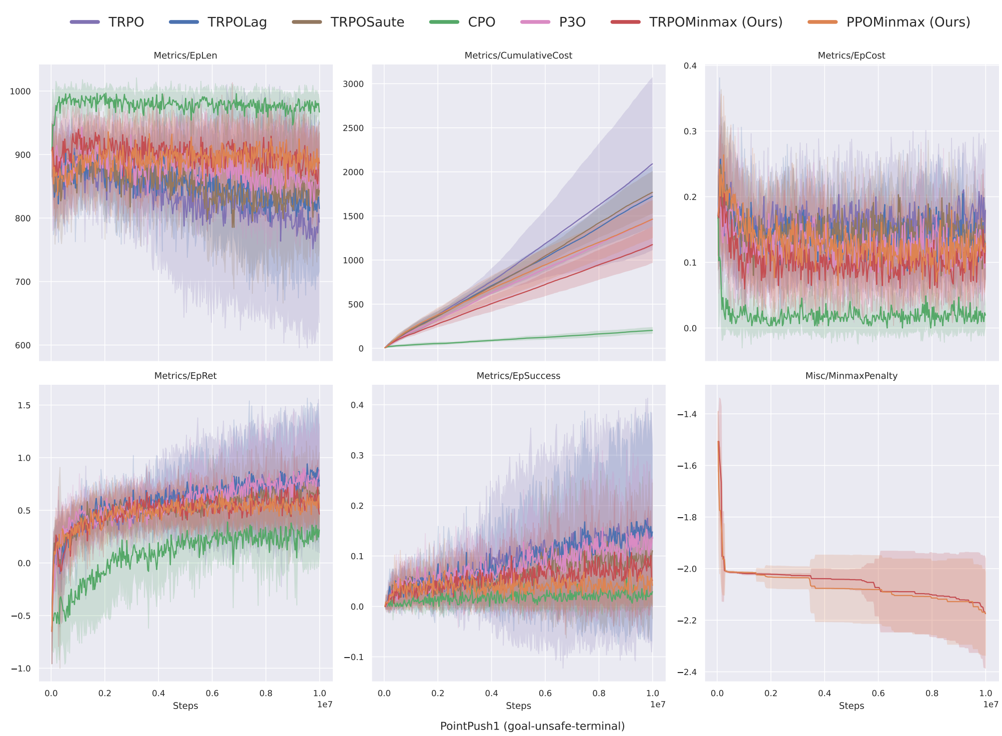


Figure 32: Training curves of models trained with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium PointPush1 environment, modified to terminate in $\mathcal{G} = \{ , , \}$ where $\mathcal{G}^! = \{ , \}$.

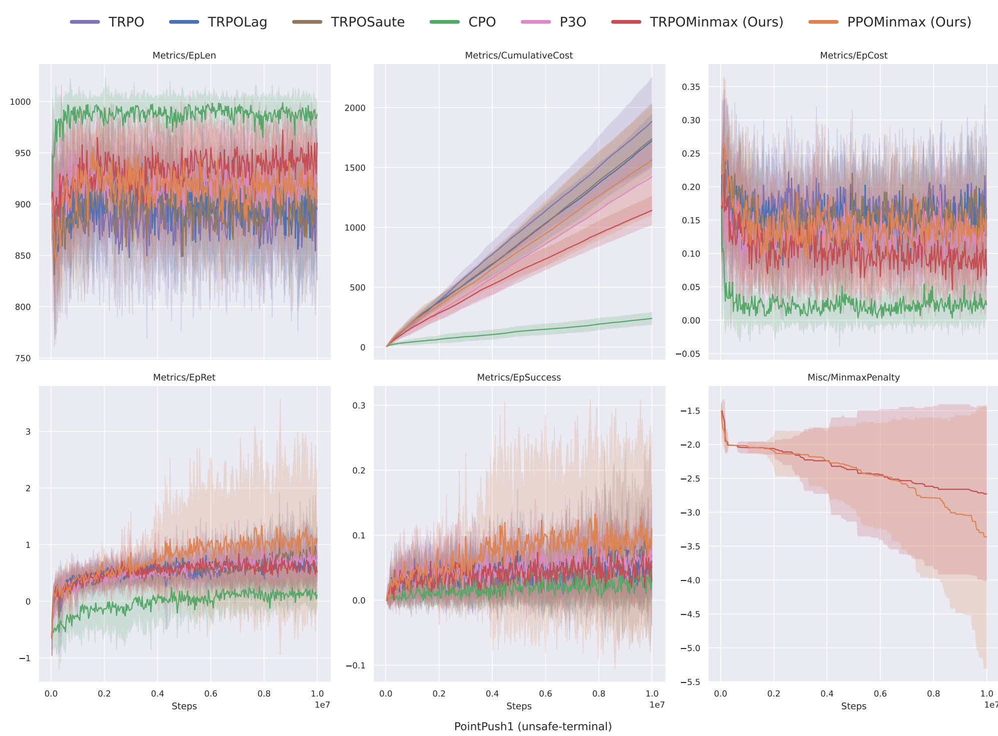


Figure 33: Training curves of models trained with Ji et al. (2024) OmniSafe baselines in the Safety-Gymnasium POINTPUSH1 environment, modified to terminate in $\mathcal{G} = \mathcal{G}^{!} = \{,\}$.