ROSARL: Reward-Only Safe Reinforcement Learning

Geraud Nangue Tasse[1], **Tamlin Love**[1,2], **Mark Nemecek**[3], **Steven James**[1] **& Benjamin Rosman**[1]
[1] School of Computer Science and Applied Mathematics, University of the Witwatersrand
[2] Institut de Robòtica i Informática Industrial, Universidad Politécnica de Cataluna
[3] Department of Computer Science, Duke University
tlove@iri.upc.edu, mark.nemecek@duke.edu
{geraud.nanguetasse1, steven.james, benjamin.rosman1}@wits.ac.za

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 *controllability*. 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,

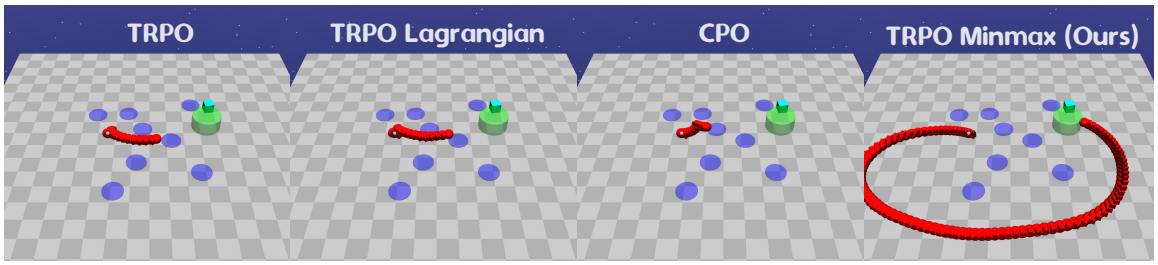


Figure 1: Example trajectories of prior work—TRPO (Schulman et al., 2015) (left-most), TRPO-Lagrangian (Ray et al., 2019) (middle-left), CPO (Achiam et al., 2017) (middle-right)—compared to ours (right-most) in the Safety Gym domain (Ray et al., 2019). For each, a point mass agent learns to reach a goal 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).

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.

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. See Appendix C for further discussion of related works.

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 *controllability* of an environment, and a practical estimate of it can be learned by an agent using its current value estimates. We make the following main contributions:

(i) **Bounding the Minmax penalty (Section 3.3)**: We obtain the analytical form of an upper and lower bound on the Minmax penalty and prove that using the upper bound results in learned behaviours that minimise the probability of visiting unsafe states (Theorem 2); We also show that these bounds can be accurately estimated using policy evaluation (Sutton & Barto, 2018) (Theorem 1).

(ii) **Learning safe policies (Section 4)**: We show that accurately estimating the Minmax penalty or bounds is NP-hard (Theorem 3). Hence, we propose a simple model-free algorithm for learning a practical estimate of the Minmax penalty while learning the task policy. Since the approach only modifies the rewards for unsafe transitions with the estimated penalty (keeping the rewards for other transitions unchanged), it can be integrated into any RL pipeline that learns value functions.

(iii) **Experiments (Section 5)**: Finally, we investigate the behaviour of agents that only rely on their learned Minmax penalty to solve tasks safely. Our results demonstrate that these reward-only agents are capable of learning to solve tasks while avoiding unsafe states. Additionally, while prior methods often violate safety constraints, we observe that reward-only agents consistently learn safer policies.

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 $\mathcal{G} \subset \mathcal{S}$ while avoiding unsafe absorbing states $\mathcal{G}^! \subset \mathcal{G}$. 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.

A *policy* $\pi : \mathcal{S} \to \mathcal{A}$ is a mapping from states to actions. The *value function* $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} R(s_t, a_t, s_{t+1})]$ associated with a policy specifies the expected return under that policy starting from state s. The goal of an agent is to learn an optimal policy π^* that maximises the value function $V^{\pi^*}(s) = V^*(s) = \max_\pi V^\pi(s)$ 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.

3 Avoiding Unsafe Absorbing States

Given an environment, we aim to bound the smallest penalty (hence the largest reward) to use as feedback for unsafe transitions to guarantee safe optimal policies. We formally define a safe policy as a proper policy that minimises the probability of reaching any unsafe terminal states:

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

Remark 1 *Since proper policies reach $\mathcal{G}$, Definition 1 equivalently says that safe policies are those that maximise the probability of reaching safe goal states $\mathcal{G} \setminus \mathcal{G}^!$. Since optimal policies are also proper, this means that safe optimal policies also maximise the probability of reaching $\mathcal{G} \setminus \mathcal{G}^!$. For example, looping forever in a non-absorbing region of the state space is neither proper, nor safe, nor optimal.*

We now define the Minmax penalty R_{Minmax} as the largest reward for unsafe transitions that lead to safe optimal policies:

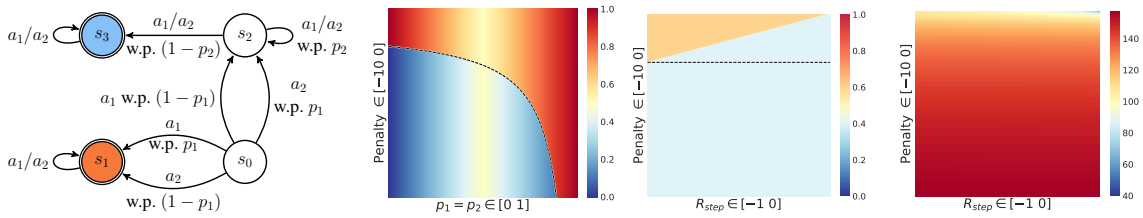
Definition 2 *Consider an environment $\langle \mathcal{S}, \mathcal{A}, P \rangle$ where task rewards $R(s, a, s')$ are bounded by $[R_{MIN}\ R_{MAX}]$ for all $s' \notin \mathcal{G}^!$. Let π^* be an optimal policy for one such task $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$. We define the Minmax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

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.



(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 penalties 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.

Since the transitions per action are 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. In fact, for $p_1 = p_2 = 0$ and -1 penalty for transitions into s_1, the optimal policy 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 is to always pick action a_1, which always reaches s_3. However, for $p_1 = p_2 = 0.4$ (Figure 2c), a higher penalty is required for a_1 to stay optimal. To capture this relationship between the stochasticity of an environment and the required penalty to obtain safe policies, we introduce a notion of *controllability*, which measures the ability of an agent to reach safe goals. Additionally, observe that as p_2 increases, the probability that the agent can 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 Controllability 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* $D \coloneqq \max_{s \in \mathcal{S} \setminus \mathcal{G}} \max_{\pi \in \Pi} \mathbb{E}\left[T(s_T \in \mathcal{G} | \pi)\right]$, *where* $T(s_T \in \mathcal{G}|\pi)$ *is the timesteps taken to reach* $\mathcal{G}$ *from* s *when following a proper policy* π.

Given the diameter of an environment, 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 $D' \coloneqq \max_{s \in \mathcal{S} \setminus \mathcal{G}} \max_{\pi \in \Pi} \mathbb{E}\left[T(s_T \in \mathcal{G} \setminus \mathcal{G}^! | \pi)\right]$. However, while $\bar{R}_{\text{MAX}}$ aims to make reaching unsafe states worse than reaching safe goals, it does not consider the controllability of an environment, nor the possibility that an unsafe policy receives R_{MAX} everywhere in its trajectory. We can formally define the controllability of an environment as follows:

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

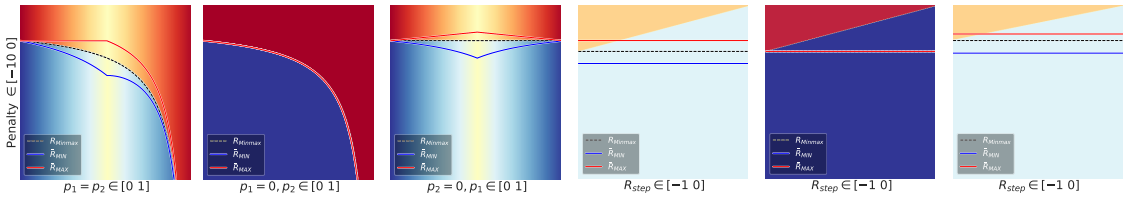
C measures the degree of controllability of the environment by simply taking the smallest non-zero probability of reaching safe goal states by following a proper policy. For example, if the dynamics are deterministic, then any deterministic policy π will either reach a safe goal or not. That is, $P_s^\pi(s_T \notin \mathcal{G}^!)$ will either be 0 or 1. Since we require $P_s^\pi(s_T \notin \mathcal{G}^!) \neq 0$, 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 $P_{s_1}^{\pi_1}(s_T \notin \mathcal{G}^!) = (1 - p_1)\mathbb{1}(p_2 = 1)$ and $P_{s_1}^{\pi_2}(s_T \notin \mathcal{G}^!) = p_1 \mathbb{1}(p_2 = 1)$. 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 controllable when $p = 1$ since s_3 is unreachable from s_2.

Remark 2 *We can think of* $C = 0$ *as the limit of* C *when safe goals are unreachable.*

Given the diameter and controllability of an environment, we can now define a choice for the Minmax penalty that takes into account both D, C, and R_{MAX}: $\bar{R}_{\text{MIN}} \coloneqq (R_{\text{MIN}} - R_{\text{MAX}})\frac{D}{C}$. 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 controllability, from which the penalties are then obtained. We describe the method here and present the pseudo-code in **Algorithm**



(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 controllability and diameter for the bounds are estimated using Algorithm 1.

1 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 controllability 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 rewards otherwise. This approach converges via the convergence of policy evaluation (**Theorem 1**).

Theorem 1 (Estimation) *Algorithm 1 converges to D and C for any given controllable 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 safe optimal 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 controllable 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 safe optimal 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 controllability 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 1), 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. We describe the method here and present the pseudo-code in **Algorithm 2** in Appendix B. This algorithm requires initial estimates of R_{MIN} and R_{MAX}, which in this work are initialised to 0. The agent 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. Since the controllability C is also expensive to estimate, it is not explicitly considered in this estimate of $\bar{R}_{\text{MIN}}$. 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 r_t is 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 $\bar{R}_{\text{MIN}}$.

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 Appendix D for additional experiments): How does Algorithm 2 (i) behave when the theoretical assumptions are satisfied? (ii) behave when the theoretical assumptions are *not* satisfied? (iii) compare to prior approaches towards Safe RL? For each result, we report the mean (solid line) and one standard deviation around it (shaded region).

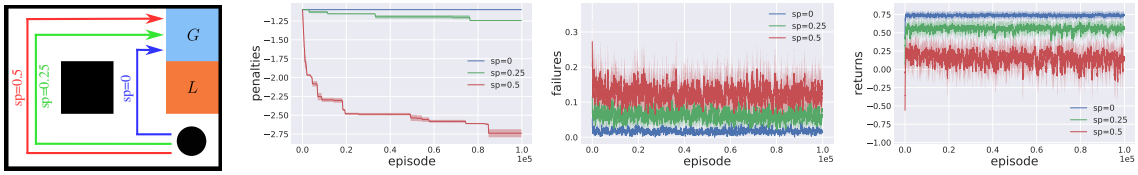
5.1 How does Algorithm 2 behave when the theoretical assumptions are satisfied?

Domain (LAVA GRIDWORLD) This is a simple gridworld environment 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 $\alpha = 0.1$. The experiments are run over 10,000 episodes and averaged over 70 runs.

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 1 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. Given the starting position of the agent next to the lava, the failure rate inevitably increases with increased stochasticity. The resulting optimal policy then chooses the longer path that passes to the left of the centre wall ($sp = 0.25$ 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 2, this algorithm can still learn optimal safe policies when the theoretical setting holds.

5.2 How does Algorithm 2 behave when the theoretical assumptions are not satisfied?

Domain (Safety Gym PILLAR) This is a custom Safety Gym environment (Ray et al., 2019), in which the simple point robot must navigate to a goal location around a large pillar (hence $\mathcal{G} = \{$, $\}$ and

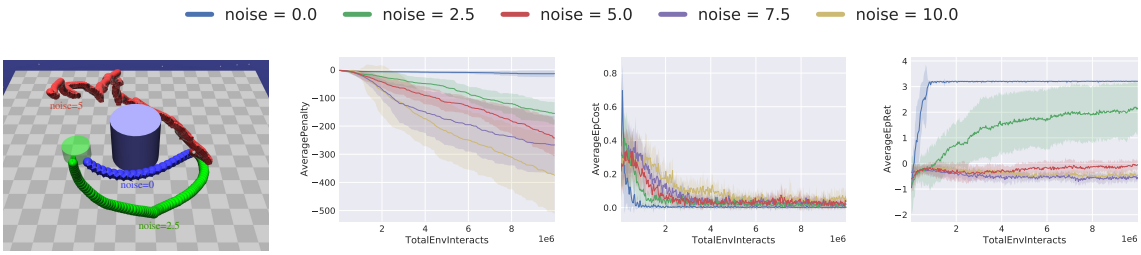


(a) Trajectories (b) Learned penalty (c) Failure rate (d) Average returns

Figure 4: Effect of increase in the slip probability of the LAVA GRIDWORLD on the learned Minmax penalty and corresponding failure rate and returns. The black circle in (a) represents the agent.

$\mathcal{G}^! = \{$$\}$). Just as in Ray et al. (2019), the agent uses *pseudo-lidar* to observe the distance to objects around it ($|\mathcal{S}| = \mathbb{R}^{60}$), and the action space is continuous over two actuators controlling the direction and forward velocity ($|\mathcal{A}| = \mathbb{R}^2$). The goal, pillar, and agent locations remain unchanged for all episodes. The discount factor is $\gamma = 0.99$, and the agent is rewarded for reaching the goal (with a reward of 1) as well as for moving towards it (the default dense distance-based reward). 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. This domain does not satisfy the shortest path setting we assume since: it is discounted, optimal policies are not guaranteed to reach $\mathcal{G}$ and policies that do not reach $\mathcal{G}$ are not guaranteed to have value functions that are unbounded from below (due to the dense rewards). 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 2. The experiments are run over 10 million steps and averaged over 10 runs.

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. Here, the value of the noise denotes the number by which a random action vector is scaled before vector addition with the agent's action. 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 2 can learn safe policies even in discounted high-dimensional continuous-control domains requiring function approximation.



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

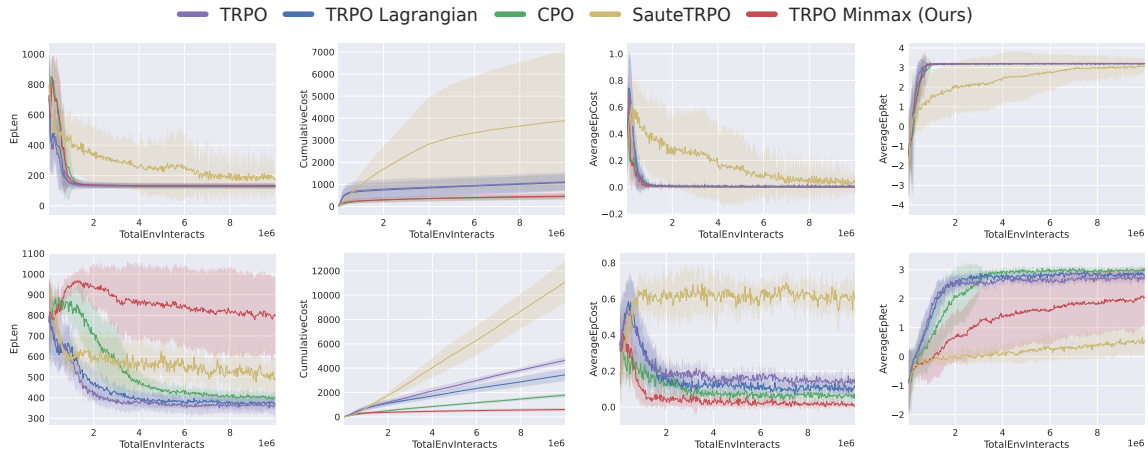
Figure 5: Performance of TRPO-Minmax in the PILLAR environment with varying noise levels.

5.3 How does Algorithm 2 compare to prior approaches towards Safe RL?

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, in the stochastic case $noise = 2.5$, we can observe that all the baselines except Sauté-TRPO achieve significantly high returns (Figure 6d **bottom**) at the expense of a rapidly increasing cumulative cost (Figure 6b **bottom**). 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 one ($noise = 2.5$). Finally, by examining the episode length (Figure 6a) and failure rates (Figure 6c) for all the baselines in the stochastic case, we can conclude that they have all learned risky policies that maximise rewards over short trajectories that are highly likely to result in collisions. We also provide additional results in the appendix for $noise \geq 5$ (Figures 9-11) to further demonstrate this point. In contrast, the results obtained show that TRPO-Minmax successfully maximises returns while minimising cost for both deterministic and stochastic environments. In addition, when the noise level is too high $noise \geq 5$, TRPO-Minmax consistently prioritises maintaining low failure rates over maximising returns.



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

Figure 6: Comparison with baselines in the PILLAR environment. **(top)** noise $= 0$, **(bottom)** noise $= 2.5$.

6 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 controllability 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. 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.

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.

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.

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.

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 1 converges to D and C for any given controllable environment.*

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

Theorem 2 (Safety Bounds) *Consider a controllable 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.
\end{aligned}$$

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

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

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

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 1: Estimating the Diameter and Controllability

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 $D = 0$, Controllability $C = 1$, 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
```

Algorithm 2: RL while learning Minmax penalty

Input : RL algorithm $\mathbf{A}$, max timesteps T

Initialise : $R_{\text{MIN}} = 0, R_{\text{MAX}} = 0, V_{\text{MIN}} = R_{\text{MIN}}, V_{\text{MAX}} = R_{\text{MAX}}$, π and V as per $\mathbf{A}$

```
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
```

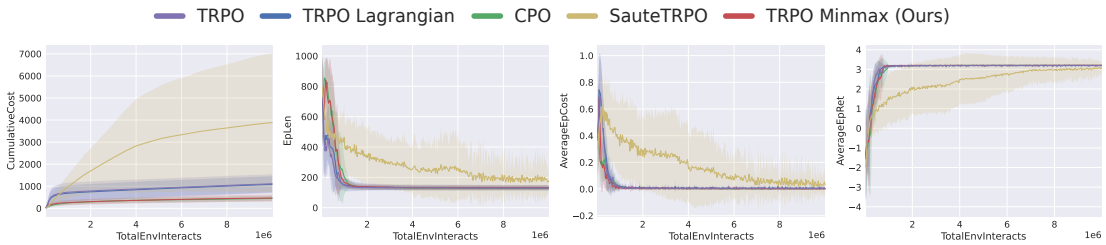
C Related Work

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.

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, 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.

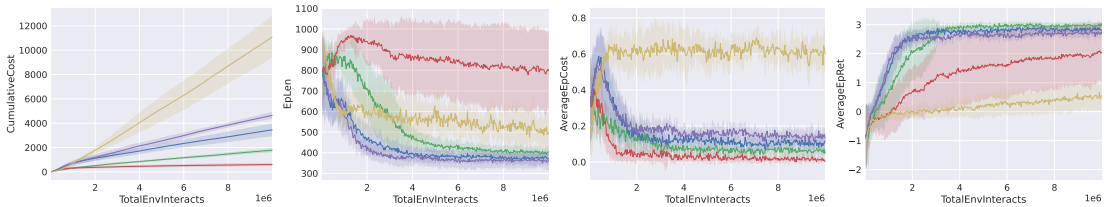
Shielding: Another important line of work involves relying on interventions from a model (Dalal et al., 2018; Wagener et al., 2021) or human (Tennenholtz et al., 2022) to prevent unsafe actions from being considered by the agent (shielding the agent) or prevent the environment from executing those unsafe 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 Additional Experiments and Figures



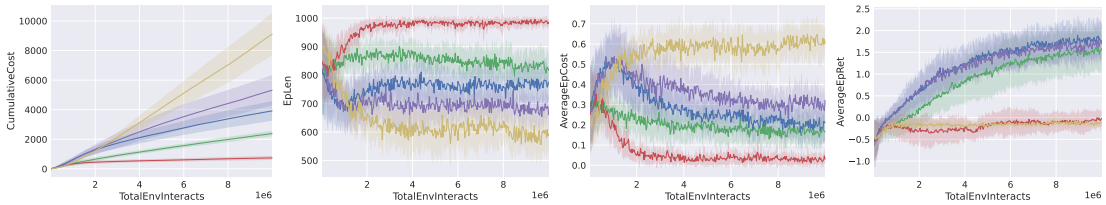
(a) Cumulative cost (b) Episode length (c) Failure rate (d) Average returns

Figure 7: Performance comparison with baselines in the PILLAR environment with noise $= 0$.



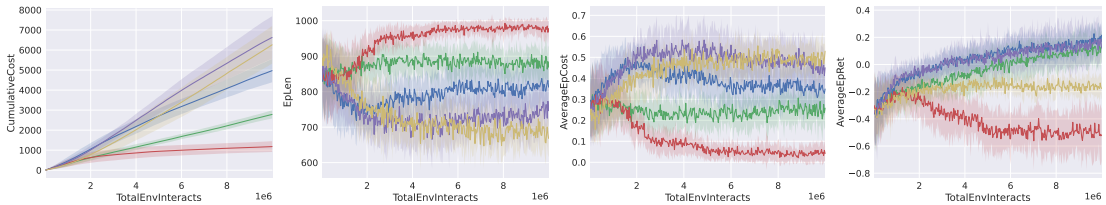
(a) Cumulative cost (b) Episode length (c) Failure rate (d) Average returns

Figure 8: Performance comparison with baselines in the PILLAR environment with noise $= 2.5$.



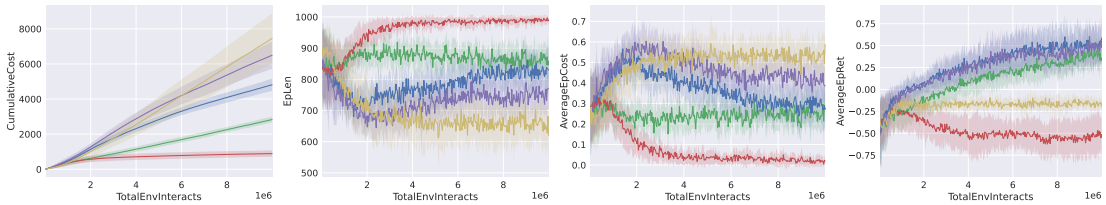
(a) Cumulative cost (b) Episode length (c) Failure rate (d) Average returns

Figure 9: Performance comparison with baselines in the PILLAR environment with noise $= 5$.



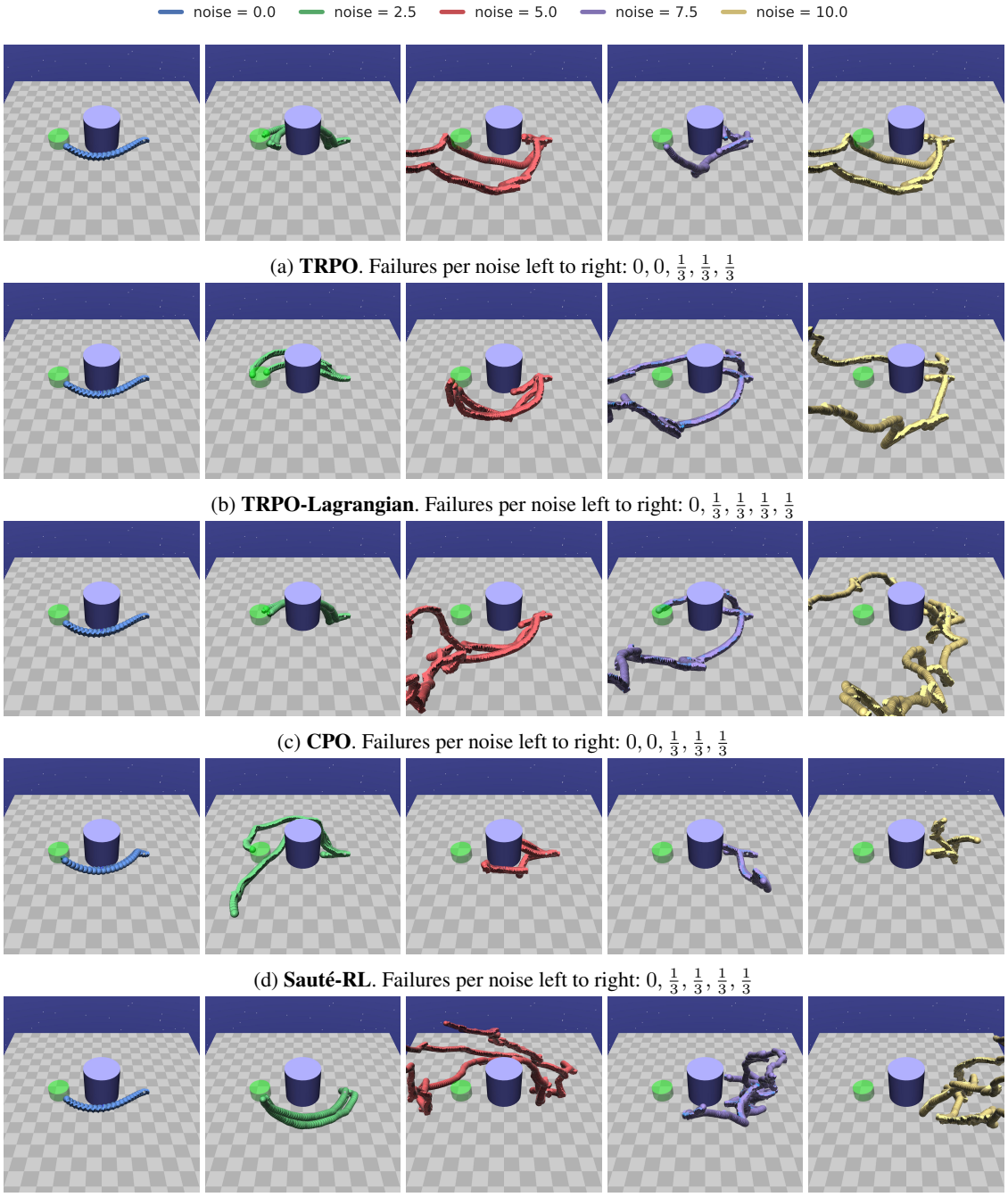
(a) Cumulative cost (b) Episode length (c) Failure rate (d) Average returns

Figure 10: Performance comparison with baselines in the PILLAR environment with noise $= 7.5$.



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

Figure 11: Performance comparison with baselines in the PILLAR environment with noise $= 10$.



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

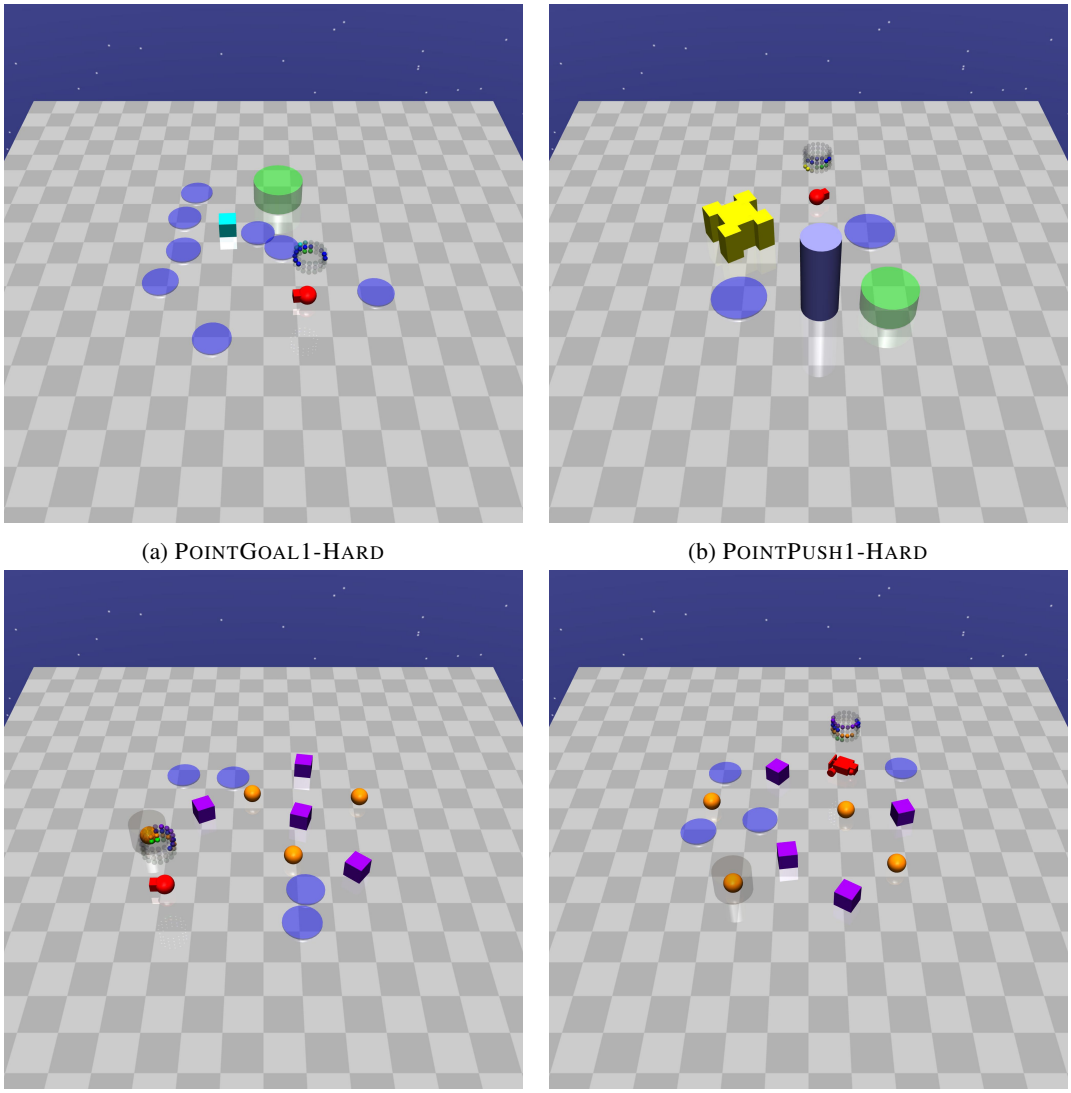
(b) **TRPO-Lagrangian**. Failures per noise left to right: $0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}$

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

(d) **Sauté-RL**. 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: $0, 0, 0, \frac{1}{3}, 0$

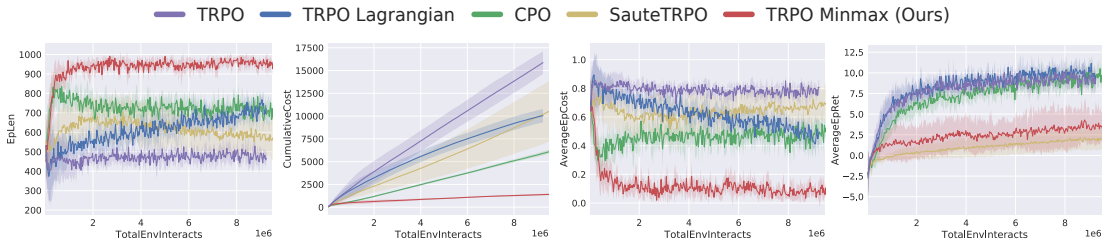
Figure 12: Sample trajectories of policies learned by each baseline and our **TRPO-Minmax** approach in the Safety Gym PILLAR environment with varying noise levels. To sample the trajectories for each noise level, we use the same three environment random seeds across all the algorithms.



(a) POINTGOAL1-HARD (b) POINTPUSH1-HARD

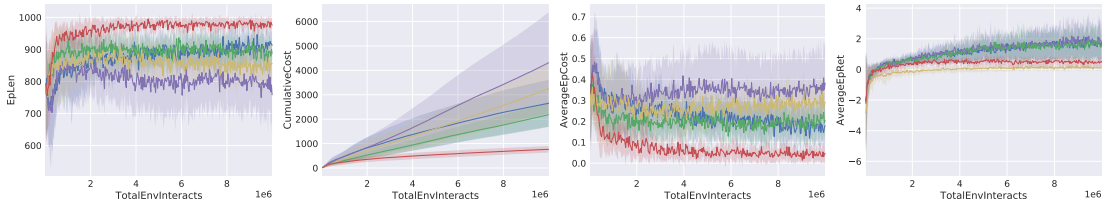
(c) POINTBUTTON1-HARD (d) CARBUTTON1-HARD

Figure 13: Additional Safety-Gym domains. (a) is a modified version of the POINTGOAL1 task from OpenAI's Safety Gym environments (Ray et al., 2019), which represents complex, high-dimensional, continuous control tasks. In all of the original domains, $\mathcal{G} = \emptyset$ by default. We only modify POINTGOAL1 to make unsafe transitions terminal $\mathcal{G} = \mathcal{G}^! = \{\text{states with cost} > 0\}$, leaving the safe goal states non-terminal ($\mathcal{G} \setminus \mathcal{G}^! = \emptyset$). Here, a simple robot must navigate to a goal location across a 2D plane while avoiding several hazards (where $\mathcal{G} = \mathcal{G}^! = \{\}$). The agent's sensors, actions, and rewards are identical to the PILLAR domain. Unlike the PILLAR domain, the goal's location is randomly reset when the agent reaches it, but does not terminate the episode. (b-d) are modified similarly to the POINTGOAL1-HARD environment. POINTPUSH1-HARD is similar to POINTGOAL1-HARD, 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 (where $\mathcal{G} = \mathcal{G}^! = \{,\}$). Finally, POINTBUTTON1-HARD and CARBUTTON1-HARD are also similar to POINTGOAL1-HARD, but with the more complex car robot for CARBUTTON1-HARD and the addition of these to both: (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 (where $\mathcal{G} = \mathcal{G}^! = \{,,\}$).



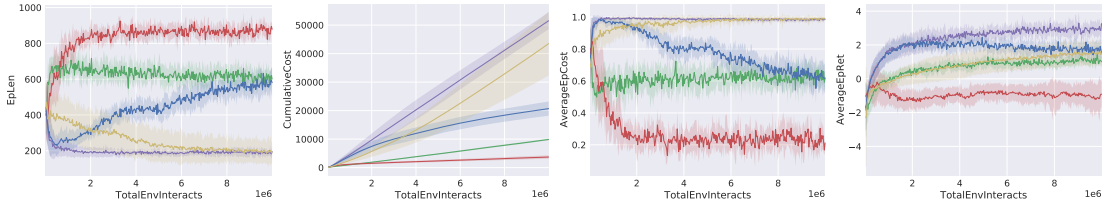
(a) Episode length. (b) Cumulative cost. (c) Failure rate (d) Average returns

Figure 14: Performance in POINTGOAL1-HARD (where $\mathcal{G} = \mathcal{G}^! = \{$●$\}$). Here, higher episode lengths are better (in addition to higher returns) since episodes only terminate when the agent reaches a hazard 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 paths to the goals but the dense reward function incentivises moving towards the goal despite the large number of hazards in-between.



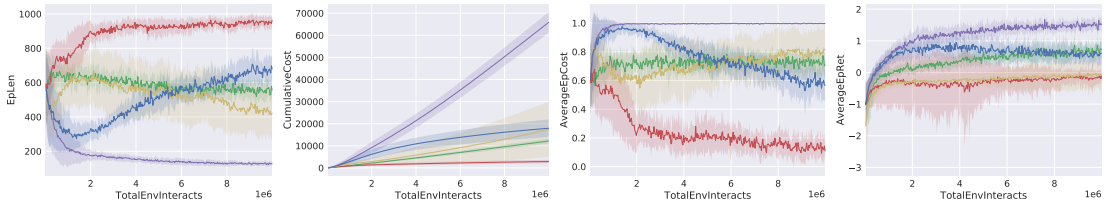
(a) Episode length. (b) Cumulative cost. (c) Failure rate (d) Average returns

Figure 15: Performance in POINTPUSH1-HARD (where $\mathcal{G} = \mathcal{G}^! = \{$●, ◒$\}$). Here, higher episode lengths are better (in addition to higher returns) since episodes only terminate when the agent reaches a hazard 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.



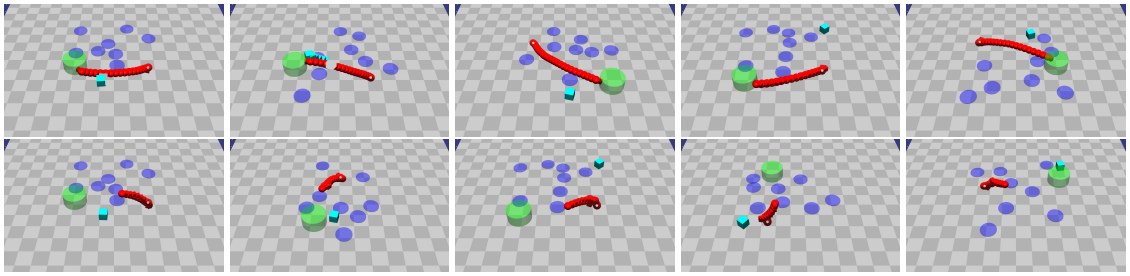
(a) Episode length. (b) Cumulative cost. (c) Failure rate (d) Average returns

Figure 16: Performance in POINTBUTTON1-HARD (where $\mathcal{G} = \mathcal{G}^! = \{$●, ■, ●$\}$). Here, higher episode lengths are better (in addition to higher returns) since episodes only terminate when the agent reaches a hazard 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.

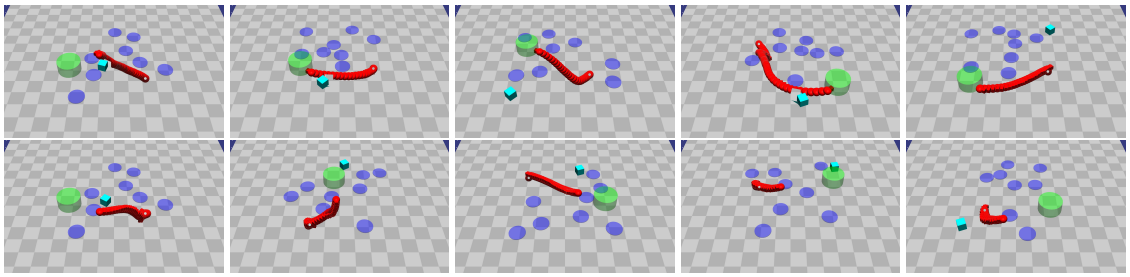


(a) Episode length. (b) Cumulative cost. (c) Failure rate (d) Average returns

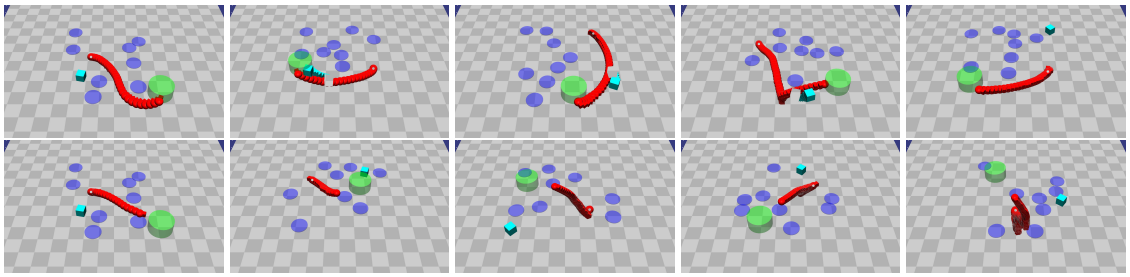
Figure 17: Performance in CARBUTTON1-HARD (where $\mathcal{G} = \mathcal{G}^! = \{$●, ■, ●$\}$). Here, higher episode lengths are better (in addition to higher returns) since episodes only terminate when the agent reaches a hazard or after 1000 timesteps. Similar to Figure 6, the baselines achieve significantly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



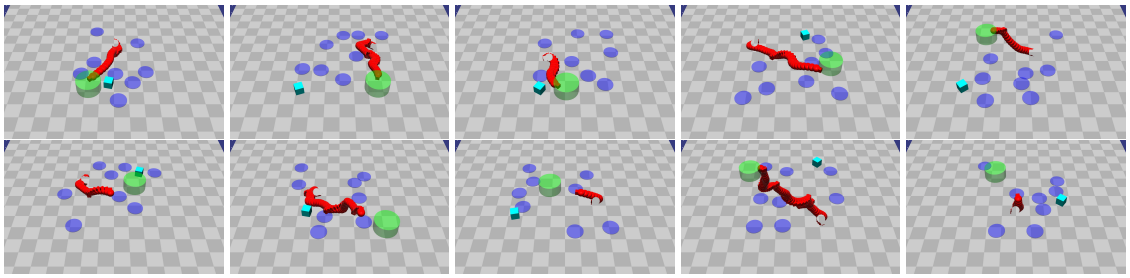
(a) **TRPO** successes (top) and failures (bottom)



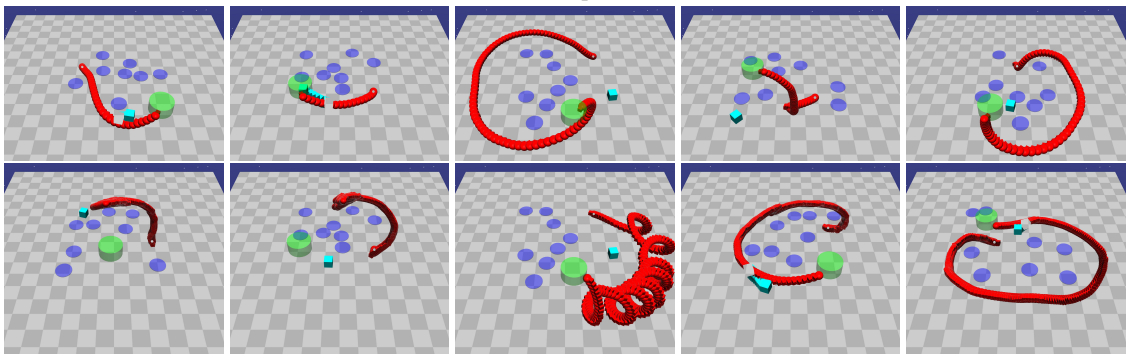
(b) **TRPO-Lagrangian** successes (top) and failures (bottom)



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

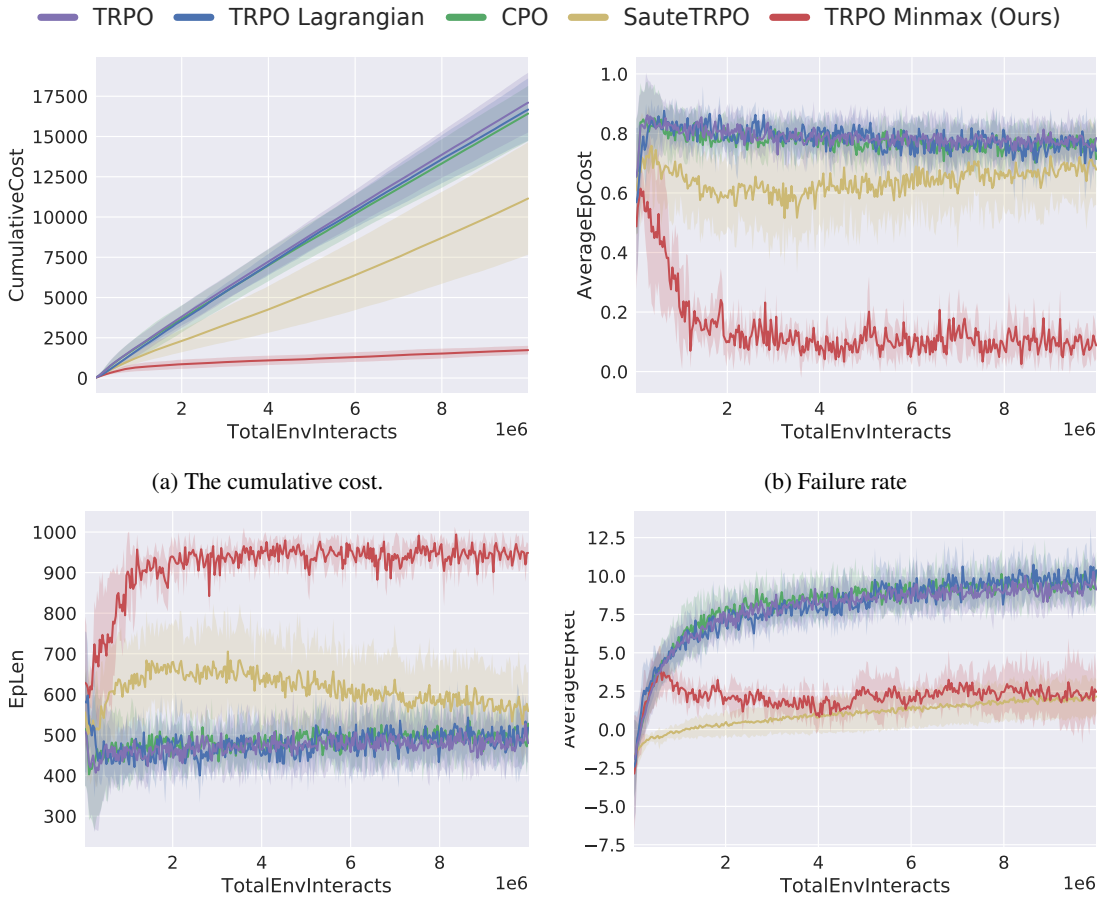


(d) **Sauté-RL** successes (top) and failures (bottom)



(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-HARD 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.



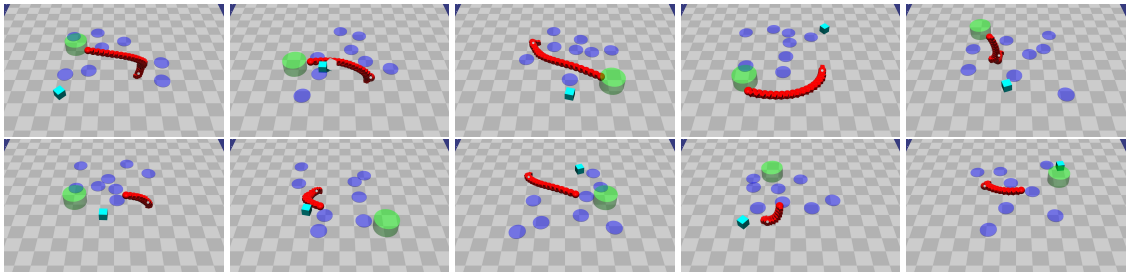
(a) The cumulative cost.

(b) Failure rate

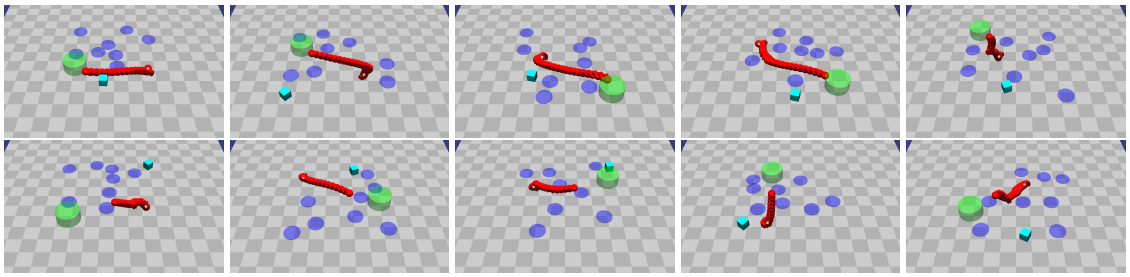
(c) Average episode length

(d) Average returns

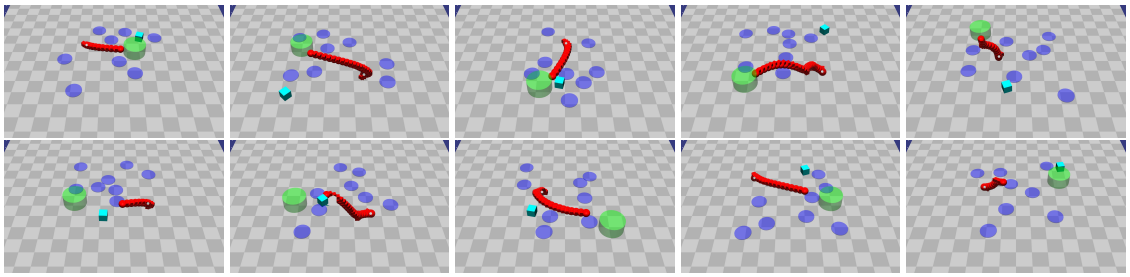
Figure 19: Comparison with baselines in the POINTGOAL1-HARD environment (where $\mathcal{G} = \mathcal{G}^! = \{$, $\}$). Here, higher episode lengths are better (in addition to higher returns) since episodes only terminate when the agent reaches a hazard or after 1000 timesteps. This experiment is similar to Figure 14, but 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 (where the cost threshold was 0). Similar results were 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.



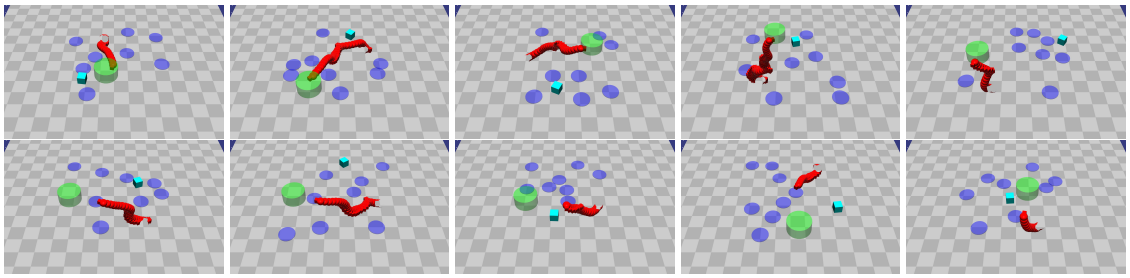
(a) **TRPO** successes (top) and failures (bottom)



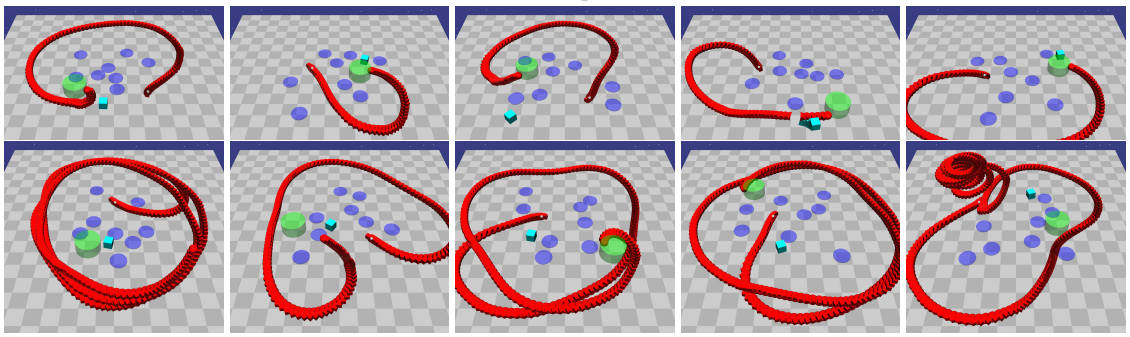
(b) **TRPO-Lagrangian** successes (top) and failures (bottom)



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

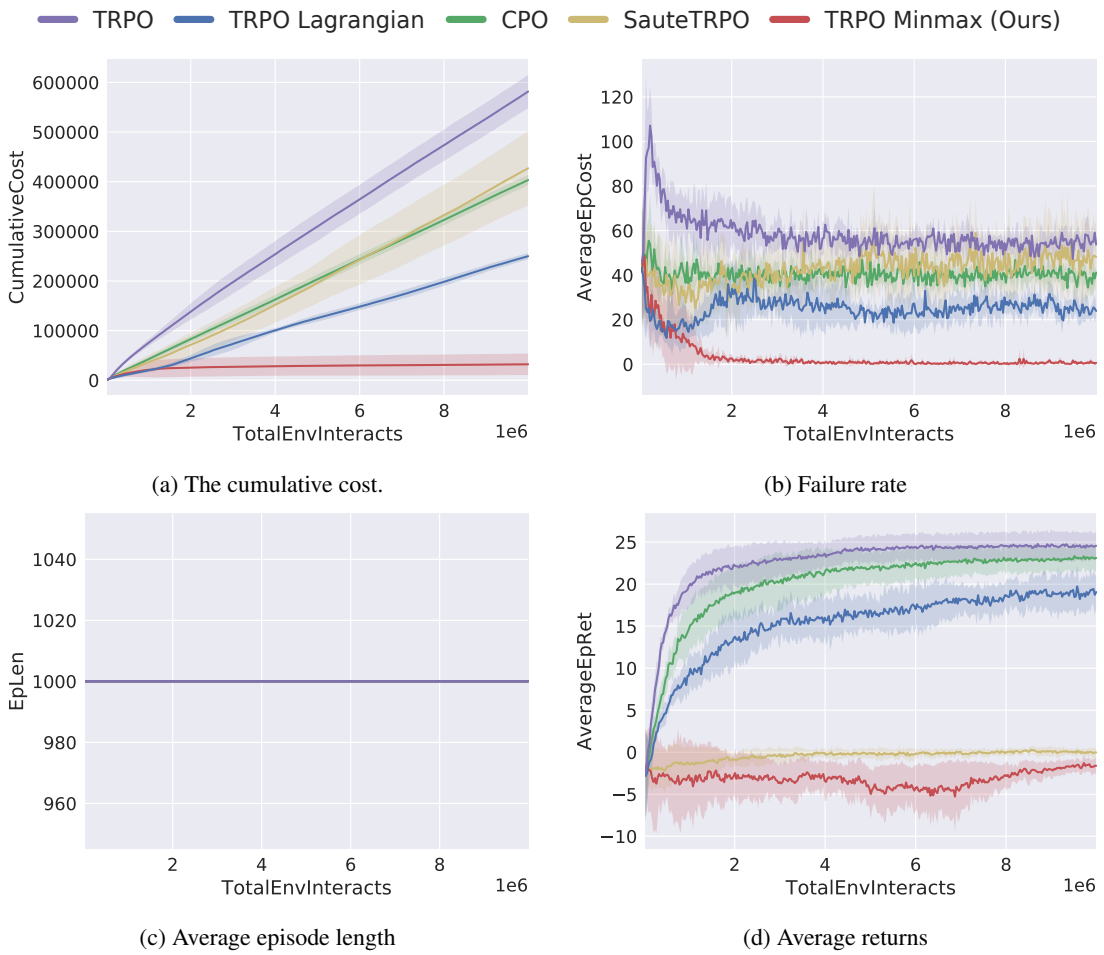


(d) **Sauté-RL** successes (top) and failures (bottom)



(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-HARD 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.



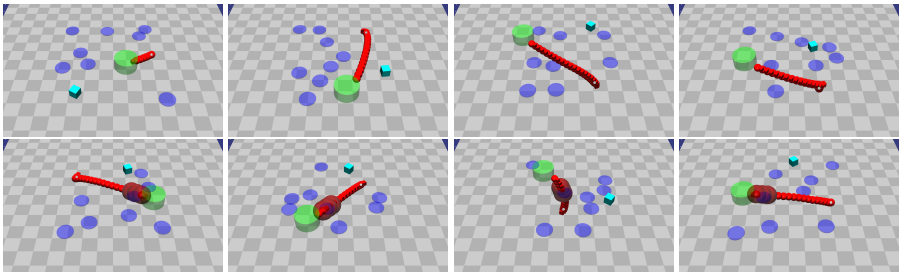
(a) The cumulative cost.

(b) Failure rate

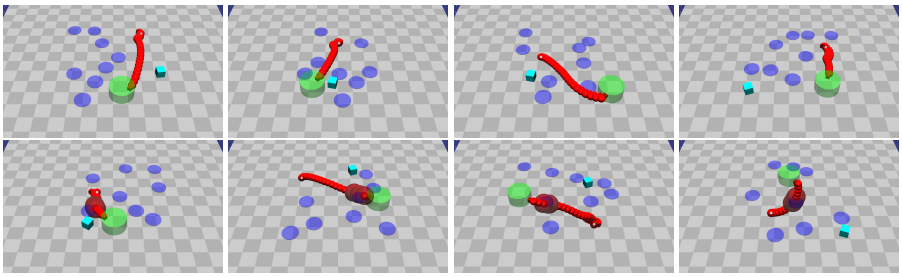
(c) Average episode length

(d) Average returns

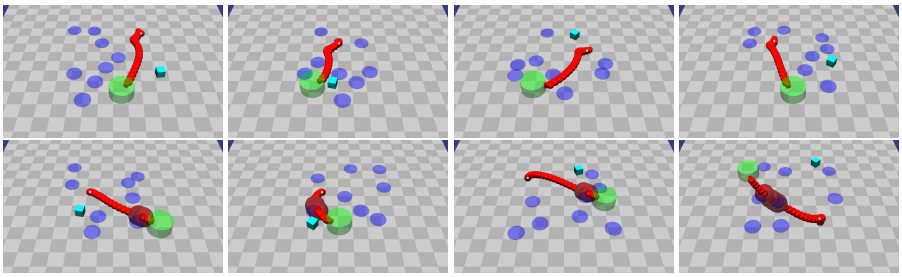
Figure 21: Comparison with baselines in the original Safety Gym POINTGOAL1 environment. This domain is the same as POINTGOAL1-HARD, except that episodes do not terminate when a hazard is hit (hence every episode only terminates after 1000 steps). We set the cost threshold for the baselines to 25 as in 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 2. While TRPO Minmax still beats the baselines in safe exploration (a-b), it struggles to maximise rewards while avoiding unsafe states (d).



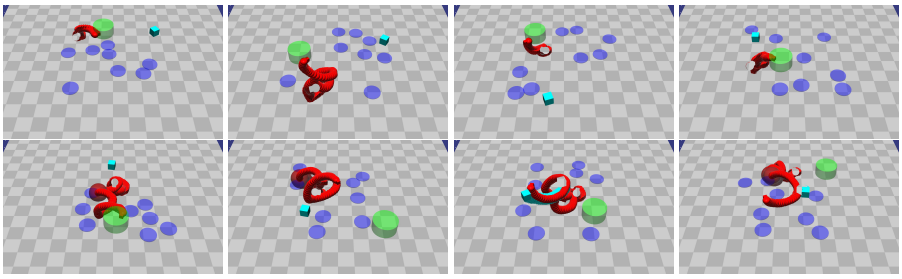
(a) **TRPO** successes (top) and failures (bottom)



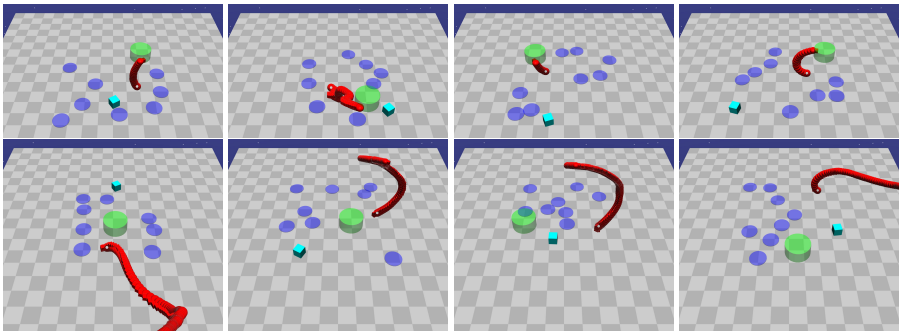
(b) **TRPO-Lagrangian** successes (top) and failures (bottom)



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



(d) **Sauté-RL** 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-HARD 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.