Learn With Imagination: Safe Set Guided State-wise Constrained Policy Optimization

Yifan Sun* YIFANSU2@ANDREW.CMU.EDU
Feihan Li* FEIHANL@ANDREW.CMU.EDU
Weiye Zhao* WEIYEZHA@ANDREW.CMU.EDU
Rui Chen RUIC3@ANDREW.CMU.EDU
Tianhao Wei TWEI2@ANDREW.CMU.EDU
Changliu Liu CLIU6@ANDREW.CMU.EDU
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA [*]

Editors: A. Abate, L. Balzano, N. Ozay, D. Panagou

Abstract

Deep reinforcement learning (RL) has achieved remarkable success across various control tasks. However, its reliance on exploration through trial and error often results in safety violations during training. To mitigate this, safety filters are commonly employed to correct unsafe actions generated by the RL policy. Yet, a key challenge remains: how to enable safety-filter-guided learning to produce a policy that remains optimally safe even after the filter is removed. In this paper, we propose Safe Set Guided State-wise Constrained Policy Optimization (S-3PO) — a novel algorithm designed to generate optimal safe RL policies with zero training violations while maintaining safety during evaluation even without any safety filter. S-3PO integrates a safety-oriented monitor operating on black-box dynamics to ensure safe exploration and introduces an imaginary cost mechanism that guides the safe RL agent toward optimal behavior under safety constraints. This imaginary cost inherits the interpretability of the safety filter while outperforming conventional imitation-based cost designs. Equipped with state-of-the-art components, S-3PO demonstrates superior performance on high-dimensional robotic control tasks, effectively handling expected state-wise constraints and ensuring safety throughout the training process.

1. Introduction

Safe Reinforcement Learning (safe RL) has emerged as a powerful approach in domains such as games and robotic control, where ensuring safety during or after training is critical. While objective-based methods aim to optimize reward, they often lack formal guarantees on safety performance Bohez et al. (2019). To address this, many approaches enforce hard constraints Bouvier et al. (2024b,a); however, these methods are typically effective only in low-dimensional systems. More recent advances Zhao et al. (2023b, 2024b) leverage trust-region techniques combined with Maximum Markov Decision Processes (MMDP) to enforce simultaneous improvement of worst-case performance and adherence to cost constraints.

Despite these developments, RL-based methods fundamentally depend on trial-and-error exploration, making it difficult to guarantee safety throughout the training process. A common strategy to mitigate this issue is the use of safety filters Alshiekh et al. (2018), which is designed to correct unsafe actions generated by the RL policy. These safety filters are often constructed using principles from safe control theories Shao et al. (2021), where energy function-based methods remain the most

[*] * Equal Contribution

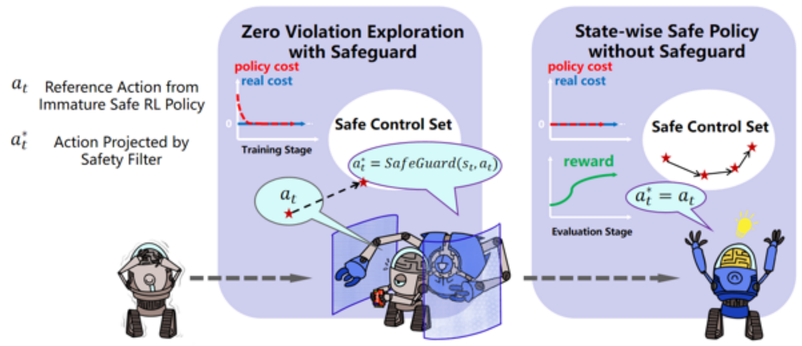

Zero Violation Exploration with Safeguard
State-wise Safe Policy without Safeguard
a_t Reference Action from Immature Safe RL Policy
a_t^* Action Projected by Safety Filter
policy cost
real cost
Training Stage
Safe Control Set
$a_t^* = SafeGuard(s_t, a_t)$
a_t
reward
Evaluation Stage
$a_t^* = a_t$

Figure 1: Overview of the principles of the S-3PO algorithm.

widely adopted approach Khatib (1986); Ames et al. (2014); Liu and Tomizuka (2014); Gracia et al. (2013); Wei and Liu (2019). Although safety filters can ensure safety during training by overriding unsafe actions, they also prevent the policy from learning how to avoid unsafe behaviors on its own. This creates a fundamental dilemma: *how can an agent learn to avoid unsafe scenarios if it is always shielded from experiencing them*?

To enable the policy to learn from the safety filter and generate safe actions by itself, rather than only being protected by the safety filter, Cheng et al. (2019) use Gaussian Process (GP) models to estimate unknown system dynamics and construct a safety filter that implicitly guides the policy updates based on the history of safety filter interventions. To explicitly imitate the effect of the safety filter at each step, other works have considered using reward penalties to learn from safety filters. Krasowski et al. (2022) introduce a constant penalty when the safety filter is activated. Yet, this approach does not account for how unsafe the proposed action was. In contrast, Wabersich and Zeilinger (2021) penalize the reward with magnitude of the action correction applied by the safety filter. However, the magnitude of the correction alone may not accurately capture the true impact of the action on system safety, and the reward penalty can only impose a soft constraint during the learning process.

These limitations highlight a critical gap: **to enable a policy to maintain safety after the safety filter is removed, it is essential to introduce a cost term that explicitly measures how the policy's actions influence the system's safety level if the safe filter is removed**. To this end, we introduce *Safe Set Guided State-wise Constrained* ***P****olicy* ***O****ptimization* (S-3PO). S-3PO safeguards the exploration of immature policies through a black-box safe control mechanism and formulates a novel constrained optimization framework where RL learns an optimal safe policy by constraining *imaginary safety violations*—violations that would have occurred without the filter.

2. Problem Formulation

2.1. Assumptions

Dynamics We consider a robot system described by its state $s_t \in \mathcal{S} \subset \mathbb{R}^{n_s}$ at time step t, with n_s denoting the dimension of the state space $\mathcal{S}$, and its action input $a_t \in \mathcal{A} \subset \mathbb{R}^{n_a}$ at time step t, where n_a represents the dimension of the control space $\mathcal{A}$. The system dynamics are defined as follows:

$$s_{t+1} = f(s_t, a_t), \tag{1}$$

where $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a deterministic function that maps the current robot state and control to the robot's state in the next time step.

To maintain simplicity, our approach focuses on deterministic dynamics, although the proposed method can be easily extended to stochastic dynamics Zhao et al. (2021). Additionally, we assume the access to the dynamics model f is only in the training phase and restricted to an black-box form, such as an implicit digital twin simulator or a deep neural network model Zhao et al. (2021). We also assume there is no model mismatch. Model mismatch can be addressed by robust safe control Wei et al. (2022) and is left for future work. Post training, the knowledge of the dynamics model is concealed—a benefit of using "imaginary cost"—aligning with practical scenarios where digital twins of real-world environments are too costly to access during deployment. Based on these, our core target is to figure out how can safety-filter-guided learning be used to produce a policy that remains optimally safe even without the filter.

Markov Decision Process In this research, our primary focus lies in ensuring safety for episodic tasks, which falls within the purview of finite-horizon Markov Decision Processes (MDP). An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \gamma, R, P, \mu)$. The reward function is denoted by $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, the discount factor by $0 \leq \gamma < 1$, the initial state distribution by $\mu : \mathcal{S} \rightarrow \mathbb{R}$, and the transition probability function by $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$.

The transition probability $P(s'|s, a)$ represents the likelihood of transitioning to state s' when the previous state was s, and the agent executed action a at state s. This paper assumes deterministic dynamics, implying that $P(s_{t+1}|s_t, a_t) = 1$ when $s_{t+1} = f(s_t, a_t)$. We denote the set of all stationary policies as Π, and we further denote π_θ as a policy parameterized by the parameter θ.

In the context of an MDP, our ultimate objective is to learn a policy π that maximizes a performance measure $\mathcal{J}(\pi)$, computed via the discounted sum of rewards, as follows:

$$\mathcal{J}(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \right], \tag{2}$$

where $H \in \mathbb{N}$ denotes the horizon, $\tau = [s_0, a_0, s_1, \cdots]$, and $\tau \sim \pi$ indicates that the distribution over trajectories depends on π, i.e., $s_0 \sim \mu$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim P(\cdot|s_t, a_t)$.

Safety Specification The safety specification requires that the system state remains within a closed subset in the state space, denoted as the "safe set" $\mathcal{S}_S$. This safe set is defined by the zero-sublevel set of a continuous and piecewise smooth function $\phi_0 : \mathbb{R}^{n_s} \rightarrow \mathbb{R}$, where $\mathcal{S}_S = \{s \mid \phi_0(s) \leq 0\}$, usually specified by users. For instance, for collision avoidance, ϕ_0 can be specified as $d_{min} - d$ where d is the closest distance between the robot and environmental obstacles and d_{min} is the distance margin.

2.2. Problem

We are interested in the safety imperative of averting collisions for mobile robots navigating 2D planes. We aim to persistently satisfy safety specifications at every time step while solving MDP, following the intuition of State-wise Constrained Markov Decision Process (SCMDP) Zhao et al. (2023c). Formally, the set of feasible stationary policies for SCMDP is defined as

$$\bar{\Pi}_C = \{\pi \in \Pi \big| \, \forall s_t \sim \tau, s_t \in \mathcal{S}_S\}, \tag{3}$$

where $\tau \sim \pi$. Then, the objective for SCMDP is to find a feasible stationary policy from $\bar{\Pi}_C$ that maximizes the performance measure. Formally,

$$\max_\theta \mathcal{J}(\pi_\theta), \text{ s.t. } \pi_\theta \in \bar{\Pi}_C. \tag{4}$$

State-wise Safe Policy with Zero Violation Training The primary focus of this paper centers on solving (4), i.e., ensuring no safety violation during the training process, while achieving convergence of the policy to the optimal solution of (4).

3. Preliminary

3.1. Implicit Safe Set Algorithm

As a deterministic safety filter, Implicit Safe Set Algorithm (ISSA) Zhao et al. (2021, 2024a) ensures the persistent satisfaction of safety specifications for systems with black-box dynamics (e.g., digital twins or neural networks) through energy function-based optimization. Leveraging energy function $\phi = \phi_0^* + k_1\dot{\phi}_0 + \cdots + k_n\phi_0^{(n)}$ and theoretical results from SSA Liu and Tomizuka (2014), ISSA synthesizes a safety index to guarantee that the safe control set $\mathcal{A}_S(s) := \{a \in \mathcal{A} \mid \dot{\phi} \leq -\eta(\phi)\}$ is nonempty. And then the set $\bar{S} := \{s \mid \phi(s) \leq 0\} \cap \{s \mid \phi_0(s) \leq 0\}$ is forward invariant under Assumption 1. Here $\eta(\phi)$ is designed to be a positive constant when $\phi \geq 0$ and $-\infty$ when $\phi < 0$.

Assumption 1 *The system (1) is a second-order mobile platform that avoids 2D obstacles. 1) The state space is bounded, and the relative acceleration w and angular velocity z to the obstacle are bounded and both can achieve zeros, i.e., $w \in [w_{min}, w_{max}]$ for $w_{min} \leq 0 \leq w_{max}$ and $z \in [z_{min}, z_{max}]$ for $z_{min} \leq 0 \leq z_{max}$; 2) For all possible values of z and w, there always exists a control a to realize such z and w; 3) The discrete-time system time step $dt \to 0$; 4) At any given time, there can at most be one obstacle becoming safety critical (Sparse Obstacle Environment).*

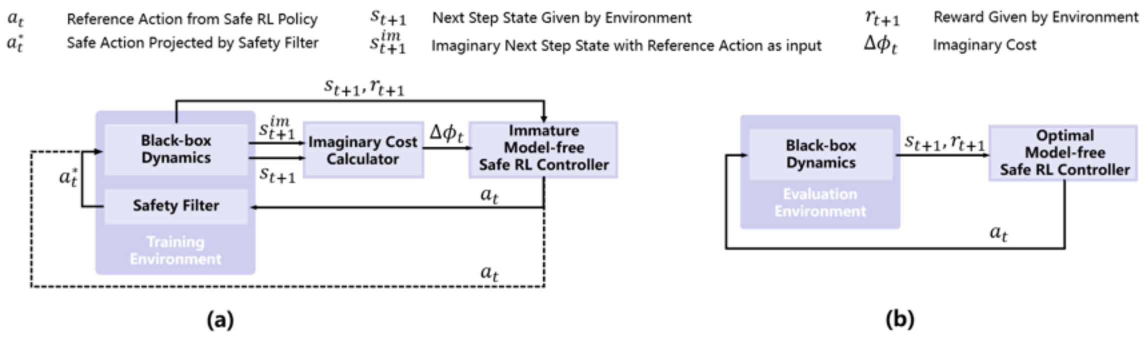
Remark 1 *The bounds in the first assumption will be directly used to synthesize ϕ. The second assumption enables us to turn the question on whether there exists a feasible control in $\mathcal{A}_S^D$ to the question on whether there exists z and w to decrease ϕ. The third assumption ensures that the discrete time approximation error is small. The last assumption ensures that the safety index design rule is applicable to multiple moving obstacles.*

The deterministic ISSA could be used as a safety filter in the discrete-time MDP and is treated as part of the environment during training. We define the discrete-time safe control set as $\mathcal{A}_S^D(s) := \{a \in \mathcal{A} \mid \phi(f(s,a)) \leq \max\{\phi(s) - \eta, 0\}\}$. The ISSA mechanism ensures safety by projecting the nominal control action a_t, proposed by the RL policy π_θ, onto the safe control set $\mathcal{A}_S^D(s_t)$ by solving the optimization problem:

$$\begin{aligned} &\min_{a_t^* \in \mathcal{A}} \|a_t^* - a_t\|^2 \\ &\text{s.t. } \phi(f(s_t, a_t)) \leq \max\{\phi(s_t) - \eta, 0\}. \end{aligned} \tag{5}$$

3.2. State-wise Constrained Policy Optimization

Safe RL algorithms under the framework of Constrained Markov Decision Process (CMDP) do not consider state-wise constraints. To address this gap, State-wise Constrained Policy Optimization (SCPO) was proposed Zhao et al. (2023b) to provide guarantees for state-wise constraint satisfaction in expectation, which is under the framework of State-wise CMDP (SCMDP). To achieve this, SCPO directly constrain the expected maximum state-wise cost along the trajectory. And they introduced Maximum MDP (MMDP). In this setup, a running maximum cost value is associated with each state, and a non-discounted finite MDP is utilized to track and accumulate non-negative increments in cost. The format of MMDP will be introduced in Section 4.



a_t Reference Action from Safe RL Policy
a_t^* Safe Action Projected by Safety Filter
s_{t+1} Next Step State Given by Environment
s_{t+1}^{im} Imaginary Next Step State with Reference Action as input
r_{t+1} Reward Given by Environment
$\Delta\phi_t$ Imaginary Cost
Black-box Dynamics
Safety Filter
Training Environment
Imaginary Cost Calculator
Immature Model-free Safe RL Controller
s_{t+1}, r_{t+1}
s_{t+1}^{im}
s_{t+1}
$\Delta\phi_t$
a_t
a_t^*
(a)
Black-box Dynamics
Evaluation Environment
Optimal Model-free Safe RL Controller
(b)

Figure 2: (a) S-3PO pipeline during training. (b) S-3PO pipeline during evaluation.

4. Safety Index Guided State-wise Constrained Policy Optimization

The core idea of S-3PO is to enforce zero safety violations during training by projecting unsafe actions to the safe set, and then constraining the "imaginary" safety violations to ensure convergence of the policy to an optimal safe policy. As shown in Figure 2, the safety filter is part of the training environment, while the evaluation environment does not have the safety filter.

4.1. Learn with Imaginary Cost

Zero Violation Exploration To ensure zero violation exploration, we adopt ISSA as the safety filter and safeguard nominal control via solving (5) at every time step during policy training. With the safety index synthesis rule in Zhao et al. (2021), ISSA is guaranteed to find a feasible solution of (5), making the system forward invariance in the set $\bar{\mathcal{S}}$. It is worth mentioning that any energy-function-based method that ensures forward invariance could be used as the safety filter, and the scalar energy function could be used to evaluate the imaginary cost. The integration with other energy-function-based methods will be left for future work.

Learning Safety Measures Safely While eliminating safety violations during training is beneficial, it also poses challenges for RL policy training, as RL relies on a trial-and-error process. To address this, our key insight is that instead of directly encountering unsafe states ($s \notin \mathcal{S}_S$), the policy can leverage an "imaginary cost" to learn about unsafe scenarios without actually experiencing them.

Imaginary Cost Define "imaginary cost" as $\Delta\phi_t = \Delta\phi(s_t, a_t, s_{t+1}) \doteq \phi(f(s_t, a_t)) - \phi(f(s_t, a_t^*))$, i.e. the degree of required correction to safeguard a_t. Here a_t^* is the projected action by ISSA. Therefore, $\Delta\phi_t$ can be treated as an imagination on how unsafe the reference action would be, where $\Delta\phi_t \leq 0$ means $a_t \in \mathcal{A}_S^D(s_t)$.

Following the definition, Equation (4) can be translated to:

$$\max_{\theta} \mathcal{J}(\pi_\theta), \text{ s.t. } \pi_\theta \in \{\pi \in \Pi \big| \, \forall \Delta\phi_t \sim \tau, \Delta\phi_t \leq 0\}. \tag{6}$$

Remark 2 *Policies satisfying (6) ensure there is no imaginary safety violation in expectation for any possible a_t, making π_θ a safe policy as required by (4), to be proved by lemma 5.*

4.2. Transfrom State-wise Constraint into Maximum Constraint

For (6), each state-action transition pair corresponds to a constraint, which is intractable to solve. Inspired by Zhao et al. (2023c), we constrain the expected maximum state-wise $\Delta\phi$ along the trajectory instead of individual state-action transition $\Delta\phi$.

Next, by treating $\Delta\phi_t$ as an "imaginary" cost, we define a MMDP Zhao et al. (2023c) by introducing (i) an up-to-now maximum state-wise cost M within $\mathcal{M} \subset \mathbb{R}$, and (ii) a "cost increment" function D, where $D : (\mathcal{S}, \mathcal{M}) \times \mathcal{A} \times (\mathcal{S}, \mathcal{M}) \to [0, \mathbb{R}^+]$ maps the augmented state-action transition tuple to non-negative cost increments. We define the augmented state $\hat{s} = (s, M) \in (\mathcal{S}, \mathcal{M}) \doteq \hat{\mathcal{S}}$, where $\hat{\mathcal{S}}$ is the augmented state space. Formally,

$$D\big(\hat{s}_t, a_t, \hat{s}_{t+1}\big) = \max\{\Delta\phi(s_t, a_t, s_{t+1}) - M, 0\}. \tag{7}$$

By setting $D\big(\hat{s}_0, a_0, \hat{s}_1\big) = \Delta\phi(s_0, a_0, s_1)$, we have $M = \sum_{k=0}^{t-1} D\big(\hat{s}_k, a_k, \hat{s}_{k+1}\big)$ for $t \geq 1$. Hence, we define *expected maximum state-wise cost* (or D-return) for S-3PO policy π:

$$\mathcal{J}_D(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{H} D\big(\hat{s}_t, a_t, \hat{s}_{t+1}\big)\right]. \tag{8}$$

With (8), (6) can be rewritten as:

$$\max_{\pi} \mathcal{J}(\pi), \ \textbf{s.t.} \ \mathcal{J}_D(\pi) \leq 0, \tag{9}$$

where $\mathcal{J}(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{H} \gamma^t R(\hat{s}_t, a_t, \hat{s}_{t+1})\right]$ and $R(\hat{s}, a, \hat{s}') \doteq R(s, a, s')$. With $R(\tau)$ being the discounted return of a trajectory, we define the on-policy value function as $V^\pi(\hat{s}) \doteq \mathbb{E}_{\tau\sim\pi}[R(\tau)|\hat{s}_0 = \hat{s}]$, the on-policy action-value function as $Q^\pi(\hat{s}, a) \doteq \mathbb{E}_{\tau\sim\pi}[R(\tau)|\hat{s}_0 = \hat{s}, a_0 = a]$, and the advantage function as $A^\pi(\hat{s}, a) \doteq Q^\pi(\hat{s}, a) - V^\pi(\hat{s})$.

Lastly, we define on-policy value functions, action-value functions, and advantage functions for the cost increments in analogy to V^π, Q^π, and A^π, with D replacing R, respectively. We denote those by V_D^π, Q_D^π and A_D^π.

Remark 3 *Equation* (6) *is difficult to solve since there are as many constraints as the size of trajectory* τ. *With* (9)*, we turn all constraints in* (6) *into only a single constraint on the maximal* $\Delta\phi$ *along the trajectory, yielding a practically solvable problem.*

4.3. S-3PO

To solve (9), we propose S-3PO under the framework of trust region optimization methods Schulman et al. (2015). S-3PO uses KL divergence distance to restrict the policy search in (9) within a trust region around the most recent policy π_k. Moreover, S-3PO uses surrogate functions for the objective and constraints, which can be easily estimated from sample trajectories by π_k. Mathematically, S-3PO updates policy via solving the following optimization:

$$\begin{aligned}
\pi_{k+1} = \ & \mathop{\textbf{argmax}}_{\pi\in\Pi_\theta} \mathop{\mathbb{E}}_{\substack{\hat{s}\sim d^{\pi_k}\\ a\sim\pi}}[A^{\pi_k}(\hat{s}, a)] \\
\textbf{s.t.} \ & \mathbb{E}_{\hat{s}\sim\bar{d}^{\pi_k}}[\mathcal{D}_{KL}(\pi\|\pi_k)[\hat{s}]] \leq \delta, \\
& \mathcal{J}_D(\pi_k) + \mathop{\mathbb{E}}_{\substack{\hat{s}\sim\bar{d}^{\pi_k}\\ a\sim\pi}}\left[A_D^{\pi_k}(\hat{s}, a)\right] + 2(H+1)\epsilon_D^\pi\sqrt{\frac{1}{2}\delta} \leq 0.
\end{aligned} \tag{10}$$

where $\mathcal{D}_{KL}(\pi'\|\pi)[\hat{s}]$ is KL divergence between two policy (π', π) at state $\hat{s}$, the set $\{\pi \in \Pi_\theta : \mathbb{E}_{\hat{s}\sim\bar{d}^{\pi_k}}[\mathcal{D}_{KL}(\pi\|\pi_k)[\hat{s}]] \leq \delta\}$ is called *trust region*, $d^{\pi_k} \doteq (1-\gamma)\sum_{t=0}^{H}\gamma^t P(\hat{s}_t = \hat{s}|\pi_k)$, $\bar{d}^{\pi_k} \doteq \sum_{t=0}^{H} P(\hat{s}_t = \hat{s}|\pi_k)$ and $\epsilon_D^\pi \doteq \mathbf{max}_{\hat{s}}|\mathbb{E}_{a\sim\pi}[A_D^{\pi_k}(\hat{s}, a)]|$.

Remark 4 *Despite the complex forms, the objective and constraints in (10) can be interpreted in two steps. First, maximizing the objective (expected reward advantage) within the trust region (marked by the KL divergence constraint) theoretically guarantees the worst performance degradation. Second, $J_D(\pi)$ can not be computed at step $k+1$ since the state s_{k+1} is inaccessible, thus we leverage a surrogate function to upper bound the $J_D(\pi)$ to guarantee the worst-case "imagionary" cost is non-positive at all steps as in (6).*

4.4. Practical Implementation

The pseudocode of S-3PO is give as algorithm 1. Here we summarize two techniques that helps with S-3PO's practical performance. (i) **Weighted loss for cost value targets**: A critical step in S-3PO involves fitting the cost increment value function, $V_D^\pi(\hat{s}_t)$, which represents the maximum future cost increment relative to the highest state-wise cost observed so far. This function follows a non-increasing staircase pattern along the trajectory. Thus, we adopt a weighted loss function, L_{weight}, to penalize predictions that violate the non-increasing property: $L_{weight} = L(\hat{y}_t - y_t) * (1 + w * \mathbb{1}[(\hat{y}_t - y_{t-1}) > 0])$, where L denotes Mean Squared Error, $\hat{y}_t$ is the prediction, y_t is the fitting target and w is the penalty weight. (ii) **Line Search scheduling**: Constraints in (10) might become infeasible due to approximation errors. In this case, we perform a recovery update, enforcing the cost advantage A_D^π to decrease from early training steps k_{safe} while focusing on reward improvements of A^π towards the end of training, prioritizing safety first and reward performance later. Check Appendix C.3 for furthur details.

5. Theoretical Results

In this section, we first present the lemma to show the equivalence between constraining the *imaginary cost* and constraining the safety violation. Then we present the main conclusion for S3PO.

Lemma 5 (Safety Equivalence under Imaginary Cost) *For a given policy π and any initially safe state s_0 ($\phi(s_0) \leq 0$), the following two conditions are equivalent: 1) the corresponding trajectory in the training environment has zero imaginary cost $\max_t \Delta\phi_t \leq 0$; 2) the corresponding trajectory in the evaluation environment has zero safety violation $\max_t \phi_t \leq 0$. And if either condition holds, the two trajectories are the same.*

Theorem 6 (Safety and Optimality of S-3PO) *S-3PO will converge in the training environment to a policy π with no imaginary cost in expectation, and bounded worst case reward performance. In particular, if π_k and π_{k+1} are related by applying S-3PO (10), then with $\epsilon^{\pi_{k+1}} \doteq \mathbf{max}_{\hat{s}}|\mathbb{E}_{a\sim\pi_{k+1}}[A^{\pi_k}(\hat{s},a)]|$, the performance of π_{k+1} in the training environment satisfies:*

$$\mathcal{J}(\pi_{k+1}) - \mathcal{J}(\pi_k) \geq -\frac{\sqrt{2\delta}\gamma\epsilon^{\pi_{k+1}}}{1-\gamma}.$$

Since the training environment is deterministic and all the assumptions in the theoretical results for SCPO are satisfied, the proof for the theorem directly follows from Proposition 2 from SCPO Zhao et al. (2023b).

By lemma 5, no imaginary cost in the training environment implies no safety violation in the evaluation environment. The theorem then implies that the converged policy could achieve zero safety violation in expectation in the evaluation environment. Nevertheless, formally establishing their equivalence in the probability space (e.g., in expectation) will be left for future work.

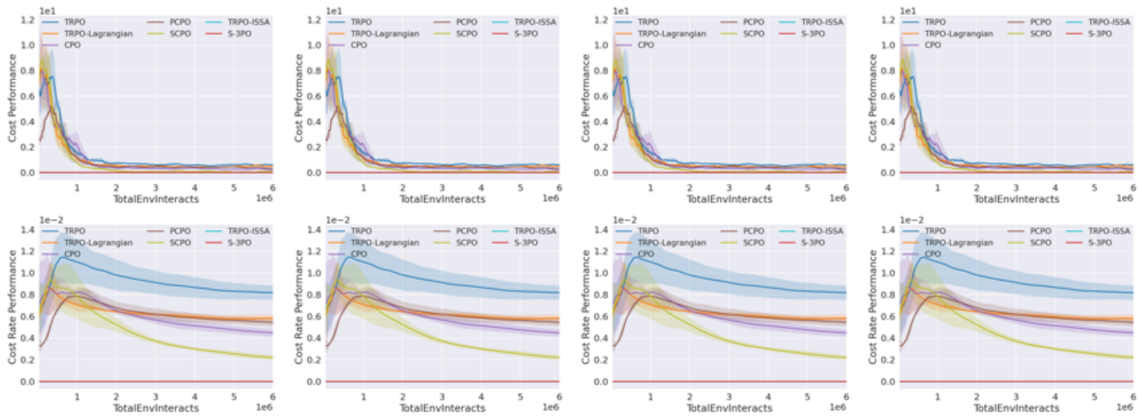

(*a*) Point_1Hazard (*b*) Point_8Hazard (*c*) Point_1Pillar (*d*) Swimmer_1Hazard

Figure 3: Illustration of training-time cost performance from four representative test suites.

6. Experiments

In our experiments, we aim to answer: **Q1:** Does S-3PO achieve zero-violation during the training? **Q2:** How does S-3PO without safeguard compare with other advanced safe RL methods? **Q3:** Does S-3PO learn to act without safeguard? **Q4:** How does weighted loss trick impact the performance of S-3PO? **Q5:** Is “imaginary” cost necessary to make the RL policy learn to achieve zero violation by itself? **Q6:** How does S-3PO scale to high dimensional robots?

6.1. Experiments Setup

To answer these questions, we conducted experiments on the safe reinforcement learning benchmark GUARD Zhao et al. (2023a) which is based on Mujoco and Gym interface.

Environment Setting We design experimental environments with different task types, constraint types, constraint numbers, and constraint sizes. We name these environments as {Robot}_{Constraint Number}{Constraint Type}. All of the environments are based on `Goal` where the robot must navigate to a goal. Three different robots that can be categorized into two types are included in our experiments: (i) Wheel Robot: **Point**: (fig. 8(*a*)) A point robot ($\mathcal{A} \subseteq \mathbb{R}^2$) that maintains seamless interaction with the environment. (ii) Link Robot: (a) **Swimmer**: (fig. 8(*b*)) A three-link robot ($\mathcal{A} \subseteq \mathbb{R}^2$) that interacts intermittently with the surroundings. (b) **Ant**: (fig. 8(*c*)) A quadrupedal robot ($\mathcal{A} \subseteq \mathbb{R}^8$). Two different types of constraints are considered. (i) **Hazard**: (fig. 8(*d*)) Trespassable circles on the ground. (ii) **Pillar**: (fig. 8(*e*)) Fixed obstacles. All tasks are trained over 200 epochs, with each epoch consisting of 30,000 steps. More details about the experiments are discussed in Appendix C.1.

Comparison Group The methods in the comparison group include: (i) unconstrained RL algorithm TRPO (Schulman et al., 2015) and TRPO-ISSA. (ii) end-to-end constrained safe RL algorithms CPO (Achiam et al., 2017), TRPO-Lagrangian (Bohez et al., 2019), PCPO (Yang et al., 2020), SCPO (Zhao et al., 2023b). (iii) We select TRPO as our baseline method since it already has safety-constrained derivatives that can be tested off-the-shelf. The full list of parameters of all methods compared can be found in Appendix C.2.

Metrics For comparison, we evaluate algorithm performance based on (i) reward performance, (ii) average episode cost, and (iii) cost rate. More details are provided in Appendix C.4. We set the limit of cost to 0 for all safe RL algorithms since no violation of the constraints is allowed.

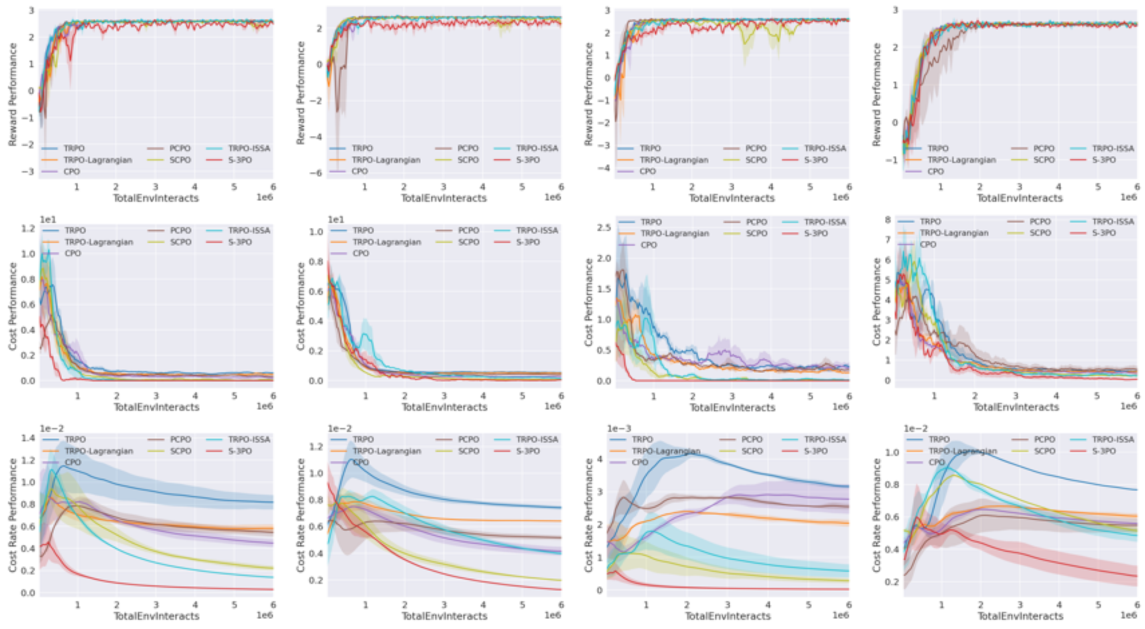

(*a*) Point_1Hazard (*b*) Point_8Hazard (*c*) Point_1Pillar (*d*) Swimmer_1Hazard

Figure 4: Results from four representative test suites (evaluated without the safeguard).

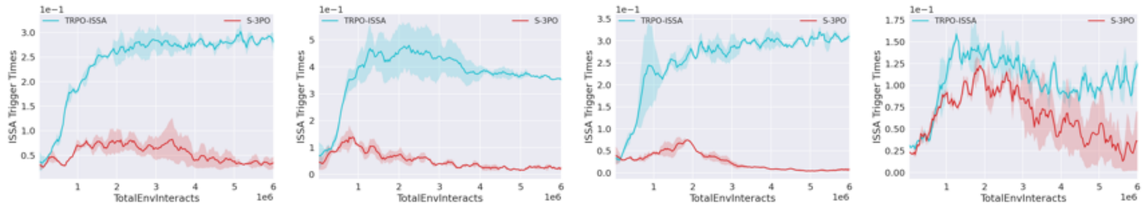

(*a*) Point_1Hazard (*b*) Point_8Hazard (*c*) Point_1Pillar (*d*) Swimmer_1Hazard

Figure 5: Triggering frequency of the safeguard from four representative test suites.

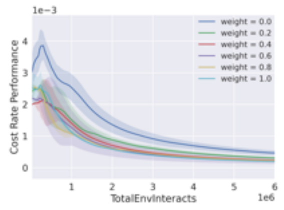
6.2. Evaluating S-3PO and Comparison Analysis

Zero Violation During Training The training performance of four representative test suites are summarized in Figure 3, where the S-3PO algorithm clearly outperforms other baseline methods by achieving zero violations, consistent with the safety guarantee outlined in Zhao et al. (2021). For more experiments, please check Appendix C. This superior performance is attributed to the safeguard mechanism within the S-3PO framework, which effectively corrects unsafe actions at every step, particularly during training. Furthermore, as demonstrated in Figure 4, the reward performance remains comparable to advanced baselines. This distinct capability of S-3PO ensures safe reinforcement learning with zero safety violations, addressing **Q1**.

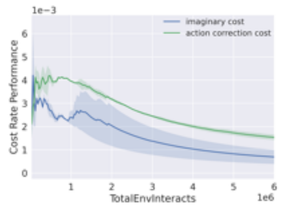
State-wise Safety Without Safety Monitor At the end of each epoch, the S-3PO policy is tested over 10,000 steps without the safeguard. This allows us to determine whether S-3PO effectively learns a state-wise safe policy through the guidance of the safe set-guided cost. As shown in Figure 4, S-3PO demonstrates superior performance even without the safeguard, achieving (i) near-zero average episode cost and (ii) significantly reduced cost rates, all while maintaining competitive reward performance. These findings highlights that by minimizing imaginary safety violations, the policy rapidly learns to act safely, which addresses **Q2**.

Learn to Act without Safeguard As highlighted in Section 4.1, the key concept behind penalizing imaginary safety violations is to minimize the activation of the safeguard, thereby significantly reducing its computational complexity and enabling real-time implementation. To illustrate this, we visualize the average number of times the ISSA-based safeguard is triggered per step in Figure 5. For comparison, TRPO-ISSA is included as a baseline, which relies continuously on the safeguard to maintain safe control. Figure 5 shows that S-3PO dramatically reduces the frequency of safeguard activations, approaching zero, indicating that a state-wise safe policy has been effectively learned, thus addressing **Q3**.

Ablation on Weighted Loss for Fitting Cost Increment Value Targets As pointed in Section 4.4, fitting $V_{D_i}(\hat{s}_t)$ is a critical step towards solving S-3PO, which is challenging due to non-increasing stair shape of the target sequence. To elucidate the necessity of weighted loss for solving this challenge, we evaluate the cost rate of S-3PO under six distinct weight settings (0.0, 0.2, 0.4, 0.6, 0.8, 1.0) on `Point_4Hazard` test suite. The results shown in Figure 6(*a*) validates that a larger weight (hence higher penalty on predictions that violate the characteristics of value targets) results in better cost rate performance. This ablation study answers **Q4**.



(*a*) Comparison of cost rate performance with 6 different weights.



(*b*) Comparison between "imaginary" cost and action correction cost.

Figure 6: Comparison results of S-3PO

Necessity of "Imaginary" Cost To understand the importance of the "imaginary" cost within the S-3PO framework, we compare it to another cost based on the magnitude of action correction Chen and Liu (2021). This empirical analysis is conducted using the `Point_4Hazard` test suite. As shown in Figure 6(*b*), the "imaginary" cost yields superior cost rate performance. This suggests that the "imaginary" cost offers deeper insights into the complex dynamics between the robot and its environment, thereby addressing **Q5**.

Scale S-3PO to High-Dimensional Link Robots To showcase S-3PO's scalability and performance with complex, high-dimensional link robots, we conducted additional tests on `Ant_1Hazard` featuring 8 dimensional control spaces. As shown in Figure 7, S-3PO effectively drives the cost to zero and rapidly reduces the cost rate, showcasing its superiority in high-dimensional safety policy learning and highlighting its exceptional scalability to more complex systems, which answers **Q6**.

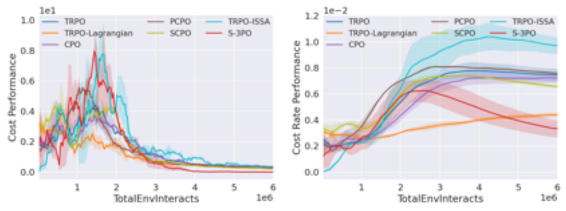


Figure 7: Cost performance of Ant_1Hazard.

7. Conclusion and Future Prospectus

In this study, we introduce Safe Set Guided State-wise Constrained Policy Optimization (S-3PO), a novel algorithm pioneering state-wise safe optimal policies. This distinction is underlined by the absence of training violations, signifying an error-free learning paradigm. S-3PO employs a safeguard anchored in black-box dynamics to ensure secure exploration. Subsequently, it integrates a novel "imaginary" safety cost to guide the RL agent towards optimal safe policies. In the following work, we will try to implement our algorithm in real robot implementation.

Acknowledgement

This work is supported by the National Science Foundation under grant No. 2144489.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 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.

Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278. IEEE, 2014.

Steven Bohez, Abbas Abdolmaleki, Michael Neunert, Jonas Buchli, Nicolas Heess, and Raia Hadsell. Value constrained model-free continuous control. *arXiv preprint arXiv:1902.04623*, 2019.

Jean-Baptiste Bouvier, Kartik Nagpal, and Negar Mehr. Learning to provably satisfy high relative degree constraints for black-box systems, 2024a. URL `https://arxiv.org/abs/2407.20456`.

Jean-Baptiste Bouvier, Kartik Nagpal, and Negar Mehr. Policed rl: Learning closed-loop robot control policies with provable satisfaction of hard constraints, 2024b. URL `https://arxiv.org/abs/2403.13297`.

Hongyi Chen and Changliu Liu. Safe and sample-efficient reinforcement learning for clustered dynamic environments. *IEEE Control Systems Letters*, 6:1928–1933, 2021.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3387–3395, 2019.

Luis Gracia, Fabricio Garelli, and Antonio Sala. Reactive sliding-mode algorithm for collision avoidance in robotic systems. *IEEE Transactions on Control Systems Technology*, 21(6):2391–2399, 2013.

Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

Hanna Krasowski, Jakob Thumm, Marlon Müller, Lukas Schäfer, Xiao Wang, and Matthias Althoff. Provably safe reinforcement learning: Conceptual analysis, survey, and benchmarking. *arXiv preprint arXiv:2205.06750*, 2022.

Changliu Liu and Masayoshi Tomizuka. Control in a safe set: Addressing safety in human-robot interactions. In *ASME 2014 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2014.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *CoRR*, abs/1910.01708, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

Yifei Simon Shao, Chao Chen, Shreyas Kousik, and Ram Vasudevan. Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control. *IEEE Robotics and Automation Letters*, 6(2):3663–3670, 2021.

Kim Peter Wabersich and Melanie N Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021.

Tianhao Wei and Changliu Liu. Safe control algorithms using energy functions: A unified framework, benchmark, and new directions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 238–243. IEEE, 2019.

Tianhao Wei, Shucheng Kang, Weiye Zhao, and Changliu Liu. Persistently feasible robust safe control by safety index synthesis and convex semi-infinite programming. *IEEE Control Systems Letters*, 7:1213–1218, 2022.

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

Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

Weiye Zhao, Rui Chen, Yifan Sun, Ruixuan Liu, Tianhao Wei, and Changliu Liu. Guard: A safe reinforcement learning benchmark. *arXiv preprint arXiv:2305.13681*, 2023a.

Weiye Zhao, Rui Chen, Yifan Sun, Tianhao Wei, and Changliu Liu. State-wise constrained policy optimization. *arXiv preprint arXiv:2306.12594*, 2023b.

Weiye Zhao, Tairan He, Rui Chen, Tianhao Wei, and Changliu Liu. State-wise safe reinforcement learning: A survey. *arXiv preprint arXiv:2302.03122*, 2023c.

Weiye Zhao, Tairan He, Feihan Li, and Changliu Liu. Implicit safe set algorithm for provably safe reinforcement learning, 2024a. URL `https://arxiv.org/abs/2405.02754`.

Weiye Zhao, Feihan Li, Yifan Sun, Yujie Wang, Rui Chen, Tianhao Wei, and Changliu Liu. Absolute state-wise constrained policy optimization: High-probability state-wise constraints satisfaction, 2024b. URL `https://arxiv.org/abs/2410.01212`.

Appendix A. Algorithms

A.1. S-3PO Algorithm

Algorithm 1 S-3PO

```
Input: Initial policy $\pi_0 \in \Pi_\theta$.
for $k = 0, 1, 2, \dots$ do
  for $t = 0, 1, 2, \dots$ do
    Sample nominal action $a_t \sim \pi_k(s_t)$
    Compute and execute $a_t^* = \text{ISSA}(s_t, a_t^r)$
    Log $\tau \leftarrow \tau \cup \{(s_t, a_t, r_t, s_{t+1}, \Delta\phi_t)\}$
  end for
  $g \leftarrow \nabla_\theta \mathbb{E}_{\hat{s},a\sim\tau}\left[A^{\pi}(\hat{s},a)\right]\big|_{\theta=\theta_k}$
  $b \leftarrow \nabla_\theta \mathbb{E}_{\hat{s},a\sim\tau}\left[A_D^{\pi}(\hat{s},a)\right]\big|_{\theta=\theta_k}$
  $c \leftarrow \mathcal{J}_D(\pi_k) + 2(H+1)\epsilon_D^{\pi}\sqrt{\delta/2}$
  $H \leftarrow \nabla_\theta^2 \mathbb{E}_{\hat{s}\sim\tau}[\mathcal{D}_{KL}(\pi\|\pi_k)[\hat{s}]]\big|_{\theta=\theta_k}$
  $\theta_{k+1}^* = \underset{\theta}{\textbf{argmax}}\ g^\top(\theta-\theta_k)$ s.t.
    $\frac{1}{2}(\theta-\theta_k)^\top H(\theta-\theta_k) \le \delta, c + b^\top(\theta-\theta_k) \le 0$
  Get search direction $\Delta\theta^* \leftarrow \theta_{k+1}^* - \theta_k$
  for $j = 0, 1, 2, \dots$ do
    $\theta' \leftarrow \theta_k + \xi^j \Delta\theta^*$
    if $\mathbb{E}_{\hat{s}\sim\tau}[\mathcal{D}_{KL}(\pi_{\theta'}\|\pi_k)[\hat{s}]] \le \delta$ and
    $\mathbb{E}_{\hat{s},a\sim\tau}\left[A_D^{\pi_{\theta'}}(\hat{s},a) - A_D^{\pi_k}(\hat{s},a)\right] \le \max(-c, 0)$ and
    $(k \le k_{\text{safe}}$ or $\mathbb{E}_{\hat{s},a\sim\tau}\left[A^{\pi_{\theta'}}(\hat{s},a)\right] \ge \mathbb{E}_{\hat{s},a\sim\tau}\left[A^{\pi_k}(\hat{s},a)\right])$ then
      $\theta_{k+1} \leftarrow \theta'$ {Update policy}
      break
    end if
  end for
end for
```

Appendix B. Proof of Lemma 5

Proof To prove that the two conditions are equivalent, we show that condition 2 can be derived from condition 1, and vice versa.

(Condition 1 ⇒ Condition 2):

By the property of ISSA, $\Delta\phi > 0$ only when the safety filter is triggered. Under condition 1, we have $\Delta\phi \leq 0$ throughout the trajectory in the training environment, which implies that ISSA is never activated during training. As a result, the trajectory remains entirely within the safe state set, satisfying $\max_t \phi_t \leq 0$.

Moreover, since ISSA is inactive, the applied action coincides with the nominal policy action, i.e., $a_t^* = a_t$ for all t. Therefore, the same sequence of actions will be executed in the evaluation environment (which uses the same initial state), producing the same trajectory that stays within the safe set, with $\max_t \phi_t \leq 0$ and without triggering ISSA.

(Condition 2 ⇒ Condition 1):

Conversely, if a given policy starting from a certain initial state generates a trajectory in the evaluation environment that never triggers ISSA and satisfies $\max_t \phi_t \leq 0$, the same trajectory will be produced in the training environment when initialized from the same state. Along this trajectory, ISSA remains inactive, implying that the imaginary cost remains zero, i.e., $\Delta\phi_t \leq 0$ for all t.

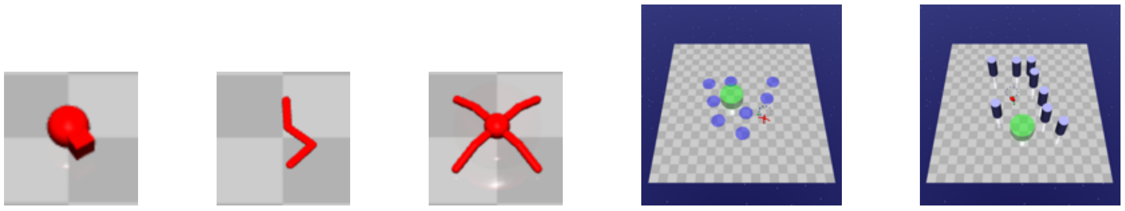
Since both conditions ensure that ISSA is never triggered along the entire trajectory, the actions remain unchanged, $a_t^* = a_t$ for all t, and the resulting trajectories are identical in both the training and evaluation environments.

■

It is important to note that the guarantee in Lemma 5 holds only when both conditions are strictly satisfied over the entire trajectory distribution induced by the policy. In contrast, for the probability space where the conditions are satisfied in expectation, the trajectories in the evaluation environment may deviate from those in the training environment, even when initialized from the same initial state. Such deviation can occur if either the imaginary cost or the safety index ϕ becomes positive at any step along the trajectory. The deviation may lead to discrepancies in the policy performance during evaluation, which will be analyzed in future work.

Appendix C. Expeiment Details

C.1. Environment Settings



(a) Point (b) Swimmer (c) Ant (d) Hazard (e) Pillar

Figure 8: Robots and constraints for benchmark problems in our environment.

Goal Task In the Goal task environments, the reward function is:

$$r(x_t) = d^g_{t-1} - d^g_t + \mathbf{1}[d^g_t < R^g] ,$$

where d^g_t is the distance from the robot to its closest goal and R^g is the size (radius) of the goal. When a goal is achieved, the goal location is randomly reset to someplace new while keeping the rest of the layout the same. The test suites of our experiments are summarized in Table 1.

Hazard Constraint In the Hazard constraint environments, the cost function is:

$$c(x_t) = \max(0, R^h - d^h_t) ,$$

where d^h_t is the distance to the closest hazard and R^h is the size (radius) of the hazard.

Pillar Constraint In the Pillar constraint environments, the cost $c_t = 1$ if the robot contacts with the pillar otherwise $c_t = 0$.

Additional high dimensional link robot To scale our method to high dimensional link robots. We additionally adopt **Walker** shown in 9 as a bipedal robot ($\mathcal{A} \subseteq \mathbb{R}^{10}$) in our experiments.

Figure 9: Walker

State Space The state space is composed of two parts. The internal state spaces describe the state of the robots, which can be obtained from standard robot sensors (accelerometer, gyroscope, magnetometer, velocimeter, joint position sensor, joint velocity sensor and touch sensor). The details of the internal state spaces of the robots in our test suites are summarized in Table 2. The external state spaces are describe the state of the environment observed by the robots, which can be obtained from 2D lidar or 3D lidar (where each lidar sensor perceives objects of a single kind). The state spaces of all the test suites are summarized in Table 3. Note that Vase and Gremlin are two other constraints in Safety Gym Ray et al. (2019) and all the returns of vase lidar and gremlin lidar are zero vectors (i.e., $[0, 0, \cdots, 0] \in \mathbb{R}^{16}$) in our experiments since none of our test suites environments has vases.

Table 1: The test suites environments of our experiments

Task Setting	Low dimension		High dimension	
	Point	Swimmer	Walker	Ant
Hazard-1	✓	✓	✓	✓
Hazard-4	✓			
Hazard-8	✓			
Pillar-1	✓			
Pillar-4	✓			
Pillar-8	✓			

Table 2: The internal state space components of different test suites environments.

Internal State Space	Point	Swimmer	Walker	Ant
Accelerometer ($\mathbb{R}^3$)	✓	✓	✓	✓
Gyroscope ($\mathbb{R}^3$)	✓	✓	✓	✓
Magnetometer ($\mathbb{R}^3$)	✓	✓	✓	✓
Velocimeter ($\mathbb{R}^3$)	✓	✓	✓	✓
Joint position sensor ($\mathbb{R}^n$)	$n = 0$	$n = 2$	$n = 10$	$n = 8$
Joint velocity sensor ($\mathbb{R}^n$)	$n = 0$	$n = 2$	$n = 10$	$n = 8$
Touch sensor ($\mathbb{R}^n$)	$n = 0$	$n = 4$	$n = 2$	$n = 8$

Control Space For all the experiments, the control space of all robots are continuous, and linearly scaled to [-1, +1].

C.2. Policy Settings

The hyper-parameters used in our experiments are listed in Table 4 as default.

Our experiments use separate multi-layer perception with $tanh$ activations for the policy network, value network and cost network. Each network consists of two hidden layers of size (64,64). All of the networks are trained using $Adam$ optimizer with learning rate of 0.01.

We apply an on-policy framework in our experiments. During each epoch the agent interact B times with the environment and then perform a policy update based on the experience collected from the current epoch. The maximum length of the trajectory is set to 1000 and the total epoch number N is set to 200 as default. In our experiments the Walker was trained for 250 epochs due to the high dimension.

The policy update step is based on the scheme of TRPO, which performs up to 100 steps of backtracking with a coefficient of 0.8 for line searching.

Table 3: The external state space components of different test suites environments.

External State Space	Goal-Hazard	Goal-Pillar
Goal Compass ($\mathbb{R}^3$)	✓	✓
Goal Lidar ($\mathbb{R}^{16}$)	✓	✓
3D Goal Lidar ($\mathbb{R}^{60}$)	✗	✗
Hazard Lidar ($\mathbb{R}^{16}$)	✓	✗
3D Hazard Lidar ($\mathbb{R}^{60}$)	✗	✗
Pillar Lidar ($\mathbb{R}^{16}$)	✗	✓
Vase Lidar ($\mathbb{R}^{16}$)	✓	✓
Gremlin Lidar ($\mathbb{R}^{16}$)	✓	✓

For all experiments, we use a discount factor of $\gamma = 0.99$, an advantage discount factor $\lambda = 0.95$, and a KL-divergence step size of $\delta_{KL} = 0.02$.

For experiments which consider cost constraints we adopt a target cost $\delta_c = 0.0$ to pursue a zero-violation policy.

Other unique hyper-parameters for each algorithms are hand-tuned to attain reasonable performance.

Each model is trained on a server with a 48-core Intel(R) Xeon(R) Silver 4214 CPU @ 2.2.GHz, Nvidia RTX A4000 GPU with 16GB memory, and Ubuntu 20.04.

C.3. Practical Implementation Details

In this section, we summarize implementation techniques that helps with S-3PO's practical performance. The pseudocode of S-3PO is give as algorithm 1.

Weighted loss for cost value targets A critical step in S-3PO requires fitting of the cost increment value functions, denoted as $V^{\pi}_{D}(\hat{s}_t)$. By definition, $V^{\pi}_{D}(\hat{s}_t)$ is equal to the maximum cost increment in any future state over the maximal state-wise cost so far. In other words, $V^{\pi}_{D}(\hat{s}_t)$ forms a non-increasing stair shape along the trajectory. Here we visualize an example of $V^{\pi}_{D}(\hat{s}_t)$ in Figure 10. To enhance the accuracy of fitting this stair shape function, a weighted loss strategy is adopted, capitalizing on its monotonic property. Specifically, we define a weighted loss L_{weight}:

$$L_{weight} = L(\hat{y}_t - y_t) * (1 + w * \mathbb{1}[(\hat{y}_t - y_{t-1}) > 0])$$

where L denotes Mean Squared Error (MSE), $\hat{y}_t$ is the prediction, y_t is the fitting target and w is the penalty weight. To account for the initial step ($t = 0$), we set y_{t-1} to sufficiently large, thereby disregarding the weighted term associated with the first step. In essence, the rationale is to penalize any prediction that violates the non-increasing characteristics of the target sequence, thereby leading to an improved fitting quality.

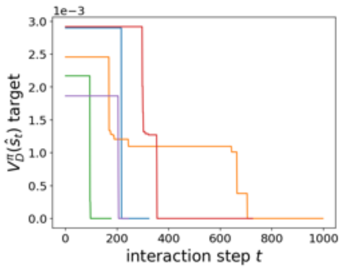


Figure 10: $V_{[D_i]\pi}(\hat{s})$ target of five sampled episodes.

Table 4: Important hyper-parameters of different algorithms in our experiments

Policy Parameter		TRPO	TRPO-ISSA	TRPO-Lagrangian	CPO	PCPO	SCPO	S-3PO
Epochs	N	200	200	200	200	200	200	200
Steps per epoch	B	30000	30000	30000	30000	30000	30000	30000
Maximum length of trajectory	L	1000	1000	1000	1000	1000	1000	
Policy network hidden layers		(64, 64)	(64, 64)	(64, 64)	(64, 64)	(64, 64)	(64, 64)	(64, 64)
Discount factor	γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Advantage discount factor	λ	0.97	0.97	0.97	0.97	0.97	0.97	0.97
TRPO backtracking steps		100	100	100	100	-	100	100
TRPO backtracking coefficient		0.8	0.8	0.8	0.8	-	0.8	0.8
Target KL	δ_{KL}	0.02	0.02	0.02	0.02	0.02	0.02	0.02
Value network hidden layers		(64, 64)	(64, 64)	(64, 64)	(64, 64)	(64, 64)	(64, 64)	(64, 64)
Value network iteration		80	80	80	80	80	80	80
Value network optimizer		Adam	Adam	Adam	Adam	Adam	Adam	Adam
Value learning rate		0.001	0.001	0.001	0.001	0.001	0.001	0.001
Cost network hidden layers		-	-	(64, 64)	(64, 64)	(64, 64)	(64, 64)	(64, 64)
Cost network iteration		-	-	80	80	80	80	80
Cost network optimizer		-	-	Adam	Adam	Adam	Adam	Adam
Cost learning rate		-	-	0.001	0.001	0.001	0.001	0.001
Target Cost	δ_c	-	-	0.0	0.0	0.0	0.0	0.0
Lagrangian learning rate		-	-	0.005	-	-	-	-
Cost reduction		-	-	-	0.0	-	0.0	0.0

Line search scheduling Note that in (10), there are two constraints: (a) the trust region and (b) the bound on expected advantages. In practice, due to approximation errors, constraints in (10) might become infeasible. In that case, we perform a recovery update that only enforces the cost advantage A^{π}_{D} to decrease starting from early training steps (in first k_{safe} updates), and starts to enforce reward improvements of A^{π} towards the end of training. This is different from Zhao et al. (2023b), where the reward improvements are enforced at all times. This is because SCPO only guarantees safety (constraint satisfaction) after convergence, while S-3PO prioritizes constraining imaginary safety violation. With our line search scheduling, S-3PO is able to first grasp a safe policy, and then improve the reward performance. In that way, S-3PO achieves zero safety violation both during training and in testing with a worst-case performance degradation guarantee.

C.4. Metrics Comparison

In Tables 5 to 7, we report all the 9 results of our test suites by three metrics:

- The average episode return J_r.
- The average episodic sum of costs M_c.
- The average cost over the entirety of training ρ_c.

Both the evaluation performance and training performance are reported based on the above metrics. Besides, we also report the ISSA triger times as ISSA performance of TRPO-ISSA and S-3PO. All of the metrics were obtained from the final epoch after convergence. Each metric was averaged over two random seed. The evaluation performance curves of all experiments are shown in Figures 11, 14 and 17, the training performance curves of all experiments are shown in Figures 12, 15 and 18 and the ISSA performance curves of all experiments are shown in Figures 13, 16 and 19

Table 5: Metrics of three **Point_Hazard** environments obtained from the final epoch.

Point_1Hazard

Algorithm	Evaluation Performance			Training Performance			ISSA Performance
	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	
TRPO	2.5738	0.5078	0.0082	2.5738	0.5078	0.0082	-
TRPO-Lagrangian	**2.6313**	0.5977	0.0058	**2.6313**	0.5977	0.0058	-
CPO	2.4988	0.1713	0.0045	2.4988	0.1713	0.0045	-
PCPO	2.4928	0.3765	0.0054	2.4928	0.3765	0.0054	-
SCPO	2.5457	0.0326	0.0022	2.5457	0.0326	0.0022	-
TRPO-ISSA	2.5113	0.0000	0.0000	2.5981	0.0000	0.0000	0.2714
S-3PO	2.4157	**0.0000**	**0.0000**	2.2878	**0.0000**	**0.0000**	**0.0285**

Point_4Hazard

Algorithm	Evaluation Performance			Training Performance			ISSA Performance
	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	
TRPO	**2.6098**	0.2619	0.0037	**2.6098**	0.2619	0.0037	-
TRPO-Lagrangian	2.5494	0.2108	0.0034	2.5494	0.2108	0.0034	-
CPO	2.5924	0.1654	0.0024	2.5924	0.1654	0.0024	-
PCPO	2.5575	0.1824	0.0025	2.5575	0.1824	0.0025	-
SCPO	2.5535	0.0523	0.0009	2.5535	0.0523	0.0009	-
TRPO-ISSA	2.5014	0.0712	0.0000	2.5977	0.0135	0.0000	0.1781
S-3PO	2.3868	**0.0000**	**0.0000**	2.3550	**0.0000**	**0.0000**	**0.0117**

Point_8Hazard

Algorithm	Evaluation Performance			Training Performance			ISSA Performance
	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	
TRPO	2.5535	0.5208	0.0074	2.5535	0.5208	0.0074	-
TRPO-Lagrangian	2.5851	0.5119	0.0064	2.5851	0.5119	0.0064	-
CPO	**2.6440**	0.2944	0.0041	**2.6440**	0.2944	0.0041	-
PCPO	2.6249	0.3843	0.0052	2.6249	0.3843	0.0052	-
SCPO	2.5126	**0.0703**	0.0020	2.5126	0.0703	0.0020	-
TRPO-ISSA	2.5862	0.0865	0.0000	2.5800	0.0152	0.0000	0.3431
S-3PO	2.4207	0.1710	**0.0000**	2.3323	**0.0000**	**0.0000**	**0.0337**

Table 6: Metrics of three **Pillar_Hazard** environments obtained from the final epoch.

Point_1Pillar							
Algorithm	Evaluation Performance			Training Performance			ISSA Performance
	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	
TRPO	**2.6065**	0.2414	0.0032	**2.6065**	0.2414	0.0032	-
TRPO-Lagrangian	2.5772	0.1218	0.0020	2.5772	0.1218	0.0020	-
CPO	2.5464	0.2342	0.0028	2.5464	0.2342	0.0028	-
PCPO	2.5857	0.2088	0.0025	2.5857	0.2088	0.0025	-
SCPO	2.5928	0.0040	0.0003	2.5928	0.0040	0.0003	-
TRPO-ISSA	2.5985	0.0000	0.0000	2.5909	0.0020	0.0000	0.3169
S-3PO	2.5551	**0.0000**	**0.0000**	2.5241	**0.0000**	**0.0000**	**0.0060**

Point_4Pillar							
Algorithm	Evaluation Performance			Training Performance			ISSA Performance
	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	
TRPO	2.5671	0.4112	0.0063	2.5671	0.4112	0.0063	-
TRPO-Lagrangian	**2.6040**	0.2786	0.0050	**2.6040**	0.2786	0.0050	-
CPO	2.5720	0.5523	0.0062	2.5720	0.5523	0.0062	-
PCPO	2.5709	0.3240	0.0052	2.5709	0.3240	0.0052	-
SCPO	2.5367	**0.0064**	0.0005	2.5367	0.0064	0.0005	-
TRPO-ISSA	2.5739	0.1198	0.0001	2.5881	0.0427	0.0001	0.2039
S-3PO	2.2513	0.0114	**0.0000**	2.3459	**0.0000**	**0.0000**	**0.0116**

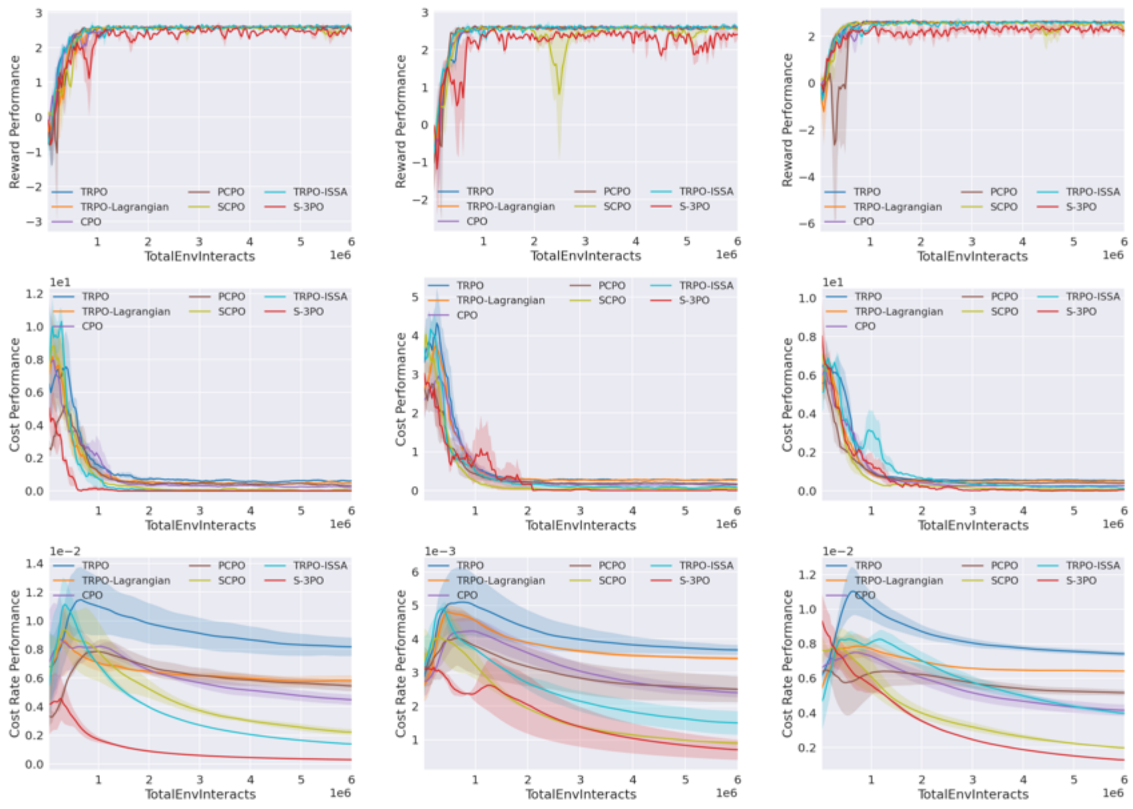
Point_8Pillar							
Algorithm	Evaluation Performance			Training Performance			ISSA Performance
	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	
TRPO	2.6140	3.1552	0.0201	2.6140	3.1552	0.0201	-
TRPO-Lagrangian	2.6164	0.6632	0.0129	2.6164	0.6632	0.0129	-
CPO	**2.6440**	0.5655	0.0166	**2.6440**	0.5655	0.0166	-
PCPO	2.5704	6.6251	0.0219	2.5704	6.6251	0.0219	-
SCPO	2.4162	0.2589	0.0024	2.4162	0.2589	0.0024	-
TRPO-ISSA	2.6203	0.6910	0.0009	2.5921	0.0709	0.0009	0.3517
S-3PO	2.0325	**0.0147**	**0.0002**	2.3371	**0.0000**	**0.0002**	**0.0231**

Table 7: Metrics of three link robots environments obtained from the final epoch.

Swimmer_1Hazard							
Algorithm	Evaluation Performance			Training Performance			ISSA Performance
	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	
TRPO	**2.7049**	0.3840	0.0076	**2.7049**	0.3840	0.0076	-
TRPO-Lagrangian	2.6154	0.3739	0.0060	2.6154	0.3739	0.0060	-
CPO	2.5817	0.3052	0.0056	2.5817	0.3052	0.0056	-
PCPO	2.5418	0.6243	0.0055	2.5418	0.6243	0.0055	-
SCPO	2.6432	0.3919	0.0051	2.6432	0.3919	0.0051	-
TRPO-ISSA	2.5826	0.2595	0.0000	2.5955	**0.0000**	0.0000	0.1240
S-3PO	2.6032	**0.0313**	**0.0000**	2.6239	0.0001	**0.0000**	**0.0378**

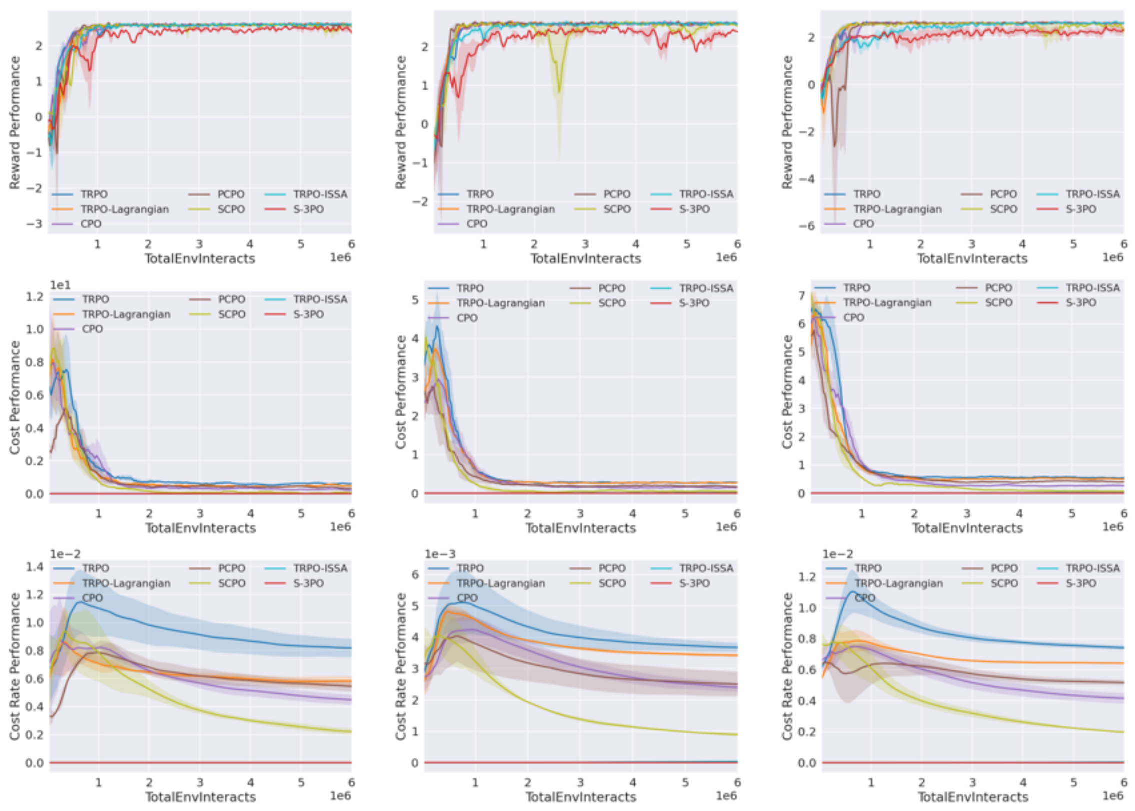
Ant_1Hazard							
Algorithm	Evaluation Performance			Training Performance			ISSA Performance
	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	
TRPO	2.6390	0.3559	0.0074	**2.6390**	0.3559	0.0074	-
TRPO-Lagrangian	2.5866	0.2169	0.0044	2.5866	0.2169	0.0044	-
CPO	2.6175	0.2737	0.0072	2.6175	0.2737	0.0072	-
PCPO	2.6103	0.2289	0.0076	2.6103	0.2289	0.0076	-
SCPO	2.6341	0.2384	0.0065	2.6341	0.2384	0.0065	-
TRPO-ISSA	**2.6509**	0.3831	0.0032	2.6318	0.3516	0.0032	0.0279
S-3PO	2.2047	**0.0000**	**0.0002**	2.2031	**0.0000**	**0.0002**	**0.0001**

Walker_1Hazard							
Algorithm	Evaluation Performance			Training Performance			ISSA Performance
	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$	
TRPO	2.5812	0.2395	0.0075	2.5812	0.2395	0.0075	-
TRPO-Lagrangian	2.6227	0.1666	0.0041	2.6227	0.1666	0.0041	-
CPO	2.6035	0.3068	0.0062	2.6035	0.3068	0.0062	-
PCPO	2.5775	0.2414	0.0059	2.5775	0.2414	0.0059	-
SCPO	2.6352	**0.1423**	0.0051	**2.6352**	**0.1423**	0.0051	-
TRPO-ISSA	**2.6419**	0.3544	0.0037	2.5787	0.2060	0.0037	0.0316
S-3PO	2.6117	0.3437	**0.0025**	2.6055	0.2665	**0.0025**	**0.0319**



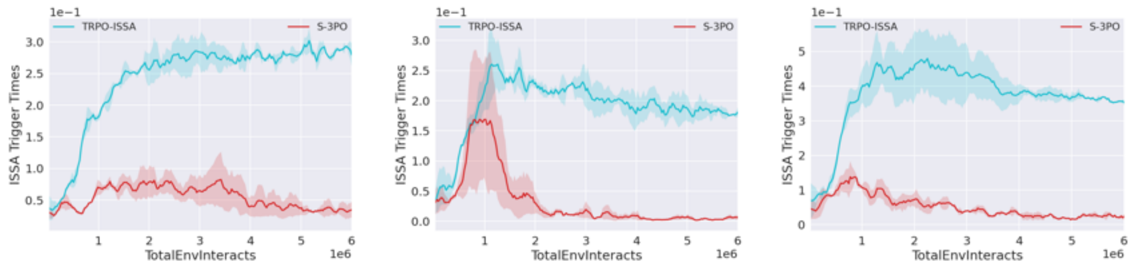
(*a*) Point_1Hazard (*b*) Point_4Hazard (*c*) Point_8Hazard

Figure 11: Evaluation performance of Point_Hazard



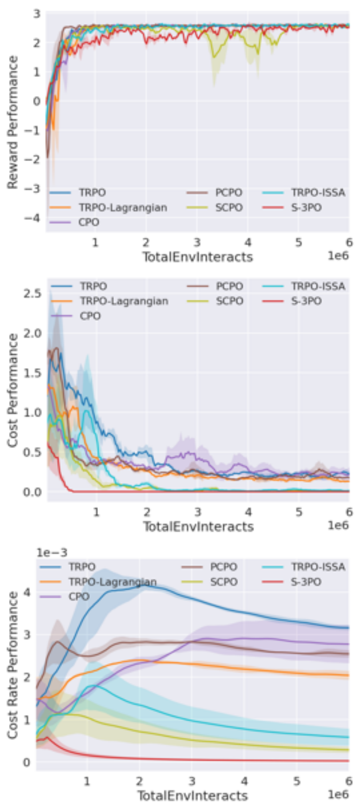
(*a*) Point_1Hazard (*b*) Point_4Hazard (*c*) Point_8Hazard

Figure 12: Training performance of Point_Hazard

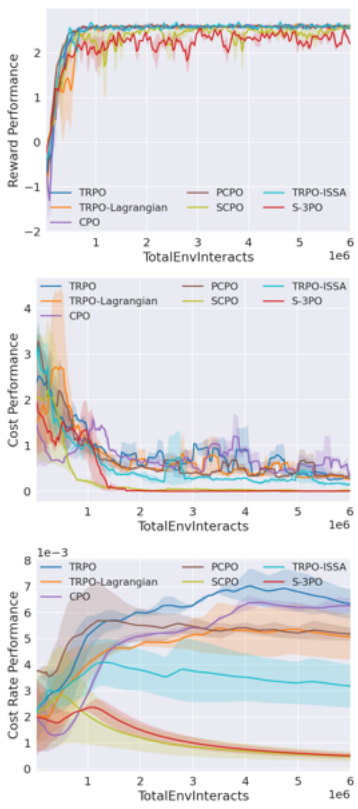


(*a*) Point_1Hazard (*b*) Point_4Hazard (*c*) Point_8Hazard

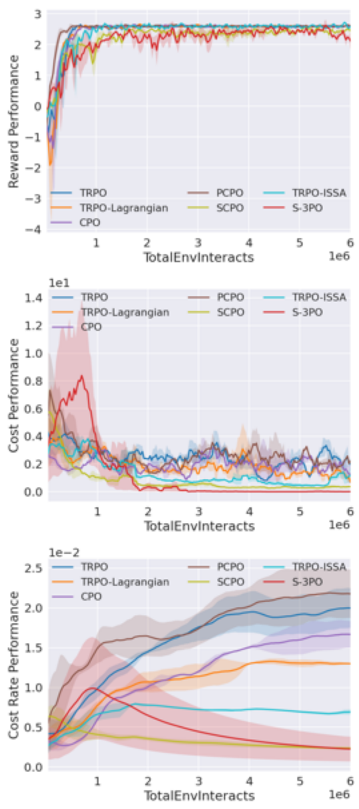
Figure 13: ISSA performance of Point_Hazard



(*a*) Point_1Pillar

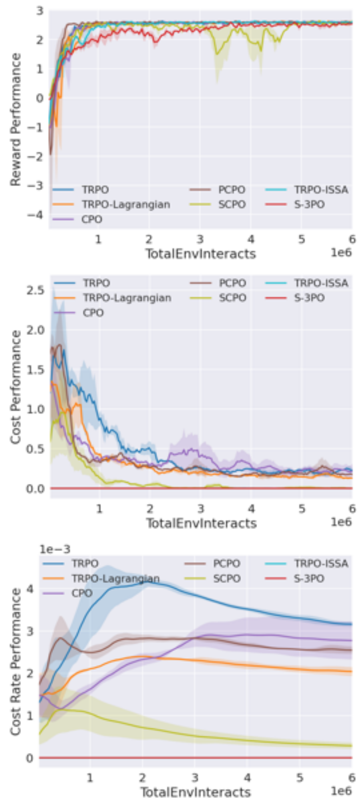


(*b*) Point_4Pillar

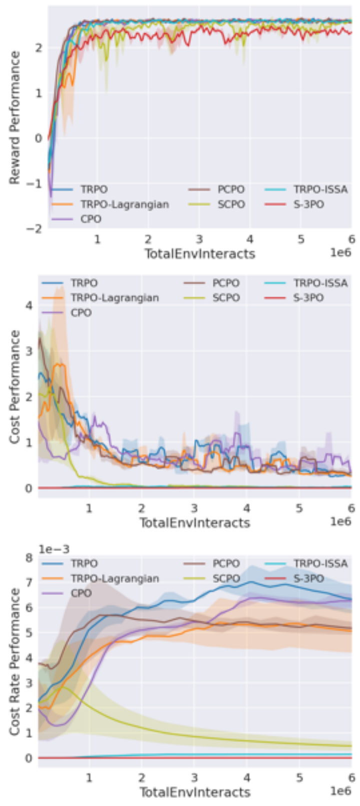


(*c*) Point_8Pillar

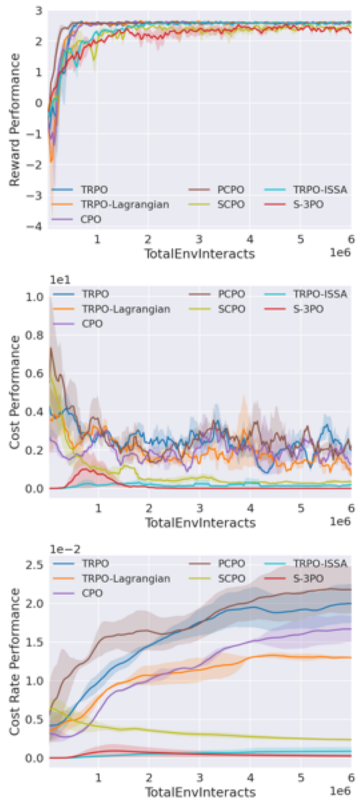
Figure 14: Evaluation performance of Point_Pillar



(*a*) Point_1Pillar

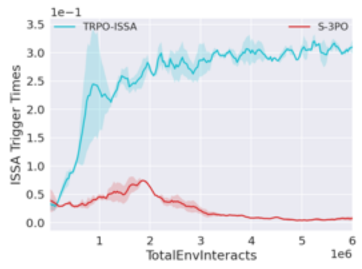


(*b*) Point_4Pillar

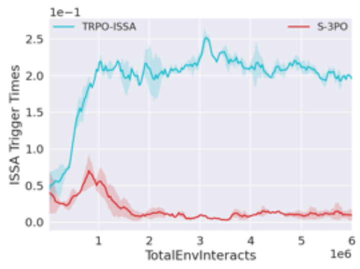


(*c*) Point_8Pillar

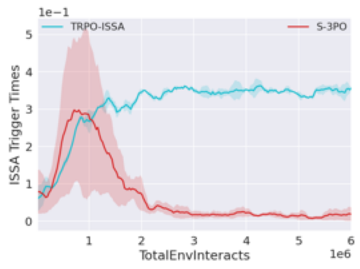
Figure 15: Training performance of Point_Pillar



(*a*) Point_1Pillar

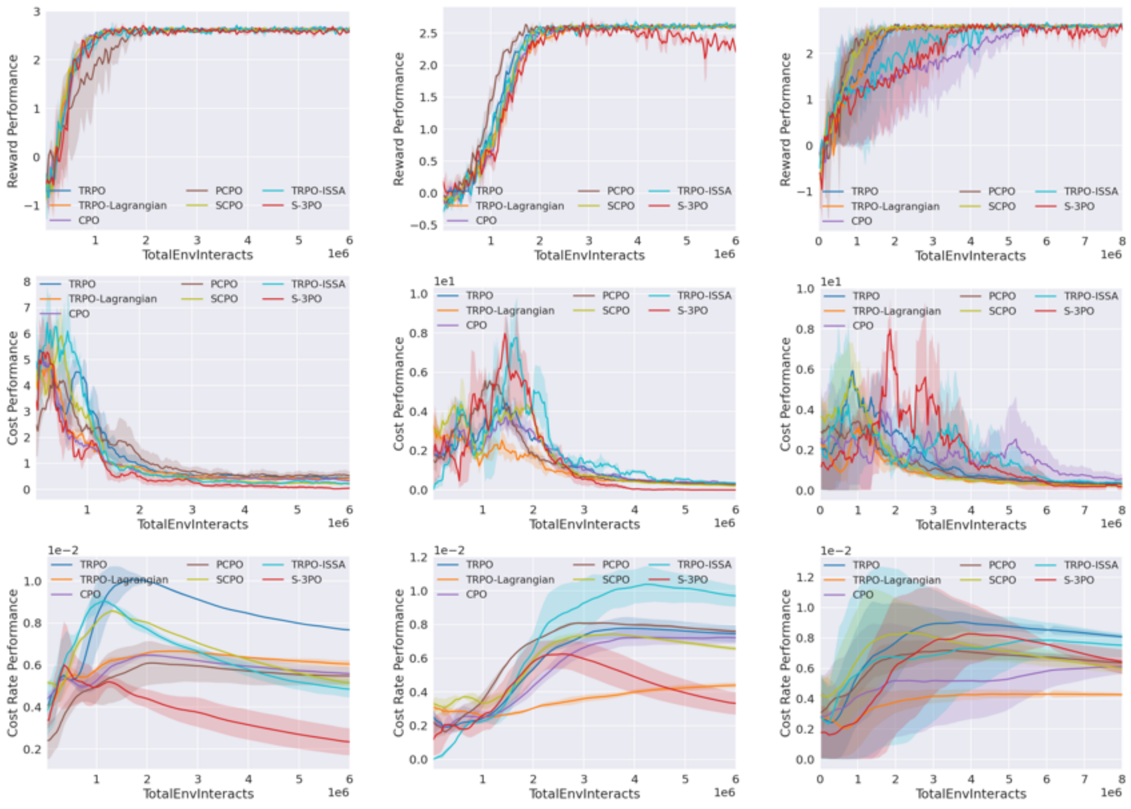


(*b*) Point_4Pillar



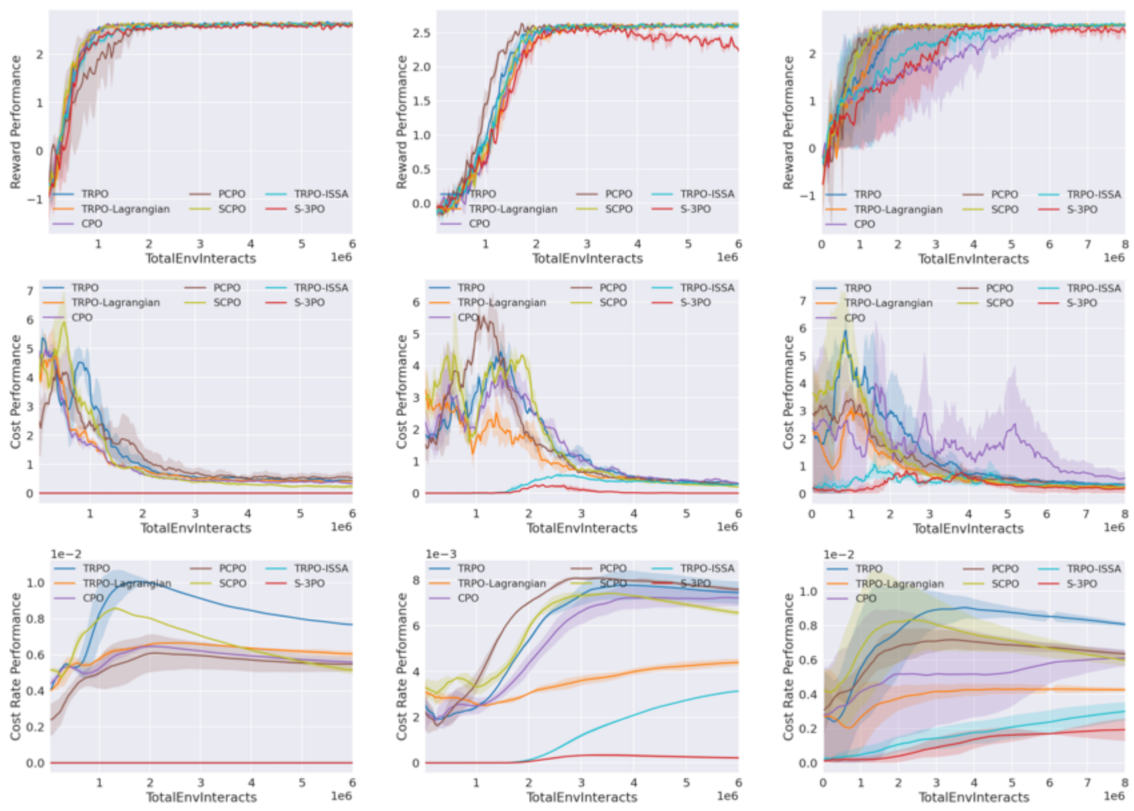
(*c*) Point_8Pillar

Figure 16: ISSA performance of Point_Pillar



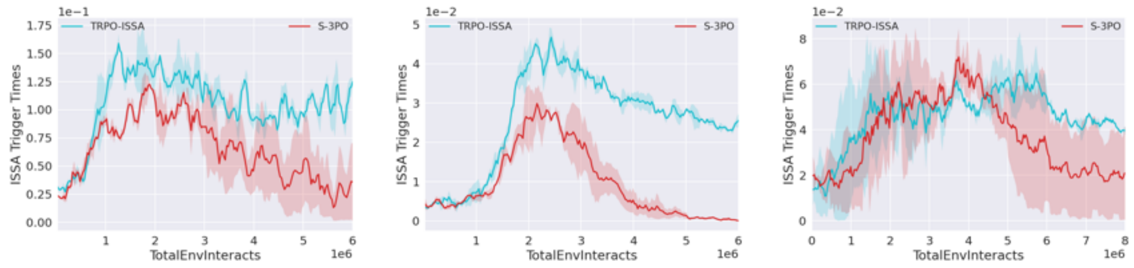
(*a*) Swimmer_1Hazard (*b*) Ant_1Hazard (*c*) Walker_1Hazard

Figure 17: Evaluation performance of link robots



(a) Swimmer_1Hazard (b) Ant_1Hazard (c) Walker_1Hazard

Figure 18: Training performance of link robots



(a) Swimmer_1Hazard (b) Ant_1Hazard (c) Walker_1Hazard

Figure 19: ISSA performance of link robots