

Singapore Management University

## Institutional Knowledge at Singapore Management University

---

Research Collection School Of Computing and Information Systems

School of Computing and Information Systems

---

2-2024

### Imitate the good and avoid the bad: An incremental approach to safe reinforcement learning

Minh Huy HOANG

Mai Anh TIEN

Pradeep VARAKANTHAM

Singapore Management University, pradeepv@smu.edu.sg

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)



Part of the [Databases and Information Systems Commons](#), and the [Theory and Algorithms Commons](#)

---

#### Citation

HOANG, Minh Huy; TIEN, Mai Anh; and VARAKANTHAM, Pradeep. Imitate the good and avoid the bad: An incremental approach to safe reinforcement learning. (2024). *Proceedings of the 38th Annual AAAI Conference on Artificial Intelligence, Vancouver, Canada, 2024 February 20-27*.

Available at: [https://ink.library.smu.edu.sg/sis\\_research/8594](https://ink.library.smu.edu.sg/sis_research/8594)

This Conference Proceeding Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [cherylids@smu.edu.sg](mailto:cherylids@smu.edu.sg).

# Imitate the Good and Avoid the Bad: An Incremental Approach to Safe Reinforcement Learning

Huy Hoang, Tien Mai, Pradeep Varakantham

School of Computing and Information Systems  
Singapore Management University, Singapore  
{mhhoang, atmai, pradeepv}@smu.edu.sg

## Abstract

A popular framework for enforcing safe actions in Reinforcement Learning (RL) is Constrained RL, where *trajectory* based constraints on expected cost (or other cost measures) are employed to enforce safety and more importantly these constraints are enforced while maximizing expected reward. Most recent approaches for solving Constrained RL convert the trajectory based cost constraint into a surrogate problem that can be solved using minor modifications to RL methods. A key drawback with such approaches is an over or under-estimation of the cost constraint at each state. Therefore, we provide an approach that does not modify the trajectory based cost constraint and instead imitates “good” trajectories and avoids “bad” trajectories generated from incrementally improving policies. We employ an oracle that utilizes a reward threshold (which is varied with learning) and the overall cost constraint to label trajectories as “good” or “bad”. A key advantage of our approach is that we are able to work from any starting policy or set of trajectories and improve on it. In an exhaustive set of experiments, we demonstrate that our approach is able to outperform top benchmark approaches for solving Constrained RL problems, with respect to expected cost, CVaR cost, or even unknown cost constraints.

## 1 Introduction

Reinforcement learning (RL) is widely acknowledged as a powerful paradigm, thanks to its exceptional ability to learn and adapt by interacting with the environment. This adaptability has been demonstrated through numerous studies that highlight its practical applications across diverse domains. For example, reinforcement learning has been successfully employed in video games to achieve groundbreaking results (Mnih et al. 2016; Firoiu, Whitney, and Tenenbaum 2017), robot manipulation tasks have been enhanced using this approach (Hoang, Dinh, and Nguyen 2023; Kilinc and Montana 2022), and even the field of healthcare has benefited from its potential (Weng et al. 2017; Raghu et al. 2017). In light of the notable achievements of reinforcement learning, it is crucial to acknowledge the practical limitations that come with this approach when applied to real-world situations. The constraints of limited resources, budgetary restrictions, and safety concerns pose significant challenges in implementing reinforcement learning effectively.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

**Constrained RL:** To address these challenges, Constrained Markov Decision Processes (CMDPs) have been developed as an extension of Markov Decision Processes (MDPs) (Altman 1999). CMDPs have emerged as a valuable framework for decision-making in various domains, as they allow for the optimization of objectives while ensuring the fulfillment of trajectory-based constraints over expected cost and other measures (e.g., CVaR). In order to tackle the challenges posed by these constraints, several Constrained RL algorithms have been proposed (Yang et al. 2022; Zhang, Vuong, and Ross 2020). State-of-the-art constrained RL approaches (Satija, Amortila, and Pineau 2020; Chow et al. 2019; Achiam et al. 2017) convert trajectory-based cost constraints into local cost constraints that can be solved easily while guaranteeing the enforcement of trajectory-based constraints. One potential issue with such local cost constraints in challenging constrained RL problems is the estimation of cost value functions. Due to the difficulty involved in estimating the costs of partial (or full) trajectories, output policies can either be conservative or aggressive with regard to costs.

In this work, we develop a novel principled framework that avoids the use of local cost constraints and, instead, focuses on directly solving the original constrained MDP problem, thereby avoiding cost estimation. Our innovation is rooted in the observation that, within the context of CMDP, from a given set of trajectories, it is easy to identify “good” trajectories that are feasible with respect to the cost constraints and offer high rewards. In contrast, “bad” trajectories would be identified as infeasible with respect to the cost constraints and/or yield low rewards. Subsequently, a policy that assigns high probabilities to good trajectories becomes a strong candidate for effectively addressing the CMDP problem. Hence, our approach to address CMDP involves learning a policy that replicates the actions of the good trajectories while steering clear of the bad ones. We do this by employing imitation learning, a framework designed to imitate an expert’s policy based on their demonstrations.

**Imitation Learning:** Imitation learning (IL) has been recognized as a compelling approach for making sequential decisions (Ng, Russell et al. 2000; Abbeel and Ng 2004). In IL, a set of expert trajectories is provided, and the aim is to train a policy that replicates the behavior of the expert’s policy. One of the simplest IL methods is Behavioral

Cloning (BC), which mimics an expert’s policy by maximizing the likelihood of the expert’s actions under the learned policy. BC is simple to implement but it disregards environmental dynamics, making it unable to perform as well as an expert in unseen states (Ross, Gordon, and Bagnell 2011). To address this issue, Generative Adversarial Imitation Learning (Ho and Ermon 2016) and Adversarial Inverse Reinforcement Learning (Fu, Luo, and Levine 2017) were introduced. These methods use adversarial training to make the agent’s behavior match the expert’s occupancy distribution as estimated by their discriminator. However, the adversarial training often hinders the agent from achieving expert-level performance, especially in continuous settings. ValueDICE (Kostrikov, Nachum, and Tompson 2019) learns a value function based on the KL divergence of the learner and expert occupancy distributions and performs well in offline settings while still incorporating adversarial training. More recent methods like PWIL (Dadashi et al. 2020) and IQ-learn (Garg et al. 2021) use different statistical distances for occupancy distribution and successfully eliminate the need for adversarial training.

It’s important to note that imitation within our context differs from the conventional IL approaches from the aforementioned works. Here, our approach not only involves mimicking the behavior of “good” demonstrations but also actively avoiding the bad ones. To the best of our knowledge, this marks the first time the concept of learning to avoid bad demonstrations is introduced within the realm of IL. Additionally, in a standard IL algorithm, the set of expert demonstrations is fixed beforehand. In contrast, in our context, the set of demonstrations is generated by a pre-trained or learning policy, thus allowing it to expand as training progresses. These factors collectively pave the way for the development of a novel IL algorithm that is well-suited to our specific context.

**Contributions:** We make the following contributions:

- *New framework for Constrained RL:* We propose a novel training framework for Constrained RL that incrementally improves an agent policy by imitating “good” trajectories and avoiding “bad” trajectories. The sets of “good” and “bad” trajectories are selected based on their accumulated rewards and costs and are updated as the policy is improved.
- *Theoretical insights:* We show that our way of imitating the good trajectories and avoiding the bad ones can be shown to ensure no deterioration in the output policy performance.
- *New Learning algorithm:* We develop a non-adversarial *imitate and avoid* algorithm that is able to imitate “good” trajectories and avoid “bad” trajectories. Due to the non-adversarial nature of the algorithm, it provides higher stability while being scalable.
- *Experimental results:* We provide an extensive experimental results section, where we demonstrate that our approach outperforms existing best approaches on all six different environments<sup>1</sup> within the highly challeng-

<sup>1</sup>Existing works have typically showed results only on the sim-

ing Safety-Gym benchmark. Furthermore, we also provide results for expected cost, CVaR cost, and *unknown cost* settings.

## 2 Background

We present a description of the Constrained MDP problem and some popular IL approaches.

### 2.1 Constrained Markov Decision Process

The Markov Decision Process (MDP) described in (Altman 1999) can be represented as  $\mathcal{M} = \langle S, A, r, P, s_0 \rangle$ . Here,  $S$  denotes the set of states,  $s_0$  represents the initial state set,  $A$  is the set of actions,  $r : S \times A \rightarrow \mathbb{R}$  defines the reward function for each state-action pair, and  $P : S \times A \rightarrow S$  is the transition function.

By introducing an additional constraint set  $\mathcal{C} = \langle d, c_{\max} \rangle$  to the MDP, we can formulate a Constrained Markov Decision Process (CMDP). The constraint set includes a cost function  $d : S \rightarrow \mathbb{R}$  and a maximum allowed accumulated cost  $c_{\max}$ . The objective of the CMDP is to maximize the return while ensuring that the expected accumulated cost remains below the specified maximum. Mathematically, the objective function and constraint can be expressed as follows:

$$\begin{aligned} \max_{\pi} \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0, \pi \right] \\ \text{s.t.} \quad \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t d(s_t) | s_0, \pi \right] \leq c_{\max}. \end{aligned} \tag{CMDP}$$

where  $\pi$  represents a policy,  $\gamma$  is the discount factor, and the expectation is taken with respect to the initial state and the policy. From now, to simplify the notion, we define  $R(\tau)$  and  $C(\tau)$  be the expectation of return and accumulated cost on trajectories  $\tau$ , i.e.,  $R(\tau) = \sum_{(s_t, a_t) \in \tau} \gamma^t r(s_t, a_t)$ ,  $C(\tau) = \sum_{s_t \in \tau} \gamma^t d(s_t)$ .

### 2.2 Imitation Learning

**Behavioral Cloning.** In BC, the objective is to maximize the likelihood of the demonstrations.

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi^E} \left[ \sum_{(s,a) \in \tau} \ln(\pi(a|s)) \right] \tag{BC}$$

BC has a strong theoretical foundation, as it can guarantee to return the exact expert policy  $\pi^E$ , i.e.,  $\pi^E$  is the unique optimal solution to BC. However, BC ignores environmental dynamics and only works with offline learning, so it often requires a huge number of samples to achieve a desired performance (Ross, Gordon, and Bagnell 2011).

**Distribution matching.** A popular and useful approach for IL is based on state-action distribution matching. Specifically, let  $\rho^{\pi}(s, a)$  be the occupancy measure of visiting state  $s$  and taking action  $a$ , under policy  $\pi$ . Let  $\rho^{\pi^E}$  the state-action distribution given by expert policy  $\pi^E$ . The distribution matching approach proposes to learn  $\pi$  to minimize the

plest environment, Safety Point Goal-v0.

discrepancy between  $\rho^\pi$  and  $\rho^{\pi^E}$  such as KL-divergence:

$$\min_{\pi} KL\left(\rho^\pi \parallel \rho^{\pi^E}\right) = \min_{\pi} \left\{ \mathbb{E}_{(s,a) \sim \rho^\pi} \left[ \ln \frac{\rho^{\pi^E}(s,a)}{\rho^\pi(s,a)} \right] \right\} \quad (1)$$

Approaches based on distributional matching include some state-of-the-art IL algorithms such as adversarial IL (Ho and Ermon 2016; Fu, Luo, and Levine 2017) or IQ-learn (Garg et al. 2021).

### 3 Self-Imitation Learning Approach

Before describing our learning approach, we define ‘‘Good’’ and ‘‘Bad’’ trajectories:

**Definition 1.** A trajectory,  $\tau$  is a good trajectory if:  $R(\tau) \geq R_G$  and  $C(\tau) \leq c_{max}$ . On the other, a trajectory,  $\tau$  is a bad trajectory if  $R(\tau) < R_B$  or  $C(\tau) > c_{max}$ .

Here,  $R_G$  and  $R_B$  represent some predefined<sup>2</sup> thresholds for selecting good and bad trajectories, respectively. We denote  $\Omega^G$  and  $\Omega^B$  as the set of good and bad trajectories respectively.

#### 3.1 Learning from Good and Bad Demonstrations

Our aim is to train an RL agent to imitate the good behavior from a set of good demonstrations (trajectories) and avoid the bad demonstrations. In other words, we try to mimic the good part of the pre-trained policy and avoid the bad part.

**Behavior Cloning GB:** When using a Behavior Cloning, BC type approach to achieve the above objective, the aim is to maximize the likelihood of the good set while minimizing the likelihood of the bad one. The training objective can be written as:

$$\max_{\pi} \left\{ \lambda \mathbb{E}_{\substack{\tau \sim \pi^0 \\ \tau \in \Omega^G}} \left[ \sum_{(s,a) \in \tau} \ln(\pi(a|s)) \right] - (1 - \lambda) \mathbb{E}_{\substack{\tau \sim \pi^0 \\ \tau \in \Omega^B}} \left[ \sum_{(s,a) \in \tau} \phi(\ln(\pi(a|s))) \right] \right\} \quad (\text{BC-GB})$$

where  $\phi(\cdot)$  is a monotone regularizer mapping  $(-\infty, 0)$  to a finite interval, and  $\lambda \in [0, 1]$  is a parameter capturing the impact of each good or bad set on the objective function, and  $\pi^0$  is a starting policy that we want to improve upon. We use  $\pi^0$  instead of  $\pi^E$  as the starting policy is not necessarily an expert one. If  $\lambda = 1$ , then we only learn from good demonstrations and ignore bad ones, and  $\lambda = 0$  otherwise. We incorporate the regularization term  $\phi(\cdot)$  in this context to address a critical concern. Without this regularization, the maximization process could drive the value of  $\ln(\pi(a|s))$  in the second term of equation (BC-GB) towards negative infinity, leading to an unbounded and numerically unstable objective. Intuitively, to improve the objective in (BC-GB), it is necessary for the policy to allocate higher probabilities to trajectories in the good set while assigning lower probabilities to trajectories in the bad set.

<sup>2</sup>We provide an in-depth analysis on the selection of these hyperparameters and changing them during the learning for certain problems.

**Distribution Matching GB:** In the realm of distribution matching, the learning process entails a delicate balance. It involves minimizing the Kullback-Leibler (KL) divergence between the occupancy measures of the policy under consideration, denoted as  $\rho^\pi$ , and the good trajectories, represented as  $\rho^G$ . Simultaneously, the goal is to maximize the KL divergence between  $\rho^\pi$  and the occupancy measure corresponding to bad trajectories, denoted as  $\rho^B$ . This dual divergence approach aims to shape the policy by aligning it closely with the good trajectories while also distancing it from the bad ones. These ‘‘good’’ and ‘‘bad’’ occupancy measures can be computed as:  $\rho^G(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t p_t(s, a | \Omega^G)$ , where  $p_t(s, a | \Omega^G)$  is the probability that  $(s_t, a_t)$  is in the set  $\Omega^G$  and  $(s_t, a_t) = (s, a)$ . Similarly,  $\rho^B(s, a)$  can be computed in the same way. Then, the training objective becomes

$$\min_{\pi} \left\{ \lambda \text{KL}(\rho^\pi \parallel \rho^G) - (1 - \lambda) \text{KL}(\rho^\pi \parallel \rho^B) \right\} \quad (\text{DM-GB})$$

Intuitively, to minimize the objective function in (DM-GB), it is necessary for the occupancy distribution to move towards  $\rho^G$  and far away from  $\rho^B$ . Consequently,  $\rho^\pi$  will allocate a higher probability to a pair  $(s, a)$  that appears more frequently in  $\Omega^G$  than in  $\Omega^B$ , and vice-versa.

#### 3.2 Theoretical Insights

We investigate the theoretical properties of our concept of learning from good and bad demonstrations. Our aim is to explore the question whether we can obtain improved policies by learning from good and bad demonstrations. Since BC-GB works directly with trajectories, we will employ it to present our theory on why intuitively using good and bad trajectories is useful. Distribution Matching GB, on the other hand, works with state-action pairs, thus is much more challenging to analyze theoretically. That is why we develop our algorithm based on Distribution Matching GB, and show extensive empirical results with it in our experimental results to demonstrate that it is a more practical algorithm and it outperforms existing work.

We first note that, in the context of maximum likelihood estimation,  $\pi^E$  is optimal for (BC). In other words, if we have sufficient samples from the expert policy, it is guaranteed that we can recover the expert policy by solving (BC).

We now look at the BC with good and bad trajectories in (BC-GB). The following lemma says that a policy that allocates zero probabilities to bad trajectories in  $\Omega^B$  will be optimal for (BC-GB).

**Lemma 1.** For any  $\lambda > 0$ , if there exists a policy  $\pi^*$  such that  $P_{\pi^*}(\tau) = 0$  for all  $\tau \in \Omega^B$ , and

$$P_{\pi^*}(\tau) = \frac{P_{\pi^0}(\tau)}{\sum_{\tau' \in \Omega^G} P_{\pi^0}(\tau')}; \quad \forall \tau \in \Omega^G$$

then  $\pi^*$  is an optimal policy to (BC-GB).

where  $P_{\pi^*}(\tau)$  is the probability of  $\tau$  given by  $\pi^*$ , i.e.,  $P_{\pi^*}(\tau) = \sum_{(s_t, a_t, s_{t+1}) \in \tau} \pi^*(a_t | s_t) P(s_{t+1} | a_t, s_t)$ . There might be no policy that allocates exactly  $P_{\pi}(\tau) = 0$  for all  $\tau \in \Omega^B$ , due to, for instance, the dynamic of the environment or the structure of  $\Omega^B$ . However, intuitively, a policy trying to assign small probabilities to  $(s, a)$  that appear more

frequently in  $\Omega^B$  than in  $\Omega^G$  will move towards  $\pi^*$  (so closer to the optimal policy).

In the proposition below we show that, if we construct a bad set consisting of trajectories having low reward values and violating the cost constraints, then it is guaranteed that the optimal policy mentioned in Lemma 1 will perform better than the initial policy  $\pi^0$  in terms of both reward and cost constraint satisfaction.

**Proposition 1.** *For any  $\lambda > 0$ , let  $\pi^0$  be a pre-trained feasible policy,  $\mathbf{R}^E = \mathbb{E}_{\tau \sim \pi^0} [R(\tau)]$ , and  $\Omega^B$  be a collection of trajectories of low reward and high-cost values*

$$\Omega^B = \left\{ \tau \mid R(\tau) \leq \mathbf{R}^E, C(\tau) > c_{\max} \right\}$$

*the optimal policy mentioned in Lemma 1 is feasible to the cost constraint while offering a better expected reward than the pre-trained policy  $\pi^0$ , specifically,*

$$\mathbb{E}_{\pi^*} [R(\tau)] - \mathbb{E}_{\pi^0} [R(\tau)] = \frac{\sum_{\tau} P_{\pi^0}(\tau)(\mathbf{R}^E - R(\tau))}{1 - P_{\pi^0}(\Omega^B)} \geq 0 \quad (2)$$

$$\mathbb{E}_{\tau \sim \pi^*} [C(\tau)] \leq c_{\max}$$

where  $P_{\pi^0}(\Omega^B) = \sum_{\tau \in \Omega^B} P_{\pi^0}(\tau)$ .

The inequality in (2) suggests that increasing the proportion or total probability of the bad set  $\Omega$  will result in a larger gap, thereby leading to improved policy enhancement. In other words, as more bad policies are identified, the quality of  $\pi^*$  improves.

The above results hold for  $\lambda > 0$ , also indicating that one might obtain a better policy by eliminating the probabilities of bad trajectories (trajectories with low reward and high-cost values). When  $\lambda = 0$ , the BC is about to learn only from the bad trajectories. Interestingly, we can show that by just learning from bad trajectories, it is not necessary to obtain a better policy. We first state the following lemma.

**Lemma 2.** *If  $\lambda = 0$ , any policy  $\pi^*$  such that  $P_{\pi^*}(\tau) = 0$  for all  $\tau \in \Omega^B$  is optimal for (BC-GB).*

The following proposition tells us that learning with  $\lambda = 0$  would not offer a policy improvement as in the case of  $\lambda > 0$ .

**Proposition 2.** *If  $\lambda = 0$ , and the bad set  $\Omega^B$  is selected in the same manner as in Proposition 1, then the optimal policy  $\pi^*$  from Lemma (2) is feasible, but it does not necessarily provide a higher expected reward than the policy  $\pi^0$ .*

In Propositions 1 and 2, it is assumed that the pre-trained policy  $\pi^0$  is feasible. It is then relevant to discuss other scenarios where the expected policy is not feasible, or even when the cost function is unknown. We summarize our claims below.

**Proposition 3.** *The following hold*

- (i) *If we select the bad set as  $\Omega^B = \left\{ \tau \mid R(\tau) \leq \mathbf{R}^E, C(\tau) > \mathbf{C}^E \right\}$ , then it is guaranteed that  $\pi^*$  offers a higher (or equal) expected reward and lower (or equal) expected cost, compared to those from  $\pi^0$ , where  $\mathbf{C}^E = \mathbb{E}_{\tau \sim \pi^0} [C(\tau)]$*

	$s_0$	$s_1$	$s_2$	$s_3$	$s_4$	$s_5$
$r$	0	2	3	1	2	0
$c$	0	1	1	3	1	0
$\pi^0$	1/2, 1/2	1/3, 1/3, 1/3	1	1	1	1

Table 1: Rewards, Costs and Policy

- (ii) *If the pre-trained policy  $\pi^0$  is not feasible, then if we select the bad set as  $\Omega^B = \left\{ \tau \mid C(\tau) > c_{\max} \right\}$ , then it is guaranteed that  $\pi^*$  is feasible*
- (iii) *If the cost function is not accessible, but there is an oracle that can tell us which trajectories are violating the constraint, then by selecting,  $\Omega^B = \left\{ \tau \mid \tau \text{ is violated} \right\}$ , then  $\pi^*$  is feasible.*

The above results provide some interesting insights to understand the framework. It is evidenced that if we learn from both good and bad trajectories, the policy will be trained towards a better one, compared to the pre-trained policy  $\pi^0$ . If we only use bad trajectories, then it is possible that we cannot get a better policy than  $\pi^*$ . This remark will be further validated in our later experiments, as we observe that one cannot learn a good policy by just using bad trajectories. Moreover, according to Proposition 3, our framework can be used to train towards policies with lower cost or even feasible policies by selecting different bad sets  $\Omega^E$ , even when the cost function is not known beforehand.

The theory also tells us that if we are more selective in choosing good and bad trajectories, we will tend to obtain better policies, as long as there are policies that can eliminate the probabilities of the bad ones. However, the selection process can be tricky and may not be easy to achieve in practice. So, it is better to not be too selective (or conservative) in classifying good and bad demonstrations.

### 3.3 Example

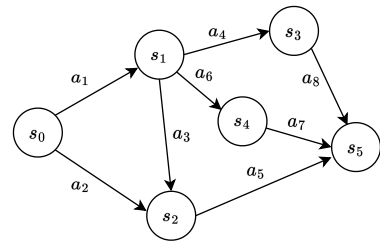


Figure 1: Example

We give a small example to demonstrate how our framework returns a better policy by learning from bad and good trajectories. We consider the small deterministic MDP given in Figure 1. The rewards,  $r$ , costs,  $c$  and pre-trained policy  $\pi^0$  are as shown in Table 1. The probabilities are over feasible actions from the state. There are 4 possible trajectories  $\tau_1 = \{s_0, s_1, s_3, s_5\}$ ,  $\tau_2 = \{s_0, s_1, s_4, s_5\}$ ,  $\tau_3 = \{s_0, s_1, s_2, s_5\}$ , and  $\tau_4 = \{s_0, s_2, s_5\}$ . We then see that  $\mathbb{E}_{\pi^0} [R(\tau)] = \sum_{\tau} P_{\pi^0}(\tau) = 3.5$ ;  $\mathbb{E}_{\pi^0} [C(\tau)] =$

$\sum_{\tau} P_{\pi^0}(\tau) = 2$ , implying that  $\pi^0$  is feasible for the CMDP problem.

Under our good-bad scheme, trajectory  $\tau_1 = \{s_0, s_1, s_3, s_5\}$  has the accumulated reward and cost as  $R(\tau_1) = 3 < \mathbb{E}_{\pi^0}[R(\tau)]$ ,  $C(\tau_1) = 4 > c_{\max}$ . So, according to the criteria in Theorem 1,  $\tau_1$  should be considered a bad trajectory (the others are good). The BC objective can be written as  $F(\pi) = \lambda \sum_{i \in \{2,3,4\}} P_{\pi^0}(\tau_i) \ln P_{\pi}(\tau_i)$ . The following policy  $\pi^*$  such that  $\pi^*(a_4|s_1) = 0$ ,  $\pi^*(a_6|s_1) = \pi^*(a_3|s_1) = 1/2$ ,  $\pi^*(a_1|s_0) = 2/5$  and  $\pi^*(a_2|s_0) = 3/5$  will satisfy the condition in Lemma 1, i.e.,  $P_{\pi^*}(\tau_1); P_{\pi^*}(\tau_2) = 1/5 = \frac{P_{\pi^0}(\tau_2)}{5/6}$ ;  $P_{\pi^*}(\tau_3) = 1/5 = \frac{P_{\pi^0}(\tau_3)}{5/6}$ ;  $P_{\pi^*}(\tau_4) = 3/5 = \frac{P_{\pi^0}(\tau_4)}{5/6}$  and  $P_{\pi^0}(\tau_2) + P_{\pi^0}(\tau_3) + P_{\pi^0}(\tau_4) = 5/6$ , thus  $\pi^*$  is optimal for  $\max_{\pi} \{F(\pi)\}$ . On the other hand,  $\mathbb{E}_{\pi^*}[R(\tau)] = 3.6$ ;  $\mathbb{E}_{\pi^*}[C(\tau)] = 1.6$ . So  $\pi^*$  offers a better expected reward and a lower cost compared to the pre-trained policy  $\pi^0$ .

## 4 Self-Imitation based Safe RL

In this section, we present a practical IL-based algorithm for constrained RL. A BC-based algorithm can be developed using (BC-GB). However, this approach (or the BC in general) would not be practical and would necessitate a huge number of samples to attain the desired performance. In contrast, Distribution Matching proves to be a more practical alternative. Taking inspiration from the GAIL algorithm, to address (DM-GB), one can construct two discriminators: one for  $\text{KL}(\rho^{\pi} || \rho^G)$  and another for  $\text{KL}(\rho^{\pi} || \rho^B)$ . Nonetheless, this approach involves two adversaries and would be highly unstable (as demonstrated in our experiments). To get rid of adversarial training, let us put the occupancy measures of the learning policy and the good demonstrations together, and consider the following mixed state-action distribution  $\rho^{G,\pi} = (\rho^{\pi} + \rho^G)/2$ . We then set our aim to maximize the KL divergence between  $\rho^{G,\pi}$  and the occupancy measure of the bad trajectories  $\rho^B$  (thus making  $\rho^{G,\pi}$  far away from the ‘‘bad’’ occupancy measure  $\rho^B$ ).

$$\max_{\pi} \{ \text{KL}(\rho^{G,\pi} || \rho^B) \} = \max_{\pi} \mathbb{E}_{(s,a) \sim \rho^{G,\pi}} \left[ \ln \frac{\rho^B(s,a)}{\rho^{G,\pi}(s,a)} \right] \quad (3)$$

To estimate distribution ratio  $\frac{\rho^B(s,a)}{\rho^{G,\pi}(s,a)}$ , we propose the following surrogate maximization problem

$$\max_{K: S \times A \rightarrow (0,1)} \left\{ J(K, \pi) := \mathbb{E}_{\rho^B} [\ln(K(s,a))] + \frac{1}{2} \mathbb{E}_{\rho^{\pi}} [\ln(1 - K(s,a))] + \frac{1}{2} \mathbb{E}_{\rho^G} [\ln(1 - K(s,a))] \right\} \quad (4)$$

Here, (4) is connected to (3) through the following result:

**Proposition 4.** *The maximization in (4) is achieved at  $K^*(s,a)$  such that*

$$\ln \left( \frac{K^*(s,a)}{1 - K^*(s,a)} \right) = \ln \frac{\rho^B(s,a)}{\rho^{G,\pi}(s,a)}$$

This implies that the distribution ratio can be estimated as  $\ln \frac{K^*(s,a)}{1 - K^*(s,a)}$ . As a result, the policy can be updated by max-

imizing  $\max_{\pi} \mathbb{E}_{(s,a) \sim \rho^{G,\pi}} \left[ \ln \frac{K^*(s,a)}{1 - K^*(s,a)} \right]$ , which is equivalent to  $\max_{\pi} \mathbb{E}_{(s,a) \sim \rho^{\pi}} \left[ \ln \frac{K^*(s,a)}{1 - K^*(s,a)} \right]$  as the occupancy measure of the good demonstrations is constant.

In practice,  $K(s,a)$  need not be fully optimized. Instead,  $K$  and  $\pi$  can be updated alternatively by gradient ascent. It is important to note that we update  $K(s,a)$  by maximizing  $J(K, \pi)$  and update  $\pi$  by maximizing  $\text{KL}(\rho^{G,\pi} || \rho^B)$ , so our algorithm is *non-adversarial*. In other words,  $K(s,a)$  operates in a cooperative manner rather than an adversarial one – it collaborates with the policy  $\pi$  to estimate the distribution ratio and make the mixed distribution  $\rho^{G,\pi}$  far away from the bad one  $\rho^B$ . Here, the non-adversarial nature of our method stems from our approach of maximizing the KL divergence, in contrast to the minimizing aspect employed in GAIL.

Drawing from the above analyses, we proceed to outline our algorithm. Let  $w$  and  $\theta$  denote the parameters of  $K(s,a)$  and  $\pi(s,a)$  respectively. The core concept involves iteratively enhancing  $K$  and  $\pi$  through alternating gradient ascent updates. While  $K_w$  can be updated by using the derivatives of  $J(K_w, \pi_{\theta})$ ,  $\pi_{\theta}$  can be updated by a policy gradient method, e.g., PPO (Schulman et al. 2017). During the training process, we generate additional trajectories and update the good and bad sets. The key steps of our method are described in Algorithm 1.

---

### Algorithm 1: Self-imitation Safe Reinforcement Learning

---

**Require:**  $\pi^0, K_w, R_G, R_B, c_{\max}$ , learning rates  $\kappa_{\theta}, \kappa_w$

$\Omega_G \leftarrow \emptyset; \Omega_B \leftarrow \emptyset; \pi_{\theta} \leftarrow \pi^0$

**while** not converge **do**

  # Sample new set trajectories

$T = \{\tau_0, \tau_1, \dots, \tau_n \sim \pi_{\theta}\}$

  # Update the ‘‘good’’ and ‘‘bad’’ sets,  $\sigma$  is the deviation

$R_B \leftarrow \mathbb{E}_{\tau \sim T} [R(\tau)] - \sigma_{\tau \sim T} [R(\tau)]$

$\Omega_G \leftarrow \Omega_G \cup \{\tau \in T | R(\tau) \geq R_G \cap C(\tau) \leq c_{\max}\}$

$\Omega_B \leftarrow \Omega_B \cup \{\tau \in T | R(\tau) < R_B \cup C(\tau) > c_{\max}\}$

  # Update  $K_w$

$w \leftarrow w + \kappa_w \nabla_w J(K_w, \pi_{\theta})$

  where  $J(K_w, \pi_{\theta}) = \mathbb{E}_{\Omega_B} [\ln(K_w(s,a))] + \frac{1}{2} \mathbb{E}_T [\ln(1 - K_w(s,a))] + \frac{1}{2} \mathbb{E}_{\Omega_G} [\ln(1 - K_w(s,a))]$

  # Update  $\pi_{\theta}$

$\theta = \theta + \kappa_{\theta} \mathbb{E}_{\tau \sim T} [\nabla_{\theta} \ln \pi_{\theta}(s,a) Q^K(s,a)]$

  where  $Q^K(s,a) = \mathbb{E}_T \left[ \sum_t \gamma^t \ln \frac{K_w(s_t, a_t)}{1 - K_w(s_t, a_t)} \Big|_{\substack{s_0=s \\ a_0=a}} \right]$

**end while**

---

## 5 EXPERIMENTS

We conduct experiments to compare our method against some state-of-the-art Constrained RL algorithms: FOCOPS (Zhang, Vuong, and Ross 2020), CUP (Yang et al. 2022), CPO (Achiam et al. 2017)<sup>3</sup>. For the sake of completeness, we also include PPO-Lagrangian (Ray, Achiam,

<sup>3</sup>FOCOPS, CUP, and CPO implementations can be found on <https://github.com/PKU-Alignment/omnisafe>.

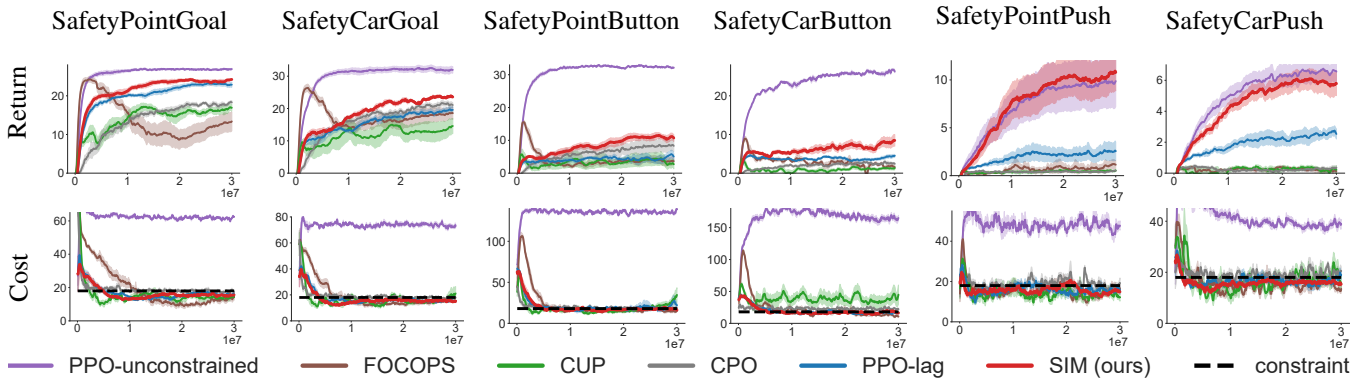


Figure 2: Training curves for 6 different SafetyGym environments. Every lines in calculated by the mean with shaded by the standard error of 6 independent seeds.

and Amodei 2019) and unconstrained PPO. We use PPO-Lagrangian to train our pre-trained policy and name our algorithm as SIM, standing for **S**elf-**I**Mitation based safe RL algorithm.

Through the experiments, we aim to address the following questions.

- (Q1) Would SIM outperform state-of-the-art constrained RL algorithms?
- (Q2) Is it necessary to use both good and bad demonstrations in the training?
- (Q3) How is SIM compared to a BC-based and GAIL-based algorithm?
- (Q4) How does SIM perform with different expertise levels of the initial policy  $\pi^0$ ? Can it benefit from a not well-trained policy

In the following, we provide experiments to address Q1-Q4. Additional experiments addressing Q5-Q8 can be found in the appendix. We set the cost limit as  $c_{max} = 18$ . We, however, train the initial policy  $\pi^0$  with a higher cost limit of  $c_{max} = 28$ , which allows us to generate more trajectories of high rewards and more unsafe trajectories. We test our method on 6 SafetyGym environments (Ji et al. 2023). We also simplify the names of environments, e.g., SafetyPointGoal1-v0 is renamed as SafetyPointGoal.

### 5.1 SIM vs other Constrained RL methods on SafetyGym

We compare our algorithm with prior safe RL ones using six different SafetyGym environments (Ray, Achiam, and Amodei 2019; Ji et al. 2023). The learning curves are shown in Figure 2 where the experiments are repeated over 6 independent seeds. For the sake of comparison, we include the PPO-unconstrained. Here are the key observations. In all the 6 environments, including the very challenging ones (SafetyPointButton and SafetyCarButton), SIM achieves the best performance – it offers the highest expected rewards while being safe. The PPO-unconstrained gives the highest rewards but is unsafe by a huge margin. Notably, in the last two push tasks, SIM achieves competitive or even higher

rewards, compared to the unconstrained one. Overall PPO-Lagrangian had the second best performance.

### 5.2 SIM vs GAIL, and the Importance of “Good” and “Bad” Demonstrations



Figure 3: Comparisons with GAIL-based algorithms and other versions of SIM.

We aim to assess the importance of having both “good” and “bad” demonstrations in our IL-based approach, as well as to demonstrate the advantages of our non-adversarial method. To this end, we compared SIM with three versions of GAIL that use (i) only good demonstrations  $\Omega^G$ , (ii) only “bad” demonstrations  $\Omega^B$ , and (iii) both “good” and “bad” demonstrations, but using two discriminators as described in Section 4. For the sake of comparison, we also include two versions of SIM with only good demonstrations and only bad demonstrations. We do this by just removing the “good” (or “bad”) part from (4). Figure 3 shows the comparisons



on 2 SafetyGym environments, which clearly demonstrates the superior performance of SIM, compared to SIM versions with only good (or bad) demonstrations, and the 3 different GAIL versions, highlighting the importance of having both good and bad trajectories in the training. Furthermore, it can be observed that our non-adversarial algorithm is highly stable and consistent in returning high-reward and safe policies.

### 5.3 SIM vs Behavioral Cloning

As mentioned earlier, one can design an IL-based method for constrained RL based on (BC-GB). In this section, we aim to compare our algorithm with a BC-based approach. We implement two BC algorithms: one is based on (BC) with one “good” demonstration, and another is based on (BC-GB) with both sets. The comparison results are presented in Figure 4. For the two BC algorithms, since their training curves are not comparable with those from SIM, we only draw horizon lines representing their expected reward and cost at convergence (their training curves are provided in the appendix).

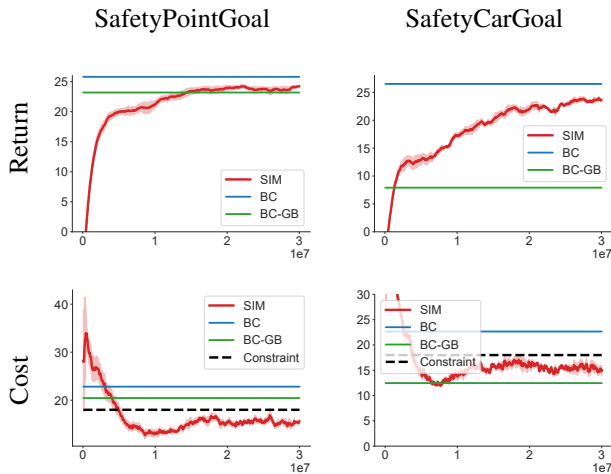


Figure 4: Comparison with BC-based algorithms.

In both environments, the BC achieves the highest expected rewards, but it fails to satisfy the constraint. BC-GB either gives low-reward or unsafe policies. On the other hand, SIM consistently achieves high rewards while satisfying the constraint in all the experiments.

### 5.4 Varying Expertise Level

In this section, our goal is to comprehend the influence of the training extent of the pre-trained policy  $\pi^0$  on the efficiency of SIM. To this end, we trained the initial policy  $\pi^0$  using varying numbers of environmental steps: specifically, 10 million (1e7), 20 million (2e7), and 30 million (3e7) steps, corresponding to what we term “entry-level”, “medium-level” and “expert-level”, respectively. The comparison results are shown in Figure 5, revealing that both the “entry-level” and “medium-level” pre-trained policies achieve lower expected rewards compared to the “expert-level” SIM. Nevertheless, with either the “entry-level” or

“medium-level”  $\pi^0$ , SIM outperforms the original PPO-Lagrangian baseline, which was the second best among all the baselines.

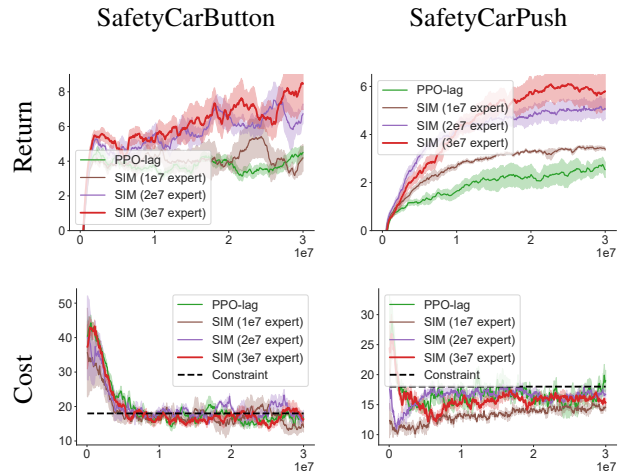


Figure 5: Comparison results for different expertise levels of the pre-trained policy.

These results demonstrate that even without a well-trained initial policy, SIM is able to efficiently improve it and outperform the traditional PPO-Lagrangian method. These also indicate that, for SIM to achieve the best performance, one should start with a well-trained initial policy. As mentioned previously, SIM would greatly benefit from learning from good trajectories generated by a well-trained policy.

## 6 CONCLUSION

We introduced a novel framework to solve Constrained RL without relying on cost estimations or cost penalties, as commonly done in prior work. Our new algorithm, based on the idea of learning to mimic the behavior of good demonstrations and avoid bad demonstrations, is non-adversarial and allows learning from demonstration sets to evolve during the training process. Extensive experiments on several challenging benchmark tasks demonstrate that our approach achieves superior performance compared to prior constrained RL algorithms. Our IL-based framework would open new directions to address safe RL problems without explicitly considering the reward or cost function. Our algorithm relies on sets of good demonstrations generated by a pre-trained policy, so a limitation would be that our algorithm will not work if it is difficult to generate feasible trajectories due to, for instance, strict constraints. A future direction would be to develop new IL-based algorithms to address such issues.

## References

Abbeel, P.; and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 1.



- Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *International conference on machine learning*, 22–31. PMLR.
- Altman, E. 1999. *Constrained Markov decision processes*, volume 7. CRC press.
- Chow, Y.; Nachum, O.; Faust, A.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2019. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*.
- Dadashi, R.; Hussenot, L.; Geist, M.; and Pietquin, O. 2020. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*.
- Firoiu, V.; Whitney, W. F.; and Tenenbaum, J. B. 2017. Beating the world’s best at Super Smash Bros. with deep reinforcement learning. *arXiv preprint arXiv:1702.06230*.
- Fu, J.; Luo, K.; and Levine, S. 2017. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*.
- Garg, D.; Chakraborty, S.; Cundy, C.; Song, J.; and Ermon, S. 2021. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34: 4028–4039.
- Ho, J.; and Ermon, S. 2016. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29.
- Hoang, M.-H.; Dinh, L.; and Nguyen, H. 2023. Learning from Pixels with Expert Observations. *arXiv preprint arXiv:2306.13872*.
- Ji, J.; Zhang, B.; Pan, X.; Zhou, J.; Dai, J.; and Yang, Y. 2023. Safety-Gymnasium. *GitHub repository*.
- Kilinc, O.; and Montana, G. 2022. Reinforcement learning for robotic manipulation using simulated locomotion demonstrations. *Machine Learning*, 1–22.
- Kostrikov, I.; Nachum, O.; and Tompson, J. 2019. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*.
- Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937. PMLR.
- Ng, A. Y.; Russell, S.; et al. 2000. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, 2.
- Raghu, A.; Komorowski, M.; Ahmed, I.; Celi, L.; Szolovits, P.; and Ghassemi, M. 2017. Deep reinforcement learning for sepsis treatment. *arXiv preprint arXiv:1711.09602*.
- Ray, A.; Achiam, J.; and Amodei, D. 2019. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1): 2.
- Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635. JMLR Workshop and Conference Proceedings.
- Satija, H.; Amortila, P.; and Pineau, J. 2020. Constrained markov decision processes via backward value functions. In *International Conference on Machine Learning*, 8502–8511. PMLR.
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Weng, W.-H.; Gao, M.; He, Z.; Yan, S.; and Szolovits, P. 2017. Representation and reinforcement learning for personalized glycemic control in septic patients. *arXiv preprint arXiv:1712.00654*.
- Yang, L.; Ji, J.; Dai, J.; Zhang, L.; Zhou, B.; Li, P.; Yang, Y.; and Pan, G. 2022. Constrained update projection approach to safe policy optimization. *Advances in Neural Information Processing Systems*, 35: 9111–9124.
- Yang, Q.; Simão, T. D.; Tindemans, S. H.; and Spaan, M. T. 2021. WCSAC: Worst-case soft actor critic for safety-constrained reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10639–10646.
- Zhang, Y.; Vuong, Q.; and Ross, K. 2020. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 33: 15338–15349.

## A Missing Proofs

### A.1 Proof of Lemma 1

**Lemma 1:** For any  $\lambda > 0$ , if there exists a policy  $\pi^*$  such that  $P_{\pi^*}(\tau) = 0$  for all  $\tau \in \Omega^B$ , and

$$P_{\pi^*}(\tau) = \frac{P_{\pi^0}(\tau)}{\sum_{\tau' \in \Omega^G} P_{\pi^0}(\tau')}; \quad \forall \tau \in \Omega^G$$

then  $\pi^*$  is an optimal policy to (BC-GB).

*Proof.* To simplify the proof, let us first prove the following result:

**Lemma 3.** Given  $\hat{p}_0, \hat{p}_1, \dots, \hat{p}_N \in [0, 1]$  such that  $\sum_{n=1}^N \hat{p}_n \leq 1$ , then vector  $p^*$  such that  $p_n^* = \frac{\hat{p}_n}{\sum_{n'} \hat{p}_{n'}}$  is a unique optimal solution to the following optimization problem

$$\max_{p \in [0,1]^N} \left\{ f(p) = \sum_n \hat{p}_n \ln p_n \mid \sum_n p_n \leq 1 \right\} \quad (5)$$

*Proof.* We first see that the objective function  $f(p)$  is strictly concave in  $(0, 1)^N$ , implying that (5) always has a unique optimal solution. We write the Lagrange dual of (5) as

$$\mathcal{L}(p, \eta) = \sum_n \hat{p}_n \ln p_n - \eta \left( \sum_n p_n - 1 \right)$$

Let  $\bar{p}$  be the optimal solution of (5) and  $\bar{\eta}$  be its associated Lagrange multiplier. The KKT conditions imply that the following hold

$$\begin{cases} \frac{\partial \mathcal{L}(\bar{p}, \bar{\eta})}{\partial p_n} = 0, \quad \forall n = 1, \dots, N \\ \eta(\sum_n \bar{p}_n) = 0 \end{cases}$$

which is equivalent to

$$\begin{cases} \frac{\bar{p}_1}{\bar{p}_1} = \frac{\bar{p}_2}{\bar{p}_2} = \dots = \frac{\bar{p}_N}{\bar{p}_N} = \eta \\ \eta(\sum_n \bar{p}_n - 1) = 0 \end{cases}$$

We then see that  $\eta > 0$ , thus  $\sum_n \bar{p}_n - 1 = 0$ . On the other hand  $\eta = \frac{\sum_n \hat{p}_n}{\sum_n \bar{p}_n} = \sum_n \hat{p}_n$ , which implies that  $\bar{p}_n = \frac{\hat{p}_n}{\sum_{n'} \hat{p}_{n'}}$ . Thus  $p^* = \bar{p}$  is a unique optimal solution to (5), as desired.  $\square$

We now get back to the main proof. Recall that the objective function of the training with good and bad trajectories, under BC, is

$$F(\pi) = \lambda \sum_{\tau \in \Omega^G} P_{\pi^0}(\tau) \ln P_{\pi}(\tau) - (1 - \lambda) \sum_{\tau \in \Omega^B} P_{\pi^0}(\tau) \phi(\ln P_{\pi}(\tau))$$

We now assume that the regularizer  $\phi(\cdot)$  map  $(-\infty, 0]$  to a finite interval  $[a, b]$ . Since  $\phi(\cdot)$  is monotone, we see that, for any  $\tau \in \Omega$ ,  $P_{\pi}(\tau) \geq P_{\pi^*}(\tau)$ , thus  $\phi(P_{\pi}(\tau)) \geq \phi(P_{\pi^*}(\tau))$ . Moreover, from Lemma 3, the first term of  $F(\pi)$  can be bounded as

$$\sum_{\tau \in \Omega^G} P_{\pi^0}(\tau) \ln P_{\pi}(\tau) \leq \sum_{\tau \in \Omega^G} P_{\pi^0}(\tau) \ln P_{\pi^*}(\tau)$$

implying  $F(\pi) \leq F(\pi^*)$  for any policy  $\pi$ . So, if  $\pi^*$  exists, it will be optimal to (BC-GB).  $\square$

### A.2 Proof of Proposition 1

**Proposition 1:** For any  $\lambda > 0$ , let  $\pi^0$  be a pre-trained feasible policy,  $\mathbf{R}^E = \mathbb{E}_{\tau \sim \pi^0} [R(\tau)]$ , and  $\Omega^B$  be a collection of trajectories of low reward and high-cost values

$$\Omega^B = \left\{ \tau \mid R(\tau) \leq \mathbf{R}^E, C(\tau) > c_{\max} \right\}$$

the optimal policy mentioned in Lemma 1 is feasible to the cost constraint while offering a better expected reward than the pre-trained policy  $\pi^0$ , specifically,

$$\mathbb{E}_{\pi^*} [R(\tau)] - \mathbb{E}_{\pi^0} [R(\tau)] = \frac{\sum_{\tau} P_{\pi^0}(\tau) (\mathbf{R}^E - R(\tau))}{1 - P_{\pi^0}(\Omega^B)} \geq 0 \quad (6)$$

$$\mathbb{E}_{\tau \sim \pi^*} [C(\tau)] \leq c_{\max}$$

where  $P_{\pi^0}(\Omega^B) = \sum_{\tau \in \Omega^B} P_{\pi^0}(\tau)$ .

*Proof.* Recall that  $P_{\pi^*}(\tau) = 0$  for all  $\tau \in \Omega^B$  and  $P_{\pi^*}(\tau) = \frac{P_{\pi^0}(\tau)}{\sum_{\tau'} P_{\pi^0}(\tau')}$  for all  $\tau \in \Omega^G$ . We write the expected reward under  $\pi^*$  as

$$\mathbb{E}_{\tau \sim \pi^*}[R(\tau)] = \sum_{\tau \in \Omega^G} P_{\pi^*}(\tau)R(\tau) = \frac{\sum_{\tau \in \Omega^G} P_{\pi^0}(\tau)}{1 - P_{\pi^0}(\Omega^B)}$$

Thus

$$\begin{aligned} \mathbb{E}_{\tau \sim \pi^*}[R(\tau)] - \mathbb{E}_{\tau \sim \pi^0}[R(\tau)] &= \sum_{\tau \in \Omega^G} P_{\pi^*}(\tau)R(\tau) - \sum_{\tau \in \Omega^G} P_{\pi^0}(\tau)R(\tau) = \frac{\sum_{\tau \in \Omega^G} P_{\pi^0}(\tau)R(\tau) - \sum_{\tau \in \Omega^G} P_{\pi^0}(\tau)R(\tau)(1 - P_{\pi^0}(\Omega^B))}{1 - P_{\pi^0}(\Omega^B)} \\ &= \frac{\sum_{\tau \in \Omega^G} P_{\pi^0}(\tau)R(\tau) - \sum_{\tau \in \Omega^G} P_{\pi^0}(\tau)R(\tau) + P_{\pi^0}(\Omega^B)\mathbf{R}^E}{1 - P_{\pi^0}(\Omega^B)} \\ &= \frac{\sum_{\tau \in \Omega^B} P_{\pi^0}(\tau)(\mathbf{R}^E - R(\tau))}{1 - P_{\pi^0}(\Omega^B)} \stackrel{(a)}{\geq} 0 \end{aligned}$$

where (a) is due to the fact that  $R(\tau) \leq \mathbf{R}^E$  for all  $\tau \in \Omega^B$ . We now consider the expected cost given by  $\pi^*$ . Let  $\mathbf{C}^E = \mathbb{E}_{\pi^0}[C(\tau)]$ , we write

$$\begin{aligned} \mathbb{E}_{\pi^*}[C(\tau)] - \mathbf{C}^E &= \sum_{\tau} P_{\pi^*}(\tau)C(\tau) - \mathbf{C}^E \\ &= \frac{\sum_{\tau \in \Omega^G} P_{\pi^0}(\tau)C(\tau) - \sum_{\tau \in \Omega^G} P_{\pi^0}(\tau)C(\tau) + \mathbf{C}^E P(\Omega^B)}{1 - P_{\pi^0}(\Omega^B)} \\ &= \frac{\mathbf{C}^E P(\Omega^B) - \sum_{\tau \in \Omega^B} P_{\pi^0}(\tau)C(\tau)}{1 - P_{\pi^0}(\Omega^B)} \\ &\stackrel{(b)}{\leq} \frac{\mathbf{C}^E P(\Omega^B) - \sum_{\tau \in \Omega^B} P_{\pi^0}(\tau)c_{max}}{1 - P_{\pi^0}(\Omega^B)} = \frac{(\mathbf{C}^E - c_{max})P(\Omega^B)}{1 - P_{\pi^0}(\Omega^B)} \stackrel{(c)}{\leq} 0 \end{aligned}$$

where (b) is because  $C(\tau) > c_{max}$  (according to the way we choose  $\Omega^B$ ), and  $\mathbf{c}^E \leq c_{max}$  ( $\pi^0$  is feasible w.r.t the cost constraint). So, we have  $\mathbb{E}_{\pi^*}[C(\tau)] \leq \mathbf{C}^E \leq c_{max}$ , implying that  $\pi^*$  is safe, as desired.  $\square$

### A.3 Proof of Lemma 2

**Lemma 2:** *If  $\lambda = 0$ , any policy  $\pi^*$  such that  $P_{\pi^*}(\tau) = 0$  for all  $\tau \in \Omega^B$  is optimal for (BC-GB).*

*Proof.* This can be obviously seen, as if  $\lambda = 0$ , then the objective function becomes  $F(\pi) = -\sum_{\tau \in \Omega^B} P_{\pi^0}(\tau)\phi(\ln P_{\pi}(\tau))$ . Since  $\phi(\cdot)$  is monotone,  $F(\pi) \geq F(\pi^*)$  for any policy  $\pi$ .  $\square$

### A.4 Proof of Proposition 2

**Proposition 2:** *If  $\lambda = 0$ , and the bad set  $\Omega^B$  is selected in the same manner as in Theorem 1, then the optimal policy  $\pi^*$  from Lemma (2) is feasible, but it does not necessarily provide a higher expected reward than the policy  $\pi^0$ .*

*Proof.* According to the Lemma 2, any  $\pi^*$  such that  $P_{\pi^*}(\tau) = 0$  is optimal for (BC-GB). To prove that  $\pi^*$  may not offer a higher expected reward than  $\pi^0$ , we will use the counter-example shown in Figure 6. There are 5 states and the MDP is deterministic. The rewards and cost are set as  $r(s_0) = 0, r(s_4) = 0, r(s_1) = 2, r(s_2) = 3, r(s_3) = 8$ , and  $d(s_0) = 0, d(s_4) = 0, d(s_1) = 5, d(s_2) = d(s_3) = 1$ . The initial policy is set as  $\pi^0(a_1|s_0) = 1/5, \pi^0(a_2|s_0) = 1/5$  and  $\pi^0(a_3|s_0) = 3/5$ . We also choose  $c_{max} = 2$ . The expected reward is  $\mathbf{R}^E = 5.6$  and expected cost is  $\mathbf{C}^E = 1.8$ . It is then clear that the trajectory  $\{s_0, s_1, s_4\}$  should be classified in the bad set. Policy  $\pi^*$  such that  $\pi^*(a_1|s_0) = 0, \pi^*(a_2|s_0) = 4/5$  and  $\pi^*(a_3|s_0) = 1/5$  is definitely optimal for (BC-GB), according to Lemma 2. We however see that  $\mathbb{E}_{\pi^*}[R(\tau)] = 4 < \mathbf{R}^E$ , implying that  $\pi^*$  offers a worse expected reward than the initial policy  $\pi^0$ . We complete the proof.  $\square$

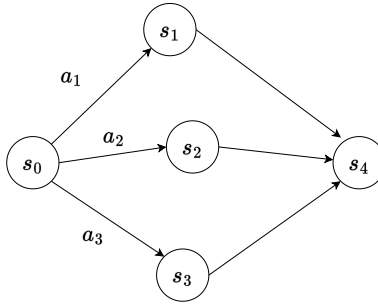


Figure 6: Example

### A.5 Proof of Proposition 3

**Proposition 3:** *The following hold*

- (i) *If we select the bad set as  $\Omega^B = \left\{ \tau \mid R(\tau) \leq \mathbf{R}^E, C(\tau) > \mathbf{C}^E \right\}$ , then it is guaranteed that  $\pi^*$  offers a higher (or equal) expected reward and lower (or equal) expected cost, compared to those from  $\pi^0$ , where  $\mathbf{C}^E = \mathbb{E}_{\tau \sim \pi^0} [C(\tau)]$*
- (ii) *If the pre-trained policy  $\pi^0$  is not feasible, then if we select the bad set as  $\Omega^B = \left\{ \tau \mid C(\tau) > c_{\max} \right\}$ , then it is guaranteed that  $\pi^*$  is feasible*
- (iii) *If the cost function is not accessible, but there is an oracle that can tell us which trajectories are violating the constraint, then by selecting,  $\Omega^B = \left\{ \tau \mid \tau \text{ is violated} \right\}$ , then  $\pi^*$  is feasible.*

*Proof.* The proof is similar to the proof of Proposition 1. For (i), we also write

$$\begin{aligned} \mathbb{E}_{\pi^*} [R(\tau)] - \mathbf{R}^E &= \frac{\sum_{\tau \in \Omega^B} P_{\pi^0}(\tau) (\mathbf{R}^E - R(\tau))}{1 - P_{\pi^0}(\Omega^B)} \\ \mathbb{E}_{\pi^*} [C(\tau)] - \mathbf{C}^E &= \frac{\sum_{\tau \in \Omega^B} (\mathbf{C}^E - C(\tau)) P_{\pi^0}(\tau)}{1 - P_{\pi^0}(\Omega^B)} \end{aligned}$$

Then according to the way we select  $\Omega^B$  in (i), we should have  $\mathbb{E}_{\pi^*} [R(\tau)] \geq \mathbf{R}^E$  and  $\mathbb{E}_{\pi^*} [C(\tau)] \leq \mathbf{C}^E$ , implying that  $\pi^*$  yields a higher expected reward and lower expected cost, compared to  $\pi^0$ .

For (ii), since  $C(\tau) > c_{\max}$  for all  $\tau \in \Omega^B$ ,  $C(\tau) \leq c_{\max}$  for all  $\tau \in \Omega^G$ . We write the expected cost under  $\pi^*$  as

$$\mathbb{E}_{\pi^*} [C(\tau)] = \frac{\sum_{\tau \in \Omega^G} P_{\pi^0}(\tau) C(\tau)}{1 - P_{\pi^0}(\Omega^B)} \leq \frac{\sum_{\tau \in \Omega^G} P_{\pi^0}(\tau) c_{\max}}{1 - P_{\pi^0}(\Omega^B)} = c_{\max}$$

So,  $\pi^*$  is safe.

Claim (iii) is the same as (ii), in the sense that the oracle can correctly select the bad set  $\Omega^B = \{ \tau \mid C(\tau) > c_{\max} \}$ . Thus, the policy  $\pi^*$ , if exists, will be safe.  $\square$

### A.6 Proof of Proposition 4

**Proposition 4:** *The maximization in (4) is achieved at  $K^*(s, a)$  such that*

$$\ln \left( \frac{K^*(a, s)}{1 - K^*(s, a)} \right) = \ln \frac{\rho^B(s, a)}{\rho^{G, \pi}(s, a)}$$

*Proof.* We first look at the training objective of  $K(s, a)$  and write

$$\begin{aligned} J(K, \pi) &= \mathbb{E}_{\rho^B} [\ln(K(s, a))] + \frac{1}{2} \mathbb{E}_{\rho^\pi} [\ln(1 - K(s, a))] + \frac{1}{2} \mathbb{E}_{\rho^G} [\ln(1 - K(s, a))] \\ &= \mathbb{E}_{\rho^B} [\ln(K(s, a))] + \sum_{(s, a)} \ln(K(s, a)) \frac{\rho^\pi(s, a) + \rho^G(s, a)}{2} \\ &= \mathbb{E}_{\rho^B} [\ln(K(s, a))] + \mathbb{E}_{\rho^{\pi, G}} [\ln(1 - K(s, a))] \\ &= \sum_{(s, a)} \ln(K(s, a)) \rho^B(s, a) + \ln(1 - K(s, a)) \rho^{\pi, G}(s, a) \end{aligned} \tag{7}$$

So, to maximize  $J(K, \pi)$ , each component  $\ln(K(s, a))\rho^B(s, a) + \ln(1 - K(s, a))\rho^{\pi, G}(s, a)$  needs to be maximized. To study this maximization problem, we consider the following simple optimization problem  $\max_{x \in (0,1)} \{f(x) = \ln(x)a + \ln(1-x)b\}$ , where  $a, b \geq 0$ . We first see that  $f'(x) = \frac{a}{x} - \frac{b}{1-x}$ . Thus if we set  $f'(x) = 0$ , this equation has a unique solution as  $x^* = \frac{a}{a+b}$ . Moreover  $f'(x) \leq 0$  if  $x \leq x^*$  and  $f'(x) \geq 0$  if  $x \geq x^*$ , thus  $x^*$  is a unique solution to  $\max_{x \in (0,1)} \{f(x) = \ln(x)a + \ln(1-x)b\}$ .

We now get back to the maximization

$$\max_K \{ \ln(K(s, a))\rho^B(s, a) + \ln(1 - K(s, a))\rho^{\pi, G}(s, a) \} \quad (8)$$

From the above small problem, we know that (8) has a unique optimization solution  $K^*(s, a)$  such that

$$K^*(s, a) = \frac{\rho^B(s, a)}{\rho^B(s, a) + \rho^{\pi, G}(s, a)}$$

implying

$$\frac{K^*(s, a)}{1 - K^*(s, a)} = \frac{\rho^B(s, a)}{\rho^{\pi, G}(s, a)}.$$

as desired. □

## B Additional Details

### B.1 Method Overview

In Figure 7 we show a diagram illustrating in detail our algorithm SIM.

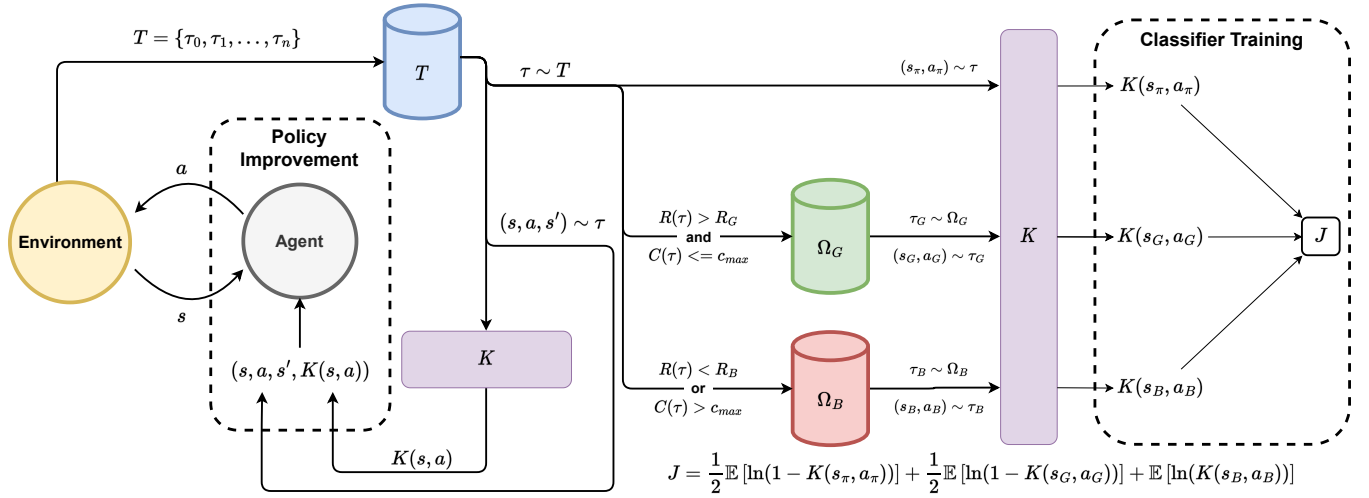


Figure 7: Overview of SIM

### B.2 Additional Settings for the Experiment with Varying Expertise Level

We provide additional details for Section 5.4 (Varying Expertise Level) in the main paper. Table 2 shows the expected rewards and the chosen thresholds  $R^G$  (those for selecting the good trajectories) of the three levels of  $\pi^0$ .

Steps	SafetyCarButton		SafetyCarPush	
	Expected return	$R_G$	Expected return	$R_G$
1e7	5.3	7.0	2.9	3.0
2e7	8.92	9.0	5.07	5.0
3e7	14.4	15.0	6.85	8.0

Table 2: Expected rewards and thresholds  $R_G$  of for different expertise levels of  $\pi^0$ .

### B.3 Relaxed Constraints

We provide a more detailed explanation of why relaxing the constraints is beneficial for the training of  $\pi^0$ . In practical scenarios, enforcing strict constraints on trajectory generation may hinder the achievement of good trajectories due to exploration challenges and limitations in obtaining high rewards. Conversely, adopting a more relaxed constraint (constraint with higher  $c_{max}$ ) setting could lead to higher returns, but it might also reduce the chances of satisfying the strict constraint. To address this, we initiate the training process with relaxed constraints (i.e., higher  $c_{max}$ ) that allow us to generate a better set of good trajectories (Figure 8 illustrate an advantage of using relaxed-constrained initial policy). Our experiments clearly demonstrate the significant advantages of employing relaxed constraints on the algorithm’s final performance.

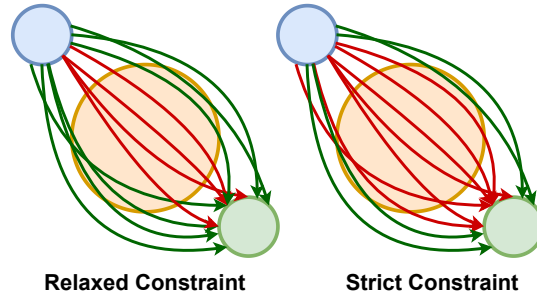


Figure 8: Although a significant number of trajectories do not satisfy the constraints (red lines), the relaxed-constraint setting is still able to offer a considerable number of good trajectories (green lines).

### B.4 Environmental Details

**Safety-gym** The Safety-gym benchmark (Ray, Achiam, and Amodei 2019), has emerged as a highly challenging benchmark for Constraint RL. Previous research mostly focused on the easiest environment, *SafetyPointGoal*, with some providing results for even simpler variations (Yang et al. 2021). In contrast, we conducted comprehensive experiments, exploring all six challenging environments within this benchmark. These environments are illustrated in Figure 9 below.

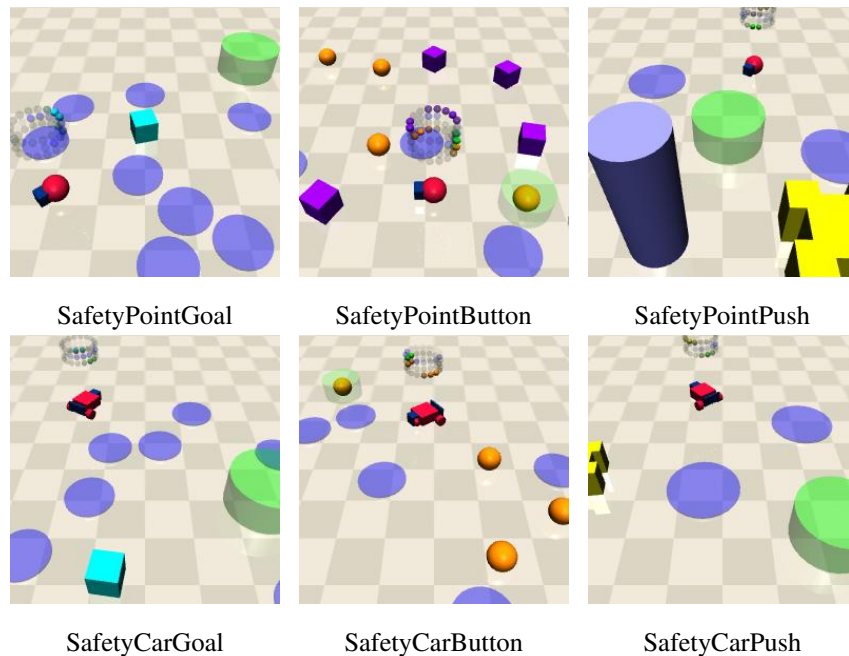


Figure 9: Six different environments in Safety-Gym.

In the first pair of environments, *SafetyPointGoal* and *SafetyCarGoal*, the agent’s primary objective is to reach the designated goal position, represented by the green area in the visuals. This must be accomplished with skillful navigation to avoid both hazardous areas (blue regions) and obstacles (cyan blocks). The *SafetyPointGoal* task features a point agent, which is relatively



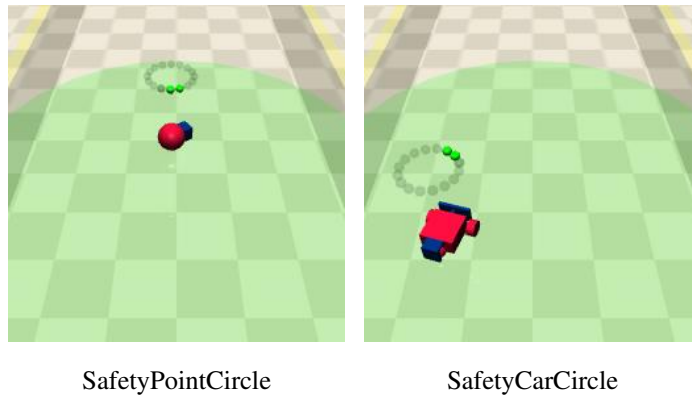


Figure 10: Mujoco Circle

easier to control, allowing for efficient training. On the other hand, the *SafetyCarGoal* task poses a greater challenge due to the more demanding control requirements of the car agent.

Moving on to the next set of environments, *SafetyPointButton* and *SafetyCarButton*, the agent encounters a fresh set of challenges. In *SafetyPointButton*, the primary goal is to navigate to the correct button, indicated by the green button, while carefully avoiding incorrect buttons, hazardous areas (blue regions), and maneuvering around moving obstacles (purple blocks). The *SafetyCarButton* environment shares a similar objective, but with the removal of moving obstacles to reduce training difficulty. Despite this adjustment, controlling the car agent remains challenging.

Lastly, in the last pair of environments, *SafetyPointPush* and *SafetyCarPush*, the agent’s main task is to push the yellow block to the goal area (green region) while skillfully evading hazard areas (blue regions) and the blocking pillar (dark-blue cylinder) to increase the task difficulty. Similar to the button tasks, the pillar is removed to ease the task difficulty for the car agent.

**Mujoco Circle** The *Mujoco Circle* task was developed by (Achiam et al. 2017), involving agents moving along a circle centered at the origin. However, there is a constraint that the agent must remain in a area within a safety region, which is smaller than the radius of the circle and represented by the green area. To further challenge the agent, two walls are introduced that hinder its ability to move freely. Compared to the *Safety-Gym* environments, these tasks are considered less difficult because there is no randomness in the constraints imposed on the agent. The constraints are well-defined and consistent throughout the task.

To evaluate the performance of different agents under increasing difficulty, two types of agents are tested: Point and Car. Each agent faces the same task but with varying degrees of complexity. The Point agent is presumably the easier to control, while the Car agent poses a higher level of difficulty due to more demanding control requirements. The illustration is in Figure 10.

By conducting experiments with these agents in the *Mujoco Circle* task, we can gain valuable insights into the agents’ abilities to navigate the circular environment while adhering to the constraints, allowing for a comparative analysis of their performance under increasing difficulty levels.

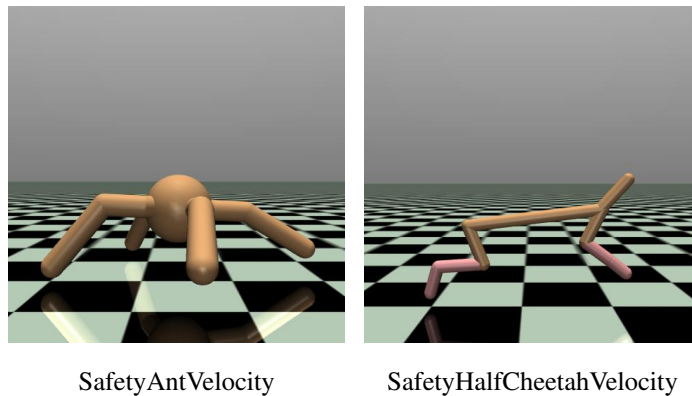


Figure 11: Mujoco Velocity

**Mujoco-velocity** We also test our algorithm with the *Mujoco Velocity* domains. MuJoCo is an advanced framework specialized in simulating intricate physical systems that feature multi-joint mechanisms and interactions. A key aspect of MuJoCo’s

capabilities involves its integration of velocity constraints. In our experiments, these constraints play a crucial role as we impose specific velocity limits on the agent’s movements. This action allows us to exert significant control over the motion of articulated entities within the simulation, effectively replicating real-world constraints and behaviors. The illustration is in Figure 11.

It’s worth noting that in our experimental setups, the MuJoCo environments that emphasize velocity demonstrate a relatively lower level of challenge due to the absence of external obstacles and random elements. Furthermore, achieving a high performance score doesn’t solely rely on achieving high velocity. As a result, all algorithms tested within this context exhibit impressive learning capabilities.

## C Additional Experiments

In this section, we provide experiments to answer 4 additional questions:

- (Q5) Can SIM provide a high-reward and safe policy using a relaxed-constraint expert?
- (Q6) What happens if the cost function is inaccessible?
- (Q7) Would an unconstrained problem benefit from our approach?
- (Q8) Would our approach work with CVaR constrained problems (Yang et al. 2021)?

### C.1 Hyper-parameter selection

We conducted all experiments on a total of 4 NVIDIA RTX A5000 GPUs and 96 core CPUs. The detailed hyper-parameters are reported in Table 3.

Hyper Parameter	Safety-gym	Mujoco-circle	Mujoco-velocity
Actor Network	[256, 256, 256]	[256, 256, 256]	[64, 64]
Critic Network	[256, 256, 256]	[256, 256, 256]	[64, 64]
Cost Critic Network	[256, 256, 256]	[256, 256, 256]	[64, 64]
Classifier Network	[100, 100, 100]	[100, 100]	[100, 100]
Gamma	0.99	0.99	0.99
lr actor	0.0001	0.0001	0.0003
lr Critic	0.0001	0.0001	0.0001
lr Cost Critic	0.0001	0.0001	0.0001
lr Classifier	0.01	0.01	0.01
lr Penalty	0.01	0.01	0.01
max KL	0.05	0.05	0.2
max iteration per update	80	80	120
buffer size	50,000	50,000	20,000
max episode length	1,000	500	1,000
Classifier batch size	4,096	4,096	4,096
$R_G$ (fixed)	$\mathbb{E}_{\tau \sim \pi^E} [R(\tau)]$	$\mathbb{E}_{\tau \sim \pi^E} [R(\tau)]$	$\mathbb{E}_{\tau \sim \pi^E} [R(\tau)]$
max $R_B$	$\max(R_G/2, R_G - 5.0)$	$\max(R_G/2, R_G - 10.0)$	$\max(R_G/2, R_G - 1000.0)$

Table 3: Hyper parameters.

Moreover, in practice, instead of solving the theoretical problem (4), our implementation applies a more stable version with the contribution of the Chi-square divergence  $\chi^2$ :

$$\begin{aligned}
 \max_{K: S \times A \rightarrow (0,1)} \left\{ J(K, \pi) := \mathbb{E}_{\rho^B} \left[ -K(s, a) - \frac{1}{a} K(s, a)^2 \right] \right. \\
 \left. + \frac{1}{2} \mathbb{E}_{\rho^\pi} \left[ K(s, a) - \frac{1}{a} K(s, a)^2 \right] + \frac{1}{2} \mathbb{E}_{\rho^G} \left[ K(s, a) - \frac{1}{a} K(s, a)^2 \right] \right\} \quad (9)
 \end{aligned}$$

### C.2 Training Curves of BC and BC-GB

Figure 12 shows the training curves of the BC and BC-GB approaches. This supplements our experimental results in Section 5.3, where we compare SIM against BC-based approaches.

### C.3 Unknown Cost

To answer Q6 (what happens if the cost function is inaccessible?), we demonstrate the capability of our method in handling the situation that the cost function is unknown. In this setting, we assume that there is an oracle telling us which trajectories are violated (i.e., the accumulated cost is greater than  $c_{max}$ ). In this scenario, other constrained RL algorithms do not apply, as they all rely on the cost function. On the other hand, our algorithm utilizes the identification of good and bad trajectories.

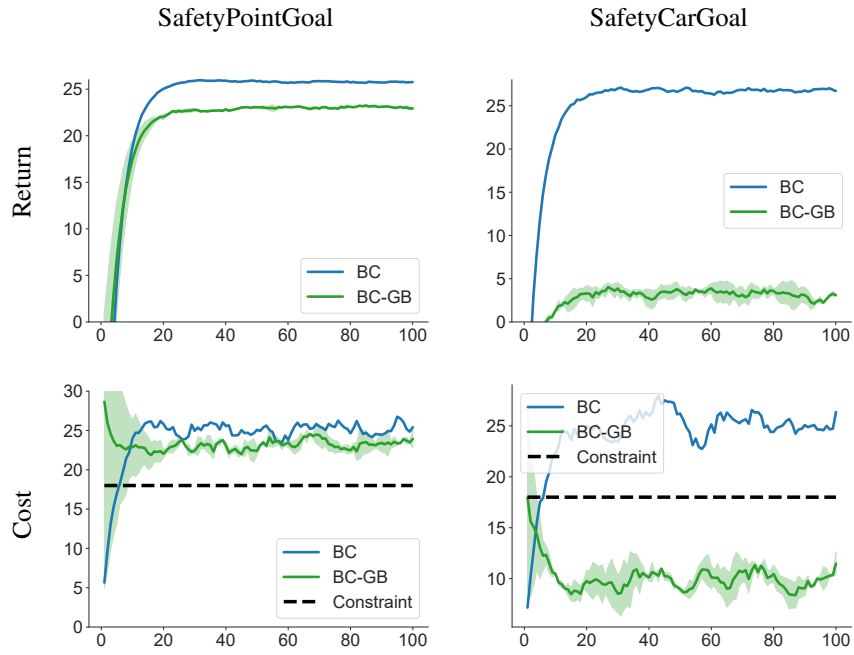


Figure 12: Training Curves of BC and BC-GB

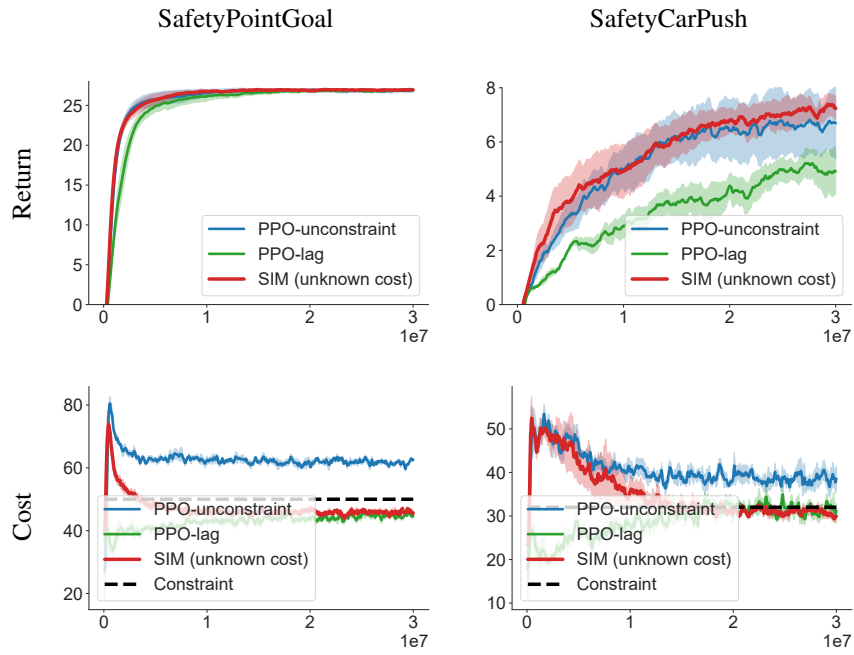


Figure 13: Results for the unknown-cost scenario.

Hence, it can be employed directly with the oracle’s assistance. However, it’s worth noting that in this situation, the oracle only aids in identifying bad trajectories, and the accessibility to good trajectories might be less potent compared to scenarios with a known cost function. We test our method on two Safety-Gym environments: *SafetyPointGoal* and *SafetyCarPush*. Since other constrained RL algorithms can be used, we just compare our algorithm with *PPO-unconstrained* and *PPO-Lag*, where the later still works with the cost function. We use *PPO-unconstrained* to train the initial policy  $\pi^0$ . The results shown in Figure 13 indicate that SIM is able to give safe policies while offering competitive expected rewards.

The ability to work with an unknown-cost setting to improve the safety of unconstrained policies would be valuable in may real-life situations where costs might be difficult or even impossible to get. This enhanced adaptability opens up new opportunities for applying RL in real-world settings.

### C.4 Enhancing Unconstrained Agent

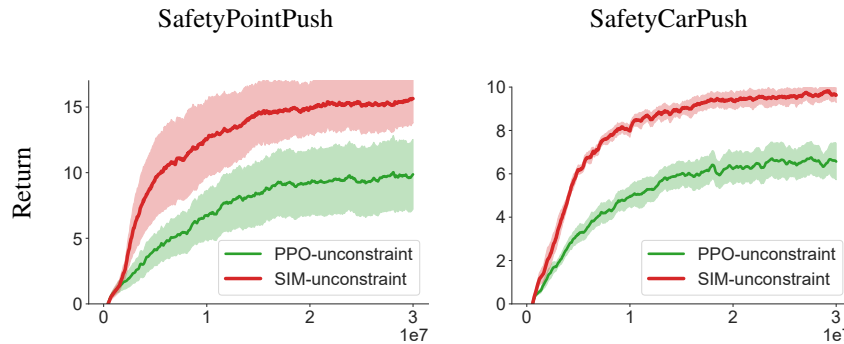


Figure 14: Comparison results for unconstrained tasks.

In this experiment, we want to answer **Q7** (would an unconstrained problem benefit from our approach?). We aim to see if SIM can improve the quality of a policy trained by an unconstrained RL algorithm (e.g., *PPO-unconstrained*). To this end, we choose two Safety-Gym environments *SafetyPointPush* and *SafetyCarPush* and set the threshold  $R_G$  for them as 11.0 and 8.0, respectively. The comparison results are shown in Figure 14, which show that algorithm was successful in significantly improving PPO under such unconstrained settings. In fact, by removing the constraint, our algorithm was able to focus solely on maximizing the reward without worrying about the costs associated with its actions. As a result, it could explore more freely and achieve better results in challenging scenarios.

Overall, this experiment showcases the efficacy of our approach in enhancing the performance of an unconstrained agent, especially in tasks where achieving high returns is challenging.

### C.5 Conditional Value at Risk

So far, we have focused on expected cost constraints. In this section, we expand our experiments to CVaR constraints (Yang et al. 2021) (to address **Q8** - would our approach work with CVaR constrained problems?). We implemented a SIM version that works with CVaR constraints by using the following criteria to select good trajectories:  $R(\tau) \geq R_G$  and  $C(\tau) + \alpha^{-1}\phi(\Phi^{-1}(\alpha))\sigma(C) \leq c_{max}$ . Here,  $\alpha = 0.5$  represents the risk level,  $\phi$  and  $\Phi$  denote the probability density function (PDF) and cumulative distribution function (CDF) of the standard normal distribution, respectively, and  $\sigma(C)$  is the standard deviation of the cost of the collected trajectories. We compare our approach with WC-SAC (Yang et al. 2021) (a state-of-the-art CVaR constrained RL algorithm). Additionally, we implement a PPO version with CVaR constrained (denoted as PPO-CVaR) for the sake of comparison.

The comparison results are shown in Figure 15. Interestingly, the original version of the WC-SAC struggled to achieve satisfactory results. However, PPO-CVaR approach performed exceptionally well in both environments, achieving improved performance while still maintaining lower costs than PPO-Lagrangian. Furthermore, our algorithm SIM (CVaR) outperformed all other curves, achieving the same expected cost while offering even higher expected rewards. This indicates the superior performance and effectiveness of our proposed approach compared to the other baseline methods considered. Overall, our experiments demonstrate that incorporating CVaR and SIM significantly enhances the performance of prior algorithms.

### C.6 Low-quality Initial Policy

In practical scenarios, there is no guarantee that an initial policy would be able to generate a sufficient set of good trajectories. In particular, a low-quality initial policy would even struggle with exploring good actions. To showcase such a situation, we run our SIM with six different random initial policies and plot their training curves in Figure 16 clearly shows that SIM was unable to achieve high rewards for Seed #3. This problem have raised a question

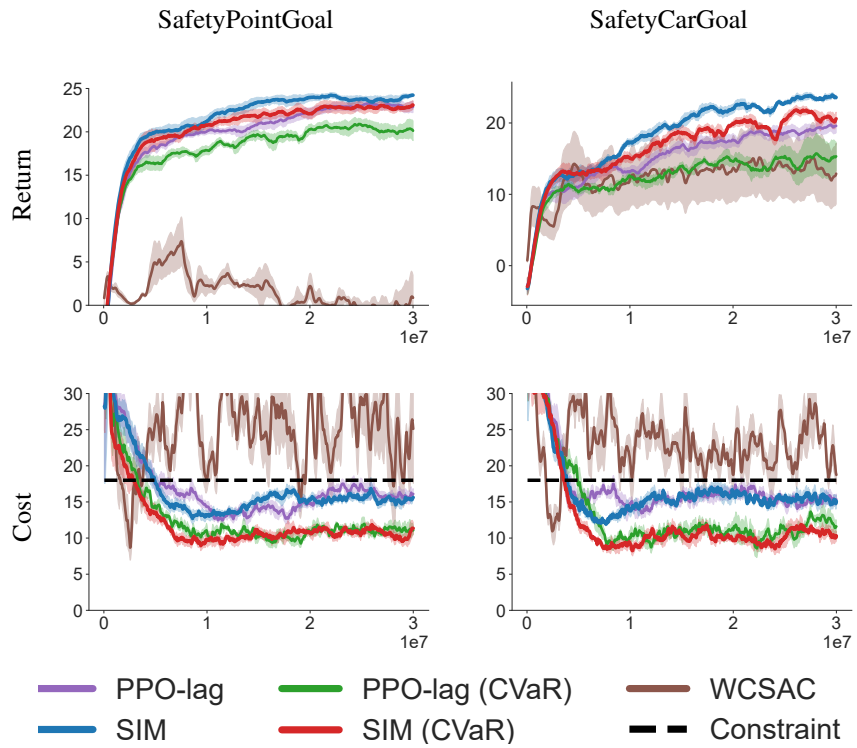


Figure 15: Comparison results with CVaR constraints.

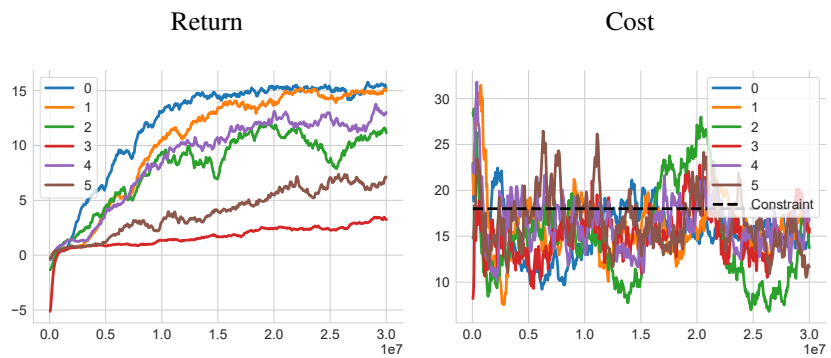


Figure 16: Results of 6 different seeds of SIM in SafetyPointPush.

Taking the above into consideration, we will show below that the issue can be addressed by using dynamic thresholds  $R_G$ . Our approach is to dynamically adjust the “good” threshold  $R_G$  during each update step to incentivize the policy to perform at par with the highest-ranking of return trajectories within the collected trajectory set  $T$ :  $R_G = \mathbb{E}_{\tau \sim T}[R(\tau)] + 2\sigma_{\tau \sim T}[R(\tau)]$ . The comparison of the original SIM and SIM with dynamic  $R_G$  for Seed #3 is shown in Figure 17, which clearly indicates the superiority of the dynamic SIM, compared to the static version (as well as the PPO baseline). Notably, it’s essential to acknowledge that due to reducing the threshold  $R_G$ , the training can achieve higher rewards when generating good trajectories is challenging. However, when the highest  $R_G$  is attainable, it can not replicate the performance of fixed  $R_G$ .

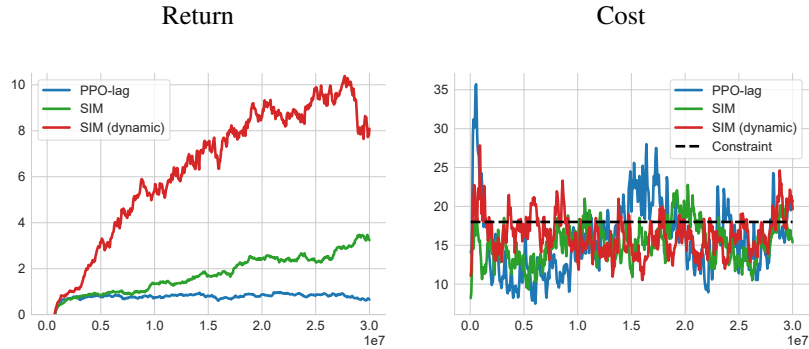


Figure 17: dynamic experiment in SafetyPointPush.

### C.7 Experiments on Mujoco domains

**Mujoco Circle** In this section, we present additional comparisons on Mujoco-Circle environments (Achiam et al. 2017; Ji et al. 2023). Figure 18 shows our comparison results, which clearly compare the strength of SIM over other baseline methods.

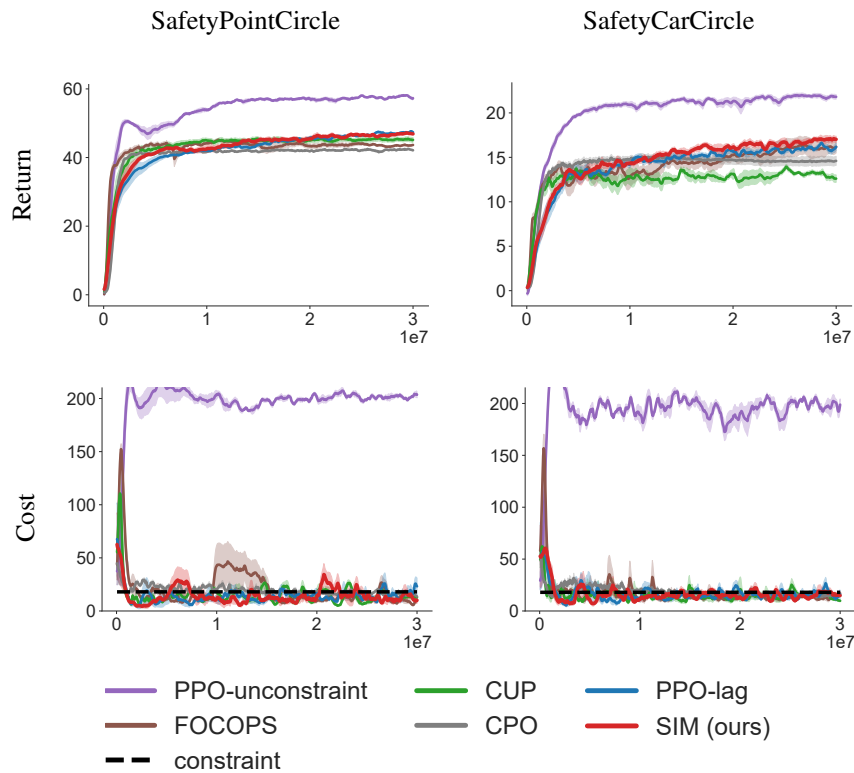


Figure 18: Results for *Mujoco Circle* environments.



**Mujoco Velocity** We further provide experiments on Mujoco-Velocity environments. The performance curves reported in Figure 19 shows our comparison results, which clearly demonstrate the superiority of SIM over other baseline methods in satisfying the constraint during the training as well as having a high return.

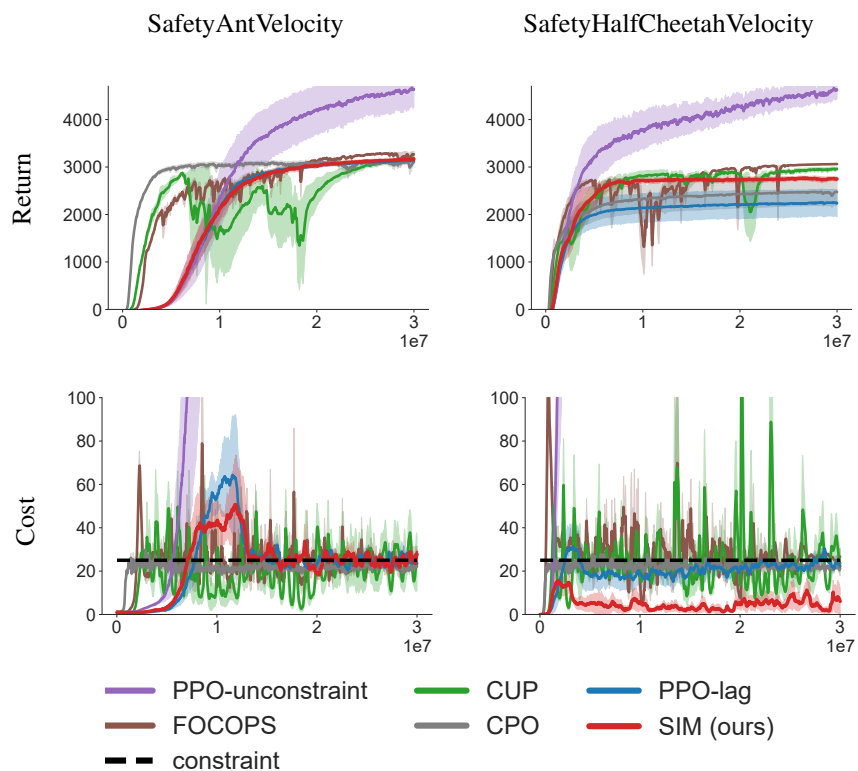


Figure 19: Results for *Mujoco Velocity* environments.