1 Notation

r	reward function
h	safety loss function
V^π	cumulative discounted reward value function
V_c^π	cumulative discounted cost value function
V_h^π	reachability value function
$\mathbb{1}_{s\in\mathcal{S}_v}$	instantaneous violation indicator function
ϕ^π	reachability estimation function (REF) of a given policy
ϕ^*	reachability estimation function (REF) of safest policy
p	predicted reachability estimation function (REF) of safest policy
$H_{\max}$	upper bound of function h
$H_{\min}$	minimum non-zero value of of function h
$\lambda_{\max}$	maximum value to clip lagrange multiplier
$R_{\max}$	upper bound of function r
$\mathcal{S}_s$	safe set
$\mathcal{S}_v$	unsafe set
$\mathcal{S}_f$	feasible set (persistent safe set)
$\mathbb{E}_{s'\sim\pi,P}$	expectation taken over possible next states
$\mathbb{E}_{\tau\sim\pi,P}$	expectation taken over possible trajectories
$\mathbb{E}_{s\sim d_0}$	expectation taken over initial distribution

Table 1: Notation used in the paper.

2 Gradient estimates

The Q value losses based on the MSE between the Q networks and the respective sampled returns result in the gradients:

$$\hat{\nabla}_\eta J_Q(\eta) = \nabla_\eta Q(s_t, a_t; \eta) \cdot [Q_\eta(s_t, a_t) - (r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}; \eta))]$$
$$\hat{\nabla}_\kappa J_{Q_c}(\kappa) = \nabla_\kappa Q_c(s_t, a_t; \kappa) \cdot [Q_\kappa(s_t, a_t) - (h(s_t) + \gamma Q_c(s_{t+1}, a_{t+1}; \kappa))]$$

Similarly the REF gradient update is:

$$\hat{\nabla}_\eta J_p(\xi) = \nabla_\xi p(s_t; \xi) \cdot [p(s_t; \xi) - \max\{\mathbb{1}_{s_t\in S_v}, \gamma p(s_{t+1}; \xi)\}]$$

From the policy gradient theorem in [1], we get the policy gradient loss as:

$$\hat{\nabla}_\theta J_\pi(\theta) = \gamma^t \Big[- Q_\eta(s_t, a_t)[1 - p_\xi(s_t)] + Q_c(s_t, a_t)[\lambda_\omega(1 - p_\xi(s_t)) + p_\xi(s_t)] \Big] \nabla_\theta \log \pi_\theta(a_t|s_t)$$

and the stochastic gradient of the multiplier is

$$\hat{\nabla}_\omega J_\lambda(\omega) = Q_c(s_t, a_t; \kappa)(1 - p_\xi(s_t))\nabla_\omega \lambda_\omega$$

and λ_ω is clipped to be in range $[0, \lambda_{\max}]$ (in particular, projection operator $\Gamma_\Omega(\lambda_\omega) = \arg\min_{\hat{\lambda}_\omega\in[0,\lambda_{\max}]} ||\lambda_\omega - \hat{\lambda}_\omega||^2$).

3 Proofs

3.1 Theorem 1 with Proof

Theorem 1. The REF can be reduced to the following recursive Bellman formulation:

$$\phi^\pi(s) = \max\{\mathbb{1}_{s\in\mathcal{S}_v}, \mathop{\mathbb{E}}_{s'\sim\pi,P(s)} \phi^\pi(s')\},$$

where $s' \sim \pi, P(s)$ is a sample of the immediate successive state (i.e., $s' \sim P(\cdot|s, a \sim \pi(\cdot|s))$) and the expectation is taken over all possible successive states.

Proof.

$$\begin{aligned}
\phi^\pi(s) &:= \mathop{\mathbb{E}}_{\tau\sim\pi,P(s)} \max_{s_t\in\tau} \mathbb{1}_{s_t^\pi\in S_v} \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi,P(s)} \max\{\mathbb{1}_{s\in S_v}, \max_{s_t\in\tau\setminus\{s\}} \mathbb{1}_{s_t^\pi\in S_v}\} \\
&= \max\{\mathbb{1}_{s\in S_v}, \mathop{\mathbb{E}}_{\tau\sim\pi,P(s)} \max_{s_t\in\tau\setminus\{s\}} \mathbb{1}_{s_t^\pi\in S_v}\} \\
&= \max\{\mathbb{1}_{s\in S_v}, \mathop{\mathbb{E}}_{s'\sim\pi,P(s)} \mathop{\mathbb{E}}_{\tau'\sim\pi,P(s')} \max_{s_t\in\tau'} \mathbb{1}_{s_t^\pi\in S_v}\} \\
&= \max\{\mathbb{1}_{s\in S_v}, \mathop{\mathbb{E}}_{s'\sim\pi,P(s)} \phi^\pi(s')\}
\end{aligned}$$

Note that we use the notation $\tau \sim \pi, P(s)$ to indicate a trajectory sampled from the MDP with transition probability P under policy π starting from state s, and use the notation $s' \sim \pi, P(s)$ to indicate the next immediate state from the MDP with transition probability P under policy π starting from state s. The third line holds because the indicator function is either 0 or 1, so if it's 1 then $\phi^\pi(s) = \mathbb{E}_{\tau\sim\pi,P(s)}\, 1 = 1$ else $\phi^\pi(s) = \mathbb{E}_{\tau\sim\pi,P(s)} \max_{s_t\in\tau\setminus\{s\}} \mathbb{1}_{s_t^\pi\in S_v}$.

□

3.2 Proposition 1 with Proof

Proposition 1. The cost value function $V_c^\pi(s)$ is zero for state s if and only if the persistent safety is guaranteed for that state under the policy π.

Proof. (IF) Assume for a given policy π, the persistent safety is guaranteed, i.e. $h(s_t|s_0 = 0, \pi) = 0$ holds for all $s_t \in \tau$ for all possible trajectories τ sampled from the environment with control policy π. We then have:

$$V_c^\pi(s) := \mathop{\mathbb{E}}_{\tau\sim\pi,P(s)}[\sum_{s_t\in\tau} \gamma^t h(s_t)] = 0.$$

(ONLY IF) Assume for a given policy π, $V_c^\pi(s) = 0$. Since the image of the safety loss function $h(s)$ is non-negative real, and $V_c^\pi(s)$ is the expectation of the sum of non-negative real values, the only way $V_c^\pi(s) = 0$ is if $h(s_t|s_0 = 0, \pi) = 0, \forall s_t \in \tau$ for all possible trajectories τ sampled from the environment with control policy π. □

3.3 Proposition 2 with Proof

Proposition 2. If $\exists\pi$ that produces trajectory $\tau = \{(s_i), i \in \mathbb{N}, s_1 = s\}$ in deterministic MDP $\mathcal{M}$ starting from state s, and $\exists m \in \mathbb{N}, m < \infty$ such that $s_m \in S_f^\pi$, then $\exists\epsilon > 0$ where if discount factor $\gamma \in (1-\epsilon, 1)$, then the optimal policy π^* of Main paper Equation 3 will produce a trajectory $\tau' = \{(s'_j), j \in \mathbb{N}, s'_1 = s\}$, such that $\exists n \in \mathbb{N}, n < \infty$, $s'_n \in S_f^{\pi^*}$ and $V_c^{\pi^*}(s) = \min_{\pi'} V_c^{\pi'}(s)$.

In other words the proposition is stating for some state s, if there is a policy that enters its feasible set in a finite number $(m-1)$ of steps, then by ensuring discount factor γ is close to 1 we can guarantee that the optimal policy π^* of Main paper Equation 3 will also enter the feasible set in a finite number of steps with the minimum cumulative discounted sum of the costs. Note that π^* will always produce trajectories with the minimum discounted sum of costs whether the state is in the feasible or infeasible set of the policy by virtue of its optimization which constrains V_c^π.

Proof. We consider two cases: (Case 1) $m = 1$ and (Case 2) $m > 1$.

Case 1 $m = 1$**:** In this case, there exists a policy π in which the the current state s is in the feasible set of that policy. By definition, that means that in a trajectory τ sampled in the MDP using that

policy, starting from state s, there are no future violations incurred in τ. Thus $V_c^{\pi}(s) = 0$. Since π^* incurs the minimum cumulative violation, $V_c^{\pi^*}(s) = 0$ trivially. Therefore, s, the first state of the trajectory, is in the feasible set of π^*.

Case 2 $m > 1$**:** Since policy π^* produces the minimum cumulative discounted cost for a given state s, the core of this proof will be demonstrating that the minimum cumulative discounted cost of *entering* the feasible set (call this value H_E) is less than the minimum cumulative discounted cost of *not entering* the feasible set (call this value H_N), and therefore π^* will choose the route of entering the feasible set.

The proof will proceed by deriving a sufficient condition for $H_E < H_N$ by establishing bounds on them.

We place an upper bound on the minimum cumulative discounted cost of entering the feasible set H_E. Since $\exists\pi$ that enters the feasible set in $m-1$ steps, entering the feasible set can be at most the highest possible cost that π incurs. Since the maximum cost at any state is $H_{\max}$, the upper bound is the discounted sum of $m-1$ steps of violations $H_{\max}$, or

$$H_E < \frac{H_{\max}(1-\gamma^{m-1})}{(1-\gamma)}$$

We place a lower bound on the minimum cumulative discounted cost of not entering the feasible set H_N. In this case, say in the sampled trajectory, the maximum gap between any two non-zero violations is w. By definition, the trajectory cannot have an infinite sequence of violation-free states since the trajectory never enters the feasible set. Therefore w is finite. Now recall $H_{\min}$ is the lower bound on the non-zero values of h. So the minimum cumulative discounted cost of not entering the feasible set must be at least the cost of the trajectory with a violation of $H_{\min}$ at intervals of w steps. That is:

$$\frac{H_{\min}(\gamma^w)}{(1-\gamma^w)} < H_N$$

Now $H_E < H_N$ will be true if the upper bound of H_E is less than the lower bound of H_N. In other words $H_E < H_N$ is true if:

$$\frac{H_{\max}(1-\gamma^{m-1})}{(1-\gamma)} < \frac{H_{\min}(\gamma^w)}{(1-\gamma^w)} \tag{1}$$

Rearranging, we get:

$$\frac{H_{\max}}{H_{\min}} < \frac{(1-\gamma)\cdot(\gamma^w)}{(1-\gamma^{m-1})\cdot(1-\gamma^w)} \tag{2}$$

Let's define the RHS of the Inequality 2 as the function $\upsilon(\gamma)$. Consider $\gamma \in (0,1)$. It is not difficult to demonstrate that $\upsilon(\gamma)$ in this domain range is a continuous function and that left directional limit $\lim_{\gamma\to 1^-} \upsilon(\gamma) = \infty$. This suggests that there is an open interval of values for γ (whose supremum is 1) for which $H_{\max}/H_{\min} < \upsilon(\gamma)$ and so $H_E < H_N$. So we establish that $\exists\epsilon > 0$ such that for $\gamma \in (1-\epsilon, 1)$, we satisfy the sufficient condition $H_E < H_N$ so that the optimal policy will enter its feasible set.

Thus, we prove that if there is a policy entering its feasible set from state s, then there is a range of values for γ that are close enough to 1 ensuring that the optimal policy of Main paper Equation 3 will enter its feasible set in a finite number of steps with minimum discounted sum of costs.

□

3.4 Theorem 2 with Proof

Theorem 2. Given Assumptions **A1**-**A3** in Main paper, the policy updates in Algorithm 1 will almost surely converge to a locally optimal policy for our proposed optimization in Equation RESPO.

We first provide an intuitive explanation behind why our REF learns to converge to the safest policy's REF, then a proof overview, and then the full proof.

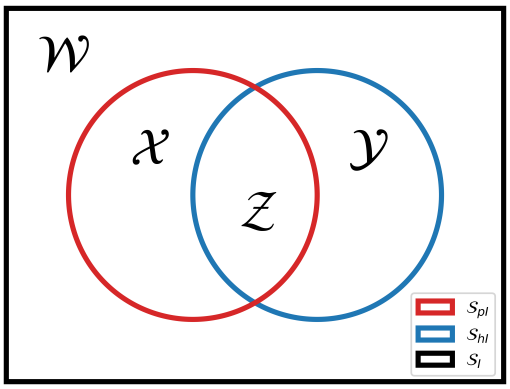

Figure 1: The predicted feasible set converges to a safest policy's feasible set since the misclassified regions $\mathcal{X}$ and $\mathcal{Y}$ are corrected over time.

3.4.1 Intuition behind REF convergence

The approach can be explained by considering what happens in the individual regions of space. Consider a deterministic environment for simplicity. As seen in Figure 1, there are two subsets of the initial state space: a safest policy's "true" feasible set $\mathcal{S}_{hI}$ and REF predicted feasible set $\mathcal{S}_{pI}$, and they create 4 regions in the initial state space $\mathcal{S}_I$: $\mathcal{W} = \overline{\mathcal{S}_{hI}} \cap \overline{\mathcal{S}_{pI}}$, $\mathcal{X} = \overline{\mathcal{S}_{hI}} \cap \mathcal{S}_{pI}$, $\mathcal{Y} = \mathcal{S}_{hI} \cap \overline{\mathcal{S}_{pI}}$, $\mathcal{Z} = \mathcal{S}_{hI} \cap \mathcal{S}_{pI}$. Consider a point during training when the lagrange multiplier λ is sufficiently large. For states in $\mathcal{W}$, the set of correctly classified infeasible states, the algorithm will simply minimize cumulative violations $V_c^{\pi_\theta}(s)$, and thereby remain as safe as possible since the policy and critics learning rates are faster than that of REF. $\mathcal{X}$, which is the set of infeasible states that are misclassified, is very small if we ensure the policy and REF are trained at much faster time scales than the multiplier and so when the agent starts in true infeasible states, it will by definition reach violations and therefore be labeled as infeasible. In $\mathcal{Y}$, the set of truly feasible states that are misclassified, the algorithm also minimizes cumulative violations, which by the definition of feasibility should be 0. It will then have no violations and enter the correctly predicted feasible set $\mathcal{Z}$. And when starting in states in $\mathcal{Z}$, the algorithm will optimize the lagrangian, and since the multiplier λ is sufficiently large, it will converge to a policy that optimizes for reward while ensuring safety, i.e. no future violations, and therefore the state will stay predictably feasible in $\mathcal{Z}$. In this manner, REF's predicted feasible set will converge to the optimal feasible set, and the agent will be safe and have optimal performance in the feasible set and be the safest behavior outside the feasible set. Thereby, the algorithm finds a locally optimal solution to the proposed optimization formulation.

3.4.2 Proof Overview

We show our algorithm convergence to the optimal policy by utilizing the proof framework of multi-time scale presented in [2, 3, 4, 5]. Specifically, we have 4 time scales for (1) the critics, (2) policy, (3) REF function, and (4) lagrange multiplier, listed in order from fastest to slowest. The overview of each timescale proof step is as follows:

1 We demonstrate the almost sure convergence of the critics to the corresponding fixed point optimal critic functions of the policy.

2 Using multi-timescale theory, we demonstrate the policy almost surely converges to a stationary point of a continuous time system, which we show has a Lyapunov function certifying its locally asymptotic stability at the stationary point.

3 We demonstrate the almost sure convergence of the REF function to the REF of the policy that is safe insofar as the lagrange multiplier is sufficiently large.

4 We demonstrate the almost sure convergence of the lagrange multiplier to a stationary point similar to the proof in the policy timecale.

Finally, we demonstrate that the stationary points for the policy and lagrange multiplier form a saddle point, and so by local saddle point theorem we almost surely achieve the locally optimal policy of our proposed optimization.

3.4.3 Proof Details

Proof. **Step 1** (convergence of the critics V_η and V_κ updates): From the multi-time scale assumption, we know that η and κ will convergence on a faster time scale than the other parameters θ, ξ, and ω. Therefore, we can leverage Lemma 1 of Chapter 6 of [3] to analyze the convergence properties while updating η_k and κ_k by treating θ, ξ, and ω as fixed parameters θ_k, ξ_k, and ω_k. In other words, the policy, REF, and lagrange multiplier are fixed while computing $Q^{\pi_{\theta_k}}(s,a)$ and $Q_c^{\pi_{\theta_k}}(s,a)$. With the Finite MDP assumption and policy evaluation convergence results of [1], and assuming sufficiently expressive function approximator (i.e. wide enough neural networks) to ensure convergence to global mininum, we can use the fact that the bellman operators $\mathcal{B}$ and $\mathcal{B}_c$ which are defined as

$$\mathcal{B}[Q](s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s',a' \sim \pi, P(s)}[Q(s',a')]$$
$$\mathcal{B}[Q_c](s,a) = h(s) + \gamma \mathop{\mathbb{E}}_{s',a' \sim \pi, P(s)}[Q_c(s',a')]$$

are γ-contraction mappings, and therefore as k approaches ∞, we can be sure that $Q(s,a;\eta_k) \to Q(s,a;\eta^*) = Q^{\pi_{\theta_k}}(s,a)$ and $Q_c(s,a;\kappa_k) \to Q_c(s,a;\kappa^*) = Q_c^{\pi_{\theta_k}}(s,a)$. So since η_k and κ_k converge to η^* and κ^*, we prove convergence of the critics in Time scale 1.

Step 2 (convergence of the policy π_θ update): Because ξ and ω updated on slower time scales than θ, we can again use Lemma 1 of Chapter 6 of [3] and treat these parameters are fixed at ξ_k and ω_k respectively when updating θ_k. Additionally in Time scale 2, we have $||Q(s,a;\eta_k) - Q(s,a;\eta^*)|| \to 0$ and $||Q_c(s,a;\kappa_k) - Q_c(s,a;\kappa^*)|| \to 0$ almost surely. Now the update of the policy θ using the gradient from Equation 4 is:

$$\begin{aligned}
\theta_{k+1} &= \Gamma_\Theta[\theta_k - \zeta_2(k)(\nabla_\theta L(\theta,\xi_k,\omega_k)|_{\theta=\theta_k})] \\
&= \Gamma_\Theta[\theta_k - \zeta_2(k)[\gamma^t[-Q_\eta(s_t,a_t)[1-p_{\xi_k}(s_t)] \\
&+ Q_c(s_t,a_t)[\lambda_\omega(1-p_{\xi_k}(s_t)) + p_{\xi_k}(s_t)]]\nabla_\theta \log \pi(a_t|s_t;\theta)|_{\theta=\theta_k}]] \\
&= \Gamma_\Theta[\theta_k - \zeta_2(k)(\nabla_\theta L(\theta,\xi_k,\omega_k)|_{\theta=\theta_k,\eta=\eta^*,\kappa=\kappa^*} + \delta\theta_{k+1} + \delta\theta_\epsilon)]
\end{aligned}$$

where

$$\begin{aligned}
\delta\theta_{k+1} = \sum_{s_i,a_i} \Big[& d_0(s_0) P^{\pi_{\theta_k}}(s_i,a_i|s_0)\gamma^i[-Q_\eta(s_i,a_i)[1-p_{\xi_k}(s_i)] \\
& + Q_c(s_i,a_i)[\lambda_\omega(1-p_{\xi_k}(s_i)) + p_{\xi_k}(s_i)]]\nabla_\theta \log \pi(a_i|s_i;\theta)|_{\theta=\theta_k} \Big] \\
& - \gamma^t[-Q_\eta(s_t,a_t)[1-p_{\xi_k}(s_t)] + Q_c(s_t,a_t)[\lambda_\omega(1-p_{\xi_k}(s_t)) + p_{\xi_k}(s_t)]] \\
& \cdot \nabla_\theta \log \pi(a_t|s_t;\theta)|_{\theta=\theta_k}
\end{aligned}$$

and

$$\begin{aligned}
\delta\theta_\epsilon = \sum_{s_i,a_i} d_0(s_0) P^{\pi_{\theta_k}}(s_i,a_i|s_0) \Big[& \\
-\gamma^i[-Q(s_i,a_i;\eta_k)[1-p_{\xi_k}(s_i)] + Q_c(s_i,a_i;\kappa_k)[\lambda_\omega(1-p_{\xi_k}(s_i)) + p_{\xi_k}(s_i)]] & \\
\cdot \nabla_\theta \log \pi(a_i|s_i;\theta)|_{\theta=\theta_k} & \\
+\gamma^i[-Q^{\pi_{\theta_k}}(s_i,a_i)[1-p_{\xi_k}(s_i)] + Q_c^{\pi_{\theta_k}}(s_i,a_i)[\lambda_\omega(1-p_{\xi_k}(s_i)) + p_{\xi_k}(s_i)]] & \\
\cdot \nabla_\theta \log \pi(a_i|s_i;\theta)|_{\theta=\theta_k} \Big] &
\end{aligned}$$

Lemma 1: We can first demonstrate that $\delta\theta_{k+1}$ is square integrable. In particular,

$$\begin{aligned}
&\mathbb{E}[||\delta\theta_{k+1}||^2|\mathcal{F}_{\theta,k}] \\
&\leq 2||\nabla_\theta \log\pi(a|s;\theta)|_{\theta=\theta_k}\mathbb{1}_{\pi(a|s;\theta_k)>0}||^2_\infty \cdot \Big(||Q(s,a;\eta_k)||^2_\infty \cdot ||1-p_{\xi_k}(s)||^2_\infty \\
&+ ||Q_c(s,a;\kappa_k)||^2_\infty \cdot \Big[||\lambda_\omega||^2_\infty \cdot ||1-p_{\xi_k}(s)||^2_\infty + ||p_{\xi_k}(s)||^2_\infty\Big]\Big) \\
&\leq 2\frac{||\nabla_\theta\pi(a|s;\theta)|_{\theta=\theta_k}||^2_\infty}{\min\{\pi(a|s;\theta_k)|\pi(a|s;\theta_k)>0\}} \cdot \Big(||Q(s,a;\eta_k)||^2_\infty \cdot ||1-p_{\xi_k}(s)||^2_\infty \\
&+ ||Q_c(s,a;\kappa_k)||^2_\infty \cdot \Big[||\lambda_\omega||^2_\infty \cdot ||1-p_{\xi_k}(s)||^2_\infty + ||p_{\xi_k}(s)||^2_\infty\Big]\Big)
\end{aligned}$$

Note that $\mathcal{F}_{\theta,k} = \sigma(\theta_m, \delta\theta_m, m \leq k)$ is the filtration for θ_k generated by different independent trajectories [2]. Also note that the indicator function is used because the expectation of $||\delta\theta_{k+1}||^2$ is taken with respect to $P^{\pi_{\theta_k}}$ and $P^{\pi_{\theta_k}}(s,a|s_0) = 0$ if $\pi(a|s;\theta_k) = 0$. From the Assumptions on Lipschitz continuity and Finite MDPs reward and costs, we can bound the values of the functions and the gradients of functions. Specifically

$$\begin{aligned}
||\nabla_\theta\pi(a|s;\theta)|_{\theta=\theta_k}||^2_\infty &\leq K_1(1+||\theta_k||^2_\infty), \\
||Q(s,a;\eta_k)||^2_\infty &\leq \frac{R_{\max}}{1-\gamma}, \\
||Q_h(s,a;\kappa_k)||^2_\infty &\leq \frac{H_{\max}}{1-\gamma}, \\
||\lambda_\omega||^2_\infty &\leq \lambda_{\max}, \\
||1-p_{\xi_k}(s)||^2_\infty &\leq 1, \\
||p_{\xi_k}(s)||^2_\infty &\leq 1
\end{aligned}$$

where K_1 is a Lipschitz constant. Furthermore, note that because we are sampling, $\pi(a|s;\theta_k)$ will take on only a finite number of values, so its nonzero values will be bounded away from zero. Thus we can say

$$\frac{1}{\min\{\pi(a|s;\theta_k)|\pi(a|s;\theta_k)>0\}} \leq K_2$$

for some large enough K_2. Thus using the bounds from these conditions, we can demonstrate

$$\mathbb{E}[||\delta\theta_{k+1}||^2|\mathcal{F}_{\theta,k}] \leq 2\cdot K_1(1+||\theta_k||^2_\infty)\cdot K_2(\frac{R_{\max}}{1-\gamma}\cdot 1 + \frac{H_{\max}}{1-\gamma}\cdot(\lambda_{\max}\cdot 1+1)) < \infty$$

Therefore $\delta\theta_{k+1}$ is square integrable.

Lemma 2: Secondly, we can demonstrate $\delta\theta_\epsilon \to 0$.

$$\begin{aligned}
\delta\theta_\epsilon &= \sum_{s_i,a_i} d_0(s_0)P^{\pi_{\theta_k}}(s_i,a_i|s_0)\Big[\gamma^i\big[(Q(s_i,a_i;\eta_k) - Q^{\pi_{\theta_k}}(s_i))[1-p_{\xi_k}(s_i)] \\
&+ (-Q_c(s_i,a_i;\kappa_k) + Q_c^{\pi_{\theta_k}}(s_i,a_i))[\lambda_\omega(1-p_{\xi_k}(s_i)) + p_{\xi_k}(s_i)]\big]\nabla_\theta\log\pi(a_i|s_i;\theta)|_{\theta=\theta_k}\Big] \\
&\leq \sum_{s_i,a_i} d_0(s_0)P^{\pi_{\theta_k}}(s_i,a_i|s_0)\Big[\gamma^i\big[(Q(s_i,a_i;\eta_k) - Q(s_i,a_i;\eta^*))[1-p_{\xi_k}(s_i)] \\
&+ (-Q_c(s_i,a_i;\kappa_k) + Q_c(s_i,a_i;\kappa^*))[\lambda_\omega(1-p_{\xi_k}(s_i)) + p_{\xi_k}(s_i)]\big]\nabla_\theta\log\pi(a_i|s_i;\theta)|_{\theta=\theta_k}\Big] \\
&\leq \sum_{s_i,a_i} d_0(s_0)P^{\pi_{\theta_k}}(s_i,a_i|s_0)\Big[\gamma^i\big[||Q(s_i,a_i;\eta_k) - Q(s_i,a_i;\eta^*)||[1-p_{\xi_k}(s_i)] \\
&+ ||-Q_c(s_i,a_i;\kappa_k) + Q_c(s_i,a_i;\kappa^*)||[\lambda_\omega(1-p_{\xi_k}(s_i)) + p_{\xi_k}(s_i)]\big]\nabla_\theta\log\pi(a_i|s_i;\theta)|_{\theta=\theta_k}\Big]
\end{aligned}$$

And because we have $||Q(s,a;\eta_k) - Q(s,a;\eta^*)|| \to 0$ and $||Q_c(s,a;\kappa_k) - Q_c(s,a;\kappa^*)|| \to 0$ almost surely, we can therefore say $\delta\theta_\epsilon \to 0$.

Lemma 3: Finally, since $\hat{\nabla}_\theta J_\pi(\theta)|_{\theta=\theta_k}$ is a sample of $\nabla_\theta L(\theta,\xi_k,\omega_k)|_{\theta=\theta_k}$ based on the history of sampled trajectories, we conclude that $\mathbb{E}[\delta\theta_{k+1}|\mathcal{F}_{\theta,k}] = 0$.

From the 3 above lemmas, the policy θ update is a stochastic approximation of a continuous system $\theta(t)$ defined by [3]

$$\dot{\theta} = \Upsilon_\Theta[-\nabla_\theta L(\theta,\xi,\omega)] \tag{3}$$

in which

$$\Upsilon_\Theta[M(\theta)] \triangleq \lim_{0<\psi\to 0} \frac{\Gamma_\Theta(\theta+\psi M(\theta)) - \Gamma_\Theta(\theta)}{\psi}$$

or in other words the left directional derivative of $\Gamma_\Theta(\theta)$ in the direction of $M(\theta)$. Using the left directional derivative $\Upsilon_\Theta[-\nabla_\theta L(\theta,\xi,\omega)]$ in the gradient descent algorithm for learning the policy π_θ ensures the gradient will point in the descent direction along the boundary of Θ when the θ update hits its boundary. Using Step 2 in Appendix A.2 from [2], we have that $dL(\theta,\xi,\omega)/dt = -\nabla_\theta L(\theta,\xi,\omega)^T \cdot \Upsilon_\Theta[-\nabla_\theta L(\theta,\xi,\omega)] \leq 0$ and the value is non-zero if $||\Upsilon_\Theta[-\nabla_\theta L(\theta,\xi,\omega)]|| \neq 0$. Now consider the continuous system $\theta(t)$. For some fixed ξ and ω, define a Lyapunov function

$$\mathcal{L}_{\xi,\omega}(\theta) = L(\theta,\xi,\omega) - L(\theta^*,\xi,\omega)$$

where θ^* is a local minimum point. Then there exists a ball centered at θ^* with a radius ρ such that $\forall\theta \in \mathfrak{B}_{\theta^*}(\rho) = \{\theta | ||\theta-\theta^*|| \leq \rho\}$, $\mathcal{L}_{\xi,\omega}(\theta)$ is a locally positive definite function, that is $\mathcal{L}_{\xi,\omega}(\theta) \geq 0$. Using Proposition 1.1.1 from [6], we can show that $\Upsilon_\Theta[-\nabla_\theta L(\theta,\xi,\omega)]|_{\theta=\theta^*} = 0$ meaning θ^* is a stationary point. Since $dL(\theta,\xi,\omega)/dt \leq 0$, through Lyapunov theory for asymptotically stable systems presented in Chapter 4 of [7], we can use the above arguments to demonstrate that with any initial conditions of $\theta(0) \in \mathfrak{B}_{\theta^*}(\rho)$, the continuous state trajectory of $\theta(t)$ converges to θ^*. Particularly, $L(\theta^*,\xi,\omega) \leq L(\theta(t),\xi,\omega) \leq L(\theta(0),\xi,\omega)$ for all $t > 0$.

Using these aforementioned properties, as well as the facts that 1) $\nabla_\theta L(\theta,\xi,\omega)$ is a Lipschitz function (using Proposition 17 from [2]), 2) the step-sizes of Assumption on steps sizes, 3) $\delta\theta_{k+1}$ is a square integrable Martingale difference sequence and $\delta\theta_\epsilon$ is a vanishing error almost surely, and 4) $\theta_k \in \Theta, \forall k$ implying that $\sup_k ||\theta_k|| < \infty$ almost surely, we can invoke Theorem 2 of chapter 6 in [3] to demonstrate the sequence $\{\theta_k\}, \theta_k \in \Theta$ converges almost surely to the solution of the ODE defined by Equation 3, which additionally converges almost surely to the local minimum $\theta^* \in \Theta$.

Step 3 (convergence of REF p_ξ updates): Since ω is updated on a slower time scale that ξ, we can again treat ω as a fixed parameter at ω_k when updating ξ. Furthermore, in Time scale 3, we know that the policy has converged to a local minimum, particularly $||\theta_k - \theta^*(\xi_k,\omega_k)|| = 0$. Now the bellman operator for REF is defined by

$$\mathcal{B}_p[p](s) = \max\{\mathbb{1}_{s\in S_v}, \gamma \mathop{\mathbb{E}}_{s'\sim\pi,P(s)}[p(s')]\}.$$

We demonstrate this is a γ contraction mapping as follows:

$$\begin{aligned}
&|\mathcal{B}_p[p](s) - \mathcal{B}_p[\hat{p}](s)| \\
&= |\max\{\mathbb{1}_{s\in S_v}, \gamma \mathop{\mathbb{E}}_{s'\sim\pi,P(s)}[p(s')]\} - \max\{\mathbb{1}_{s\in S_v}, \gamma \mathop{\mathbb{E}}_{s'\sim\pi,P(s)}[\hat{p}(s')]\}| \\
&\leq |\gamma \mathop{\mathbb{E}}_{s'\sim\pi,P(s)}[p(s')] - \gamma \mathop{\mathbb{E}}_{s'\sim\pi,P(s)}[\hat{p}(s')]| \\
&= \gamma |\mathop{\mathbb{E}}_{s'\sim\pi,P(s)}[p(s') - \hat{p}(s')]| \\
&\leq \gamma \sup_s |p(s) - \hat{p}(s)| = \gamma ||p - \hat{p}||_\infty
\end{aligned}$$

So we can say that $p(s;\xi_k)$ will converge to $p(s;\xi^*)$ as $k \to \infty$ under the same assumptions of the Finite MDP and function approximator expressiveness in Step 1. Therefore, π_{θ_k} will also converge to $\pi^\diamond = \pi_{\theta^*(\xi^*,\omega_k)}$ as $k \to \infty$. And because π_θ is the sampling policy used to compute p, $p(s;\xi^*) = p^{\pi_{\theta^*(\xi^*,\omega_k)}}(s;\xi^*) = p^\diamond(s)$.

Notice that $\pi^\diamond$ is a locally minimum optimal policy for the following optimization (recall λ_ω is treated as constant in this timescale):

$$\min_\pi \mathop{\mathbb{E}}_{s\sim d_0} \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} \Big[-Q^\pi(s,a)\cdot[1-p^\diamond(s)] + Q_c^\pi(s,a)\cdot[(1-p^\diamond(s))\lambda_\omega + p^\diamond(s)]\Big]$$

and therefore also locally minimum optimal policy for optimization:

$$\min_\pi \mathop{\mathbb{E}}_{s\sim d_0} \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} \Big[-Q^\pi(s,a) + Q_c^\pi(s,a)\cdot[\lambda_\omega + \frac{p^\diamond(s)}{(1-p^\diamond(s))}]\Big], \text{ if } p^\diamond(s) > 0$$

$$\mathop{\mathbb{E}}_{s\sim d_0} \min_\pi \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} \Big[Q_c^\pi(s,a)\Big], \text{ if } p^\diamond(s) = 0$$

Since $\frac{p^\diamond(s)}{(1-p^\diamond(s))} \geq 0$, and the Q functions are always nonnegative, we can know that $\pi^\diamond$ is at least as safe as (i.e., its expected cumulative cost is at most that of) a locally optimal policy for the optimization:

$$\min_\pi \mathop{\mathbb{E}}_{s\sim d_0} \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} \Big[-Q^\pi(s,a) + Q_c^\pi(s,a)\lambda_\omega\Big] \tag{4}$$

As λ_ω approaches $\lambda_{\max}$, which in turn approaches ∞, the local minimum optimal policies of Equation 4 approach those of the optimization $\pi^\triangle = \arg\min_\pi \mathbb{E}_{s\sim d_0}\mathbb{E}_{a\sim\pi(\cdot|s)} Q_c^\pi(s,a)\lambda_\omega = \arg\min_\pi \mathbb{E}_{s\sim d_0}\mathbb{E}_{a\sim\pi(\cdot|s)} Q_c^\pi(s,a)$. Therefore, the feasible set of the REF $p^\diamond$ will approach that of the REF $p^{\pi^\triangle}$.

Step 4 (convergence of lagrange multiplier λ_ω update): Since λ_ω is on the slowest time scale, we have that $||\theta_k - \theta^*(\omega)|| = 0$, $||\xi_k - \xi^*(\omega)|| = 0$, and $||Q_c(s,a;\kappa_k) - Q_c^{\pi_{\theta_k}}(s,a)|| = 0$ almost surely. Furthermore, due to the continuity of $\nabla_\omega L(\theta,\xi,\omega)$, we have that $||\nabla_\omega L(\theta,\xi,\omega)|_{\theta=\theta_k,\xi=\xi_k,\omega=\omega_k} - \nabla_\omega L(\theta,\xi,\omega)|_{\theta=\theta^*(\omega_k),\xi=\xi^*(\omega_k),\omega=\omega_k}|| = 0$ almost surely. The update of the multiplier using the gradient for Equation is:

$$\begin{aligned}
\omega_{k+1} &= \Gamma_\Omega[\omega_k + \zeta_4(k)(\nabla_\omega L(\theta,\xi,\omega)|_{\theta=\theta_k,\xi=\xi_k,\omega=\omega_k})] \\
&= \Gamma_\Omega[\omega_k + \zeta_4(k)(Q_c(s_t,a_t;\kappa_k)[1-p(s_t;\xi_k)]\nabla_\omega\lambda_\omega|_{\omega=\omega_k})] \\
&= \Gamma_\Omega[\omega_k + \zeta_4(k)(\nabla_\omega L(\theta,\xi,\omega)|_{\theta=\theta^*(\omega_k),\xi=\xi^*(\omega_k),\omega=\omega_k} + \delta\omega_{k+1})]
\end{aligned}$$

where

$$\begin{aligned}
\delta\omega_{k+1} &= -\nabla_\omega L(\theta,\xi,\omega)|_{\theta=\theta^*(\omega_k),\xi=\xi^*(\omega_k),\omega=\omega_k} + Q_c(s_t,a_t;\kappa_k)[1-p(s_t;\xi_k)]\nabla_\omega\lambda_\omega|_{\omega=\omega_k} \\
&= -\sum_{s_i,a_i} d_0(s_0)P^{\pi_{\theta_k}}(s_i,a_i|s_0)[Q_c^{\pi_{\theta^*}}(s_i,a_i)[1-p_{\xi^*}(s_i)]\nabla_\omega\lambda_\omega|_{\omega=\omega_k}] \\
&+ Q_c(s_t,a_t;\kappa_k)[1-p(s_t;\xi_k)]\nabla_\omega\lambda_\omega|_{\omega=\omega_k} \\
&= -\sum_{s_i,a_i} d_0(s_0)P^{\pi_{\theta_k}}(s_i,a_i|s_0)[Q_c^{\pi_{\theta^*}}(s_i,a_i)[1-p_{\xi^*}(s_i)]\nabla_\omega\lambda_\omega|_{\omega=\omega_k}] \\
&+ [Q_c(s_t,a_t;\kappa_k)[1-p(s_t;\xi_k)] - Q_c^{\pi_{\theta_k}}(s_t,a_t)[1-p(s_t;\xi_k)]+ \\
&Q_c^{\pi_{\theta_k}}(s_t,a_t)[1-p(s_t;\xi_k)] - Q_c^{\pi_{\theta_k}}(s_t,a_t)[1-p^\diamond(s_t)]+ \\
&Q_c^{\pi_{\theta_k}}(s_t,a_t)[1-p^\diamond(s_t)]]\nabla_\omega\lambda_\omega|_{\omega=\omega_k} \\
&= -\sum_{s_i,a_i} d_0(s_0)P^{\pi_{\theta_k}}(s_i,a_i|s_0)[Q_c^{\pi_{\theta^*}}(s_i,a_i)[1-p_{\xi^*}(s_i)]\nabla_\omega\lambda_\omega|_{\omega=\omega_k}] \\
&+ [(Q_c(s_t,a_t;\kappa_k) - Q_c^{\pi_{\theta_k}}(s_t,a_t))[1-p(s_t;\xi_k)]+ \\
&Q_c^{\pi_{\theta_k}}(s_t,a_t)[p^\diamond(s_t) - p(s_t;\xi_k)]+ \\
&Q_c^{\pi_{\theta_k}}(s_t,a_t)[1-p^\diamond(s_t)]]\nabla_\omega\lambda_\omega|_{\omega=\omega_k}]
\end{aligned}$$

Now, just as in the θ update convergence, we can demonstrate the following lemmas:

Lemma 4: $\delta\omega_{k+1}$ is square integrable since

$$\mathbb{E}[||\delta\omega_{k+1}||^2|\mathcal{F}_{\omega,k}] \leq 2\cdot\frac{H_{\max}}{1-\gamma}\cdot 1\cdot K_3(1+||\omega_k||_\infty^2) < \infty$$

for some large Lipschitz constant K_3. Note that $\mathcal{F}_{\omega,k} = \sigma(\omega_m, \delta\omega_m, m \leq k)$ is the filtration for ω_k generated by different independent trajectories [2].

Lemma 5: Because $||Q_c(s_t, a_t; \kappa_k) - Q_c^{\pi_{\theta_k}}(s_t, a_t)||_\infty \to 0$ and $||p^\diamond(s_t) - p(s_t; \xi_k)||_\infty \to 0$ and $Q_c^{\pi_{\theta_k}}(s_t, a_t)[1 - p_{\xi^*}(s_t)]\nabla_\omega \lambda_\omega|_{\omega=\omega_k}$ is a sample of $Q_c^{\pi_{\theta^*}}(s_i, a_i)[1 - p_{\xi^*}(s_i)]\nabla_\omega \lambda_\omega|_{\omega=\omega_k}$, we conclude that $\mathbb{E}[\delta\omega_{k+1}|\mathcal{F}_{\omega,k}] = 0$ almost surely.

Thus, the lagrange multiplier ω update is a stochastic approximation of a continuous system $\omega(t)$ defined by [3]

$$\dot{\omega} = \Upsilon_\Omega[-\nabla_\omega L(\theta, \xi, \omega)|_{\theta=\theta^*(\omega),\xi=\xi^*(\omega)}] \tag{5}$$

with Martingale difference error of $\delta\omega_k$ and where Υ_Ω is the left direction derivative defined similar to that in Time scale 2 of the convergence of θ update. Using Step 2 in Appendix A.2 from [2], we have that $dL(\theta^*(\omega), \xi^*(\omega), \omega)/dt = \nabla_\omega L(\theta, \xi, \omega)|_{\theta=\theta^*(\omega),\xi=\xi^*(\omega)}{}^T \cdot \Upsilon_\Omega[\nabla_\Omega L(\theta, \xi, \omega)|_{\theta=\theta^*(\omega),\xi=\xi^*(\omega)}] \geq 0$ and the value is non-zero if $||\Upsilon_\Omega[\nabla_\omega L(\theta, \xi, \omega)|_{\theta=\theta^*(\omega),\xi=\xi^*(\omega)}]|| \neq 0$.

For a local maximum point ω^*, define a Lyapunov function as

$$\mathcal{L}(\omega) = L(\theta^*(\omega), \xi^*(\omega), \omega^*) - L(\theta^*(\omega), \xi^*(\omega), \omega)$$

Then there exists a ball centered at ω^* with a radius ρ' such that $\forall\omega \in \mathfrak{B}_{\omega^*}(\rho') = \{\omega|||\omega - \omega^*|| \leq \rho'\}$, $\mathcal{L}(\omega)$ is a locally positive definite function, that is $\mathcal{L}(\omega) \geq 0$. Also, $d\mathcal{L}(\omega(t))/dt = -dL(\theta^*(\omega), \xi^*(\omega), \omega)/dt \leq 0$ and is equal only when $\Upsilon_\Omega[\nabla_\omega L(\theta, \xi, \omega)|_{\theta=\theta^*(\omega),\xi=\xi^*(\omega)}] = 0$, so therefore ω^* is a stationary point. By leveraging Lyapunov theory for asymptotically stable systems presented in Chapter 4 of [7] we can demonstrate that for any initial conditions of $\omega \in \mathfrak{B}_{\omega^*}(\rho')$, the continuous state trajectory of $\omega(t)$ converges to the locally maximum point ω^*.

Using these aforementioned properties, as well as the facts that 1) $\nabla_\omega L(\theta^*(\omega), \xi^*(\omega), \omega)$ is a Lipschitz function, 2) the step-sizes of Assumption on steps sizes, 3) $\{\omega_{k+1}\}$ is a stochastic approximation of $\omega(t)$ with a Martingale difference error, and 4) convex and compact properties in projections used, we can use Theorem 2 of chapter 6 in [3] to demonstrate the sequence $\{\omega_k\}$ converges almost surely to a locally maximum point ω^* almost surely, that is $L(\theta^*(\omega), \xi^*(\omega), \omega^*) \geq L(\theta^*(\omega), \xi^*(\omega), \omega)$.

From Time scales 2 and 3 we have that $L(\theta^*(\omega), \xi^*(\omega), \omega) \leq L(\theta, \xi, \omega)$ while from Time scale 4 we have that $L(\theta^*(\omega), \xi^*(\omega), \omega^*) \geq L(\theta^*(\omega), \xi^*(\omega), \omega)$. Thus, $L(\theta^*(\omega), \xi^*(\omega), \omega) \leq L(\theta^*(\omega), \xi^*(\omega), \omega^*) \leq L(\theta, \xi, \omega^*)$. Therefore, $(\theta^*, \xi^*, \omega^*)$ is a local saddle point of (θ, ξ, ω). Invoking the saddle point theorem of Proposition 5.1.6 in [6], we can conclude that $\pi(\cdot|\cdot; \theta^*)$ is a locally optimal policy for our proposed optimization formulation. □

3.4.4 Remark on Bounding Lagrange Multiplier

We can say our algorithm learns an REF that is closer to an optimally safe policy's REF as we take $\lambda_{\max} \to \infty$. Nonetheless, we want to put a bound on the $\lambda_{\max}$. This $\lambda_{\max}$ must be large enough so that choosing a policy that can reduce the expected cost returns by some non-zero amount is prioritized over increasing the reward returns. So any change in the reward critic terms must be less than any change in the cost critic term. If H_Δ is the minimum non-zero difference between any two cost values, and $P_{\min}$ is the minimum sampled non-zero likelihood of reaching a particular state and a point in the sample trajectory, then we can bound the maximum change in the reward returns and the maximum change on the weighted cost returns:

$$\Delta \mathop{\mathbb{E}}_{s\sim d_0}[V(s)] \leq \frac{R_{\max}}{1-\gamma}$$

$$\gamma^T \cdot H_\Delta \cdot P_{\min} \cdot (\lambda + \frac{\phi(s)}{(1-\phi(s))}) \leq \Delta \mathop{\mathbb{E}}_{s\sim d_0}[V_c(s) \cdot (\lambda + \frac{\phi(s)}{(1-\phi(s))})]$$

So we can find the bound for $\lambda_{\max}$:

$$\begin{aligned}
\Delta \mathop{\mathbb{E}}_{s\sim d_0}[V(s)\cdot(1-\phi(s))] &< \Delta \mathop{\mathbb{E}}_{s\sim d_0}[V_c(s)\cdot(\lambda\cdot(1-\phi(s))+\phi(s))] \\
\Delta \mathop{\mathbb{E}}_{s\sim d_0}[V(s)] &< \Delta \mathop{\mathbb{E}}_{s\sim d_0}[V_c(s)\cdot(\lambda+\frac{\phi(s)}{(1-\phi(s))})] \\
\frac{R_{\max}}{1-\gamma} &< \gamma^T\cdot H_\Delta\cdot P_{\min}\cdot(\lambda+\frac{\phi(s)}{(1-\phi(s))}) \\
\frac{R_{\max}}{(1-\gamma)\cdot\gamma^T\cdot H_\Delta\cdot P_{\min}} &< \lambda+\frac{\phi(s)}{(1-\phi(s))} \\
\frac{R_{\max}}{(1-\gamma)\cdot\gamma^T\cdot H_\Delta\cdot P_{\min}}-\frac{\phi(s)}{(1-\phi(s))} &< \lambda
\end{aligned}$$

The second line holds since we are simply rearranging the comparative weightages of the reward and cost returns. Now $-\frac{\phi(s)}{(1-\phi(s))}\leq 0$ (recall that if $\phi(s)=1$, then λ is irrelevant in the lagrangian optimization). Thus, if $\lambda > \frac{R_{\max}}{(1-\gamma)\cdot\gamma^T\cdot H_\Delta\cdot P_{\min}}$ then minimizing the cost returns is prioritized over maximizing reward returns.

4 Complete Experiment Details and Analysis

4.1 Baselines

We compare our algorithm **RESPO** with 7 other safety RL baselines, which can be divided to CMDP class and hard constraints class, and unconstrained Vanilla PPO for reference.

CMDP Approaches

Proximal Policy Optimization-Lagrangian. **PPOLag** is a primal-dual method using Proximal Policy Optimization [8] based off of the implementation found in [9]. The lagrange multiplier is a scalar learnable parameter.

Constraint-Rectified Policy Optimization. **CRPO** [10] is a primal approach that switches between optimizing for rewards and minimizing constraint violations depending on whether the constraints are violated.

Penalized Proximal Policy Optimization. **P3O** [11] is another primal approach based on applying the technique of clipping the surrogate objectives found in PPO [8] to CMDPs.

Projection-Based Constrained Policy Optimization. **PCPO** [12] is a trust-region approach that takes a step in policy parameter space toward optimizing for reward and then projects this policy to the constraint set satisfying the CMDP expected cost constraints.

Hard Constraints Approaches

Reachability Constrained Reinforcement Learning. **RCRL** [5] is a primal-dual approach where the constraint is on the reachability value function and the lagrange multiplier is represented by a neural network parameterized by state.

Control Barrier Function. This **CBF**-based approach is inspired by the various energy-based certification approaches [13, 14, 15, 16, 17, 18]. This is implemented as a primal-dual approach where the control barrier-based constraint $\dot{h}(s)+\nu\cdot h(s)\leq 0$ is to ensure stabilization toward the safe set.

Feasible Actor Critic. **FAC** [19] is another primal-dual approach similar to **RCRL** (i.e. it uses the NN representation of the lagrange multiplier parameterized by state) except that the constraint in **FAC** is based on the cumulative discount sum of costs in lieu of the reachability value function. It is important to note that **FAC** is originally meant for the CMDP framework (with some positive cost threshold), but we adapt it to hard constraints by making the cost threshold $\chi=0$. We do this to make a better comparison between an algorithm that relies on using the lagrange multiplier represented as a NN to learn feasibility with our approach of using our proposed REF function to learn the feasibility likelihood – both approaches enforce a hard constraint on the cumulative discounted costs.

4.2 Benchmarks

We compare the algorithms on a diverse suite of environments including those from the Safety Gym, PyBullet, and MuJoCo suites and a multi-constraint, multi-drone environment.

Safety Gym. In Safety Gym [20], we examine CarGoal and PointButton which have 72D and 76D observation spaces that include lidar, accelerometer, gyro, magnetometer, velocimeter, joint angles, and joint velocities sensors. In the CarGoal environment, the car agent has a 72D observation space and is supposed to reach a goal region while avoiding both hazardous spaces and contact with fragile objects. In PointButton, the point agent has a 76D observation space and must press a series of specified goal buttons while avoiding 1) quickly moving objects, 2) hazardous spaces, 3) hitting the wrong buttons.

Safety PyBullet. In Safety PyBullet [21], we evaluate in BallRun and DroneCircle environments. In the BallRun environment, the ball agent must move as fast as possible under the constraint of a speed limit, and it must be within some boundaries. In DroneCircle, the agent is based on the AscTec Hummingbird quadrotor and is rewarded for moving clockwise in a circle of a fixed radius with the constraint of remaining within a safety zone demarcated by two boundaries. Note that we use this environment to evaluate our algorithm and the baselines in a stochastic setting. We ensure the MDP is stochastic by adding a 5% gaussian noise to the transitions per step.

Safety MuJoCo. Furthermore, we compare the algorithms in with complex dynamics in MuJoCo. Specifically, we look at HalfCheetah and Reacher safety problems. In Safety HalfCheetah, the agent must move as quickly as possible in the forward direction without moving left of $x = -3$. However, unlike the standard HalfCheetah environment, the reward is based on the absolute value of the distance traveled. In this paradigm, it is easier for the agent to learn to quickly run backward rather than forward without any directional constraints. In the Safety Reacher environment, the robotic arm must reach a certain point while avoiding an unsafe region.

Multi-Drone environment. We also compare in an environment with multiple hard and soft constraints. The environment requires controlling two drones to pass through a tunnel one at a time while respecting certain distance requirements. The reward is given for quickly reaching the goal positions. The two hard constraints involve **(H1)** ensuring neither drone collides into the wall and **(H2)** the distance between the two drones is more than 0.5 to ensure they do not collide. The soft constraint is that the two drones are within 0.8 of each other to ensure real-world communication. It is preferable to prioritize hard constraint **H1** over hard constraint **H2**, since colliding with the wall may have more serious consequences to the drones rather than violations of an overly precautious distance constraint – as we will show, our algorithm **RESPO** can perform this prioritization in its optimization.

4.3 Hyperparameters/Other Details

Hyperparameters for Safe RL Algorithms	Values
On-policy parameters	
Network Architecture	MLP
Units per Hidden Layer	256
Numbers of Hidden Layers	2
Hidden Layer Activation Function	tanh
Actor/Critic Output Layer Activation Function	linear
Lagrange multiplier Output Layer Activation Function	softplus
Optimizer	Adam
Discount factor γ	0.99
GAE lambda parameter	0.97
Clip Ratio	0.2
Target KL divergence	0.1
Total Env Interactions	$9e6$
Reward/Cost Critic Learning rate	Linear Decay $1e{-}3 \rightarrow 0$
Actor Learning rate	Linear Decay $3e{-}4 \rightarrow 0$
Lagrange Multiplier Learning rate	Linear Decay $5e{-}5 \rightarrow 0$
Number Seeds per algorithm per experiment	5
RESPO specific parameters	
REF Output Layer Activation Function	sigmoid
REF Learning rate	$1e{-}4 \rightarrow 0$
CBF specific parameters	
ν	0.2
RCRL/FAC Note	
Lagrange Multiplier	2-Layer, MLP (other algs just use scalar parameter)

Table 2: Hyperparameter Settings Details

To ensure a fair comparison, the primal-dual based approaches and unconstrained Vanilla PPO were implemented based off of the same code base [22]. The other three approaches were implemented based on [23] with the similar corresponding hyperparameters as the primal-dual approaches. We run our experiments on Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz with 6 cores. For Safety Gym, PyBullet, MuJoCo, and the multi-drone environments, each algorithm, per seed, per environment, takes ~ 4 hours to train.

4.4 Double Integrator

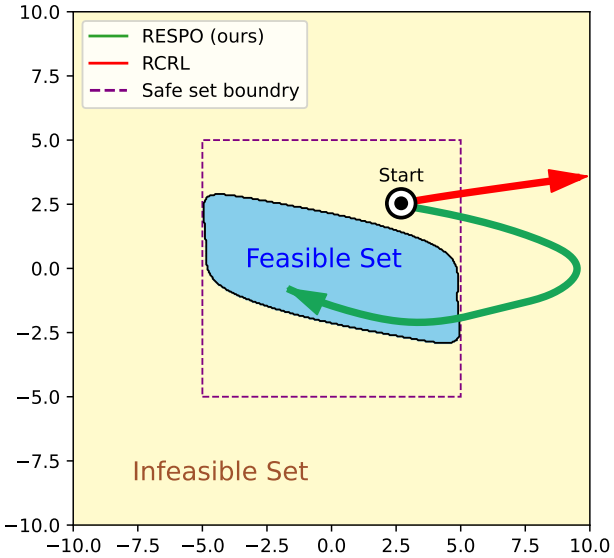

Figure 2: Comparison of the trajectories in the Double Integrator Environment of an agent controlled by policies obtained by RCRL (in red) and our proposed algorithm RESPO (in green) when starting from the infeasible set (but still within the safe set). Notice how our approach actively enters the feasible set (blue region), while RCRL fails to do so. The level set demarcating the feasible/infeasible set boundary is in black. The safe set (i.e. the set of states that have no violations) is the region within the dashed purple square. The infeasible set is in yellow.

We use the Double Integrator environment as a motivating example to demonstrate how performing constrained optimization using solely reachability-based value functions as in **RCRL** can produce nonoptimal behavior when the agent is outside the feasiblity set. Double Integrator has a 2 dimensional observation space $[x_1, x_2]$, 1 dimension action space $a \in [-0.5, 0.5]$, system dynamics is $\dot{s} = [x_2, a]$, and constraint as $||s||_\infty \leq 5$. Particularly, we make the cost as 1 if $||s||_\infty > 5$, and 0 otherwise to emphasize the importance of capturing the frequency of violation during training.

We train an **RCRL** controller and **RESPO** controller in this environment, and the results are visualized in Figure 2. The color scheme indicates the learned reachability value across the state space while the black line demarcates the border of the zero level set. We present the behavior of the trajectories of **RCRL** and **RESPO**. Because the **RCRL** optimizes for reachability value function when outside the feasible set, it simply minimizes the maximum violation, which as can be seen does not result in the agent reentering the feasible set since it is uniformly equal to or near 1 in the infeasible set. This is since it permits many violations of magnitude same or less than that of the maximum violation. On the other hand, **RESPO** optimizes for cumulative damage by considering total sum of costs, thereby re-entering the feasible set.

4.5 Safety Gym Environments

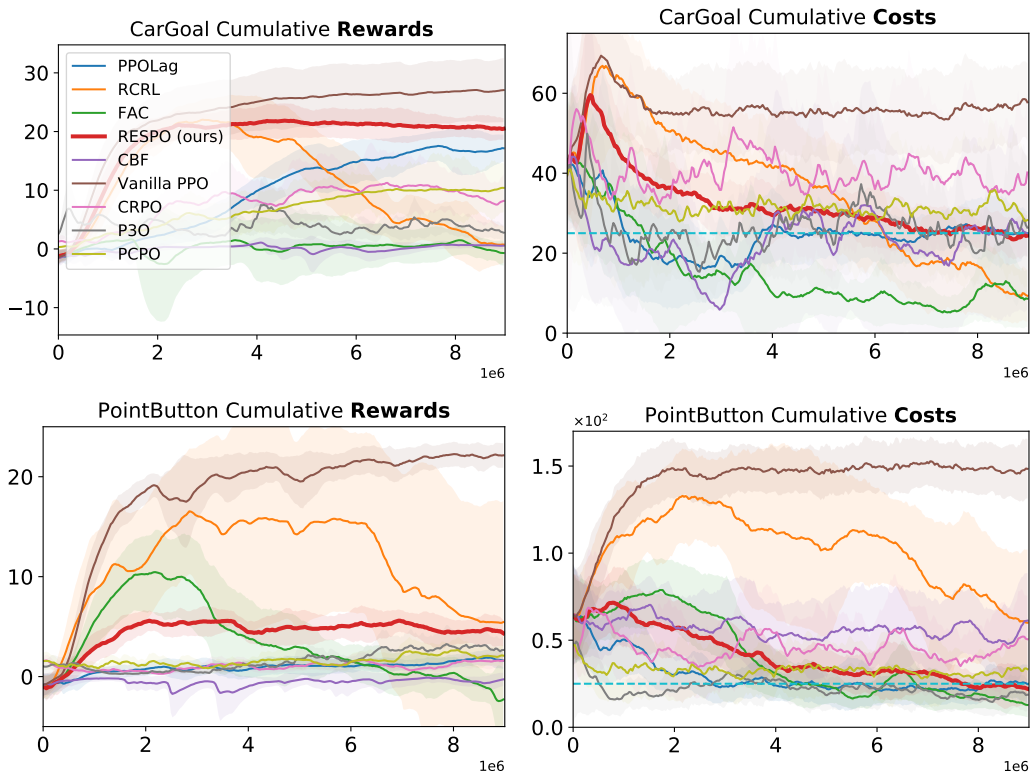

Figure 3: Closer look at comparison of algorithms in Safety Gym CarGoal and PointButton Environments.

RESPO: In the CarGoal Environment, our approach achieves the best performance among the safe RL algorithms while being within the acceptable range of cost violations. It is important to note that our algorithm *has no access to information* on the cost threshold. In PointButton, **RESPO** achieves very high reward performance among the safety baselines while maintaining among the lowest violations.

PPOLag: In both Safety Gym environments, **PPOLag** maintains relatively high performance, albeit less than our approach. Nonetheless, it always converges to the cost threshold amount of violations for the respective environments.

RCRL: **RCRL** has either high reward and high violations or low reward and low violations. It learns a very conservative behavior in CarGoal environment where the violations go down but the reward performance can also be seen to be sacrificed during training. For PointButton, **RCRL** achieves slightly higher reward performance but has over $3\times$ the number of violations as **RESPO**.

FAC: Using a NN to represent the lagrange multiplier in order to capture the feasible sets seems to produce very conservative behavior that sacrifices performance. In both CarGoal and PointButton the reward performance and cost violations are very low. This can be explained because the average observed lagrange multiplier across the states quickly grows, even becoming $9\times$ that of scalar learnable lagrange multiplier in **RESPO**.

CBF: The **CBF** approach has low reward performance in both the Safety Gym benchmarks and its cost violations are quite high.

CRPO, **P3O**, & **PCPO**: These CMDP-based primal approaches have mediocre reward performance but, with the exception of CRPO, achieve violations within the cost threshold. CRPO, however, has high cost violations in both CarGoal and PointButton.

Vanilla PPO: This unconstrained algorithm consistently has high rewards and high costs, so maximizing rewards does not improve costs in these environments.

4.6 Safety PyBullet Environments

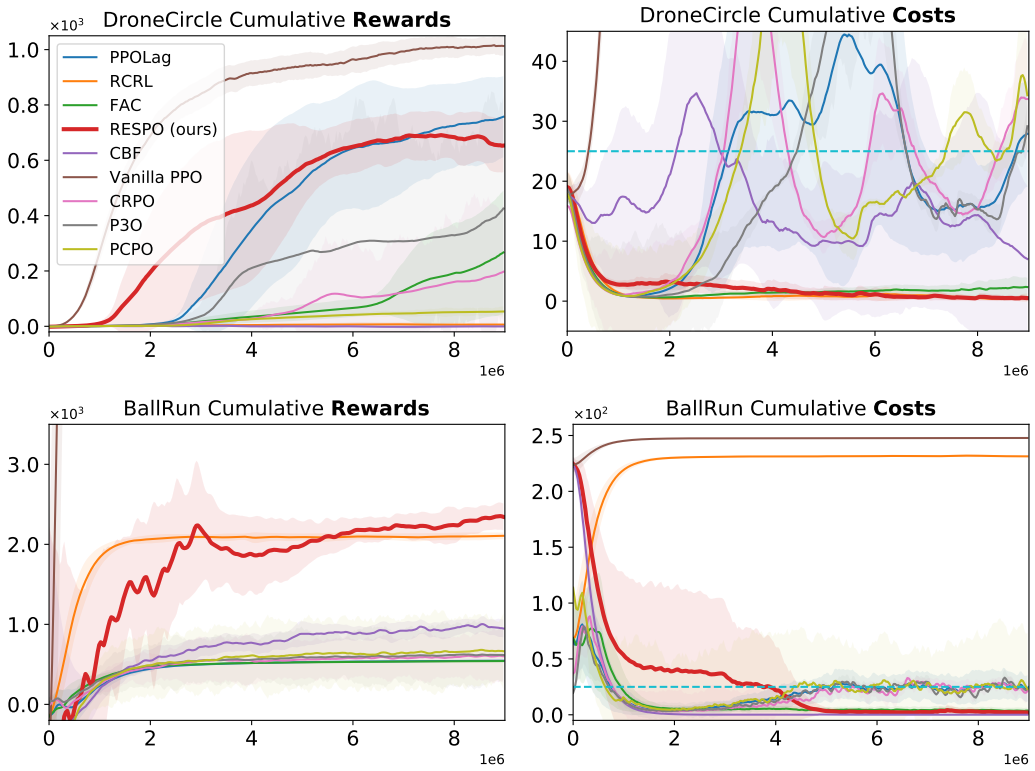

Figure 4: Closer look at comparison of algorithms in Safety PyBullet DroneCircle and BallRun Environments.

RESPO: In the both BallRun and DroneCircle, our approach achieves the highest or among the highest reward performance compared to the other safety baselines. Furthermore, **RESPO** converges to almost 0 constraint violations for both environments.

PPOLag: In DroneCircle, **PPOLag** has the best reward performance. Although its costs violations are around the cost threshold, it is much higher than **RESPO**. On the other hand, in BallRun, PPOLag has very low reward performance.

RCRL: We again see **RCRL** take on behavior with extremes – it has low reward and cost violations in Drone Circle and has high reward and cost violations in BallRun. Constraining the maximum violation with the reachability value function as **RCRL** does seems to provide poor safety in an environment with non-tangible constraints (i.e. a speed limit in BallRun).

FAC: While in BallRun, we see **FAC** have the same low reward and low violations behavior, DroneCircle shows an instance where **FAC** can achieve decently high rewards while maintaining low violations.

CBF: In DroneCircle, the **CBF** approach has low rewards and relatively low violations; in BallRun, it has a bit higher rewards compared to all the low performance algorithms with very low violations.

CRPO, **P3O**, & **PCPO**: These CMDP-based primal approaches have mediocre reward performance in DroneCircle and very low performance in BallRun. Nonetheless they achieve violations within the cost threshold.

Vanilla PPO: This unconstrained algorithm consistently has high rewards and high costs (sometimes out of the scope of the plots), so maximizing rewards does not improve costs in these environments.

4.7 Safety MuJoCo Environments

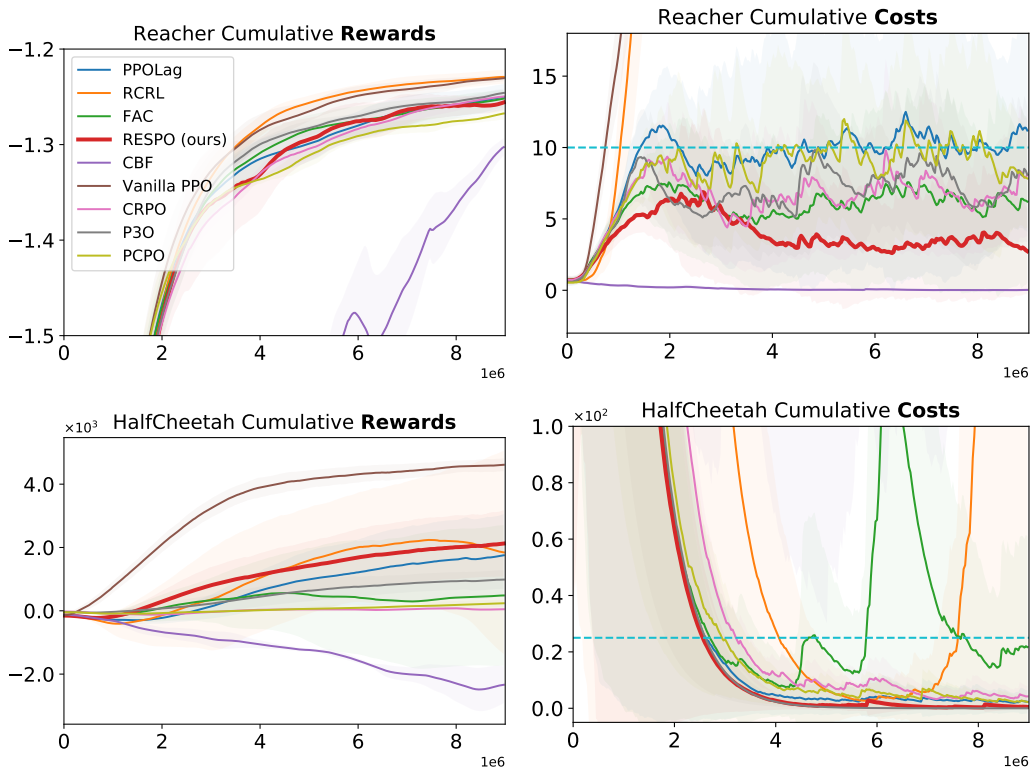

Figure 5: Closer look at comparison of algorithms in Safety MuJoCo Reacher and HalfCheetah Environments.

Note on HalfCheetah: The rewards for HalfCheetah are based on the absolute distance traveled in each step. Without the cost metric to constrain backward travel, it is easy to learn to run backward, as is the behavior learned in unconstrained PPO.

RESPO: Our approach achieves the highest reward performance among the safety baselines in HalfCheetah and decent reward performance in Reacher. Interestingly, **RESPO** also has 0 constraint violations in HalfCheetah and the second lowest constraint violations in Reacher.

PPOLag: The performance in Reacher for **PPOLag** is similar as in most of the previous environments: decently high rewrad, cost near the threshold. However, for HalfCheetah, interesting **PPOLag** learns to maintain the violations well below the cost threshold.

RCRL: We see yet again **RCRL** has high reward follow by very high constraint violations.

FAC: This approach has decent reward performance in Reacher and low reward performance in HalfCheetah. However, interestingly, **FAC** has high cost violations though below the cost threshold.

CBF: In Reacher, the **CBF** approach has conservative behavior with both low reward and low cost violations. But in HalfCheetah, it has very low reward performance and very high cost violations (not seen in the plot since its an order of magnitude larger than the visible range).

CRPO, **P3O**, & **PCPO**: These CMDP-based primal approaches have decent reward performance in Reacher while maintaining violations within cost threshold. In HalfCheetah, however, they achieve low performance and low cost violations.

Vanilla PPO: This unconstrained algorithm consistently has high rewards and high costs (sometimes out of the scope of the plots), so maximizing rewards does not improve costs in these environments.

4.8 Hard and Soft Constraints

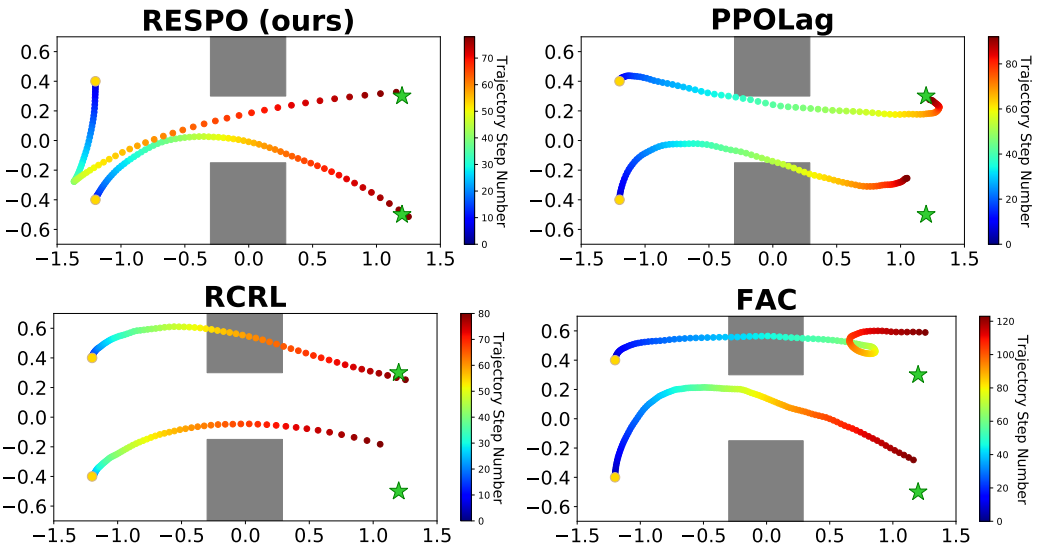

Figure 6: Closer look at comparison of RESPO with baselines trajectories in hard & soft Constraints multi-Drone control. Starting at gold circles, drones must enter the tunnel one at a time and reach green star while avoiding the wall and satisfying distance constraints. The colors indicate time along the trajectories.

RESPO: We manage multiple hard and soft Constraints by extending our framework to optimize the following Lagrangian:

$$\min_{\pi}\max_{\lambda}\Bigg(L(\pi,\lambda)=\mathop{\mathbb{E}}_{s\sim d_0}\Big[\big[[-V^{\pi}(s)+\lambda_{sc}\cdot(V^{\pi}_{sc}(s)-\chi)+\lambda_{hc2}\cdot V^{\pi}_{hc2}(s)]\cdot(1-p_{hc2}(s))\\ +V^{\pi}_{hc2}(s)\cdot p_{hc2}(s)+\lambda_{hc1}\cdot V^{\pi}_{hc1}(s)\big]\cdot(1-p_{hc1}(s))+V^{\pi}_{hc1}(s)\cdot p_{hc1}(s)\Big]\Bigg) \tag{6}$$

The subscripts $hc1$ indicates the first hard constraint (i.e. wall avoidance), $hc2$ indicates the second hard constraint (i.e. drone cannot be too close), and sc indicates soft constraint (i.e. drone cannot be too far) – they are all based on discounted sum of costs. Recall $V^{\pi}(s)$ is reward returns. We color coded the corresponding parts of the optimization. Notice how we learn a different REF for each hard constraint. Also notice that the feasible set of the first hard constraint p_{hc1} is placed in a manner so as to ensure prioritization of the first hard constraint. As can be seen in the top left plot of Figure 6, our approach successfully reaches the goals and avoids the walls. To enable mobility of the top drone to pass through the tunnel with wall collision, the second hard constraint is violated temporarily in the blue to cyan time period. Furthermore to allow the bottom drone to pass through the tunnel, the soft constraint is violated during the green to orange time period. Nonetheless, **RESPO** successfully manages the constraints and reward performance via Equation 6 optimization.

PPOLag, RCRL, FAC: The optimization formulation for these approaches is as follows:

$$\min_{\pi}\max_{\lambda}\Big(L(\pi,\lambda)=-V^{\pi}(s)+\lambda_{sc}\cdot(V^{\pi}_{sc}(s)-\chi)+\lambda_{hc2}\cdot V^{\pi}_{hc2}(s)+\lambda_{hc1}\cdot V^{\pi}_{hc1}(s)\Big) \tag{7}$$

For **PPOLag** and **FAC**, all the constraint value functions are discount sum of cost. For **RCRL**, $V^{\pi}_{sc}(s)$ is based on discount sum of costs but $V^{\pi}_{hc1}(s)$ and $V^{\pi}_{hc2}(s)$ are based on the reachability value function. Furthermore, in **PPOLag**, all the lagrange multipliers are learnable scalar parameters. In **FAC** and **RCRL** all the lagrange multipliers are NN representations parameterized by state. These formulations are not able to provide a framework for the prioritization of the constraint satisfaction – all the constraints are treated the same, weighted only on the learned lagrange multipliers. As can be seen in the other three images in Figure 6, the algorithms cannot manage the multiple constraints, and invariably collide with the wall.

4.9 Ablation – Learning Rate

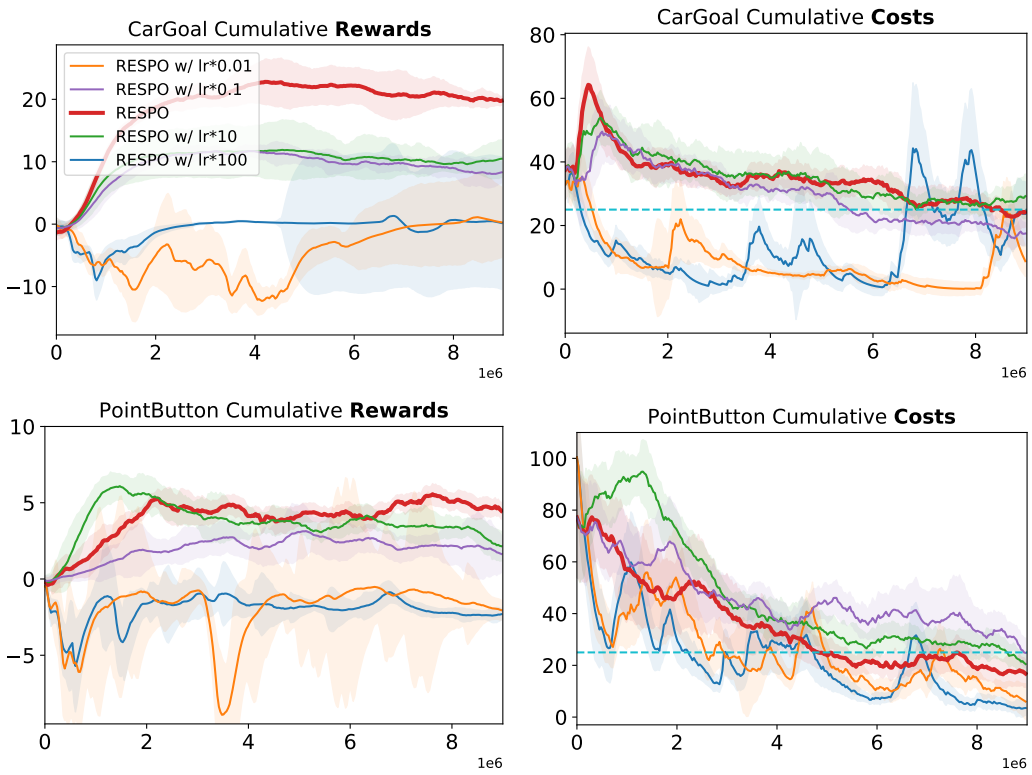

Figure 7: Closer look at Ablation study on the learning rate of REF.

We performance Ablation study of varying the learning rate of the REF function to verify the importance of the multi-timescale assumption. Particular, we compare our algorithm's approach of placing the learning rate of the REF between the policy and lagrange multiplier with making the REF's learning rate in various orders of magnitudes slower and faster. Our approach with the learning rate satisfying the multi-timescale assumption experimentally appears to still have the best balance of reward optimization and constraint satisfaction. Particularly when we change the learning rate by one order of magnitude (i.e. $\times 10$ or $\times 0.1$), we see the reward performance reduce by around half and while the cost violations generally don't change. But when we change the learning rates by another order of magnitude, there reward performance effective becomes zero and the cost violations generally reduce further. By increasing the learning rate of the REF function, we can no longer guarantee that the REF convergences to near the optimally safe REF value. Instead, it becomes the REF of the policy in question. So instead, the optimization can learn to "hack" the REF function to obtain a policy (and lagrange multiplier) that is not a local optimal for the optimization formulation. On the other hand, when the learning rate is too slow, the lagrange multiplier quickly explodes, thereby creating a very conservative solution – notice the similarity of the orange line in Figure 7 with learning rate 0.01 times that of standard in training behavior with **PPOLag** where $\chi = 0$ in the ablation study on optimization.

4.10 Ablation – Optimization

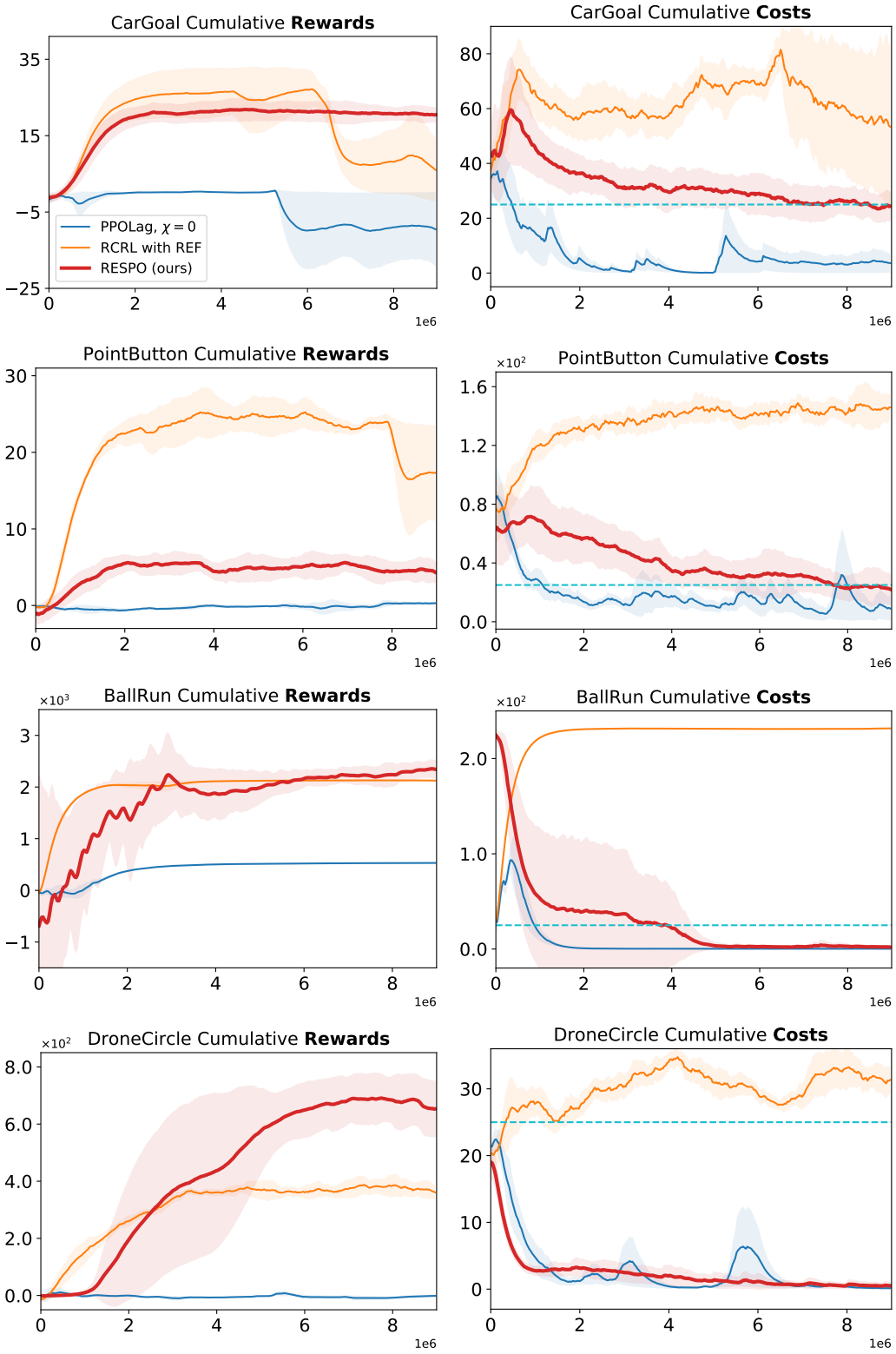

Figure 8: Closer look at Ablation study on hard constraints optimization frameworks.

In this ablation study, we examine the various optimization frameworks within the context of hard constraints. Particularly, we compare our **RESPO** framework with **RCRL** and **CMDP**. However, for **RCRL** we implement using our REF function method while still keeping the reachability value function. For **CMDP**, we make the cost threshold $\chi = 0$. These comparisons answer important questions about our design choices – specifically is it sufficient to simply to just use the REF component or to just learn the cost returns alone? From this ablation study, we propose that though we have provided theoretical support for adding each of these design components individually, in practice they are both required together in our algorithm. In **RCRL** implemented with our REF, we generally see decently high rewards but the cost violations are always very large. This highlights the problems of the reachability function again – if the agent starts or ever wanders into the infeasible set, there is no guarantee of (re)entrance into the feasible set. So the agent can indefinitely remain in the infeasible set, thereby incurring potentially an unlimited number of constraint violations. In **PPOLag** with $\chi = 0$, both the reward performance and constraint violations are very low. By using such hard constraints versions of these purely learning-based methods, even when using the cumulative discounted cost rather than reachability value function, the reward performance is very low because the lagrange multiplier becomes too large quickly and thereby overshadows the reward returns in the optimization. Ultimately, both the REF approach and the usage of the cumulative discounted costs are important components of our algorithm **RESPO** that encourage a good balance between the reward performance and safety constraint satisfaction in such stochastic settings.

References

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

[2] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

[3] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

[4] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

[5] Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International Conference on Machine Learning*, pages 25636–25655. PMLR, 2022.

[6] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[7] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[9] Chen Tessler, Daniel Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2019.

[10] Tengyu Xu, Yingbin Liang, and Guanghui Lan. Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *International Conference on Machine Learning*, pages 11480–11491. PMLR, 2021.

[11] Linrui Zhang, Li Shen, Long Yang, Shi-Yong Chen, Bo Yuan, Xueqian Wang, and Dacheng Tao. Penalized proximal policy optimization for safe reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2022.

[12] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. In *International Conference on Learning Representations*, 2020.

[13] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 2023.

[14] Ya-Chien Chang and Sicun Gao. Stabilizing neural control using self-learned almost lyapunov critics. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1803–1809, 2021.

[15] Zhizhen Qin, Tsui-Wei Weng, and Sicun Gao. Quantifying safety of learning-based self-driving control using almost-barrier functions. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12903–12910. IEEE, 2022.

[16] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *European Control Conf.*, 2019.

[17] Jason J Choi, Donggun Lee, Koushil Sreenath, Claire J Tomlin, and Sylvia L Herbert. Robust control barrier–value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6814–6821. IEEE, 2021.

[18] Eric Yang Yu, Zhizhen Qin, Min Kyung Lee, and Sicun Gao. Policy optimization with advantage regularization for long-term fairness in decision systems, 2022.

[19] Haitong Ma, Yang Guan, Shegnbo Eben Li, Xiangteng Zhang, Sifa Zheng, and Jianyu Chen. Feasible actor-critic: Constrained reinforcement learning for ensuring statewise safety. *arXiv preprint arXiv:2105.10682*, 2021.

[20] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

[21] Sven Gronauer. Bullet-safety-gym: A framework for constrained reinforcement learning. 2022.

[22] Ppo lagrangian pytorch. `https://github.com/akjayant/PPO_Lagrangian_PyTorch`, 2022.

[23] Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *arXiv preprint arXiv:2305.09304*, 2023.