

---

# Online Optimization for Offline Safe Reinforcement Learning

---

**Yassine Chemingui**

Washington State University  
yassine.chemingui@wsu.edu

**Aryan Deshwal**

University of Minnesota  
adeshwal@umn.edu

**Alan Fern**

Oregon State University  
alan.fern@oregonstate.edu

**Thanh Nguyen-Tang**

New Jersey Institute of Technology  
thanh.nguyen@njit.edu

**Janardhan Rao Doppa**

Washington State University  
jana.doppa@wsu.edu

## Abstract

We study the problem of Offline Safe Reinforcement Learning (OSRL), where the goal is to learn a reward-maximizing policy from fixed data under a cumulative cost constraint. We propose a novel OSRL approach that frames the problem as a minimax objective and solves it by combining offline RL with online optimization algorithms. We prove the approximate optimality of this approach when integrated with an approximate offline RL oracle and no-regret online optimization. We also present a practical approximation that can be combined with any offline RL algorithm, eliminating the need for offline policy evaluation. Empirical results on the DSRL benchmark demonstrate that our method reliably enforces safety constraints under stringent cost budgets, while achieving high rewards. The code is available at <https://github.com/yassineCh/03SRL>.

## 1 Introduction

Offline reinforcement learning (RL) [Levine et al., 2020] is a powerful paradigm to learn decision-making policies from logged datasets without the need for additional interaction with the environment. Offline RL has shown good success in domains including autonomous driving, robotics, control systems, and black-box optimization [Fang et al., 2022, Lin et al., 2024, Li et al., 2024, Zhan et al., 2022, Chemingui et al., 2024] where executing exploratory actions is not practical. However, in safety-critical domains such as healthcare and smart grid, the decision-making agent needs to also satisfy some cost/safety constraints. Such problems are studied under the sub-area referred to as offline safe RL (OSRL) where the goal is to learn reward-maximizing policies which satisfy cost constraints from offline datasets [Wachi et al., 2024].

OSRL inherits the challenges from both offline RL and safe RL [Wachi et al., 2024]. First, handling distributional shift when the learned policy encounters states and actions not seen in the offline dataset. Prior work in offline RL addresses this challenge by some form of penalization to realize the general principle of pessimism in the face of uncertainty [Levine et al., 2020]. Second, ensuring that the learned policy meets the cost/safety constraints after deployment. Maintaining the cost constraints typically requires off-policy evaluation (OPE) procedures which are highly unstable and typically have estimation errors [Figueiredo Prudencio et al., 2024]. OSRL is an active research topic and is relatively less-studied than offline RL. Constrained policy optimization using Lagrangian relaxation [Polosky et al., 2022, Lee et al., 2022, Xu et al., 2022b] is a popular approach for OSRL, but due to the need to solve intertwined optimization problems they are unstable in practice, and are shown to have bad performance either causing oscillation/divergence or learning overly conservative (near zero-reward) policies. Additionally, OSRL problem with stringent safety constraints (i.e., small cost

thresholds) is an important but severely under-studied problem [Zheng et al., 2024] and arises in many safety-critical control applications such as industrial process and energy system control.

This paper develops a novel offline constrained RL framework referred to as ***Online Optimization for Offline Safe RL (O3SRL)***. O3SRL formulates a minimax optimization problem for policy learning and solves it using an iterative approach that relies on two key components, namely, an offline RL oracle and a no-regret algorithm. We perform two main algorithmic steps in each iteration of O3SRL. First, we produce a distribution over policies from solving an offline RL problem by defining a new reward function that combines the original reward and cost values based on the values of the current Lagrange variable and the cost threshold. Second, we employ a no-regret algorithm [Hazan et al., 2016] to adaptively update the value of Lagrange variable based on the current distribution over policies. We prove that the O3SRL framework is guaranteed to converge to the minimax solution.

Unfortunately, the general O3SRL framework suffers from two drawbacks in practice. First, the use of no-regret algorithms over a continuous Lagrange variable requires calling OPE procedures which are typically unstable [Figueiredo Prudencio et al., 2024] and can potentially lead to error propagation over iterations. Calling an OPE procedure in each iteration also contributes to computational expense. Second, it is computationally-expensive to run an offline RL algorithm to convergence in each iteration. To overcome these challenges, we develop a practical and effective algorithm with convergence guarantees. We employ  $K$  discrete values for Lagrange variable and apply a multi-armed bandit algorithm. Since most offline RL algorithms are based on stochastic gradient descent, we perform a small number of gradient updates on the policy from the previous iteration for efficiency. One advantage of this approach is that we can leverage powerful offline RL algorithms in a plug-and-play manner to solve offline constrained RL problems.

We perform experimental evaluation of this practical approach on multiple DSRL benchmark tasks [Liu et al., 2024] and our main findings are as follows. First, even the simplest version of our approach with two arms ( $K=2$ ) is quite effective and results in state-of-the-art performance. Second, our approach achieves excellent performance for the challenging setting of small cost constraint thresholds. Third, the performance improves with higher arms ( $K > 2$ ) but shows diminishing returns beyond  $K=5$ . Finally, O3SRL performs effectively with different offline RL algorithms and higher cost limits demonstrating its robust performance.

**Contributions.** The key contribution of this paper is the development and evaluation of the O3SRL framework to solve offline safe RL problems. Specific contributions include:

- Formulating OSRL as a minimax optimization problem and providing an iterative framework based on no-regret algorithms with convergence guarantees to the minimax solution.
- Development of a practical algorithm with convergence guarantee by avoiding the usage of off-policy evaluation (can lead to error propagation) and running offline RL algorithm to convergence (high computational cost) inside each iteration of a multi-round method.
- Empirical evaluation of the practical algorithm and its variants on DSRL benchmark tasks to demonstrate its effectiveness over state-of-the-art methods.

## 2 Problem Setup

RL problems with safety constraints are typically formulated using the Constrained Markov Decision Process (CMDP) framework [Altman, 2021]. A CMDP is defined by the tuple  $(\mathcal{S}, \mathcal{A}, P, r, c, \gamma, \mu)$ , where  $\mathcal{S}$  and  $\mathcal{A}$  denote the state and action spaces respectively,  $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$  is the unknown stochastic state transition function,  $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  is the reward function,  $c : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$  is the cost function,  $\gamma \in (0, 1)$  is the discount factor, and  $\mu$  is the initial state distribution.

Let  $\pi : \mathcal{S} \rightarrow \mathcal{A}$  denote a policy and let  $\tau = \{(s_t, a_t, r_t, c_t)\}_{t=1}^T$  be a trajectory of length  $T$  induced by  $\pi$  under dynamics  $P$ . The cumulative discounted return and cost for a trajectory  $\tau$  are given by  $R(\tau) = \sum_{t=1}^T \gamma^t r_t$  and  $C(\tau) = \sum_{t=1}^T \gamma^t c_t$ , respectively. In offline safe reinforcement learning (OSRL) problems, the agent is given access only to a static dataset  $\mathcal{D}_{OSRL} = \{(s_i, a_i, r_i, c_i, s'_i)\}_{i=1}^n$  of  $n$  examples collected from an unknown behavior policy, and the overall goal is to learn a policy  $\pi$  to maximize the expected reward while satisfying a given constraint on the expected cumulative cost. Formally, the learning objective is as follows:

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi}[R(\tau)] \quad \text{subject to} \quad \mathbb{E}_{\tau \sim \pi}[C(\tau)] \leq \kappa,$$

where  $\kappa \geq 0$  is the maximum cost threshold/limit for safety constraint. This problem setting combines the challenges of offline RL (e.g., out-of-distribution states) with the added requirement of ensuring cost constraint satisfaction from static data alone, without online interaction.

Most of the prior work in OSRL cannot handle the setting where the cost threshold  $\kappa$  is small [Zheng et al., 2024] that is important for safety-critical applications with a tight budget. One of the key goals of this paper is to handle this challenging setting in a principled manner while solving the general OSRL problem effectively for higher cost constraint thresholds.

### 3 Related Work

**Offline RL.** This setting differs from the standard online RL setup in the following way: the goal is to learn reward maximizing policies from a given static dataset without interacting with the environment. The key technical challenge is to effectively handle out-of-distribution (OOD) states and actions [Levine et al., 2020, Figueiredo Prudencio et al., 2024]. Prior work has addressed this OOD challenge using improvements in value function estimation [Fujimoto and Gu, 2021, Kostrikov et al., 2022, Kumar et al., 2019, Lyu et al., 2022, Yang et al., 2022, Nguyen-Tang and Arora, 2023b,a], sequential modeling [Janner et al., 2021, Wang et al., 2022], uncertainty-awareness [An et al., 2021, Bai et al., 2022], constraining policy based on divergence [Wu et al., 2020, Jaques et al., 2020, Wu et al., 2022], by extending methods for model based RL [Kidambi et al., 2020, Yu et al., 2020, Rigter et al., 2022] and imitation learning [Xu et al., 2022a], and action selection based on filtering and reweighting [Chen et al., 2023, Hansen-Estruch et al., 2023].

**Offline safe RL.** This is a generalization of the offline RL setting where we need to learn a safe policy from a given offline dataset and a pre-specified cost constraint [Liu et al., 2024]. Offline safe RL is a relatively less-studied problem. Prior work includes methods based on constrained policy optimization [Polosky et al., 2022, Lee et al., 2022] and Lagrangian relaxation [Le et al., 2019, Xu et al., 2022b]. [Le et al., 2019] proposed one of the first batch-constrained frameworks that combines offline RL with online Lagrange-multiplier updates. However, this approach faces practical challenges (error propagation and high computational effort) due to the need to call an OPE procedure multiple times in each iteration which we address in this paper. We elaborate on these challenges and differences in the Appendix.

Extensions with convex MDP assumptions have also been explored [Zhang et al., 2024], along with approaches based on adversarial cost penalties [Wei et al., 2024], decision transformer [Liu et al., 2023], and diffusion models to create safe policies [Lin et al., 2023, Zheng et al., 2024, Yao et al., 2024]. CAPS [Chemingui et al., 2025] addresses dynamic safety constraints at deployment time by switching between a set of pre-trained policies based on the test-time cost budget. [Guo et al., 2025] introduced CCAC framework, which conditions the policy and value functions on cost thresholds to enable adaptation and generalization to unseen constraint levels. FISOR [Zheng et al., 2024] addresses safety by enforcing hard constraints—ensuring state-wise cost constraint satisfaction. Rather than optimizing toward a given cost limit, FISOR focuses on minimizing cost directly by learning policies that select actions only within the feasible region.

This paper aims to address the following two drawbacks of prior work: **1)** Lagrangian based constrained policy optimization methods solve inter-twined optimization problems and are typically unstable in practice resulting in poor performance; and **2)** While FISOR produces safe policies under tight cost constraints, it achieves low reward performance. The proposed O3SRL framework provides a theoretically-grounded and yet practically-effective solution that can be wrapped around existing offline RL methods and does not require an OPE procedure.

### 4 O3SRL Framework for Offline Constrained RL

In this section, we first provide a minimax optimization formulation of the offline constrained RL problem. Next, we describe the general O3SRL framework based on no-regret algorithms to solve this optimization problem along with theoretical guarantees for convergence to the minimax solution. Finally, we provide an approximate algorithm for practical purposes.

## 4.1 Minimax Optimization Formulation

The discounted value functions of a policy  $\pi$  for reward and cost objectives are defined as follows.

$$V_r^\pi := \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad \text{and} \quad V_c^\pi := \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \right]$$

where  $\gamma$  is the discount factor,  $r(s_t, a_t) = r_t$  and  $c(s_t, a_t) = c_t$  are the reward and cost values corresponding to state-action pair  $(s_t, a_t)$  respectively.

Let  $\Pi$  be the set of policies and  $\Delta\Pi$  be the set of all distributions over  $\Pi$ . We aim to solve the following constrained optimization problem:

$$\max_{D \in \Delta\Pi} \mathbb{E}_{\pi \sim D} [V_r^\pi] \text{ subject to } \mathbb{E}_{\pi \sim D} [V_c^\pi] \leq \kappa. \quad (1)$$

Let  $L(D, \lambda)$  be the Lagrangian of Equation (1) where  $\lambda$  is the Lagrange multiplier.

$$L(D, \lambda) := \mathbb{E}_{\pi \sim D} [V_r^\pi] - \lambda(\mathbb{E}_{\pi \sim D} [V_c^\pi] - \kappa) = \mathbb{E}_{\pi \sim D} \left[ V_{r-\lambda(c-(1-\gamma)\kappa)}^\pi \right]. \quad (2)$$

The dual problem of Equation (1) is given by:

$$\min_{\lambda \geq 0} \max_{D \in \Delta\Pi} L(D, \lambda). \quad (3)$$

Note that the objective and the constraint in Equation (1) are linear, thus convex and differentiable. Additionally, the constraint in Equation (1) satisfies Slater's condition when  $\Pi$  is expressive enough that there exists at least one policy  $\pi \in \Pi$  that is strictly safe ( $V_c^\pi \leq \kappa$ ), thus strong duality holds. This implies that solving the primal problem in Equation (1) is equivalent to solving the dual problem in Equation (3). It is also equivalent to constraining the domain of  $\lambda$  to any bounded domain  $\Lambda = [0, C]$  for some  $C > 0$ : if  $(D^*, \lambda^*)$  is a solution to the dual problem and if  $\lambda^* > 0$ , then  $(D^*, 1)$  is also a solution to the dual problem. This is because under strong duality, any solution  $(D^*, \lambda^*)$  of the dual must satisfy the KKT conditions:  $\lambda(\mathbb{E}_{\pi \sim D} [V_c^\pi] - \kappa) = 0$ .

**Definition 1** ( $\epsilon$ -approximate equilibrium).  $(\hat{D}, \hat{\lambda}) \in \Delta\Pi \times \Lambda$  is  $\epsilon$ -approximate minimax equilibrium of Equation (3) if

$$L(D, \hat{\lambda}) - \epsilon \leq L(\hat{D}, \hat{\lambda}) \leq L(\hat{D}, \lambda) + \epsilon, \forall (D, \lambda) \in \Delta\Pi \times \Lambda.$$

## 4.2 General O3SRL Framework via No-Regret Algorithms

**Overview of the O3SRL Framework.** O3SRL relies on two key elements: 1) an offline RL oracle which takes a training dataset of (state, action, next-state, and reward) tuples to produce a policy that maximizes reward, and 2) a no-regret algorithm which adaptively selects the value of Lagrange variable  $\lambda$  from a given search space  $\Lambda$ . We initialize  $\lambda$  with  $\lambda_0$  and the distribution over policies  $D$  to  $D_0$  respectively. In each iteration  $t$ , we perform two algorithmic steps. First, we employ the given offline safe RL training dataset  $\mathcal{D}_{OSRL} = \{(s_i, a_i, r_i, c_i, s'_i)\}_{i=1}^n$  of  $n$  examples and convert it into a corresponding offline RL training dataset  $\mathcal{D}_{ORL}^t = \{(s_i, a_i, r'_i, s'_i)\}_{i=1}^n$  by defining a new reward  $r'$  that combines original reward  $r$  and cost  $c$  values using  $\lambda_t$  based on the Equation 2:  $r'_i = r_i - \lambda_{t-1}(c_i - (1 - \gamma)\kappa)$  where  $\gamma$  is the discount factor and  $\kappa$  is the cost constraint threshold. We call an offline RL oracle on the dataset  $\mathcal{D}_{ORL}^t$  to get the distribution over policies  $D_t$ . Second, we call a no-regret algorithm [Hazan et al., 2016] by passing  $\lambda_{t-1}$  and  $D_t$  to get the updated Lagrange variable  $\lambda_t$ . At the end of  $T$  iterations, we return the average distribution of policies  $\bar{D}$  and the average Lagrangian value  $\bar{\lambda}$  as the solution. Algorithm 1 provides a pseudo-code of the O3SRL framework.

In what follows, we first formally define offline RL oracle and no-regret algorithm. Next, we provide a theoretical result that guarantees that the O3SRL framework converges to the minimax solution.

**Definition 2** (Offline RL oracle). Let  $\mathbb{O}_{offline\text{-}RL}$  be an offline RL oracle that takes the offline data  $\mathcal{D}_f^n$  of size  $n$  from an MDP with the expected reward functions  $f$  (and state transition function  $P$ ) and produces a policy distribution  $D = \mathbb{O}_{offline\text{-}RL}(\mathcal{D}_f^n) \in \Delta\Pi$  at the estimation error  $\epsilon_{offline\text{-}RL}(n)$ , i.e.,

$$\mathbb{E}_{\mathcal{D}_f^n} \mathbb{E}_{\pi \sim \mathbb{O}_{offline\text{-}RL}(\mathcal{D}_f^n)} [V_f^* - V_f^\pi] \leq \epsilon_{offline\text{-}RL}(n). \quad (4)$$

---

**Algorithm 1** General O3SRL Framework for Offline Constrained RL via No-Regret Algorithms

---

**Require:** Offline training data  $\mathcal{D}_{OSRL} = \{(s_i, a_i, r_i, c_i, s'_i)\}_{i=1}^n$ , Offline RL oracle  $\mathbb{O}_{\text{offline-RL}}$ , no-regret update oracle NO\_REGRET\_UPDATE, and search space of Lagrange variable  $\Lambda$

- 1: Initialize  $\lambda_0 \in \Lambda, D_0 \in \Delta\Pi$
- 2: **for**  $t = 1, \dots, T$  **do**
- 3:      $D_t \leftarrow \mathbb{O}_{\text{offline-RL}}(\mathcal{D}_{ORL}^t = \{(s_i, a_i, r'_i = r_i - \lambda_{t-1}(c_i - (1-\gamma)\kappa, s'_i)\}_{i=1}^n)$
- 4:      $\lambda_t \leftarrow \text{NO\_REGRET\_UPDATE}(\lambda_{t-1}, D_t, \Lambda)$
- 5: **end for**

**Ensure:**  $\bar{D} = \frac{1}{T} \sum_{t=1}^T D_t$  and  $\bar{\lambda} = \frac{1}{T} \sum_{t=1}^T \lambda_t$  ▷ Parameter averaging

---

**Remark 1.** The estimation error of an offline RL algorithm  $\epsilon_{\text{offline-RL}}(n)$  also depends on the coverage of the offline data for the optimal policy. This coverage in turn depends on the underlying MDP class, i.e., the stochastic transition function and reward function. In Algorithm 1, we employ an offline RL algorithm repeatedly for multiple MDPs with different reward functions. However, the same order rate  $\epsilon_{\text{offline-RL}}(n)$  can be applied to multiple such offline RL guarantees, with perhaps at the cost of a constant factor. The main reason is that, most notions of coverage depend very mildly on the reward functions, if at all. For example, the concentrability coefficients [Liu et al., 2020, Rashidinejad et al., 2021] do not depend on the reward function at all. Similarly, the data diversity measure in [Nguyen-Tang and Arora, 2023a] and the policy transfer coefficient [Nguyen-Tang and Arora, 2024] do not depend on the specific reward function but rather a class of reward functions. In particular, if we use  $\mathcal{F}$  to approximate  $V_r^\pi$  and  $\mathcal{G}$  to approximate  $V_c^\pi$ , then  $\mathcal{F} - \Lambda \cdot \mathcal{G} := \{f - \lambda \cdot g : f \in \mathcal{F}, g \in \mathcal{G}, \lambda \in \Lambda\}$  suffices to control the same policy transfer coefficient of a class of MDPs defined by any newly created reward functions used in Algorithm 1. Additionally, under good data coverage, we typically have  $\epsilon_{\text{offline-RL}}(n) = \mathcal{O}(\frac{1}{\sqrt{n}})$ .

**Definition 3** (No-regret algorithm). *Let NO\_REGRET\_UPDATE is a no-regret algorithm for  $\lambda \in \Lambda$  with regret bound  $R_T(\Lambda)$ , i.e.,*

$$\lambda_{t+1} = \text{NO\_REGRET\_UPDATE}(\lambda_t, D_t, \Lambda),$$

where

$$\sum_{t=1}^T L(D_t, \lambda_t) - \min_{\lambda \in \Lambda} \sum_{t=1}^T L(D_t, \lambda) \leq R_T(\Lambda) \quad (5)$$

and  $\lim_{T \rightarrow \infty} \frac{R_T(\Lambda)}{T} \rightarrow 0$ .

We consider a generic framework for solving offline constrained RL as shown in Algorithm 1. It relies on an offline RL oracle as per Definition 2 and a no-regret update oracle as per Definition 3.

**Remark 2.** From the given offline safe RL training dataset  $\mathcal{D}_{OSRL} = \{(s_i, a_i, r_i, c_i, s'_i)\}_{i=1}^n$  of  $n$  examples, we can create a corresponding dataset that simulates the experiences of any MDP with the same unknown state transition probability function and an arbitrary reward function  $r' = f(r, c)$ , by simply computing  $r'_i = f(r_i, c_i)$ . Specifically, in our algorithm we define the modified reward function based on the Equation 2 as:  $r' = r - \lambda_t(c - (1-\gamma)\kappa)$  where  $\lambda_t \in \Lambda$  is the Lagrange variable,  $\gamma$  is the discount factor, and  $\kappa$  is the cost constraint threshold.

The below theorem provides guarantee that Algorithm 1 will converge to the minimax solution.

**Theorem 1.** *Let  $(\bar{D}, \bar{\lambda})$  be the output of Algorithm 1. Then,  $(\bar{D}, \bar{\lambda})$  is  $\epsilon$ -approximate equilibrium, where  $\epsilon = \epsilon_{\text{offline-RL}}(n) + \frac{R_T(\Lambda)}{T}$ .*

See the detailed proof in Section A.

## 5 Practical Algorithms

In this section, we introduce an approximation to Algorithm 1 that addresses two key practical challenges. Below we outline these challenges and the corresponding modifications. The resulting Algorithm 2 with minor approximation forms the basis of our primary empirical contribution.

The first practical challenge is that Algorithm 1 assumes access to an offline RL oracle, which is computationally expensive to approximate by running offline RL algorithms to convergence. This is especially problematic because the oracle must be invoked in every round of the algorithm. To mitigate this cost, we assume a weaker oracle: a *stochastic oracle* for offline RL in Assumption 4. This assumption says that we have access to an oracle that returns a near-optimal policy and a stochastic approximation of the value function of this near-optimal policy.

**Definition 4.** Let  $\mathbb{SO}_{\text{offline-RL}}$  be a stochastic oracle for offline RL that takes the offline data  $\mathcal{D}_f^n$  of size  $n$  from an MDP with the expected reward function  $f$  (and state transition function  $P$  across all MDP instances we have considered so far) and produces a policy distribution and a stochastic approximation of the value of the optimal policy:  $(\tilde{D}, \tilde{V}) = \mathbb{SO}_{\text{offline-RL}}(\mathcal{D}_f^n)$ , where

$$\mathbb{E}_{\mathcal{D}_f^n} \mathbb{E}_{\pi \sim \tilde{D}} [V_f^* - V_f^\pi] \leq \epsilon_{\text{offline-RL}}(n), \text{ and } \mathbb{E}[\tilde{V}] = \mathbb{E}_{\pi \sim \tilde{D}} [V_f^\pi]. \quad (6)$$

This significantly relaxes the need to return an exact value of the value function of a near-optimal policy by an offline RL oracle, and, more importantly, leaves room for further practical approximation (as we elaborate in Section 5.1).

The second practical challenge arises from the use of no-regret algorithms over a continuous range of  $\lambda$ . The online optimization over a continuous domain of  $\lambda$  requires an off-policy estimate of a target policy under a mixed reward of the form  $r - \lambda c$  for every value of  $\lambda$ . This often requires an OPE procedure (to compute  $V_{c-(1-\gamma)\kappa}^{D_t}$  where  $D_t$  is the near-optimal policy distribution returned at line 4 in Algorithm 1) at every iteration. The application of OPE at every iteration is problematic in two ways: 1) Can compound the OPE estimation error exponentially over the length of the iterations, causing instability and large error; and 2) Employing OPE for every iteration is computationally expensive.

To address this, we discretize the range of  $\lambda$  into  $K$  discrete values  $\{\lambda^{(1)}, \dots, \lambda^{(K)}\}$  and optimize over this finite set. This transforms the problem into a multi-armed bandit (MAB) setting, where each of the  $K$  arms corresponds to a particular  $\lambda$  value. Crucially, many MAB algorithms do not require estimates of the underlying value functions, thus avoiding the need for OPE. In this work, we adopt the EXP3 algorithm [Auer et al., 2002a], which maintains a distribution over arms that is updated and sampled from in each round of Algorithm 2.

---

**Algorithm 2** O3SRL Approximation via Stochastic Oracle and EXP3 Multi-Arm Bandit Strategy

---

**Require:** Stochastic oracle for offline RL  $\mathbb{SO}_{\text{offline-RL}}$ , Offline safe RL training data  $\mathcal{D}_{OSRL} = \{(s_i, a_i, r_i, c_i, s'_i)\}_{i=1}^n$ , and search space  $\Lambda = \{\lambda^{(1)}, \dots, \lambda^{(K)}\}$

1: Initialize  $P_0(\lambda) = \frac{1}{K}$ ,  $\bar{c} = \kappa/H$ , where  $\kappa$  is cost limit and  $H = \frac{1}{1-\gamma}$  is the horizon

2: **for**  $t = 1, \dots, T$  **do**

3:      $(\tilde{D}_t, \tilde{V}_t) = \mathbb{SO}_{\text{offline-RL}}(\mathcal{D}_{ORL}^t = \{(s_i, a_i, r'_i = r_i - \lambda_{t-1}(c_i - (1-\gamma)\kappa, s'_i)\}_{i=1}^n)$

4:     Sample  $\lambda_t \sim P_t$  where  $P_t(\lambda) \propto P_{t-1}(\lambda) \exp\left(-\eta \frac{\tilde{V}_t 1\{\lambda = \lambda_{t-1}\}}{P_{t-1}(\lambda)}\right)$

5: **end for**

**Ensure:**  $\bar{D} = \text{Proj}_{\Delta\Pi}\left(\frac{1}{T} \sum_{t=1}^T \tilde{D}_t\right)$ ,  $\bar{\lambda} = \text{Proj}_{\Lambda}\left(\frac{1}{T} \sum_{t=1}^T \lambda_t\right)$

---

**Theorem 2.** Let  $\Lambda = \{\lambda^{(1)}, \dots, \lambda^{(K)}\}$ , where  $\lambda^1 = 0$  and  $\lambda^{(i+1)} - \lambda^{(i)} = \frac{C}{K}$ . Then, the solution  $(\bar{D}, \bar{\lambda})$  returned by Algorithm 2 is  $\epsilon$ -approximate minimax equilibrium of Equation (3), where

$$\epsilon = \mathcal{O}\left(\underbrace{\epsilon_{\text{offline-RL}}(n)}_{\text{error rate of offline RL}} + \underbrace{\sqrt{\frac{K}{T}}}_{\text{regret of EXP3}} + \underbrace{\frac{1}{K}}_{\text{discretization error}}\right).$$

**Remark 3:** If we set  $K = \Theta(T^{1/3})$  and assume the minimax rate  $\frac{1}{\sqrt{n}}$  of offline RL algorithm, then the error rate in Theorem 2 will be in the order of  $\frac{1}{\sqrt{n}} + \frac{1}{\sqrt[3]{T}}$ . This rate is a bit slower than the rate  $\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{T}}$  had we have applied a no-regret algorithm such as Follow-The-Regularized-Leader [Hazan et al., 2016] over continuous values of  $\lambda$ .

### 5.1 Final Approximate Algorithm for Empirical Evaluation

We consider an even more practical version of Algorithm 2 to form the final algorithm for our empirical evaluation. Specifically, we make two changes.

- We further approximate the stochastic oracle by leveraging the fact that most offline RL algorithms are based on stochastic gradient descent. In our approximation (Algorithm 2), rather than running the offline RL algorithm to convergence at round  $t$ , we perform only  $M$  gradient updates, after initializing from the result at the end of round  $t - 1$ . We find this truncated update scheme to be effective even for small values of  $M$ .
- Algorithm 2 (and Algorithm 1) maintains an averaged distribution over policies across iterations that is returned at the end of  $T$  rounds. In our case, this would involve storing the policy produced on each round, which would be prohibitively memory intensive. Instead, we simply return the last-iterate policy  $\pi_T$ .

In our experiments, we have found that this set of approximations leads to state-of-the-art performance even for as few as  $K=2$  arms and a small number of Offline RL iterations  $M$  per round.

## 6 Experiments and Results

In this section, we experimentally evaluate the performance of the approximate O3SRL algorithm described in Section 5.1, compare with state-of-the-art baselines, and discuss the results.

### 6.1 Experimental Research Questions

Our experiments are designed to answer the following key questions:

**Q1:** What is the influence of the number of arms in our EXP-3 based approximate O3SRL approach on both reward and safety objectives?

**Q2:** How effectively does O3SRL satisfy safety/cost constraints and achieve high reward performance for small cost constraint thresholds?

**Q3:** How does the safety and reward trade-off of O3SRL vary as the cost threshold increases?

**Q4:** Is O3SRL compatible with different offline RL algorithms, demonstrating its generality?

### 6.2 Experimental Setup

**Benchmarks.** We evaluate O3SRL and baseline methods using the DSRL Bullet benchmark [Liu et al., 2024], which provides standardized offline datasets for safe RL research and evaluation. We consider eight continuous control tasks, comprising four “Run” and four “Circle” environments. Each task involves a different agent interacting with either the Run or Circle objective, following the common Agent-Task naming convention (e.g., Car-Run, Drone-Circle).

In the Run tasks, agents are rewarded for moving quickly between two boundaries while adhering to safety constraints, such as velocity limits and positional bounds. In the Circle tasks, agents are encouraged to follow a circular trajectory within a restricted safe region. These are challenging environments which require agents to balance performance with constraint satisfaction.

**Evaluation metrics.** Following the DSRL evaluation protocol [Liu et al., 2024], we evaluate all methods using normalized cumulative reward and normalized cumulative cost. Let  $r_{\max}(M)$  and  $r_{\min}(M)$  denote the maximum and minimum empirical reward returns achievable for task  $M$ . The normalized reward is computed as  $R_{\text{normalized}} = \frac{R_\pi - r_{\min}(M)}{r_{\max}(M) - r_{\min}(M)}$ , where  $R_\pi$  is the cumulative reward of policy  $\pi$ . The normalized cost is given by  $C_{\text{normalized}} = \frac{C_\pi}{\kappa}$ , with  $\kappa$  being the cost budget.

where  $C_\pi$  is the cumulative cost of the policy  $\pi$  and  $\kappa$  is the cost threshold. A policy  $\pi$  is considered safe if  $C_{\text{normalized}} \leq 1$ . Our O3SRL method is general-purpose and can be trained for any given cost threshold  $\kappa$ . However, we report our main results under a stringent low cost threshold of  $\kappa=5$  to

highlight the method’s effectiveness in highly constrained settings. We report both mean and standard deviations over three random seeds, with each seed’s policy evaluated across twenty episodes.

**Baseline methods.** We compare O3SRL with several state-of-the-art offline safe RL methods. **(1)** BC-Safe is a behavior cloning baseline trained exclusively on safe trajectories that satisfy the given cost limit. **(2)** BEAR-Lag is a Lagrangian-based extension of BEAR [Kumar et al., 2019], which incorporates safety constraints during training. **(3)** CPQ [Xu et al., 2022b] penalizes unsafe actions by treating out-of-distribution actions as inherently risky and updates its Q-function using only safe transitions. **(4)** COptIDICE [Lee et al., 2022] builds on OptiDICE [Lee et al., 2021] by applying stationary distribution correction under cost constraints. **(5)** CDT [Liu et al., 2023] is a transformer-based method that conditions policies on returns and costs, allowing constraint-aware sequence modeling. **(6)** CCAC [Guo et al., 2025] learns a policy that adapts to changing safety budgets by jointly modeling constraints and environment dynamics. This approach conditions both the policy and value functions on constraint information. **(7)** CAPS [Chemingui et al., 2025], similar to CDT and CCAC, supports test-time adaptation to varying cost constraints by switching between a set of pre-trained policies with different cost and reward trade-offs. **(8)** FISOR [Zheng et al., 2024] is a diffusion-based method that explicitly optimizes for feasibility, aims to produce zero-violation policies, and is the state-of-the-art method for stringent cost constraints.

All baselines are evaluated under the same cost constraint threshold  $\kappa = 5$ . All results are averaged over three random seeds, with each policy evaluated across 20 episodes. Additional details are in the Appendix.

**Configuration of O3SRL.** We implement the approximate O3SRL algorithm described in Section 5.1 using TD3+BC [Fujimoto and Gu, 2021] as the underlying offline RL algorithm for our main results. We set the search space of Lagrange variable  $\Lambda = [0, C=5]$  and use  $K=5$  arms for the main results but we also present ablation results for  $K=2, 5$ , and  $10$ . We observe that increasing  $K$  beyond 5 (say to 10) yields only marginal improvements compared to the gains observed when increasing from  $K=2$  to  $K=5$ . The algorithm is trained for  $T = 100,000$  iterations, with arm probabilities updated every  $M = 10$  steps of the base offline RL algorithm (i.e.,  $M=10$  stochastic gradient updates of the policy). For baselines, we employ the DSRL [Liu et al., 2024] implementation wherever available. For CCAC, CAPS and FISOR, we use the publicly released code provided by their respective authors. We employ IQL based instantiation of CAPS in our evaluation. Full architectural and hyperparameter details are provided in the Appendix.

### 6.3 Results and Discussion

**Performance of O3SRL vs. the number of arms  $K$ .** To evaluate the impact of the granularity of discretization of  $\lambda$ , we test O3SRL using  $K \in \{2, 5, 10\}$  arms. Table 1 shows the corresponding results. Across all six benchmark tasks, O3SRL satisfies the cost constraint under every configuration and maintains consistently high reward, demonstrating that the EXP3 strategy is effective even with a coarse discretization of  $\lambda$ . With just two arms, the method keeps average cost at or near zero, but tends to underperform slightly in terms of rewards. Increasing to ten arms improves rewards marginally. The five-arm configuration achieves a good trade-off: it either matches or exceeds the highest feasible reward across all tasks while keeping costs safely below the threshold. For this reason, we adopted  $K = 5$  configuration of the approximate O3SRL approach for all main results.

Table 1: O3SRL results for different values of  $K$ : the number of arms. Each value shows the average reward ( $\uparrow$ ) and cost ( $\downarrow$ ). Higher reward is better, and lower cost (up to threshold 1) is better.

| # Arms   | BallRun           |                   | CarRun            |                   | AntRun            |                   | BallCircle        |                   | CarCircle         |                   | AntCircle         |                   |
|----------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
|          | reward $\uparrow$ | cost $\downarrow$ |
| $K = 2$  | <b>0.20</b>       | <b>0.00</b>       | <b>0.96</b>       | <b>0.01</b>       | <b>0.24</b>       | <b>0.03</b>       | <b>0.62</b>       | <b>0.00</b>       | <b>0.64</b>       | <b>0.00</b>       | <b>0.44</b>       | <b>0.04</b>       |
| $K = 5$  | <b>0.25</b>       | <b>0.00</b>       | <b>0.96</b>       | <b>0.02</b>       | <b>0.33</b>       | <b>0.14</b>       | <b>0.62</b>       | <b>0.06</b>       | <b>0.66</b>       | <b>0.11</b>       | <b>0.48</b>       | <b>0.00</b>       |
| $K = 10$ | <b>0.27</b>       | <b>0.00</b>       | <b>0.96</b>       | <b>0.00</b>       | <b>0.29</b>       | <b>0.17</b>       | <b>0.64</b>       | <b>0.28</b>       | <b>0.67</b>       | <b>0.15</b>       | <b>0.49</b>       | <b>0.00</b>       |

**O3SRL vs. Baselines.** Table 2 presents the comparative evaluation of O3SRL against baselines across eight DSRL tasks. The table reports both normalized reward (higher is better) and normalized cost (lower is better, with values less than 1 indicating constraint satisfaction). O3SRL is the *only* method that consistently satisfies the cost constraint across all eight tasks. This is particularly noteworthy given the stringent cost budget, which poses a significant challenge for all baseline algorithms.

Table 2: O3SRL normalized rewards and costs results. The cost threshold is 1. The  $\uparrow$  symbol denotes that higher rewards are better. The  $\downarrow$  symbol denotes that lower costs (up to threshold 1) are better. Each value is averaged over 20 evaluation episodes, and three random seeds. **Bold**: Safe agents whose normalized cost  $\leq 1$ . Gray: Unsafe agents. **Blue**: Safe agents with the highest reward.

| Tasks       |                   | BC Safe                         | BEAR-Lag                        | CPQ                              | COptiDICE                       | CDT                             | CCAC                            | CAPS                            | FISOR                           | O3SRL                           |
|-------------|-------------------|---------------------------------|---------------------------------|----------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| BallRun     | Reward $\uparrow$ | 0.16 $\pm$ 0.11                 | 0.53 $\pm$ 0.47                 | 0.09 $\pm$ 0.26                  | 0.53 $\pm$ 0.10                 | 0.27 $\pm$ 0.09                 | <b>0.31<math>\pm</math>0.01</b> | <b>0.07<math>\pm</math>0.05</b> | 0.09 $\pm$ 0.07                 | <b>0.25<math>\pm</math>0.03</b> |
|             | Cost $\downarrow$ | 4.50 $\pm$ 3.38                 | 18.60 $\pm$ 0.35                | 2.20 $\pm$ 2.88                  | 10.83 $\pm$ 1.68                | 2.57 $\pm$ 3.23                 | <b>0.00<math>\pm</math>0.00</b> | <b>0.00<math>\pm</math>0.00</b> | 1.28 $\pm$ 1.70                 | <b>0.00<math>\pm</math>0.00</b> |
| CarRun      | Reward $\uparrow$ | <b>0.92<math>\pm</math>0.01</b> | 0.25 $\pm$ 1.23                 | <b>0.93<math>\pm</math>0.01</b>  | <b>0.91<math>\pm</math>0.04</b> | <b>0.99<math>\pm</math>0.00</b> | 1.82 $\pm$ 0.84                 | <b>0.97<math>\pm</math>0.00</b> | <b>0.74<math>\pm</math>0.01</b> | <b>0.96<math>\pm</math>0.01</b> |
|             | Cost $\downarrow$ | <b>0.26<math>\pm</math>0.23</b> | 30.17 $\pm$ 10.64               | <b>0.20<math>\pm</math>0.18</b>  | <b>0.00<math>\pm</math>0.00</b> | <b>0.90<math>\pm</math>0.34</b> | 24.57 $\pm$ 20.94               | <b>0.11<math>\pm</math>0.17</b> | <b>0.00<math>\pm</math>0.00</b> | <b>0.02<math>\pm</math>0.03</b> |
| DroneRun    | Reward $\uparrow$ | 0.41 $\pm$ 0.23                 | -0.21 $\pm$ 0.12                | 0.29 $\pm$ 0.11                  | 0.68 $\pm$ 0.01                 | <b>0.58<math>\pm</math>0.00</b> | 0.50 $\pm$ 0.08                 | 0.41 $\pm$ 0.06                 | 0.31 $\pm$ 0.04                 | <b>0.32<math>\pm</math>0.05</b> |
|             | Cost $\downarrow$ | 1.62 $\pm$ 1.73                 | 14.85 $\pm$ 8.77                | 2.35 $\pm$ 4.08                  | 15.02 $\pm$ 0.10                | <b>0.07<math>\pm</math>0.07</b> | 16.29 $\pm$ 9.35                | 5.70 $\pm$ 3.08                 | 2.52 $\pm$ 1.10                 | <b>0.68<math>\pm</math>1.18</b> |
| AntRun      | Reward $\uparrow$ | 0.56 $\pm$ 0.02                 | <b>0.02<math>\pm</math>0.02</b> | <b>0.03<math>\pm</math>0.05</b>  | 0.61 $\pm$ 0.01                 | 0.70 $\pm$ 0.03                 | <b>0.03<math>\pm</math>0.10</b> | 0.53 $\pm$ 0.13                 | <b>0.43<math>\pm</math>0.02</b> | <b>0.33<math>\pm</math>0.13</b> |
|             | Cost $\downarrow$ | 1.15 $\pm$ 0.47                 | <b>0.00<math>\pm</math>0.01</b> | <b>0.05<math>\pm</math>0.08</b>  | 3.26 $\pm$ 1.39                 | 1.66 $\pm$ 0.24                 | <b>0.00<math>\pm</math>0.00</b> | 2.03 $\pm$ 2.17                 | <b>0.27<math>\pm</math>0.15</b> | <b>0.14<math>\pm</math>0.10</b> |
| BallCircle  | Reward $\uparrow$ | 0.45 $\pm$ 0.03                 | 0.85 $\pm$ 0.05                 | 0.56 $\pm$ 0.10                  | 0.71 $\pm$ 0.01                 | 0.61 $\pm$ 0.17                 | 0.47 $\pm$ 0.38                 | <b>0.33<math>\pm</math>0.02</b> | <b>0.32<math>\pm</math>0.05</b> | <b>0.62<math>\pm</math>0.01</b> |
|             | Cost $\downarrow$ | 1.21 $\pm$ 0.23                 | 10.92 $\pm$ 2.02                | 1.11 $\pm$ 1.92                  | 9.41 $\pm$ 0.69                 | 2.03 $\pm$ 1.39                 | 10.19 $\pm$ 17.65               | <b>0.01<math>\pm</math>0.02</b> | <b>0.00<math>\pm</math>0.00</b> | <b>0.06<math>\pm</math>0.07</b> |
| CarCircle   | Reward $\uparrow$ | 0.31 $\pm$ 0.06                 | 0.73 $\pm$ 0.05                 | <b>0.71<math>\pm</math>0.02</b>  | 0.49 $\pm$ 0.01                 | 0.71 $\pm$ 0.01                 | 0.70 $\pm$ 0.01                 | <b>0.40<math>\pm</math>0.03</b> | <b>0.37<math>\pm</math>0.02</b> | <b>0.66<math>\pm</math>0.03</b> |
|             | Cost $\downarrow$ | 1.57 $\pm$ 0.07                 | 7.39 $\pm$ 1.28                 | <b>0.00<math>\pm</math>0.00</b>  | 10.82 $\pm$ 1.28                | 1.58 $\pm$ 0.57                 | 1.61 $\pm$ 0.51                 | <b>0.03<math>\pm</math>0.03</b> | <b>0.00<math>\pm</math>0.00</b> | <b>0.11<math>\pm</math>0.16</b> |
| DroneCircle | Reward $\uparrow$ | 0.50 $\pm$ 0.01                 | 0.84 $\pm$ 0.07                 | <b>-0.21<math>\pm</math>0.04</b> | 0.26 $\pm$ 0.02                 | 0.55 $\pm$ 0.01                 | <b>0.43<math>\pm</math>0.04</b> | <b>0.36<math>\pm</math>0.02</b> | <b>0.48<math>\pm</math>0.01</b> | <b>0.49<math>\pm</math>0.07</b> |
|             | Cost $\downarrow$ | 1.22 $\pm$ 0.31                 | 15.40 $\pm$ 2.62                | <b>0.70<math>\pm</math>0.41</b>  | 3.68 $\pm$ 0.51                 | 1.14 $\pm$ 0.06                 | <b>0.06<math>\pm</math>0.07</b> | <b>0.00<math>\pm</math>0.00</b> | <b>0.17<math>\pm</math>0.15</b> | <b>0.23<math>\pm</math>0.34</b> |
| AntCircle   | Reward $\uparrow$ | 0.40 $\pm$ 0.02                 | 0.67 $\pm$ 0.05                 | <b>0.00<math>\pm</math>0.00</b>  | 0.18 $\pm$ 0.02                 | 0.45 $\pm$ 0.05                 | <b>0.49<math>\pm</math>0.07</b> | <b>0.33<math>\pm</math>0.05</b> | <b>0.24<math>\pm</math>0.02</b> | <b>0.48<math>\pm</math>0.06</b> |
|             | Cost $\downarrow$ | 4.72 $\pm$ 0.87                 | 19.00 $\pm$ 1.68                | <b>0.00<math>\pm</math>0.00</b>  | 17.40 $\pm$ 0.36                | 6.59 $\pm$ 1.42                 | <b>0.02<math>\pm</math>0.03</b> | <b>0.00<math>\pm</math>0.00</b> | <b>0.04<math>\pm</math>0.08</b> | <b>0.00<math>\pm</math>0.00</b> |

O3SRL’s consistent satisfaction of the cost constraint stands in contrast to methods that only occasionally achieve strong reward, often by violating safety. For instance, CDT achieves high rewards on tasks such as *CarRun* and *DroneRun*, but exceeds the safety budget in the other six tasks—making it unreliable. Similarly, CCAC achieves the best rewards on *BallRun* and *AntCircle*, yet violates the cost constraint in four out of eight tasks, further underscoring the challenge of maintaining both high performance and safety across environments. CPQ, on the other hand, stays within the budget more often, but its performance is inconsistent: it achieves a competitive reward on *CarCircle*, while falling short on rewards when it does remain safe, and it exceeds the cost threshold in three tasks.

While CAPS and FISOR maintain safe behavior in six out of eight tasks, their reward performance consistently trails behind O3SRL. For example, in *BallRun*, O3SRL is one of the only three safe methods (along with CAPS and CCAC), yet it achieves a reward of  $0.25 \pm 0.03$ , which is over three times higher than CAPS ( $0.07 \pm 0.05$ ). A similar pattern holds in *BallCircle*, where O3SRL maintains safety and attains  $0.65 \pm 0.01$  reward—more than double that of CAPS ( $0.23 \pm 0.17$ ) and FISOR ( $0.27 \pm 0.15$ ), both of which are also safe in that task. Note that FISOR, which aims for zero cost policies, fails to meet the cost threshold in the *DroneRun* and *BallRun* environments.

Overall, the results in Table 2 demonstrate that O3SRL strikes a uniquely favorable balance between maximizing rewards and satisfying safety constraints. It consistently ensures safety where many other methods fail, especially under a tight cost budget. Crucially, this safety does not come at a severe expense to the reward performance; O3SRL is the best-performing method (in terms of reward) among safe agents in two out of eight tasks, and consistently ranks second in the rest. This suggests that O3SRL’s iterative, no-regret approach to optimizing the reward and safety trade-off is more robust and effective than direct Lagrangian optimization or specialized heuristic-based safety mechanisms in these challenging offline safe RL settings.

**Performance of O3SRL vs. increasing cost limits.** To assess the flexibility of O3SRL in handling different safety constraints, we perform an ablation study across two extra cost limits beyond the default setting of  $\kappa = 5$  used in the main experiments. Table 3 shows the corresponding results. We find that O3SRL continues to perform effectively under higher cost limits (i.e., less stringent safety constraints), adjusting its behavior to make better use of the available budget. As the constraint becomes more permissive, the learned policy shifts toward higher-reward strategies, while still respecting the specified cost limit. These results demonstrate that the O3SRL method does not rely on tuning for a particular budget.

**O3SRL with an alternative Offline RL algorithm.** To evaluate the generality of O3SRL with respect to the underlying offline RL algorithm, we conduct an ablation using Implicit Q-Learning (IQL) [Kostrikov et al., 2022] in place of TD3+BC, which is used in our main experiments, on the Circle tasks. Table 4 shows the corresponding results. We observe that O3SRL combined with

Table 3: O3SRL results for different values of  $\kappa$ : cost budget. Each value shows the average reward ( $\uparrow$ ) and cost ( $\downarrow$ ). Higher reward is better, and lower cost (up to threshold 1) is better.

| Cost Limit    | BallRun           |                   | DroneRun          |                   | BallCircle        |                   | DroneCircle       |                   |
|---------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
|               | reward $\uparrow$ | cost $\downarrow$ |
| $\kappa = 5$  | <b>0.25</b>       | <b>0.00</b>       | <b>0.32</b>       | <b>0.68</b>       | <b>0.62</b>       | <b>0.06</b>       | <b>0.49</b>       | <b>0.23</b>       |
| $\kappa = 20$ | <b>0.26</b>       | <b>0.06</b>       | <b>0.38</b>       | <b>0.41</b>       | <b>0.69</b>       | <b>0.58</b>       | <b>0.53</b>       | <b>0.36</b>       |
| $\kappa = 40$ | <b>0.53</b>       | <b>0.71</b>       | <b>0.38</b>       | <b>0.44</b>       | <b>0.76</b>       | <b>0.64</b>       | <b>0.65</b>       | <b>0.85</b>       |

IQL achieves similar performance to the TD3+BC variant, both in satisfying safety constraints and attaining competitive reward. These results demonstrate that O3SRL is not tied to a specific offline RL method and can effectively incorporate different algorithms, highlighting its flexibility as a general framework for offline safe reinforcement learning.

Table 4: O3SRL results for different base offline RL methods. Each value shows the average reward ( $\uparrow$ ) and cost ( $\downarrow$ ). Higher reward is better, and lower cost (up to threshold 1) is better.

| Base       | BallCircle        |                   | CarCircle         |                   | DroneCircle       |                   | AntCircle         |                   |
|------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
|            | reward $\uparrow$ | cost $\downarrow$ |
| Offline RL |                   |                   |                   |                   |                   |                   |                   |                   |
| TD3BC      | <b>0.62</b>       | <b>0.06</b>       | <b>0.66</b>       | <b>0.11</b>       | <b>0.49</b>       | <b>0.23</b>       | <b>0.48</b>       | <b>0.00</b>       |
| IQL        | <b>0.54</b>       | <b>0.06</b>       | <b>0.63</b>       | <b>0.10</b>       | <b>0.42</b>       | <b>0.03</b>       | <b>0.41</b>       | <b>0.06</b>       |

In Appendix, we provide extended experimental results including evaluation on more benchmark tasks; treating  $\lambda$  as a fixed hyper-parameter; and ablation results for O3SRL over search space of  $\lambda$  and discretization levels, and the number of stochastic gradient updates of offline RL ( $M$ ).

## 7 Summary and Future Work

This paper developed and evaluated a novel online optimization framework called O3SRL to create safe and reward-maximizing decision policies from offline datasets. O3SRL solves a minimax optimization problem by employing an iterative approach that adaptively updates the policy distribution using an offline RL oracle and Lagrange variables in each iteration. To handle the practical challenges, we studied an approximate algorithm that works over discrete values of Lagrange variables (to avoid offline policy evaluation) and performs few gradient updates of an offline RL algorithm (for computational efficiency). Empirical results show that our approximate O3SRL algorithm outperforms state-of-the-art methods, especially for stringent safety/cost constraints. Future work includes applying O3SRL to real-world applications and extending it to the offline-to-online safe RL setting.

## Acknowledgments

The authors gratefully acknowledge the in part support by USDA-NIFA funded AgAID Institute award 2021-67021- 35344. The views expressed are those of the authors and do not reflect the official policy or position of the USDA-NIFA.

## References

- E. Altman. *Constrained Markov decision processes*. Routledge, 2021.
- G. An, S. Moon, J.-H. Kim, and H. O. Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.
- P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002a. doi: 10.1137/S0097539701398375.
- P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002b.
- C. Bai, L. Wang, Z. Yang, Z.-H. Deng, A. Garg, P. Liu, and Z. Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=Y4cs1Z3HnqL>.
- Y. Chemingui, A. Deshwal, T. N. Hoang, and J. R. Doppa. Offline model-based optimization via policy-guided gradient search. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024*, pages 11230–11239. AAAI Press, 2024.
- Y. Chemingui, A. Deshwal, H. Wei, A. Fern, and J. Doppa. Constraint-adaptive policy switching for offline safe reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 15722–15730, 2025.
- H. Chen, C. Lu, C. Ying, H. Su, and J. Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023.
- X. Fang, Q. Zhang, Y. Gao, and D. Zhao. Offline reinforcement learning for autonomous driving with real world driving data. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3417–3422. IEEE, 2022.
- R. Figueiredo Prudencio, M. R. O. A. Maximo, and E. L. Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(8):10237–10257, 2024. doi: 10.1109/TNNLS.2023.3250269.
- S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.
- Z. Guo, W. Zhou, S. Wang, and W. Li. Constraint-conditioned actor-critic for offline safe reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.
- P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.
- E. Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.
- M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.
- N. Jaques, A. Ghandeharioun, J. H. Shen, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard. Way off-policy batch deep reinforcement learning of human preferences in dialog, 2020. URL <https://openreview.net/forum?id=rJ15rRVFvH>.
- R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel: Model-based offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21810–21823. Curran Associates, Inc., 2020. URL [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/f7efa4f864ae9b88d43527f4b14f750f-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/f7efa4f864ae9b88d43527f4b14f750f-Paper.pdf).
- I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=68n2s9ZJWF8>.

- A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.
- H. Le, C. Voloshin, and Y. Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pages 3703–3712. PMLR, 2019.
- J. Lee, W. Jeon, B. Lee, J. Pineau, and K.-E. Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning*, pages 6120–6130. PMLR, 2021.
- J. Lee, C. Paduraru, D. J. Mankowitz, N. Heess, D. Precup, K.-E. Kim, and A. Guez. COptiDICE: Offline constrained reinforcement learning via stationary distribution correction estimation. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=FLA55mBee6Q>.
- S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.
- J. Li, X. Liu, B. Zhu, J. Jiao, M. Tomizuka, C. Tang, and W. Zhan. Guided online distillation: Promoting safe reinforcement learning by offline demonstration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7447–7454. IEEE, 2024.
- H. Lin, W. Ding, Z. Liu, Y. Niu, J. Zhu, Y. Niu, and D. Zhao. Safety-aware causal representation for trustworthy offline reinforcement learning in autonomous driving. *IEEE Robotics and Automation Letters*, 2024.
- Q. Lin, B. Tang, Z. Wu, C. Yu, S. Mao, Q. Xie, X. Wang, and D. Wang. Safe offline reinforcement learning with real-time budget constraints. In *International Conference on Machine Learning*, pages 21127–21152. PMLR, 2023.
- Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill. Off-policy policy gradient with stationary distribution correction. In *Uncertainty in artificial intelligence*, pages 1180–1190. PMLR, 2020.
- Z. Liu, Z. Guo, Y. Yao, Z. Cen, W. Yu, T. Zhang, and D. Zhao. Constrained decision transformer for offline safe reinforcement learning. In *International Conference on Machine Learning*, pages 21611–21630. PMLR, 2023.
- Z. Liu, Z. Guo, H. Lin, Y. Yao, J. Zhu, Z. Cen, H. Hu, W. Yu, T. Zhang, J. Tan, and D. Zhao. Datasets and benchmarks for offline safe reinforcement learning. *Journal of Data-centric Machine Learning Research*, 2024.
- J. Lyu, X. Ma, X. Li, and Z. Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1711–1724, 2022.
- T. Nguyen-Tang and R. Arora. On sample-efficient offline reinforcement learning: Data diversity, posterior sampling and beyond. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL <https://openreview.net/forum?id=sdlh4gV0j8>.
- T. Nguyen-Tang and R. Arora. Viper: Provably efficient algorithm for offline rl with neural function approximation. *arXiv preprint arXiv:2302.12780*, 2023b.
- T. Nguyen-Tang and R. Arora. On the statistical complexity of offline decision-making. In *Proceedings of the 41st International Conference on Machine Learning*, pages 37900–37928, 2024.
- N. Polosky, B. C. Da Silva, M. Fiterau, and J. Jagannath. Constrained offline policy optimization. In *International Conference on Machine Learning*, pages 17801–17810. PMLR, 2022.
- P. Rashidinejad, B. Zhu, C. Ma, J. Jiao, and S. Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, 34: 11702–11716, 2021.
- M. Rigter, B. Lacerda, and N. Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35:16082–16097, 2022.

- A. Wachi, X. Shen, and Y. Sui. A survey of constraint formulations in safe reinforcement learning. *arXiv preprint arXiv:2402.02025*, 2024.
- K. Wang, H. Zhao, X. Luo, K. Ren, W. Zhang, and D. Li. Bootstrapped transformer for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:34748–34761, 2022.
- H. Wei, X. Peng, A. Ghosh, and X. Liu. Adversarially trained weighted actor-critic for safe offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:52806–52835, 2024.
- J. Wu, H. Wu, Z. Qiu, J. Wang, and M. Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:31278–31291, 2022.
- Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning, 2020. URL <https://openreview.net/forum?id=BJg9hTNKPH>.
- H. Xu, L. Jiang, L. Jianxiong, and X. Zhan. A policy-guided imitation approach for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:4085–4098, 2022a.
- H. Xu, X. Zhan, and X. Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8753–8760, 2022b.
- R. Yang, C. Bai, X. Ma, Z. Wang, C. Zhang, and L. Han. RORL: Robust offline reinforcement learning via conservative smoothing. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL [https://openreview.net/forum?id=\\_QzJJGH\\_KE](https://openreview.net/forum?id=_QzJJGH_KE).
- Y. Yao, Z. Cen, W. Ding, H. Lin, S. Liu, T. Zhang, W. Yu, and D. Zhao. Oasis: Conditional distribution shaping for offline safe reinforcement learning. *Advances in Neural Information Processing Systems*, 37:78451–78478, 2024.
- T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142. Curran Associates, Inc., 2020. URL [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/a322852ce0df73e204b7e67cbbef0d0a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/a322852ce0df73e204b7e67cbbef0d0a-Paper.pdf).
- X. Zhan, H. Xu, Y. Zhang, X. Zhu, H. Yin, and Y. Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4680–4688, 2022.
- H. Zhang, X. Peng, H. Wei, and X. Liu. Safe and efficient: A primal-dual method for offline convex cmdps under partial data coverage. *Advances in Neural Information Processing Systems*, 37:34239–34269, 2024.
- Y. Zheng, J. Li, D. Yu, Y. Yang, S. E. Li, X. Zhan, and J. Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=j5JvZCaDM0>.

## A Proofs for Theorem 1 and Theorem 2

*Proof of Theorem 1.* For any  $D \in \Delta\Pi$ , we have

$$\begin{aligned}
L(D, \bar{\lambda}) &= \frac{1}{T} \sum_{t=1}^T L(D, \lambda_t) && (L(D, \lambda) \text{ is linear in } \lambda) \\
&\leq \frac{1}{T} \sum_{t=1}^T (L(D_t, \lambda_t) + \epsilon_{\text{offline-RL}}(n)) && (\text{due to Equation (4)}) \\
&\leq \frac{1}{T} \sum_{t=1}^T L(D_t, \bar{\lambda}) + \epsilon_{\text{offline-RL}}(n) + \frac{R_T(\Lambda)}{T} && (\text{due to Equation (5)}) \\
&= L(\bar{D}, \bar{\lambda}) + \epsilon_{\text{offline-RL}}(n) + \frac{R_T(\Lambda)}{T} && (L(D, \lambda) \text{ is linear in } D).
\end{aligned}$$

For any  $\lambda \in \Lambda$ , we have

$$\begin{aligned}
L(\bar{D}, \lambda) &= \frac{1}{T} \sum_{t=1}^T L(D_t, \lambda) \\
&\geq \frac{1}{T} \sum_{t=1}^T L(D_t, \lambda_t) - \frac{R_T(\Lambda)}{T} && (\text{due to Equation (5)}) \\
&\geq \frac{1}{T} \sum_{t=1}^T L(\bar{D}, \lambda_t) - \epsilon_{\text{offline-RL}}(n) - \frac{R_T(\Lambda)}{T} && (\text{due to Equation (4)}) \\
&= L(\bar{D}, \bar{\lambda}) - \epsilon_{\text{offline-RL}}(n) - \frac{R_T(\Lambda)}{T}.
\end{aligned}$$

□

*Proof of Theorem 2.* The result directly follows from the assumption of the stochastic oracle in Definition 4, the regret bound of EXP3 [Auer et al., 2002b], and the approximation error for the projection from  $[0, C]$  into the discrete domain  $\Lambda$  with resolution  $1/K$  is  $1/K$ .

In particular, by [Auer et al., 2002b], we have

$$\sum_{t=1}^T L(\tilde{D}_t, \lambda_t) - \min_{\lambda \in \Lambda} \sum_{t=1}^T L(\tilde{D}_t, \lambda) \leq H\sqrt{2}\sqrt{TK \ln K}.$$

Applying the result in Theorem 1, we have

$$\max_{D \in \Delta\Pi} L(D, \hat{\lambda}) + \epsilon_1 \leq L(\bar{D}, \hat{\lambda}) \leq \min_{\lambda \in \Lambda} L(\bar{D}, \lambda) + \epsilon_1,$$

where

$$\epsilon_1 = \epsilon_{\text{offline-RL}}(n) + H\sqrt{\frac{2K \log K}{T}}, \text{ and } \hat{\lambda} := \frac{1}{T} \sum_{t=1}^T \lambda_t.$$

Note that the running average  $\hat{\lambda}$  does not guarantee to be in  $\Lambda$ . Thus, with  $\bar{\lambda} = \text{Proj}_{\Lambda}(\hat{\lambda})$  – a projection of  $\hat{\lambda}$  onto  $\Lambda$ , we have that,

$$|L(D, \hat{\lambda}) - L(D, \bar{\lambda})| \leq \frac{H}{K}.$$

Overall, we have

$$\max_{D \in \Delta\Pi} L(D, \bar{\lambda}) + \epsilon_1 + \frac{H}{K} \leq L(\bar{D}, \bar{\lambda}) \leq \min_{\lambda \in \Lambda} L(\bar{D}, \lambda) + \epsilon_1 + \frac{H}{K},$$

Thus,  $(\bar{D}, \bar{\lambda})$  is  $\epsilon$ -approximate equilibrium, where

$$\epsilon = \epsilon_{\text{offline-RL}}(n) + H\sqrt{\frac{2K \log K}{T}} + \frac{H}{K}.$$

□

## B Extended Discussion of Related Work

Lagrangian formulation is quite natural for offline safe RL that both our work and [Le et al., 2019] share. [Le et al., 2019] and our work also share the general idea that we can use online optimization algorithms to tune the Lagrangian multiplier. The critical difference is in the design of efficient and effective algorithmic frameworks: our framework requires *one call to an offline RL oracle per loop* and *does not rely on any OPE oracle*. On the other hand, [Le et al., 2019] requires *two calls to an offline RL oracle and four calls to an OPE oracle per loop*: four calls to OPE oracle in each iteration can lead to error propagation and exponential error over the number of iterations posing stability challenges; and six calls to oracles of different nature in each iteration leads to huge computational expense posing scalability challenges. Our approximate O3SRL algorithmic framework overcomes these challenges in a principled manner.

## C Ablation Results for O3SRL

We provide additional ablation results for O3SRL beyond those presented in the main paper.

### C.1 Extended Evaluation of O3SRL

To further evaluate the generalization capability of O3SRL, we extended our experiments to additional environments from the *Gymnasium Safety Suite*, following the FISOR [Zheng et al., 2024] setup with a tight cost limit of 10. The results are summarized in Table 5.

Table 5: O3SRL normalized rewards and costs results across SafetyGym environments.  $\uparrow$  denotes that higher rewards are better;  $\downarrow$  denotes that lower costs (up to threshold 1) are better. **Bold**: Safe agents whose normalized cost  $\leq 1$ . Gray: Unsafe agents. **Blue**: Safe agents with the highest reward.

| Environment      |                   | CAPS                            | FISOR                            | CCAC                             | O3SRL                           |
|------------------|-------------------|---------------------------------|----------------------------------|----------------------------------|---------------------------------|
| PointCircle1     | Reward $\uparrow$ | <b>0.31<math>\pm</math>0.05</b> | 0.43 $\pm$ 0.05                  | 0.57 $\pm$ 0.08                  | 0.53 $\pm$ 0.02                 |
|                  | Cost $\downarrow$ | <b>0.94<math>\pm</math>1.53</b> | 14.93 $\pm$ 4.24                 | 6.65 $\pm$ 3.51                  | 1.31 $\pm$ 0.62                 |
| PointCircle2     | Reward $\uparrow$ | <b>0.44<math>\pm</math>0.06</b> | 0.76 $\pm$ 0.05                  | 0.03 $\pm$ 0.75                  | <b>0.52<math>\pm</math>0.04</b> |
|                  | Cost $\downarrow$ | <b>0.10<math>\pm</math>0.09</b> | 18.02 $\pm$ 4.17                 | 3.99 $\pm$ 4.30                  | <b>0.55<math>\pm</math>0.41</b> |
| AntVelocity      | Reward $\uparrow$ | <b>0.87<math>\pm</math>0.04</b> | <b>0.89<math>\pm</math>0.01</b>  | <b>-1.01<math>\pm</math>0.00</b> | <b>0.45<math>\pm</math>0.43</b> |
|                  | Cost $\downarrow$ | <b>0.19<math>\pm</math>0.06</b> | <b>0.00<math>\pm</math>0.00</b>  | <b>0.00<math>\pm</math>0.00</b>  | <b>0.05<math>\pm</math>0.06</b> |
| HalfCheetah      | Reward $\uparrow$ | <b>0.86<math>\pm</math>0.01</b> | <b>0.89<math>\pm</math>0.01</b>  | <b>0.91<math>\pm</math>0.04</b>  | <b>0.93<math>\pm</math>0.02</b> |
|                  | Cost $\downarrow$ | <b>0.27<math>\pm</math>0.07</b> | <b>0.00<math>\pm</math>0.00</b>  | <b>0.97<math>\pm</math>0.12</b>  | <b>0.00<math>\pm</math>0.00</b> |
| HopperVelocity   | Reward $\uparrow$ | 0.29 $\pm$ 0.14                 | <b>0.12<math>\pm</math>0.03</b>  | <b>-0.01<math>\pm</math>0.02</b> | <b>0.51<math>\pm</math>0.31</b> |
|                  | Cost $\downarrow$ | 1.61 $\pm$ 1.53                 | <b>0.90<math>\pm</math>1.07</b>  | <b>0.00<math>\pm</math>0.00</b>  | <b>0.20<math>\pm</math>0.35</b> |
| SwimmerVelocity  | Reward $\uparrow$ | 0.38 $\pm$ 0.10                 | <b>-0.02<math>\pm</math>0.05</b> | <b>0.06<math>\pm</math>0.25</b>  | <b>0.05<math>\pm</math>0.02</b> |
|                  | Cost $\downarrow$ | 2.11 $\pm$ 3.01                 | <b>0.23<math>\pm</math>0.20</b>  | <b>0.91<math>\pm</math>1.57</b>  | <b>0.00<math>\pm</math>0.01</b> |
| Walker2dVelocity | Reward $\uparrow$ | <b>0.78<math>\pm</math>0.01</b> | 0.51 $\pm$ 0.03                  | 0.40 $\pm$ 0.17                  | <b>0.77<math>\pm</math>0.02</b> |
|                  | Cost $\downarrow$ | <b>0.08<math>\pm</math>0.07</b> | 1.92 $\pm$ 0.57                  | 5.20 $\pm$ 1.16                  | <b>0.02<math>\pm</math>0.02</b> |

O3SRL satisfies the cost constraint in 6 out of the 7 evaluated tasks, demonstrating strong generalization across environments with varying dynamics and cost structures. The next best-performing method, CAPS, remains safe on 5 out of 7 tasks. In terms of reward, O3SRL achieves the highest normalized return on three environments and remains competitive on the remaining tasks. For example, O3SRL attains performance close to the best methods on *SwimmerVelocity* (0.05 vs. 0.06) and *Walker2dVelocity* (0.77 vs. 0.78) while maintaining safety.

Collectively, these results show that O3SRL maintains safety under strict cost constraints while achieving competitive or superior performance across diverse safety-critical environments.

### C.2 Alternative Approach: Offline RL with Fixed $\lambda$ as a Hyperparameter

To assess the effect of treating the Lagrange multiplier  $\lambda$  as a fixed hyperparameter, we conducted an additional experiment in which  $\lambda$  was held constant throughout training, and the resulting policy

was evaluated at the end. This setup departs from the intended offline RL formulation, as choosing an appropriate  $\lambda$  would typically require access to online validation data or environment rollouts. Moreover, an exhaustive search over possible  $\lambda$  values is computationally impractical and inconsistent with the offline setting.

In our proposed approach (O3SRL), we instead define a discrete set of candidate  $\lambda$  values, each corresponding to an “arm” in a multi-armed bandit framework. During training, a  $\lambda$  value is sampled from a learned distribution over these arms and used to shape the reward. The sampling distribution is updated at every batch following the EXP3 algorithm, enabling adaptive reweighting based on the observed performance of each arm. This mechanism allows the policy to dynamically adjust the effective penalty strength, balancing constraint satisfaction and reward maximization throughout training rather than committing to a single fixed  $\lambda$ .

Empirically, as shown in Table 6 fixed- $\lambda$  policies tend to either ignore the cost constraints when  $\lambda$  is too small or over-penalize and sacrifice reward when  $\lambda$  is too large. By contrast, O3SRL adapts its choice of  $\lambda$  during offline training, achieving a more favorable trade-off between reward and cost across different constraint budgets.

Table 6: DroneCircle results with fixed arms under varying cost limits. The  $\uparrow$  symbol denotes that higher rewards are better;  $\downarrow$  denotes that lower costs (up to threshold 1) are better. **Bold**: Safe agents whose normalized cost  $\leq 1$ . **Gray**: Unsafe agents. **Blue**: Safe agents with the highest reward.

| Cost Limit |                   | Arm 0 | Arm 1 | Arm 2 | Arm 3       | Arm 4       | O3SRL       |
|------------|-------------------|-------|-------|-------|-------------|-------------|-------------|
| 5          | Reward $\uparrow$ | 0.90  | 0.86  | 0.55  | <b>0.47</b> | <b>0.39</b> | <b>0.49</b> |
|            | Cost $\downarrow$ | 18.18 | 16.52 | 2.47  | <b>0.00</b> | <b>0.00</b> | <b>0.23</b> |
| 20         | Reward $\uparrow$ | 0.90  | 0.87  | 0.75  | <b>0.49</b> | <b>0.40</b> | <b>0.53</b> |
|            | Cost $\downarrow$ | 4.54  | 4.33  | 2.80  | <b>0.04</b> | <b>0.00</b> | <b>0.36</b> |
| 40         | Reward $\uparrow$ | 0.90  | 0.87  | 0.82  | <b>0.52</b> | <b>0.42</b> | <b>0.65</b> |
|            | Cost $\downarrow$ | 2.27  | 2.21  | 1.88  | <b>0.08</b> | <b>0.00</b> | <b>0.85</b> |

### C.3 Search space of Lagrange variable $\lambda$

Recall that our search space for  $\lambda$  is  $[0, C]$ . Table 7 presents an ablation study on the impact of varying  $C$ , the maximum allowable value for the Lagrange variable  $\lambda$ , which determines how strongly constraint violations are penalized in O3SRL. Results are shown for  $C = 2, 5$ , and  $10$ .

The ablation results highlight that small values of  $C$  (e.g.,  $C = 2$ ) often lead to insufficient constraint enforcement, resulting in catastrophic safety violations in some environments. For example, in *CarCircle* and *DroneCircle*, the average cost exceeds the threshold by a factor of over 8 and 13 respectively, completely defeating the purpose of safe learning. These cases show that when  $\lambda$  is capped too low, the algorithm lacks the leverage to penalize unsafe behavior adequately.

In contrast, increasing  $C$  to 5 or 10 significantly improves safety across all tasks. For instance, in *DroneCircle*, the cost drops from 13.32 at  $C = 2$  to 0.23 at  $C = 5$ , and further to 0.00 at  $C = 10$ . However, in tasks with already low costs (e.g., *AntCircle*), large  $C$  values can lead to reduced reward, due to over-penalization.

Overall,  $C = 5$  emerges as a robust choice, offering strong safety guarantees while maintaining high reward across environments. This ablation underscores the necessity of a sufficiently expressive  $\lambda$  range to enable O3SRL to satisfy constraints in practice.

### C.4 Discretization of $\lambda$

Table 8 presents an ablation comparing two  $\lambda$  set choices used in O3SRL: a uniformly spaced set (e.g.,  $[0, 1.25, 2.5, 3.75, 5]$ ) and an adaptive set that retains the extreme values 0 and 5 but distributes intermediate values more densely near lower penalty regions (e.g.,  $[0, 0.5, 1.12, 2.5, 5]$ ). The purpose of the adaptive grid is to modulate constraint penalization based on the cost threshold  $\kappa$ , allowing the algorithm to apply smaller penalties in more permissive cost budget scenarios. The results show that while both grids perform similarly under strict cost threshold constraints ( $\kappa = 5$ ), the adaptive grid provides clear advantages as the constraint becomes more relaxed. For instance, under  $\kappa = 40$ , the

Table 7: O3SRL results for different values of  $C$ :  $\lambda$  maximum value. Each value shows the average reward ( $\uparrow$ ) and cost ( $\downarrow$ ). Higher reward is better, and lower cost (up to threshold 1) is better. **Bold**: Safe agents whose normalized cost  $\leq 1$ . Gray: Unsafe agents.

| Tasks       |                   | $C = 2$                | $C = 5$                | $C = 10$               |
|-------------|-------------------|------------------------|------------------------|------------------------|
| BallRun     | Reward $\uparrow$ | <b>0.28</b> $\pm 0.01$ | <b>0.25</b> $\pm 0.03$ | <b>0.25</b> $\pm 0.01$ |
|             | Cost $\downarrow$ | <b>0.00</b> $\pm 0.00$ | <b>0.00</b> $\pm 0.00$ | <b>0.00</b> $\pm 0.00$ |
| CarRun      | Reward $\uparrow$ | <b>0.97</b> $\pm 0.00$ | <b>0.96</b> $\pm 0.01$ | <b>0.95</b> $\pm 0.01$ |
|             | Cost $\downarrow$ | <b>0.06</b> $\pm 0.10$ | <b>0.02</b> $\pm 0.03$ | <b>0.00</b> $\pm 0.00$ |
| DroneRun    | Reward $\uparrow$ | 0.43 $\pm 0.10$        | <b>0.32</b> $\pm 0.05$ | <b>0.33</b> $\pm 0.05$ |
|             | Cost $\downarrow$ | 1.93 $\pm 1.90$        | <b>0.68</b> $\pm 1.18$ | <b>0.38</b> $\pm 0.66$ |
| AntRun      | Reward $\uparrow$ | 0.34 $\pm 0.09$        | <b>0.33</b> $\pm 0.13$ | <b>0.20</b> $\pm 0.06$ |
|             | Cost $\downarrow$ | 1.16 $\pm 0.95$        | <b>0.14</b> $\pm 0.10$ | <b>0.27</b> $\pm 0.47$ |
| BallCircle  | Reward $\uparrow$ | 0.82 $\pm 0.06$        | <b>0.62</b> $\pm 0.01$ | <b>0.58</b> $\pm 0.03$ |
|             | Cost $\downarrow$ | 7.20 $\pm 2.99$        | <b>0.06</b> $\pm 0.07$ | <b>0.01</b> $\pm 0.01$ |
| CarCircle   | Reward $\uparrow$ | 0.75 $\pm 0.11$        | <b>0.66</b> $\pm 0.03$ | <b>0.57</b> $\pm 0.08$ |
|             | Cost $\downarrow$ | 8.07 $\pm 7.53$        | <b>0.11</b> $\pm 0.16$ | <b>0.00</b> $\pm 0.00$ |
| DroneCircle | Reward $\uparrow$ | 0.79 $\pm 0.06$        | <b>0.49</b> $\pm 0.07$ | <b>0.37</b> $\pm 0.03$ |
|             | Cost $\downarrow$ | 13.32 $\pm 2.62$       | <b>0.23</b> $\pm 0.34$ | <b>0.00</b> $\pm 0.00$ |
| AntCircle   | Reward $\uparrow$ | <b>0.58</b> $\pm 0.02$ | <b>0.48</b> $\pm 0.06$ | <b>0.41</b> $\pm 0.02$ |
|             | Cost $\downarrow$ | <b>0.25</b> $\pm 0.26$ | <b>0.00</b> $\pm 0.00$ | <b>0.04</b> $\pm 0.07$ |

adaptive grid allows significantly higher reward in tasks such as *DroneCircle* (reward improves from 0.45 to 0.65), by leveraging a smoother trade-off between reward maximization and cost constraint satisfaction. These findings suggest that finer resolution at low penalty levels enables better policy selection for higher cost constraint thresholds, making the adaptive grid more effective for balancing performance and safety.

Table 8: O3SRL results for different values of the set of  $\lambda$ s and cost limits  $\kappa$ . Each value shows the average reward ( $\uparrow$ ) and cost ( $\downarrow$ ). Higher reward is better, and lower cost (up to threshold 1) is better. **Bold**: Safe agents whose normalized cost  $\leq 1$ . Gray: Unsafe agents.

| Tasks       | $\kappa = 5$      |                        | $\kappa = 20$          |                        | $\kappa = 40$          |                        |
|-------------|-------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
|             | uniform           | adaptive               | uniform                | adaptive               | uniform                | adaptive               |
| BallRun     | Reward $\uparrow$ | <b>0.25</b> $\pm 0.01$ | <b>0.25</b> $\pm 0.03$ | <b>0.25</b> $\pm 0.01$ | <b>0.26</b> $\pm 0.01$ | <b>0.25</b> $\pm 0.01$ |
|             | Cost $\downarrow$ | <b>0.00</b> $\pm 0.00$ | <b>0.00</b> $\pm 0.00$ | <b>0.00</b> $\pm 0.00$ | <b>0.06</b> $\pm 0.11$ | <b>0.00</b> $\pm 0.00$ |
| DroneRun    | Reward $\uparrow$ | 0.39 $\pm 0.04$        | <b>0.32</b> $\pm 0.05$ | <b>0.31</b> $\pm 0.11$ | <b>0.38</b> $\pm 0.05$ | <b>0.30</b> $\pm 0.05$ |
|             | Cost $\downarrow$ | 1.95 $\pm 1.97$        | <b>0.68</b> $\pm 1.18$ | <b>0.22</b> $\pm 0.33$ | <b>0.41</b> $\pm 0.41$ | <b>0.60</b> $\pm 0.72$ |
| BallCircle  | Reward $\uparrow$ | <b>0.63</b> $\pm 0.04$ | <b>0.62</b> $\pm 0.01$ | <b>0.63</b> $\pm 0.08$ | <b>0.69</b> $\pm 0.09$ | <b>0.62</b> $\pm 0.06$ |
|             | Cost $\downarrow$ | <b>0.07</b> $\pm 0.12$ | <b>0.06</b> $\pm 0.07$ | <b>0.20</b> $\pm 0.30$ | <b>0.58</b> $\pm 0.66$ | <b>0.10</b> $\pm 0.10$ |
| DroneCircle | Reward $\uparrow$ | <b>0.46</b> $\pm 0.02$ | <b>0.49</b> $\pm 0.07$ | <b>0.46</b> $\pm 0.01$ | <b>0.53</b> $\pm 0.01$ | <b>0.45</b> $\pm 0.03$ |
|             | Cost $\downarrow$ | <b>0.00</b> $\pm 0.00$ | <b>0.23</b> $\pm 0.34$ | <b>0.00</b> $\pm 0.00$ | <b>0.36</b> $\pm 0.27$ | <b>0.00</b> $\pm 0.00$ |

### C.5 Number of update steps $M$

Recall that our approximate algorithm for empirical evaluation performs  $M$  stochastic gradient updates of a given offline RL algorithm in each iteration. Table 9 presents an ablation on the number of offline RL update steps  $M$ . This parameter controls how long the policy is trained before updating the  $\lambda$  value to balance reward and constraint satisfaction. The results demonstrate that  $M = 10$  provides a stable trade-off between learning progress and responsiveness to constraint violations.

With  $M = 1$ , the frequent updates to  $\lambda$  overall increases costs adjusting the policy aggressively. Conversely, setting  $M = 100$  delays  $\lambda$  updates, which can result in policies drifting away from feasible regions, as seen in the sharp increase in costs for DroneRun and DroneCircle. Overall, the ablation results validate that safe and effective offline learning in O3SRL requires coordinated progress on both the policy and the constraint via well-timed  $\lambda$  updates. The choice of  $M = 10$  strikes this balance, supporting its use for all the results presented in the main paper.

Table 9: O3SRL results for different values of  $M$ : Offline RL update steps. Each value shows the average reward ( $\uparrow$ ) and cost ( $\downarrow$ ). Higher reward is better, and lower cost (up to threshold 1) is better. **Bold**: Safe agents whose normalized cost  $\leq 1$ . **Gray**: Unsafe agents.

| Tasks       |                   | $M = 1$                         | $M = 10$                        | $M = 100$                       |
|-------------|-------------------|---------------------------------|---------------------------------|---------------------------------|
| BallRun     | Reward $\uparrow$ | <b>0.25<math>\pm</math>0.03</b> | <b>0.25<math>\pm</math>0.03</b> | <b>0.26<math>\pm</math>0.03</b> |
|             | Cost $\downarrow$ | <b>0.37<math>\pm</math>0.64</b> | <b>0.00<math>\pm</math>0.00</b> | <b>0.00<math>\pm</math>0.00</b> |
| CarRun      | Reward $\uparrow$ | <b>0.96<math>\pm</math>0.00</b> | <b>0.96<math>\pm</math>0.01</b> | <b>0.96<math>\pm</math>0.00</b> |
|             | Cost $\downarrow$ | <b>0.00<math>\pm</math>0.01</b> | <b>0.02<math>\pm</math>0.03</b> | <b>0.01<math>\pm</math>0.01</b> |
| DroneRun    | Reward $\uparrow$ | 0.34 $\pm$ 0.06                 | <b>0.32<math>\pm</math>0.05</b> | 0.27 $\pm$ 0.08                 |
|             | Cost $\downarrow$ | 1.10 $\pm$ 0.90                 | <b>0.68<math>\pm</math>1.18</b> | 3.72 $\pm$ 4.44                 |
| AntRun      | Reward $\uparrow$ | <b>0.34<math>\pm</math>0.05</b> | <b>0.33<math>\pm</math>0.13</b> | <b>0.39<math>\pm</math>0.16</b> |
|             | Cost $\downarrow$ | <b>0.20<math>\pm</math>0.21</b> | <b>0.14<math>\pm</math>0.10</b> | <b>0.97<math>\pm</math>1.36</b> |
| BallCircle  | Reward $\uparrow$ | <b>0.67<math>\pm</math>0.00</b> | <b>0.62<math>\pm</math>0.01</b> | <b>0.67<math>\pm</math>0.05</b> |
|             | Cost $\downarrow$ | <b>0.25<math>\pm</math>0.19</b> | <b>0.06<math>\pm</math>0.07</b> | <b>0.61<math>\pm</math>0.88</b> |
| CarCircle   | Reward $\uparrow$ | <b>0.66<math>\pm</math>0.01</b> | <b>0.66<math>\pm</math>0.03</b> | <b>0.68<math>\pm</math>0.01</b> |
|             | Cost $\downarrow$ | <b>0.10<math>\pm</math>0.17</b> | <b>0.11<math>\pm</math>0.16</b> | <b>0.17<math>\pm</math>0.29</b> |
| DroneCircle | Reward $\uparrow$ | <b>0.44<math>\pm</math>0.06</b> | <b>0.49<math>\pm</math>0.07</b> | 0.58 $\pm$ 0.20                 |
|             | Cost $\downarrow$ | <b>0.11<math>\pm</math>0.10</b> | <b>0.23<math>\pm</math>0.34</b> | 4.87 $\pm$ 7.77                 |
| AntCircle   | Reward $\uparrow$ | <b>0.41<math>\pm</math>0.06</b> | <b>0.48<math>\pm</math>0.06</b> | <b>0.49<math>\pm</math>0.01</b> |
|             | Cost $\downarrow$ | <b>0.02<math>\pm</math>0.03</b> | <b>0.00<math>\pm</math>0.00</b> | <b>0.00<math>\pm</math>0.00</b> |

## C.6 Evaluation Across Multiple Random Seeds

To assess the stability of O3SRL with respect to random initialization, we extend our evaluation to include additional random seeds. Specifically, we compare results averaged over the original three seeds (10, 20, and 30) with those averaged over all six seeds (10, 20, 30, 40, 50, and 60). Table 10 reports the mean and standard deviation of rewards and costs under both settings.

The results across confirm the stability of O3SRL, showing consistent rewards and safe cost levels across all evaluated environments. The performance trends observed with three seeds largely persist when extended to six, indicating robust and reliable behavior under different random seeds.

## D Experimental Details

This section provides additional details about the experimental setup and hyper-parameters.

### D.1 Description of environments

We evaluate O3SRL performance in a set of environments from the **Bullet-Safety-Gym** suite. Each environment is defined by a combination of *robot type* and *task*, resulting in scenarios such as *BallRun*, *CarCircle*, *DroneRun*, and so on. The agent types include **Ball**, **Car**, **Drone**, and **Ant**, and each is evaluated under either the **Run** or **Circle** task. Figure 1 illustrates the Bullet-Safety-Gym environments, demonstrating various agent types (Ball, Car, Drone, Ant) performing the Run and Circle tasks.

Table 10: Comparison between 3-seed and 6-seed experiments. Reward ( $\uparrow$ ) indicates higher is better, and Cost ( $\downarrow$ ) indicates lower is better. **Bold**: Safe agents whose normalized cost  $\leq 1$ .

| Environment |                   | 3 seeds                         | 6 seeds                         |
|-------------|-------------------|---------------------------------|---------------------------------|
| BallRun     | Reward $\uparrow$ | <b>0.25<math>\pm</math>0.03</b> | <b>0.26<math>\pm</math>0.02</b> |
|             | Cost $\downarrow$ | <b>0.00<math>\pm</math>0.00</b> | <b>0.00<math>\pm</math>0.00</b> |
| CarRun      | Reward $\uparrow$ | <b>0.96<math>\pm</math>0.01</b> | <b>0.96<math>\pm</math>0.01</b> |
|             | Cost $\downarrow$ | <b>0.02<math>\pm</math>0.03</b> | <b>0.00<math>\pm</math>0.00</b> |
| DroneRun    | Reward $\uparrow$ | <b>0.32<math>\pm</math>0.05</b> | <b>0.37<math>\pm</math>0.03</b> |
|             | Cost $\downarrow$ | <b>0.68<math>\pm</math>1.18</b> | <b>0.74<math>\pm</math>0.67</b> |
| AntRun      | Reward $\uparrow$ | <b>0.33<math>\pm</math>0.13</b> | <b>0.22<math>\pm</math>0.10</b> |
|             | Cost $\downarrow$ | <b>0.14<math>\pm</math>0.10</b> | <b>0.17<math>\pm</math>0.19</b> |
| BallCircle  | Reward $\uparrow$ | <b>0.62<math>\pm</math>0.01</b> | <b>0.63<math>\pm</math>0.04</b> |
|             | Cost $\downarrow$ | <b>0.06<math>\pm</math>0.07</b> | <b>0.26<math>\pm</math>0.25</b> |
| CarCircle   | Reward $\uparrow$ | <b>0.66<math>\pm</math>0.03</b> | <b>0.66<math>\pm</math>0.03</b> |
|             | Cost $\downarrow$ | <b>0.11<math>\pm</math>0.16</b> | <b>0.04<math>\pm</math>0.10</b> |
| DroneCircle | Reward $\uparrow$ | <b>0.49<math>\pm</math>0.07</b> | <b>0.47<math>\pm</math>0.03</b> |
|             | Cost $\downarrow$ | <b>0.23<math>\pm</math>0.34</b> | <b>0.02<math>\pm</math>0.04</b> |
| AntCircle   | Reward $\uparrow$ | <b>0.48<math>\pm</math>0.06</b> | <b>0.44<math>\pm</math>0.05</b> |
|             | Cost $\downarrow$ | <b>0.00<math>\pm</math>0.00</b> | <b>0.02<math>\pm</math>0.03</b> |

**Run Environments:** In the Run task, agents are required to move quickly along a straight corridor. They receive rewards based on forward progress, encouraging high-speed locomotion. However, they are penalized for crossing lateral safety boundaries or exceeding a predefined velocity threshold. These safety constraints are not enforced through physical barriers but through a cost signal that tracks violations.

**Circle Environments:** In the Circle task, agents are trained to move in a clockwise direction around a circular track. Rewards are higher when agents maintain fast, smooth motion near the boundary of the circle. Deviating from the intended path or exiting the safety zone results in penalties.

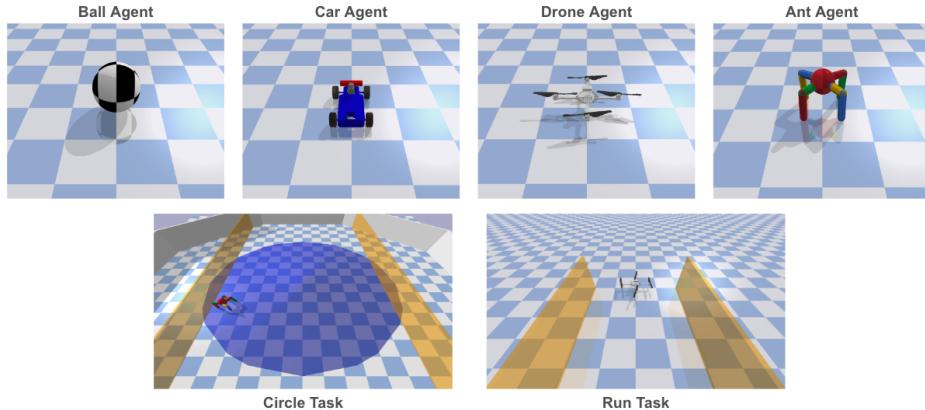


Figure 1: Visualization of the Bullet-Safety-Gym environments featuring different agents and tasks.

## D.2 Training details

**Adaptive  $\lambda$  set:** We construct the  $\lambda$  set using a geometric spacing scheme that adapts to the cost limit. The grid always includes  $\lambda = 0$  and a maximum value  $\lambda_{\max} = C$ , and places  $k - 2$  intermediate points between them. These intermediate values are spaced geometrically and scaled by a factor  $\text{shrink} = (\text{reference\_limit}/\text{cost\_limit})^{\alpha_{\text{shrink}}}$ , which compresses the grid as the cost limit increases, where  $\text{reference\_limit} = 5$  and  $\alpha_{\text{shrink}} = 0.3$ . This design ensures finer resolution among low-penalty

values when constraints are loose, and broader coverage when strict cost constraints require stronger penalization.

**Rewards scale:** We observe that reward distributions in offline datasets can be heavy-tailed, with occasional extreme values that disproportionately affect learning stability, as illustrated in Figure 2.

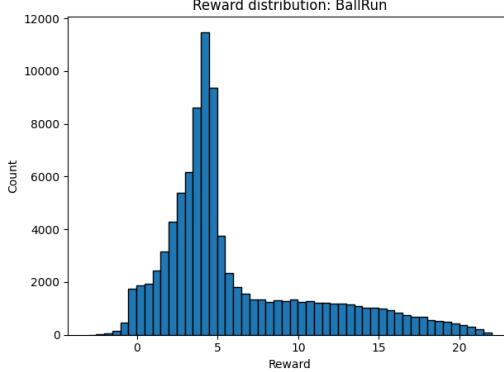


Figure 2: An example of a heavy-tailed reward distribution from the *BallRun* dataset.

To mitigate this, we clip rewards symmetrically at the  $\tau$ -th percentile of their absolute values:

$$r_{\text{clip}} = \text{percentile}(|r|, \tau), \quad r \leftarrow \text{clip}(r, -r_{\text{clip}}, r_{\text{clip}})$$

where  $\tau = 99$ . We then scale down the clipped rewards multiplying by  $0.9/r_{\text{clip}}$ .

**Hyperparameters:** We employ a batch size of 1024 for the tasks *CarCircle*, *DroneRun*, *AntCircle*, and *BallCircle*, while a batch size of 512 is used for the remaining tasks: *CarRun*, *DroneCircle*, *AntRun*, and *BallRun*. The other hyperparameters used in our experiments are listed in Table 11. For TD3+BC, we adopt the same hyperparameter settings as in the original paper.

Table 11: O3SRL Hyperparameters

| <b>EXP3 parameters</b>         |              |
|--------------------------------|--------------|
| Number of arms $k$             | 5            |
| $C = \max(\lambda)$            | 5            |
| $\lambda$ update frequency $M$ | 10           |
| $\lambda$ learning rate $\eta$ | 2e-3         |
| <b>TD3BC parameters</b>        |              |
| Discount $\gamma$              | 0.99         |
| Policy noise                   | 0.2          |
| Policy noise clip              | (0.5, 0.5)   |
| Policy update frequency        | 2            |
| $\alpha$                       | 2.5          |
| Optimizer                      | Adam         |
| Actor, Critic learning rate    | 3e-4         |
| Actor, Critic hidden size      | 256          |
| Training steps                 | 100000       |
| Seed                           | [10, 20, 30] |

**Training Curves of O3SRL:** Figure 3 shows the training curves of O3SRL across the different tasks, plotting both the average reward and constraint cost over training iterations. The curves demonstrate that O3SRL is able to quickly stabilize and achieve a favorable trade-off between reward maximization and cost minimization. In most tasks, the cost decreases steadily and remains below the constraint threshold, while the reward improves progressively.

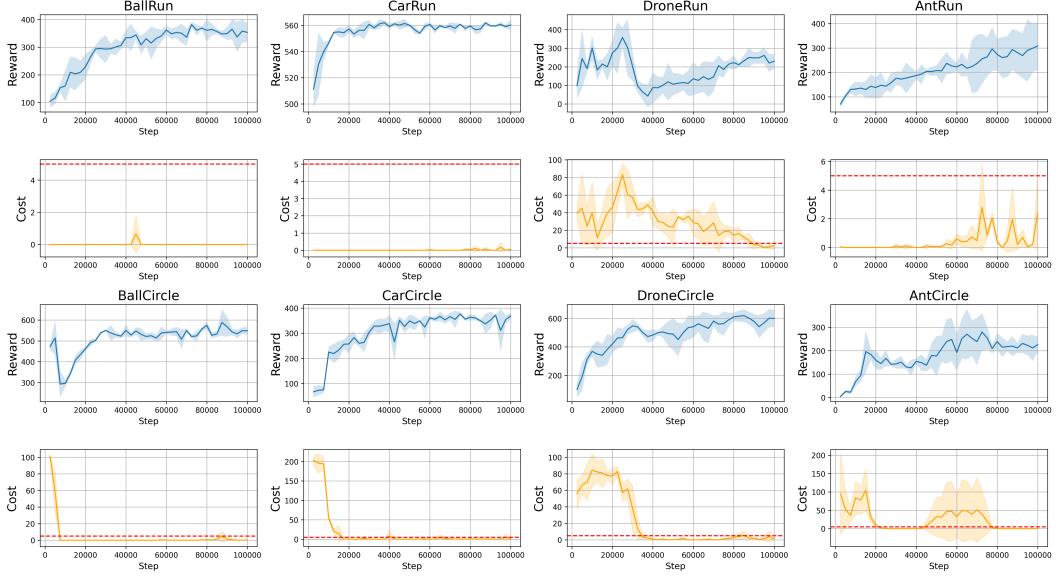


Figure 3: Training curves of O3SRL across different tasks, showing average reward (top) and cost (bottom) over training iterations.

**Computational Resources and Training Time:** All experiments were conducted using an NVIDIA A40 GPU with 48GB of memory. Training a single task for one random seed takes approximately 20 minutes when the server is not under load.