Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

Beyond Hard Constraints: Budget-Conditioned Reachability for Safe Offline Reinforcement Learning

Janaka Brahmanage and Akshat Kumar

School of Computing and Information Systems, Singapore Management University.
janakat.2022@phdcs.smu.edu.sg, akshatkumar@smu.edu.sg

Abstract

Sequential decision-making using Markov Decision Process underpins many real-world applications. Both model-based and model-free methods have achieved strong results in these settings. However, real-world tasks must balance reward maximization with safety constraints, often conflicting objectives, that can lead to unstable min–max, adversarial optimization. A promising alternative is *safety reachability* analysis, which precomputes a forward-invariant safe state–action set, ensuring that an agent starting inside this set remains safe indefinitely. Yet, most reachability-based methods address only hard safety constraints, and little work extends reachability to cumulative cost constraints. To address this, *first*, we define a safety-conditioned reachability set that *decouples* reward maximization from cumulative safety cost constraints. *Second*, we show how this set enforces safety constraints without unstable min–max or Lagrangian optimization, yielding a novel offline safe RL algorithm that learns a safe policy from a fixed dataset without environment interaction. *Finally*, experiments on standard offline safe-RL benchmarks, and a real-world maritime navigation task demonstrate that our method matches or outperforms state-of-the-art baselines while maintaining safety.

1 Introduction

Markov Decision Processes (MDPs) have demonstrated impressive success modeling across a range of domains, including robotics (Tang et al. 2024), algorithm discovery (Fawzi et al. 2022), classical board games (Silver et al. 2016), and Atari games (Mnih et al. 2013). Despite these achievements, deploying agents in real-world settings poses significant challenges. In practice, most applications utilize a Reinforcement Learning (RL) framework to train agents, and these agents must not only maximize cumulative reward but also operate safely (Altman 2021), requiring policies that strike a balance between performance and safety. Additionally, constructing fast and accurate simulators for complex environments is often computationally prohibitive. Offline safe reinforcement learning (*offline safe RL*) offers a practical solution by enabling agents to learn from pre-collected datasets without further environment interaction (Liu et al. 2024). This work aims to develop methods that address the dual goals of reward optimization and safety in offline RL, an important problem in RL research (Gu et al. 2024).

Approach overview Our goal is to solve the standard CMDP problem (Altman 2021), where the agent must maximize reward subject to a constraint on the expected cumulative cost. We propose an approach that augments the state with a dynamic budget. This dynamic budget is a quantity that controls the conservativeness of the policy; a smaller budget implies a more conservative policy. For example, we can start with the total cost threshold of the CMDP and track the exact remaining budget by subtracting immediate costs incurred as the policy executes. In a deterministic setting, we can enforce the safety constraint by tracking such a budget. However, this does not work in a stochastic environment. Instead, we show that tracking a quantity other than the exact remaining budget can enforce safety in stochastic settings. We then estimate a budget-conditioned *persistent* safety set: the set of state-action pairs from which there exists a policy that keeps future costs within the remaining budget. By restricting the agent's actions to this safety set, we decouple safety enforcement from reward optimization, avoiding the instability of Lagrangian or min-max methods.

Related work Constrained Markov decision processes (CMDPs) (Altman 2021) are a standard framework for safe RL. Prior offline safe RL methods often struggle with optimization instability or high computational overhead. Lagrangian methods like BCQ-Lagrangian (Liu et al. 2024) suffer from tuning difficulties and unstable learning. Methods like COptiDICE (Lee et al. 2022) perform distribution correction but struggle empirically on recent benchmarks (Liu et al. 2024). Recent approaches learn generative models (VAEs) as a pre-training step (Koirala et al. 2025; Guo et al. 2025), which incurs significant computational overhead. While Hamilton-Jacobi reachability (Yu et al. 2022) enforces safety independently of the policy for hard constraints, action-constrained RL ensures per-step feasibility through optimization solvers (Lin et al. 2021) or generative models (Brahmanage, Ling, and Kumar 2023, 2024). Another related approach, Saute RL (Sootla et al. 2022), enforces per-step constraints but requires online rollouts to track budgets. In contrast, our approach prunes unsafe actions offline using a dynamic budget without requiring a generative model or online samples. We provide an extended

discussion of related work in Appendix C.

Adaptive Safty Budgets for RL Standard safe RL methods enforce static, episode-level constraints (i.e., *expected* cost over all the episodes is less than a budget). In contrast, we introduce a step-wise budget that dynamically adjusts during policy execution. A key benefit is that using this step-wise budget, we can prune unsafe actions at each time step and guide value estimation (for both reward and cost), enabling policy that adapts over time.

Our contributions are:

- We propose **Budget-Conditioned Reachability**, a framework that applies reachability analysis to CMDPs with cumulative cost constraints. It uses dynamic budgets to estimate persistently safe state–action sets, enabling reward-maximizing policy learning within this safe region. We also provide rigorous theoretical justification for the framework.
- We introduce two variants of our method to address both deterministic and stochastic CMDP settings, ensuring broad applicability across different problem structures.
- Our method integrates seamlessly with existing offline RL algorithms such as IQL (Wang et al. 2023), XQL (Garg et al. 2023), and SparseQL (Xu et al. 2023), yielding a safe offline RL approach, **BCRL** (Budget-Conditioned Reachability RL). The resulting algorithms are easy to implement, require no min–max adversarial training, never query out-of-distribution actions, and generalize to any budget.
- We evaluate BCRL on interpretable grid-world environments, DSRL benchmarks (Liu et al. 2024), and a real-world maritime navigation task, where agents are trained using historical ship trajectory data to navigate safely in high-traffic conditions. Extensive ablations show consistent improvements over state-of-the-art baselines. Our code is available at[1].

2 Preliminaries

2.1 Constrained Markov Decision Process

A constrained Markov Decision Process (CMDP) is defined as a tuple $\mathcal{M} := \langle S, A, T, r, c, \gamma, \mu_0, \delta_{\text{init}} \rangle$. Here, S and A represent the state and action spaces, respectively. The transition dynamics are given by $T(s' \mid s, a)$, specifying the probability of transitioning from state s to s' under action a. The reward function is $r(s,a) : S \times A \to \mathbb{R}$, and the cost function is $c(s,a) : S \times A \to \mathbb{R}^+$. The discount factor is $\gamma \in [0,1)$, and the initial state distribution is $\mu_0 : S \to [0,1]$. The δ_{init} is the cost threshold over the total expected cost agent can incur under any policy. **The objective** is to determine a policy $\pi : S \to \Delta(A)$ that maximizes the expected discounted reward $J_R(\pi)$, subject to the cost constraint $J_C(\pi) \leq \delta_{\text{init}}$ where,

$$J_R(\pi) = \mathop{\mathbb{E}}_{\pi} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t); \quad J_C(\pi) = \mathop{\mathbb{E}}_{\pi} \sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \quad (1)$$

[1]https://janakact.github.io/bcrl

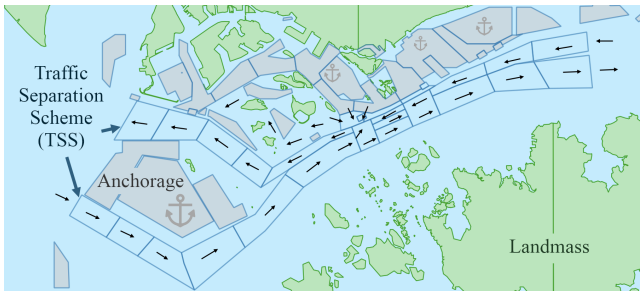

Figure 1: Electronic navigation chart for maritime traffic navigation in Singapore strait

and $\delta_{\text{init}} \in [0, \delta_{\max}]$ is the cost (or safety budget) threshold. Here $\delta_{\max} = \frac{c_{\max}}{1-\gamma}$, where $c_{\max}$ is the highest cost the environment can incur at a single step.

Learning from offline data For CMDPs, getting faithful domain model (e.g., transition, reward, cost functions) for realistic problems is challenging. Thus, we focus on the offline RL setting where we learn from a pre-collected transition dataset $\mathcal{D} = \{(s_i, a_i, r(s_i, a_i), c(s_i, a_i), s'_i)\}_{i=1}^N$ generated by one or more behavioral policies π_β. Offline safe RL aims to learn a policy π that optimizes CMDP objective subject to cost constraints using this dataset alone, without further environment interaction. As a real world example, consider the safe maritime navigation problem in figure 1. The Singapore strait is a high traffic, constrained waterway, with vessels navigating narrow lanes, TSS corridors, anchorage zones, and areas near landmasses. Ships must make sequential decisions–adjusting speed or heading–while avoiding restricted zones, minimizing collision risk, and following navigation rules. Because unsafe exploration is impossible in real waters, learning through trial-and-error is not feasible. Yet, the sea traffic generates extensive AIS (automatic identification system) data on vessel positions, speeds, headings, and interactions. This makes safe offline RL a natural fit for such problems where safe policies are learned solely from historical data without real-world experimentation.

2.2 In-Sample Q-learning Algorithms

For unconstrained reinforcement learning (i.e., standard RL without any cost constraints), several algorithms use asymmetric loss functions to estimate the value function instead of the exact maximum, learning policies from offline data without querying Q-values of unseen actions. Methods such as IQL (Kostrikov, Nair, and Levine 2022), XQL (Garg et al. 2023), and SparseQL (Xu et al. 2023) employ asymmetric regression to fit the Bellman optimal value, replacing the max operator while still following temporal-difference learning. The critic $Q_\theta(s,a)$, target critic $Q_{\theta_T}(s,a)$, and value function $V_\psi(s)$, parameterized by θ, θ_T, and ψ, are trained by minimizing the following losses:

$$\mathcal{L}^V(\psi) = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}} [\mathcal{L}^*(Q_{\theta_T}(s,a) - V_\psi(s))] \quad (2)$$

$$\mathcal{L}^Q(\theta) = \mathop{\mathbb{E}}_{(s,a,s')\sim\mathcal{D}} [(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a))^2] \quad (3)$$

Here, L^* is an asymmetric loss function that penalizes positive and negative errors differently, weighting overestimation more heavily to approximate the maximum Q-value. In IQL (Kostrikov, Nair, and Levine 2022), $L^*(u) = L^\tau(u) = |\tau - \mathbf{1}_{\{u<0\}}|u^2$ is the expectile loss (Newey and Powell 1987), with $\tau > 0.5$ pushing $V_\psi(s)$ toward the maximum. Similarly, XQL (Garg et al. 2023) uses Gumbel regression, and SparseQL (Xu et al. 2023) introduces a robust asymmetric loss. Target parameters are updated via $\theta_T \leftarrow (1-\alpha)\theta_T + \alpha\theta$. These methods are computationally efficient, avoid querying OOD actions, and have strong theoretical guarantees (Kostrikov, Nair, and Levine 2022; Xu et al. 2023). For policy extraction, they use in-sample methods similar to Advantage Weighted Regression (AWR) (Peng et al. 2019). In this work, we adopt primarily in-sample methods to avoid OOD actions.

2.3 Persistent Safety with Value Functions

In safe RL, we sometimes have access to a safety indicator for a given state, which can be viewed as a state-wise safety constraint. However, it is not sufficient to ensure only *instantaneous safety*. When an agent takes an action, we also need to guarantee that in the future the agent will always have at least one safe action available. In other words, the agent should not be driven into an unrecoverable situation or a dead end. Formally, a state is unrecoverable if there exists no sequence of future actions that can prevent an eventual safety violation. A safety signal that indicates the agent is not in an unrecoverable state is referred to as a *persistent safety signal* (Yu et al. 2022). Hamilton–Jacobi (HJ) reachability-based RL methods (Zheng et al. 2024; Yu et al. 2022; Ganai et al. 2023) address this by estimating a *forward-invariant safety set*. This set consists of states from which the agent can remain safe indefinitely, provided it follows an appropriate policy. For example, in FISOR (Zheng et al. 2024), the immediate safety indicator $h(s)$ defines the safe state set $S_f := \{s \in S \mid h(s) \leq 0\}$. A learned value function is then used to estimate

$$V_h^*(s) = \min_\pi V_h^\pi(s) = \min_\pi \max_{t\in\mathbb{N}} h(s_t) \tag{4}$$

If $V_h^*(s) \leq 0$, this implies that there exists a policy π such that the future cost satisfies $V_h^\pi(s) \leq 0$. Therefore, V_h^* acts as a persistent safety indicator. The set $S_p := \{s \in S \mid V_h^*(s) \leq 0\}$ is then used to define the persistent safety set.

3 Budget-Conditioned Reachability

Prior reachability and persistent safety estimation methods are mostly limited to hard constraints (i.e., whether safety indicator $h(s) \leq 0$). They do not extend trivially to cumulative cost based safety constraints in CMDPs defined in Section 2.1. Thus, we aim to estimate the persistent safety set *independently* of the policy being learned, using it to enforce cumulative cost constraints. Decoupling safety estimation from policy optimization ensures the learned policy remains safe while making learning more stable and tractable, without using Lagrangian or min-max adversarial optimization. Our approach builds on three key ideas: **(1)** augmenting the CMDP to explicitly track the remaining budget, **(2)** estimating a budget-conditioned persistent safety set for *any* remaining budget, and **(3)** solving the augmented CMDP.

Methodological Intuition: Rather than enforcing a static constraint over an entire episode, we augment the state with a remaining budget that dynamically depletes as costs are incurred. By pre-computing whether this budget is sufficient for safe completion (the persistent safety set), we can actively prune unsafe actions at each step before the agent acts, entirely decoupling safety enforcement from reward maximization.

3.1 A Budget-Conditioned Persistent Safety Set

Similar to the definition of persistent safety discussed in Section 2.3, we now define the budget-conditioned safety set that indicate feasibility based on the discounted future cost.

Definition 3.1 (The optimal cost-value function)**.** *The optimal state-cost-value function V_C^* and the optimal action-cost-value function Q_C^* are defined as:*

$$V_C^*(s) = \min_\pi V_C^\pi(s) = \min_\pi \mathop{\mathbb{E}}_{\substack{\pi \\ s_0=s}} \sum_{t\in\mathbb{N}} \gamma^t c(s_t, a_t), \tag{5}$$

$$Q_C^*(s,a) = \min_\pi Q_C^\pi(s,a) = \min_\pi \mathop{\mathbb{E}}_{\substack{\pi \\ s_0=s \\ a_0=a}} \sum_{t\in\mathbb{N}} \gamma^t c(s_t, a_t) \tag{6}$$

Following a similar reasoning as in Section 2.3, we claim that, given a budget δ, if $V_C^*(s) \leq \delta$, then there exists a policy π such that the future discounted cost satisfies $V_C^\pi(s) \leq \delta$. Therefore, $V_C^*(s) \leq \delta$ serves as a persistent safety indicator for the given budget, guaranteeing that there exists a policy under which the future cost remains within the budget. This allows us to define the largest set of feasible states, as well as the feasible actions for each state, that remain within the given budget.

Definition 3.2 (Budget-Conditioned Persistent Safety Sets)**.** *Given a budget δ, the* budget-conditioned persistent safety set *is defined as:*

$$S_P(\delta) := \{s \in S \mid V_C^*(s) \leq \delta\}. \tag{7}$$

Similarly, the budget-conditioned persistent safe action set *for a given state s is defined as:*

$$A_P(s,\delta) := \{a \in A \mid Q_C^*(s,a) \leq \delta\}. \tag{8}$$

Intuitively, the budget-conditioned persistent safety set consists of states where a policy exists to keep the total discounted future cost within the budget δ.

Lemma 3.3 (Safe Actions Always Exist for Persistent Safety States)**.** *Given $\delta \in \mathbb{R}^+$; for any state $s \in S_P(\delta)$, the budget-conditioned persistent safe action set $A_P(s,\delta)$ is non-empty: $A_P(s,\delta) \neq \emptyset$.*

Proof. Let $s \in S_P(\delta)$, which means by definition $V_C^*(s) \leq \delta$. By the definition of the value function,

$$V_C^*(s) = \min_\pi \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t c(s_t,a_t) \,\Big|\, s_0 = s, \pi\Big] = \min_{a\in A} Q_C^*(s,a),$$

so there exists at least one action $a^* \in A$ that attains this minimum. Therefore, $Q_C^*(s,a^*) = V_C^*(s) \leq \delta$, which implies $a^* \in A_P(s,\delta)$. Hence, $A_P(s,\delta) \neq \emptyset$. □

In the following sections, we discuss how the budget-conditioned persistent safety sets defined above can be used to enforce the cumulative cost constraint of the CMDP.

3.2 Budget Adaptive MDPs

Since the definition of the budget-conditioned persistent safety set depends on the remaining budget, it is necessary to make the budget an explicit part of the system's dynamics. To achieve this, we extend the CMDP by augmenting its state space with a dynamic budget variable. This augmentation allows the agent to reason about safety not only as a function of the environment state but also as a function of the available safety budget over time. One key contribution is our method for initializing and updating the budget via functions f and g. Specially for stochastic MDPs, these updates are non-trivial and crucial for ensuring the theoretical guarantees of our approach.

Definition 3.4 (Budget-Adaptive MDP)**.** *A* Budget-Adaptive Markov Decision Process *(BAMDP) constructed from a CMDP $\mathcal{M}$ is a tuple*

$$\bar{\mathcal{M}}(\mathcal{M}, f, g) := \langle \bar{S}, A, \bar{T}, \bar{r}, \bar{c}, \gamma, \bar{\mu}_0, \delta_{init} \rangle.$$

where the augmented state space is $\bar{S} := S \times \mathbb{R}^+$, and the action space A remain unchanged. The reward and cost functions are unchanged from the original CMDP. That is, $\bar{r}((s,\delta),a) := r(s,a),\ \bar{c}((s,\delta),a) := c(s,a)$.

The initial budget function $f : S \times \mathbb{R}^+ \to \mathbb{R}^+$ *assigns the initial budget based on the CMDP budget δ_{init} and the starting state s_0, such that the augmented initial state distribution is:*

$$\bar{\mu}_0 := \{(s_0, \delta_0) \mid s_0 \sim \mu_0,\ \delta_0 = f(s_0, \delta_{init})\}.$$

The budget update function $g : S \times A \times S \times \mathbb{R}^+ \to \mathbb{R}^+$ *updates the budget after each transition. In the most general form considered here, it depends on the current state s, action a, next state s', and the current budget δ, such that $\delta' = g(s,a,s',\delta)$. This results in the following augmented transition function:*

$$\bar{T}\big((s',\delta') \mid (s,\delta), a\big) := T(s' \mid s, a) \cdot \mathbf{1}_{\{\delta' = g(s,a,s',\delta)\}},$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function.

Intuition of f and g The role of f and g is to initialize and track a remaining budget (or a similar quantity) while executing the policy, and then use it to prune the action space, as discussed later in this section. The simplest setup is to define $f(s_0, \delta_{\text{init}}) := \delta_{\text{init}}, \quad g(s,a,s',\delta) := \frac{\delta - c(s,a)}{\gamma}$, where the budget is initialized as the CMDP cost constraint. This formulation keeps track of the exact remaining budget while dividing by γ to account for discounting. However, this is not the only possible formulation. We can also track other quantities as the budget, as long as they help enforce the original CMDP cost constraint. In Section 3.3, we show how such a quantity can be utilized in stochastic CMDPs.

While we present our formulation with a single cost objective, our approach naturally extends to multiple cost types by augmenting the state space with a budget vector. We provide the detailed formulation for multiple cost constraints in Appendix A.3.

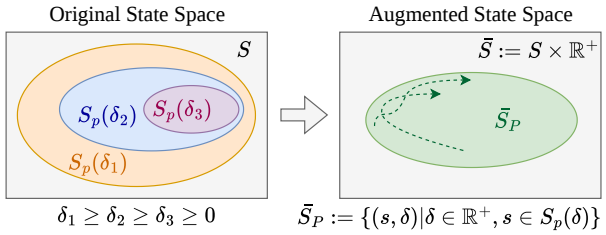

Figure 2: State augmentation with budget. A desirable property is set $\bar{S}_P$ to be persistent (trajectories starting in $\bar{S}_P$ must end in $\bar{S}_P$)

Definition 3.5 (Budget-Restricted Policy Set)**.** *Let $A_P(s,\delta)$ denote the persistent safety action set, as defined in the Eq* (8)*. The* budget-restricted policy set *is:*

$$\Pi_P := \left\{ \pi : \bar{S} \to \Delta(A) \;\middle|\; \pi(a \mid (s,\delta)) > 0 \implies a \in A_P(s,\delta) \right\}$$

The goal of defining Π_P is that, with an appropriate choice of f and g, any policy in Π_P will satisfy the CMDP constraint, as discussed in Section 3.3.

The objective of the BAMDP is to find a policy in Π_P that maximizes the reward.

$$\pi^* = \max_{\pi \in \Pi_P} \mathbb{E}_{\pi} \sum_{t=0}^{\infty} \gamma^t \bar{r}((s_t, \delta_t), a_t) \tag{9}$$

Feasible state space and Π_P A policy in Π_P may not be defined for all states, since there may exist states where $Q_C^*(s,a) > \delta$ for all actions $a \in A$, leaving no admissible action. The following definition identifies the subset of states where the budget-restricted policy set Π_P can be well-defined.

Definition 3.6 (Feasible State Subspace)**.** *The* feasible state subspace *is* $\bar{S}_P := \left\{ (s,\delta) \;\middle|\; \delta \in \mathbb{R}^+,\ s \in S_P(\delta) \right\}$*. where $S_P(\delta)$ is the persistent safety set in* (7) *.*

Augmented State Space: Figure 2 illustrates the modified state space and the feasible subspace defined within it. The feasible subspace represents the set of augmented states $((s,\delta))$ where each state (s) remains feasible under its corresponding budget (δ). As shown in Lemma 3.3, for every state $s \in S_P(\delta)$, the safe action set $A_P(s,\delta)$ is non-empty. This guarantees that the policy set Π_P is well-defined over $\bar{S}_P$, since there is always at least one admissible action for each $(s,\delta) \in \bar{S}_P$.

Intuitively, some key desirable properties associated with $\bar{S}_P$ and Π_P are: (a) the set $\bar{S}_P$ should be persistent. That, is an agent starting in $s \in \bar{S}_P$ remains in $\bar{S}_P$ following a policy $\pi \in \Pi_P$; (b) For any policy $\pi \in \Pi_P$, it is guaranteed that the original CMDP constraint $J_C(\pi) \leq \delta_{\text{init}}$ is satisfied. In the following, we discuss benefits of defining Π_P and $\bar{S}_P$, and then show how to choose appropriate budget update functions for properties (a) and (b) to hold.

Benefits of defining Π_P and $\bar{S}_P$: Both sets are constructed from the persistent safety sets in Eq. 7 and Eq. 8, which depend only on the conservative cost critic and are

independent of both the reward signal and the reward-optimizing policy. This makes enforcing $\pi \in \Pi_P$ far simpler than satisfying the original CMDP constraint in Eq. 1, which depends on the policy being learned and creates a difficult min–max coupling between the cost critic, reward critic, and policy. Our formulation removes this circular dependency, yielding a more stable learning process.

3.3 Budget Update for CMDPs

The most intuitive budget update function is to track the remaining cost budget exactly, where $\delta_0 = \delta_{\text{init}}$, and define the update rule as: $\delta_{t+1} = \frac{\delta_t - c(s_t,a_t)}{\gamma}$. In fact, such a budget update rule can result in safety for fully deterministic CMDPs, where, transition function and policy both are deterministic. We formally define and prove it in Appendix A.1. However, tracking such an exact budget and using it to restrict the policy to Π_P does not guarantee safety in stochastic CMDPs. In a stochastic setting, the cost-to-go satisfies

$$Q_C^*(s,a) = c(s,a) + \gamma \mathbb{E}_{s'}[V_C^*(s')],$$

rather than the deterministic form

$$Q_C^*(s,a) = c(s,a) + \gamma V_C^*(s').$$

As a result, the next-state value $V_C^*(s')$ can take a range of values, and the budget update defined for the deterministic setting no longer guarantees that (s', δ') lies inside $\bar{S}_P$. In such cases, we may need to follow a different policy—possibly the most conservative policy at these states—but doing so does not ensure that the agent will satisfy the CMDP constraint in Eq. (1). To address these issues, we propose an alternative budget update formulation, called *soft budget-tracking*.

Definition 3.7 (Soft Budget-Tracking)**.** *Let $\bar{\mathcal{M}}_S$ denote the Soft Budget-Tracking BAMDP defined by*

$$\bar{\mathcal{M}}_S(\mathcal{M}) := \bar{\mathcal{M}}(\mathcal{M}, f, g),$$

with

$$f(s_0, \delta_{init}) = V_C^*(s_0) + \delta_{init} - \mathop{\mathbb{E}}_{s\sim\mu_0}[V_C^*(s)], \tag{10}$$

$$g(s,a,s',\delta) = V_C^*(s') + \frac{\delta - Q_C^*(s,a)}{\gamma} \tag{11}$$

The advantage of the above formulation is that it ensures that after any transition, the next state $(s', \delta') \in \bar{S}_P$. Moreover, we can show that it satisfies the CMDP constraint. Although $\bar{\mathcal{M}}_S$ may seem unintuitive, its budget update functions reduce to the deterministic budget update in fully deterministic settings, and we formally show this in Appendix A.2.

With the soft budget-tracking BAMDP, we can reason about the behavior of policies constrained to remain within the persistent safety set $\bar{S}_P$, even in stochastic environments. Under the reasonable assumption Eq. (12) on the initial cost threshold, we can formally show that these policies maintain feasibility: they remain inside the persistent safety set and satisfy the CMDP cumulative cost constraint in expectation. The following theorem summarizes these key properties.

Theorem 3.8 (Properties of Policies in Π_P under Soft Budget-Tracking)**.** *Let $\bar{\mathcal{M}}_S$ be the soft budget-tracking BAMDP. Assume the CMDP is feasible, i.e., the cost threshold δ_{init} satisfies*

$$\delta_{init} \geq \mathbb{E}_{s\sim\mu_0}[V_C^*(s)]. \tag{12}$$

Let $\pi \in \Pi_P$, and define the expected discounted cumulative cost from an augmented state $(s,\delta) \in \bar{S}$ by

$$J_C^\pi((s,\delta)) := \mathop{\mathbb{E}}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t c(s_t,a_t) \;\Big|\; (s_0,\delta_0) = (s,\delta)\Big].$$

Then the following properties hold:

1. *Any policy $\pi \in \Pi_P$ induces trajectories that start in and remain entirely within the feasible subspace $\bar{S}_P$.*
2. *For any $(s,\delta) \in \bar{S}_P$ and any policy $\pi \in \Pi_P$, $J_C^\pi((s,\delta)) \leq \delta$.*
3. *Any policy $\pi \in \Pi_P$ satisfies the CMDP cumulative cost constraint (Eq. 1): $J_C(\pi) \leq \delta_{init}$.*

The proof of Theorem 3.8 is provided in Appendix A.2. The theorem ensures that any policy in Π_P remains within $\bar{S}_P$ and that the expected cumulative cost from any augmented state (s,δ) does not exceed its budget δ, thereby satisfying the CMDP constraint. Unlike the deterministic case, we cannot guarantee that all CMDP-feasible policies lie within Π_P, so the resulting constraint may be stricter. Nevertheless, our empirical results show that it still yields high-quality policies.

CMDP Instantiation We next discuss how to formulate and solve BAMDP from a given CMDP with known model. BAMDP is formed by augmenting the state with a discretized budget dimension and defining transitions using f or g, depending on whether the setting is stochastic or deterministic. We mask states outside $\bar{S}_P$ and restrict actions to the support of Π_P. The resulting finite MDP can then be solved with a standard LP solver. Further details appear in Appendix D.1.

In real-world settings where dynamics are unknown, offline RL is often used to learn from pre-collected data. Next, we show how our framework integrates with standard offline RL methods to solve *constrained* RL problems.

4 Safe Offline-RL With BAMDP

In this section we discuss how our approach can be used with existing *unconstrained* offline RL algorithms. Our approach can be seen as a three step process:

1. We first estimate the cost-to-go functions V_C^* and Q_C^* using offline RL algorithms, ignoring the reward, using the original dataset $\mathcal{D}$.
2. Then we use it create a new dataset $\bar{\mathcal{D}}$ from $\mathcal{D}$ that represent valid transitions of the augmented MDP $\bar{\mathcal{M}}$ with the extended state space $\bar{S}$ and dynamic budget.
3. Then we train an Offline RL agent, using the augmented dataset $\bar{\mathcal{D}}$ to maximize the reward for the augmented MDP $\bar{\mathcal{M}}$, while ensuring that the resulting policy respects the persistent safety constraints.

The first step is performed on the original MDP without any modifications to the original MDP, while the third step is performed on the BAMDP.

Learning the Persistent Safety Set For the first step, any offline RL algorithm can be used; the only difference is that we minimize the cost instead of maximizing the reward. In practice, in-sample RL algorithms such as IQL (Kostrikov, Nair, and Levine 2022), XQL (Garg et al. 2023), or SparseQL (Xu et al. 2023) are particularly convenient, as they do not require learning a policy to estimate Q-values. However, if one wishes to use algorithms that rely on out-of-distribution actions, such as SAC+BC (Fujimoto and Gu 2021), CQL (Kumar et al. 2020), our approach is fully compatible and does not impose any restrictions. For convenience and stability, we use IQL in our implementation. The following loss functions can then be used, with $\tau_C \leq 0.5$ to minimize the cost (details in (Kostrikov, Nair, and Levine 2022)):

$$\mathcal{L}_C^V(\psi) = \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}}[\mathcal{L}^{\tau_C}(Q_{\theta_T}^C(s,a) - V_\psi^C(s))] \tag{13}$$

$$\mathcal{L}_C^Q(\theta) = \mathop{\mathbb{E}}_{(s,a,s')\sim\mathcal{D}}[(c(s,a)+\gamma V_\psi^C(s') - Q_\theta^C(s,a))^2] \tag{14}$$

Offline RL within the Augmented MDP In this step, we train an RL algorithm within the persistent safety set of the augmented MDP $\bar{\mathcal{M}}(\mathcal{M}, f, g)$, ensuring the policy selects only actions from A_P by restricting the MDP to $\bar{S}_P$. To facilitate training, we define an augmented dataset as follows:

$$\bar{\mathcal{D}} := \left\{ \big(\underbrace{(s,\delta)}_{\bar{s}}, a, \underbrace{(s',\delta')}_{\bar{s}'} \big) \;\middle|\; \substack{(s,a,s')\sim\mathcal{D},\\ \delta\sim\mathcal{U}_{[Q_C^*(s,a),\delta_{\max}]},\ \delta'=g(s,a,s',\delta)} \right\}$$

Unlike a fixed dataset, this augmented dataset is not pre-generated; instead, we sample mini-batches from it dynamically during training. Here, the budget δ is sampled from a uniform distribution $\mathcal{U}_{[Q_C^*(s,a),\delta_{\max}]}$ to generate augmented transitions within the persistent safety set. Sampling $\delta \geq Q_C^*(s,a)$ ensures that the dataset extends only to the persistent safety set $\bar{S}_P$ and not to the full augmented state space $\bar{S}$. Crucially, this process acts as data augmentation rather than filtering. A valid budget range $[Q_C^*(s,a), \delta_{\max}]$ always exists for any state-action transition (s,a,s') in the offline dataset. Thus, we can always sample a valid budget δ and construct an augmented transition $((s,\delta),a,(s',\delta'))$. This ensures that the training process utilizes the entire support of the original offline dataset without discarding any samples. The rewards $r(s,a)$ and $c(s,a)$ remain unchanged, as discussed in the Definition 3.4.

Implicit Enforcement of Persistent Safety Offline RL naturally stays close to the behavior policy, thus restricting the dataset to the persistent safety set already biases the learned policy to remain within it. In practice, we found no extra penalties are required.

IQL-Instantiation We instantiate our approach using Implicit Q-Learning (IQL) by extending the offline dataset to include only transitions within the persistent safety set $\bar{S}_P$. This modification naturally integrates our safety constraint into the learning process. The value and Q-functions are trained using the standard expectile regression (with $\tau_R \geq 0.5$) and Bellman objectives defined over the augmented dataset $\bar{\mathcal{D}}$:

$$\mathcal{L}_R^V(\hat{\psi}) = \mathop{\mathbb{E}}_{(\bar{s},a,\bar{s}')\sim\bar{\mathcal{D}}}\left[\mathcal{L}^{\tau_R}\big(Q_{\hat{\theta}_T}^R(\bar{s},a) - V_{\hat{\psi}}^R(\bar{s})\big)\right] \tag{15}$$

$$\mathcal{L}_R^Q(\hat{\theta}) = \mathop{\mathbb{E}}_{(\bar{s},a,\bar{s}')\sim\bar{\mathcal{D}}}\left[(\bar{r}(\bar{s},a) + \gamma V_{\hat{\psi}}^R(\bar{s}') - Q_{\hat{\theta}}^R(\bar{s},a))^2\right] \tag{16}$$

Here, θ_T denotes the target network parameters, which are updated using a soft update. For policy extraction, we follow Advantage-Weighted Regression (AWR) as in IQL, where the policy is trained to maximize the advantage-weighted likelihood of actions under the Q-function:

$$\mathcal{L}^\pi(\phi) = -\mathop{\mathbb{E}}_{(\bar{s},a)\sim\bar{\mathcal{D}}}\left[\exp\left(\frac{A(\bar{s},a)}{\beta}\right)\log\pi_\phi(a\,|\,\bar{s})\right] \tag{17}$$

where $A(\bar{s},a) = Q_{\hat{\theta}}^R(\bar{s},a) - V_{\hat{\psi}}^R(\bar{s})$ denotes the advantage, and β controls the temperature of the weighting distribution. The exact learning process is outlined in the Algorithm 1. In our method, called BCRL, cost critics (Q^C, V^C) can be learned by ignoring rewards altogether, and policy extraction only depends on reward critics. This is significantly more stable that standard Lagrangian relaxation methods.

Algorithm 1: BCRL : IQL Instantiation Algorithm

Initialize Parameters: θ, θ_T, ψ, $\hat{\theta}$, $\hat{\theta}_T$, $\hat{\psi}$, ϕ
Inputs: Offline data $\mathcal{D}$, and other hyper parameters $\tau_c, \tau_r, \alpha, \beta$
for each gradient step **do**
- $(s,a,s') \sim \mathcal{D}$ ▷ sample mini batch of transitions
- $\theta \leftarrow \theta - \lambda_Q^C \nabla_\theta \mathcal{L}_Q^C(\theta)$ ▷ update cost-critic Eq.(14)
- $\psi \leftarrow \psi - \lambda_V^C \nabla_\psi \mathcal{L}_V^C(\psi)$ ▷ update cost-value Eq.(13)
- $\theta_T \leftarrow (1-\alpha)\theta_T + \alpha\theta$ ▷ soft update cost target-net.

- $\delta \sim \mathcal{U}_{[Q_\theta^C(s,a),\delta_{\max}]}$ ▷ sample budget from uniform dist.
- $\delta' = g(s,a,s',\delta)$ ▷ compute next budget
- $\bar{s} = (s,\delta)$ and $\bar{s}' = (s',\delta')$ ▷ compute augmented states
- $\hat{\theta} \leftarrow \hat{\theta} - \lambda_Q^R \nabla_{\hat{\theta}} \mathcal{L}_Q^R(\hat{\theta})$ ▷ update reward critic Eq.(16)
- $\hat{\psi} \leftarrow \hat{\psi} - \lambda_V^R \nabla_{\hat{\psi}} \mathcal{L}_V^R(\hat{\psi})$ ▷ update value function Eq.(15)
- $\hat{\theta}_T \leftarrow (1-\alpha)\hat{\theta}_T + \alpha\hat{\theta}$ ▷ soft update reward target-net.
- $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi \mathcal{L}^{\pi_R}(\phi)$ ▷ policy extraction (AWR) Eq.(17)

end for

5 Experiments

We evaluate BCRL on DSRL benchmarks (Liu et al. 2024) and a real-world maritime navigation task. We aim to answer following questions: **(Q1)** Does BCRL create a notable gap from the optimal CMDP (model known) solution, particularly under stochastic conditions? (Section 5.1) **(Q2)** Can BCRL outperform existing state-of-the-art offline (model-free) safe RL baselines on standard benchmarks? (Section 5.2) **(Q3)** How sensitive is BCRL to different hyperparameter settings, and how do these choices affect its performance and safety behavior? **(Q4)** How does BCRL behave on real-world-data, both qualitatively (in terms of safe and interpretable trajectories) and quantitatively (in terms of performance and safety metrics)? (Section 5.3).

	CDT		CAPS		CCAC		LSPC		BCRL (Ours)	
	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
PointCircle1	**0.59**$_{\pm 0.00}$	**0.69**$_{\pm 0.04}$	**0.50**$_{\pm 0.12}$	**0.68**$_{\pm 1.51}$	0.58$_{\pm 0.13}$	2.22$_{\pm 1.05}$	0.56$_{\pm 0.35}$	3.49$_{\pm 3.55}$	**0.55**$_{\pm 0.17}$	**0.86**$_{\pm 0.21}$
PointCircle2	0.64$_{\pm 0.01}$	1.05$_{\pm 0.08}$	**0.51**$_{\pm 0.21}$	**0.76**$_{\pm 0.47}$	0.18$_{\pm 0.50}$	1.42$_{\pm 1.09}$	0.67$_{\pm 0.24}$	4.18$_{\pm 4.56}$	**0.58**$_{\pm 0.10}$	**0.74**$_{\pm 0.34}$
PointGoal1	0.69$_{\pm 0.02}$	1.12$_{\pm 0.07}$	**0.46**$_{\pm 0.28}$	**0.63**$_{\pm 0.81}$	0.55$_{\pm 0.26}$	1.88$_{\pm 1.72}$	**0.26**$_{\pm 0.27}$	**0.23**$_{\pm 0.45}$	**0.56**$_{\pm 0.25}$	**0.70**$_{\pm 0.65}$
PointGoal2	0.59$_{\pm 0.03}$	1.34$_{\pm 0.05}$	**0.34**$_{\pm 0.23}$	**0.80**$_{\pm 0.74}$	0.41$_{\pm 0.36}$	3.60$_{\pm 3.37}$	**0.23**$_{\pm 0.24}$	**0.50**$_{\pm 0.79}$	**0.38**$_{\pm 0.21}$	**0.77**$_{\pm 0.68}$
HopperVelocity	**0.63**$_{\pm 0.06}$	**0.61**$_{\pm 0.08}$	**0.44**$_{\pm 0.26}$	**0.75**$_{\pm 0.83}$	**0.46**$_{\pm 0.38}$	**0.50**$_{\pm 0.55}$	0.21$_{\pm 0.04}$	1.25$_{\pm 1.20}$	**0.66**$_{\pm 0.23}$	**0.80**$_{\pm 0.35}$
HalfCheetahVelocity	**1.00**$_{\pm 0.01}$	**0.01**$_{\pm 0.01}$	**0.94**$_{\pm 0.04}$	**0.77**$_{\pm 0.13}$	**0.94**$_{\pm 0.04}$	**0.94**$_{\pm 0.07}$	**0.97**$_{\pm 0.02}$	**1.00**$_{\pm 1.04}$	**0.93**$_{\pm 0.07}$	**0.62**$_{\pm 0.27}$
SafetyGym Average (21 tasks)	0.54$_{\pm 0.21}$	1.06$_{\pm 0.59}$	**0.36**$_{\pm 0.34}$	**0.76**$_{\pm 1.43}$	0.41$_{\pm 0.46}$	3.18$_{\pm 5.24}$	0.34$_{\pm 0.19}$	1.10$_{\pm 1.75}$	**0.38**$_{\pm 0.17}$	**0.57**$_{\pm 0.68}$
BallRun	0.39$_{\pm 0.09}$	1.16$_{\pm 0.19}$	**0.18**$_{\pm 0.09}$	**0.94**$_{\pm 0.73}$	**0.40**$_{\pm 0.12}$	**0.84**$_{\pm 0.31}$	**0.15**$_{\pm 0.01}$	**0.00**$_{\pm 0.00}$	**0.20**$_{\pm 0.02}$	**0.06**$_{\pm 0.10}$
CarCircle	**0.75**$_{\pm 0.06}$	**0.95**$_{\pm 0.61}$	**0.70**$_{\pm 0.10}$	**0.66**$_{\pm 0.27}$	**0.72**$_{\pm 0.04}$	**0.77**$_{\pm 0.44}$	0.78$_{\pm 0.12}$	1.88$_{\pm 3.23}$	**0.53**$_{\pm 0.17}$	**0.51**$_{\pm 0.51}$
DroneCircle	**0.63**$_{\pm 0.07}$	**0.98**$_{\pm 0.27}$	**0.55**$_{\pm 0.06}$	**0.65**$_{\pm 0.29}$	**0.29**$_{\pm 0.27}$	**0.63**$_{\pm 0.70}$	0.60$_{\pm 0.02}$	1.31$_{\pm 0.86}$	**0.42**$_{\pm 0.12}$	**0.52**$_{\pm 0.32}$
AntCircle	0.54$_{\pm 0.20}$	1.78$_{\pm 4.33}$	**0.37**$_{\pm 0.18}$	**0.15**$_{\pm 0.25}$	**0.61**$_{\pm 0.14}$	**0.75**$_{\pm 0.90}$	**0.44**$_{\pm 0.14}$	**0.53**$_{\pm 0.93}$	**0.56**$_{\pm 0.17}$	**0.79**$_{\pm 0.60}$
BulletGym Average (8 tasks)	0.68$_{\pm 0.19}$	1.04$_{\pm 1.65}$	**0.57**$_{\pm 0.25}$	**0.86**$_{\pm 1.82}$	0.54$_{\pm 0.30}$	1.07$_{\pm 2.04}$	0.60$_{\pm 0.10}$	1.01$_{\pm 1.35}$	**0.58**$_{\pm 0.09}$	**0.67**$_{\pm 0.46}$
easysparse	**0.17**$_{\pm 0.14}$	**0.23**$_{\pm 0.32}$	**0.12**$_{\pm 0.20}$	**0.37**$_{\pm 0.43}$	**-0.06**$_{\pm 0.00}$	**0.10**$_{\pm 0.05}$	0.79$_{\pm 0.14}$	1.34$_{\pm 1.43}$	**0.70**$_{\pm 0.19}$	**0.94**$_{\pm 0.17}$
easydense	**0.32**$_{\pm 0.18}$	**0.62**$_{\pm 0.43}$	**0.11**$_{\pm 0.15}$	**0.16**$_{\pm 0.17}$	**-0.06**$_{\pm 0.00}$	**0.07**$_{\pm 0.04}$	0.84$_{\pm 0.10}$	1.88$_{\pm 1.20}$	**0.70**$_{\pm 0.17}$	**0.90**$_{\pm 0.16}$
mediumdense	0.88$_{\pm 0.12}$	2.41$_{\pm 0.71}$	**0.75**$_{\pm 0.29}$	**0.65**$_{\pm 0.57}$	**-0.08**$_{\pm 0.00}$	**0.06**$_{\pm 0.03}$	0.82$_{\pm 0.30}$	1.07$_{\pm 0.71}$	**0.78**$_{\pm 0.28}$	**0.68**$_{\pm 0.31}$
hardmean	**0.33**$_{\pm 0.21}$	**0.97**$_{\pm 0.31}$	**0.29**$_{\pm 0.13}$	**0.28**$_{\pm 0.31}$	**-0.05**$_{\pm 0.00}$	**0.06**$_{\pm 0.03}$	0.48$_{\pm 0.14}$	1.16$_{\pm 1.19}$	**0.46**$_{\pm 0.15}$	**0.84**$_{\pm 0.54}$
harddense	**0.08**$_{\pm 0.15}$	**0.21**$_{\pm 0.42}$	**0.36**$_{\pm 0.18}$	**0.66**$_{\pm 0.92}$	**-0.03**$_{\pm 0.01}$	**0.11**$_{\pm 0.08}$	0.47$_{\pm 0.19}$	1.43$_{\pm 0.93}$	**0.44**$_{\pm 0.17}$	**0.63**$_{\pm 0.24}$
MetaDrive Average (9 tasks)	**0.42**$_{\pm 0.31}$	**0.80**$_{\pm 0.61}$	**0.38**$_{\pm 0.34}$	**0.54**$_{\pm 0.69}$	**-0.06**$_{\pm 0.02}$	**0.07**$_{\pm 0.06}$	0.74$_{\pm 0.13}$	1.24$_{\pm 0.89}$	**0.69**$_{\pm 0.16}$	**0.81**$_{\pm 0.27}$

Table 1: **Results for normalized cost and normalized reward:** ↑ indicates that a higher value is better, while ↓ indicates that a lower value is preferable. **Bold** text denotes safe policies, gray indicates unsafe policies, and **blue** highlights the highest reward among the safe policies for each task. Each result is obtained by running three random seeds across three distinct cost thresholds and evaluating over 20 episodes. Some tasks are omitted to save space. Complete results are provided in Table 3 of Appendix B.

	ADE (m) ↓	Close-quaters Rate ↓	Avg. Acceleration (ms^{-2})	Avg. Speed (ms^{-1})	Success Rate ↑
EXPERT	0.0	0.3	0.0001	4.44	1.0
CAPS	87.3 ± 19.82	0.23 ± 0.05	0.0034 ± 0.0006	5.67 ± 0.07	0.73 ± 0.16
CCAC	371.07 ± 26.74	**0.11 ± 0.06**	0.7549 ± 0.1339	21.41 ± 0.0	0.0 ± 0.0
LSPC	251.61 ± 22.68	0.17 ± 0.02	0.0087 ± 0.0157	8.47 ± 0.37	0.23 ± 0.1
BCRL	**52.08 ± 10.64**	0.26 ± 0.03	-0.0006 ± 0.0003	3.65 ± 0.14	**0.88 ± 0.03**

Table 2: Performance metrics in the Maritime Navigation Task (section 5.3)

Baselines and benchmarks: For our main results, we compare against state-of-the-art baselines that also generalize to different cost budgets similar to our method. These methods include **CDT** (Liu et al. 2023), **CAPS** (Chemingui et al. 2025), **CCAC** (Guo et al. 2025), and **LSPC** (Koirala et al. 2025). An additional comparison with methods that do not adapt to different cost budgets is provided in the supplementary. We use the DSRL (Liu et al. 2024) dataset and environments. We evaluate our approach on all 38 tasks, which include three types of environments: *SafetyGym*, *BulletGym*, and *MetaDrive*. The evaluation metrics include *Normalized-Reward* and *Normalized-Cost*, where the cost is normalized with respect to the cost threshold; that is, *Normalized-Cost* > 1 indicates an unsafe policy. More details on these metrics are in the Appendix.

5.1 Synthetic CMDPs

Under deterministic dynamics, our method matches the optimal solution; in stochastic settings, it can be slightly conservative. We test this on a discrete grid world where the optimal CMDP policy is computed via Linear Programming (LP) solvers. The agent starts at a fixed point and must reach a goal (appendix B.7 provides environment details.). Figure 3 compares our method with the LP solution and an unconstrained baseline. The X-axis is the probability of executing the intended action; lower values introduce more randomness. When this probability is too low, no feasible policy satisfies the budget, so the LP solver yields no solution (shown as missing plot segments). For all the cases, both methods satisfy the cost constraint, and our approach incurs only a small reward gap. As the environment becomes less noisy, our solution converges to the LP optimum. This shows that even in a highly stochastic setting, BCRL is fairly accurate.

5.2 Offline Safe RL

Main Results. Table 1 reports the main results under the standard DSRL evaluation settings. All algorithms are trained following the official DSRL benchmark protocols, with additional implementation details provided in the Appendix. We instantiate BCRL using IQL. In stochastic domains (e.g., SafetyGym, Bullet-SafetyGym), we use the stochastic variant, while for deterministic environments like MetaDrive, we report the deterministic version. Both variant results are detailed in the Appendix for completeness. Overall, the results show BCRL consistently outperforms baselines while satisfying safety across all domains. BCRL produces safe policies in all 38 out of 38 tasks (indicated in **bold**), outperforming all baselines in 16 out of 38 tasks, and achieving strongly consistent results as highlighted in blue. BCRL attains **highest average performance across all three benchmarks**.

Table 1 focuses on baselines capable of adapting to different cost budgets. Additional comparisons with non–budget-

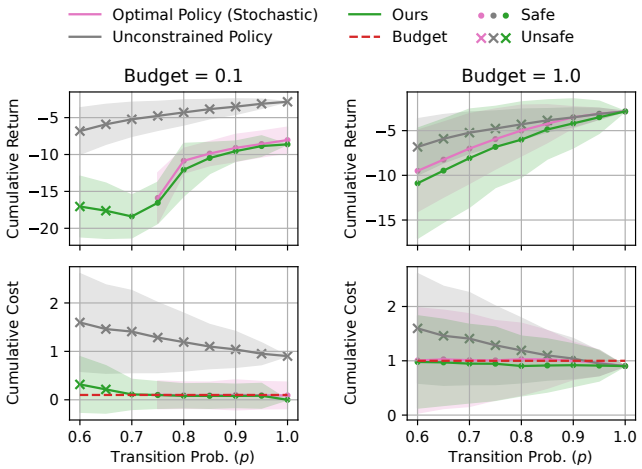

Figure 3: **Comparison with Optimal Solution:** Grid-world results with X-axis as the intended-movement probability p (higher values indicate less noise). Top plots show total return; bottom plots show cost for two budget levels.

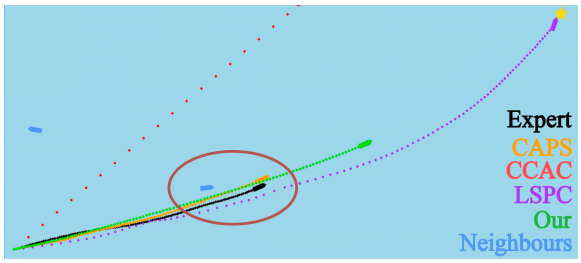

Figure 4: Expert, learned trajectories in marine navigation

conditioned baselines such as BC-Safe (Liu et al. 2024), BCQ-Lag, COptiDICE (Lee et al. 2022), and CPQ (Xu, Zhan, and Zhu 2022) are in the Appendix B. **Runtime:** Our algorithm was tested on an NVIDIA RTX 3090 GPU. As shown in Table 9 (in appendix), it completes training and evaluation within a few minutes—significantly faster than baseline methods that require 2–3 hours under the same conditions. Although runtime comparisons are approximate, the results indicate notable efficiency gains of our method.

Ablations: Our IQL variant is governed by three hyperparameters: the cost critic expectile (τ_C), the reward critic expectile (τ_R), and the AWR policy temperature (β). Among these, the expectile values influence performance more, as also noted in (Kostrikov, Nair, and Levine 2022). Expectile ablation results are in Table 6, appendix B.4. Additionally, the appendix discusses how the quality of the learned cost critic affects overall performance and how inclusive the learned persistent safety set is.

5.3 Evaluation on Real-World Maritime Data

We test BCRL on a real-world maritime navigation task, learning a policy that imitates expert captains navigating congested routes while minimizing collision risk. We use the maritime traffic simulator, and the imitation-learning dataset from (Pham, Brahmanage, and Kumar 2025), augmented with reward and cost signals. The agent receives a sparse reward of 100 for reaching the goal, and a cost of 1 for entering a **close-quarter scenario** (getting close to another vessel within 555,m (Pham, Brahmanage, and Kumar 2025)).

The dataset contains ∼2 years of AIS trajectories from vessels operating in the Singapore Strait. We simulate a **multi-agent environment** where surrounding vessels replay their historical (log-play) paths, while the RL-controlled ego agent starts from a historical position but must follow its learned policy. The agent observes the past five steps of its own and nearby vessels' states and uses a delta-action space (Gulino et al. 2023). We evaluate performance using the **average displacement error (ADE)**, the **close-quarter rate**, and the **success rate**. To model dense traffic, we select the top 20% most congested scenarios. We additionally report **average acceleration and speed** to assess how closely the policy matches real-world maneuvering. Further environment and dataset details appear in Appendix B.3.

As shown in Table 2, BCRL reduces close-quarter events from 30% to 26%—a meaningful improvement given the high-risk nature of close encounters involving large cargo and tanker vessels. BCRL also achieves the **lowest ADE** (∼52 m) and the highest **success rate** (88%). Some baselines reduce close-quarter rates further but at the cost of unrealistic deviations from expert trajectories or reduced task success. In contrast, BCRL maintains expert-like **acceleration and speed** profiles. Results are averaged over 3 seeds with 100 evaluation episodes each.

Figure 4 illustrates qualitative behavior: the expert's trajectory (dotted **black**), neighboring vessels (blue), and learned-policy rollouts—BCRL, LSPC, CCAC, and CAPS. The red ellipse shows a close-quarter scenario where the expert gets too close to a neighbor. BCRL avoids near-misses through smooth path and speed adjustments, whereas LSPC and CCAC exhibit unrealistic deviations. CAPS performs more reasonably but shows higher acceleration and lower success than BCRL. Additional results and videos are included in the supplementary material.

6 Conclusion

We introduced Budget-Conditioned Reachability (BCR), a novel framework that fundamentally decouples reward maximization from cumulative safety constraints in Constrained Markov Decision Processes (CMDPs). By defining a dynamic, safety-conditioned reachability set—tracked through a step-wise remaining budget—our approach prunes unsafe actions at each timestep and ensures reliable constraint satisfaction without resorting to unstable min-max adversarial optimization or computationally heavy generative models.

A significant advantage of our framework is its plug-and-play compatibility with any standard offline RL algorithm (e.g., IQL), enabling the cost critics to be learned independently of the reward for improved training stability. Through extensive cross-domain validation—ranging from interpretable grid-worlds and the high-dimensional DSRL benchmark to a complex, real-world maritime navigation task—BCRL consistently matches or exceeds state-of-the-art performance while reliably ensuring safety.

References

Altman, E. 2021. *Constrained Markov Decision Processes: Stochastic Modeling*. Boca Raton: Routledge, 1 edition. ISBN 978-1-315-14022-3.

Brahmanage, J. C.; Ling, J.; and Kumar, A. 2023. FlowPG: Action-constrained Policy Gradient with Normalizing Flows. In *NeurIPS*.

Brahmanage, J. C.; Ling, J.; and Kumar, A. 2024. Leveraging Constraint Violation Signals For Action Constrained Reinforcement Learning. In *AAAI*.

Chemingui, Y.; Deshwal, A.; Wei, H.; Fern, A.; and Doppa, J. R. 2025. Constraint-Adaptive Policy Switching for Offline Safe Reinforcement Learning. In *AAAI*.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *NeurIPS*.

Fawzi, A.; Balog, M.; Huang, A.; Hubert, T.; Romera-Paredes, B.; Barekatain, M.; Novikov, A.; R. Ruiz, F. J.; Schrittwieser, J.; Swirszcz, G.; Silver, D.; Hassabis, D.; and Kohli, P. 2022. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610: 47–53.

Fujimoto, S.; and Gu, S. 2021. A Minimalist Approach to Offline Reinforcement Learning. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *NeurIPS*.

Ganai, M.; Gao, S.; and Herbert, S. L. 2024. Hamilton-Jacobi Reachability in Reinforcement Learning: A Survey. *IEEE OJ-CS*, 3: 310–324.

Ganai, M.; Gong, Z.; Yu, C.; Herbert, S. L.; and Gao, S. 2023. Iterative Reachability Estimation for Safe Reinforcement Learning. In *NeurIPS*.

Garg, D.; Hejna, J.; Geist, M.; and Ermon, S. 2023. Extreme Q-Learning: MaxEnt RL without Entropy. In *ICLR*.

Gu, S.; Yang, L.; Du, Y.; Chen, G.; Walter, F.; Wang, J.; and Knoll, A. 2024. A Review of Safe Reinforcement Learning: Methods, Theory and Applications. In *IEEE TPAMI*.

Gulino, C.; Fu, J.; Luo, W.; Tucker, G.; Bronstein, E.; Lu, Y.; Harb, J.; Pan, X.; Wang, Y.; Chen, X.; Co-Reyes, J. D.; Agarwal, R.; Roelofs, R.; Lu, Y.; Montali, N.; Mougin, P.; Yang, Z.; White, B.; Faust, A.; McAllister, R.; Anguelov, D.; and Sapp, B. 2023. Waymax: An Accelerated, Data-Driven Simulator for Large-Scale Autonomous Driving Research. In *NeurIPS*.

Guo, Z.; Zhou, W.; Wang, S.; and Li, W. 2025. Constraint-Conditioned Actor-Critic for Offline Safe Reinforcement Learning. In *ICLR*.

Jang, Y.; Kim, G.-H.; Lee, J.; Sohn, S.; Kim, B.; Lee, H.; and Lee, M. 2023. SafeDICE: Offline Safe Imitation Learning with Non-Preferred Demonstrations. In *NeurIPS*.

Kim, G.-H.; Seo, S.; Lee, J.; Jeon, W.; Hwang, H.; Yang, H.; and Kim, K.-E. 2022. DemoDICE: Offline Imitation Learning with Supplementary Imperfect Demonstrations. In *ICLR*.

Kingma, D. P.; Welling, M.; et al. 2019. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4): 307–392.

Koirala, P.; Jiang, Z.; Sarkar, S.; and Fleming, C. 2025. Latent Safety-Constrained Policy Approach for Safe Offline Reinforcement Learning. In *ICLR*.

Kostrikov, I.; Nair, A.; and Levine, S. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *ICLR*.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. *Advances in NeurIPS*, 33: 1179–1191.

Lee, J.; Jeon, W.; Lee, B.-J.; Pineau, J.; and Kim, K.-E. 2021. OptiDICE: Offline Policy Optimization via Stationary Distribution Correction Estimation. In *ICML*.

Lee, J.; Paduraru, C.; Mankowitz, D. J.; Heess, N.; Precup, D.; Kim, K.-E.; and Guez, A. 2022. COptiDICE: Offline Constrained Reinforcement Learning via Stationary Distribution Correction Estimation. In *ICLR*.

Lin, J.-L.; Hung, W.; Yang, S.-H.; Hsieh, P.-C.; and Liu, X. 2021. Escaping from zero gradient: Revisiting action-constrained reinforcement learning via Frank-Wolfe policy optimization. In *UAI*, 397–407. PMLR.

Liu, Z.; Guo, Z.; Lin, H.; Yao, Y.; Zhu, J.; Cen, Z.; Hu, H.; Yu, W.; Zhang, T.; Tan, J.; and Zhao, D. 2024. Datasets and Benchmarks for Offline Safe Reinforcement Learning. *JMLR*.

Liu, Z.; Guo, Z.; Yao, Y.; Cen, Z.; Yu, W.; Zhang, T.; and Zhao, D. 2023. Constrained Decision Transformer for Offline Safe Reinforcement Learning. In *ICML*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning.

Nachum, O.; Chow, Y.; Dai, B.; and Li, L. 2019. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Advances in NeurIPS*, 32.

Newey, W. K.; and Powell, J. L. 1987. Asymmetric Least Squares Estimation and Testing. *Econometrica*, 55: 819–847.

Peng, X. B.; Kumar, A.; Zhang, G.; and Levine, S. 2019. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning.

Pham, Q. A.; Brahmanage, J. C.; and Kumar, A. 2025. ShipNaviSim: Data-Driven Simulation for Real-World Maritime Navigation. In *AAMAS*, 1641–1649.

Pham, Q. A.; Brahmanage, J. C.; Mai, T. A.; and Kumar, A. 2025. IOSTOM: Offline Imitation Learning from Observations via State Transition Occupancy Matching. In *NeurIPS*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489.

Sootla, A.; Cowen-Rivers, A. I.; Jafferjee, T.; Wang, Z.; Mguni, D.; Wang, J.; and Bou-Ammar, H. 2022. Saute RL: Almost Surely Safe Reinforcement Learning Using State Augmentation. In *ICML*.

Tang, C.; Abbatematteo, B.; Hu, J.; Chandra, R.; Martín-Martín, R.; and Stone, P. 2024. Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes. In *ARCRAS*.

Wang, X.; Xu, H.; Zheng, Y.; and Zhan, X. 2023. Offline Multi-Agent Reinforcement Learning with Implicit Global-to-Local Value Regularization. In *NeurIPS*.

Xu, H.; Jiang, L.; Li, J.; Yang, Z.; Wang, Z.; Chan, V. W. K.; and Zhan, X. 2023. Offline RL with No OOD Actions: In-Sample Learning via Implicit Value Regularization. In *ICLR*.

Xu, H.; Zhan, X.; and Zhu, X. 2022. Constraints Penalized Q-learning for Safe Offline Reinforcement Learning. In *NeurIPS*.

Yu, D.; Ma, H.; Li, S. E.; and Chen, J. 2022. Reachability Constrained Reinforcement Learning. In *ICML*.

Zheng, Y.; Li, J.; Yu, D.; Yang, Y.; Li, S. E.; Zhan, X.; and Liu, J. 2024. Safe Offline Reinforcement Learning with Feasibility-Guided Diffusion Model. In *ICLR*.