SAFETY-AWARE VALUE ALIGNMENT FOR OFFLINE SAFE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline safe reinforcement learning (OSRL) often extracts policies from learned value funcitons, but learned reward values can rank dataset-supported actions differently from the constrained objective. We introduce Safety-Aware Value Alignment (SAVA), which makes learned long-horizon cost information part of the Bellman target for reward value learning. Using cost values to identify high-cost regions, SAVA reshapes reward value learning when transitions enter these regions, allowing the resulting penalized Bellman targets to propagate to preceding state–action pairs and to reshape earlier action preferences before policy extraction. We theoretically establish sufficient conditions under which the penalized objective favors safe policies over unsafe ones and derive a bound on the reward gap to the optimal safe policy under value-estimation and policy-extraction errors. Empirically, controlled ablations show that the region-aware penalized reward value is the primary component responsible for constraint satisfaction. On the DSRL benchmark, SAVA achieves the highest normalized return on 27 of 32 tasks while satisfying the common normalized-cost safety criterion.

1 INTRODUCTION

Offline safe reinforcement learning (OSRL) aims to learn high-reward policies from fixed datasets while satisfying safety constraints at deployment. A central challenge inherited from offline reinforcement learning (RL) is distributional shift, where out-of-distribution actions can receive overestimated values and induce behavior poorly represented in the dataset (Fujimoto et al., 2019; Levine et al., 2020; Janner et al., 2021). Behavior regularization and conservative value estimation can mitigate this risk, but may also exclude rewarding actions that are only sparsely supported by the offline data (Fujimoto et al., 2019; Kumar et al., 2020; An et al., 2021; Kumar et al., 2019; Fujimoto & Gu, 2021; Mao et al., 2023). In-sample methods such as implicit Q-learning (IQL) avoid explicit out-of-distribution action queries by estimating values only for dataset-supported actions, which makes this value-learning paradigm particularly suitable for OSRL (Kostrikov et al., 2022; Hansen-Estruch et al., 2023; Xu et al., 2023; Garg et al., 2023; Xiao et al., 2023).

Safe RL augments reward maximization with a cost signal representing constraint violation (García & Fernández, 2015; Ray et al., 2019; Brunke et al., 2022; Andersen et al., 2020; Shi et al., 2021; Gu et al., 2024), and commonly incorporates this signal through primal–dual optimization (Paternain et al., 2023; Ying et al., 2023; Wu et al., 2024; Ding & Lavaei, 2023), reward penalties (Laud & DeJong, 2003; Thomas et al., 2021; Araújo & Braga, 1998; Nilaksh et al., 2024), and feasibility analysis (Chow et al., 2018; Fisac et al., 2019; Yu et al., 2022; Ganai et al., 2023). In the offline setting, the long-term reward and cost consequences of candidate actions must be estimated from fixed data without additional environment interaction, making reliable value estimation and policy improvement particularly challenging.

In OSRL, a fundamental challenge lies in how to incorporate safety information into policy improvement while preserving reward optimization from fixed offline data. Existing OSRL methods incorporate safety information through different approaches, including cost-penalty formulations (Xu et al., 2022), constrained optimization (Lee et al., 2022; Liu et al., 2023b; Zheng et al., 2024), and deployment-time safety intervention (Gong et al., 2025; Chemingui et al., 2025). While existing approaches incorporate safety information at different stages of policy improvement, they generally do not allow expected cumulative cost to reshape the reward optimization trajectory before the policy

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

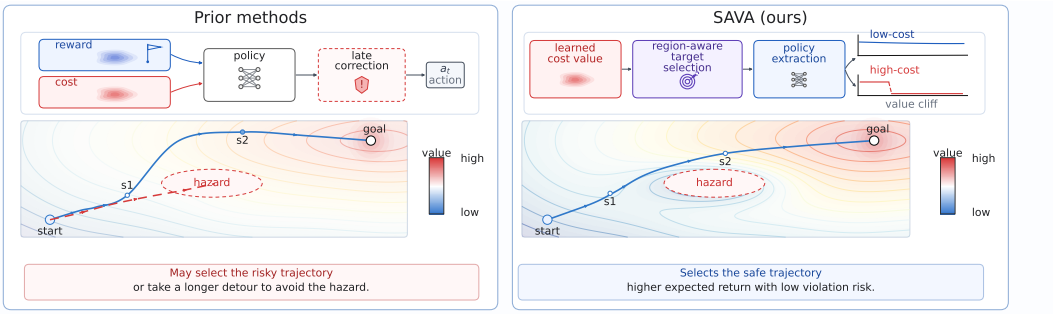


Figure 1: **Conceptual illustration of SAVA in a reach–avoid setting.**

is obtained. As a result, the learned policies may not fully account for future constraint violations, leading to suboptimal reward–safety trade-offs. These limitations motivate a fundamental question: **how can offline policy improvement incorporate cumulative cost into reward optimization at the value-learning stage, before policy optimization?**

SAVA addresses this problem by incorporating learned long-horizon cost information directly into reward value learning before policy extraction. Different from existing approaches that treat safety as an external penalty or post-hoc correction, SAVA uses the estimated long-horizon cost value to identify future high-cost regions and modifies the reward Bellman target according to the predicted safety consequences of subsequent trajectories. Through Bellman backups, this penalized target propagates to preceding state–action pairs, reducing the learned reward values of actions that lead toward the estimated high-cost region and creating a sharp value drop near the estimated cost boundary, thereby allowing predicted future cost to influence earlier action preferences. Figure 1 provides a conceptual comparison with late safety correction in a reach–avoid setting, where delayed correction may lead to a longer route or fail to avoid the hazard, whereas SAVA changes the earlier action preference before policy extraction.

To analyze the effect of the proposed penalization, we study the corresponding penalized objective. We show that, when the penalty coefficient is sufficiently large, safe policies are preferred to unsafe policies under this objective. We further bound the reward gap between the learned policy and the optimal safe policy in terms of the penalty magnitude, the discounted high-cost region occupancy of the optimal safe policy, value-estimation error, and policy-extraction error.

Empirically, controlled ablations identify the region aware penalized reward value as the primary component responsible for constraint satisfaction among the evaluated variants, while the remaining components mainly affect finite-data robustness, expressiveness, and performance. On the DSRL benchmark (Liu et al., 2023a), SAVA achieves the highest normalized return on 27 of 32 tasks while satisfying the same normalized cost criterion used for all compared methods.

2 Related Work

Offline Safe Reinforcement Learning. Offline safe reinforcement learning learns policies from fixed datasets under safety constraints, typically formulated as constrained Markov decision processes (Altman, 2021). Existing approaches combine offline RL with safety mechanisms including cost penalties and constrained optimization (Xu et al., 2022; Lee et al., 2022), trajectory conditioning (Liu et al., 2023b; Lin et al., 2023; Zhang et al., 2023; Gong et al., 2025), and feasibility- or deployment-time intervention (Zheng et al., 2024; Chemingui et al., 2025). Representative methods include CPQ, which restricts reward backups using cost estimates (Xu et al., 2022), and COptiDICE, which optimizes constrained stationary distributions (Lee et al., 2022). Recent work further explores latent safety-constrained policy classes, dataset shaping, and constraint-conditioned learning (Koirala et al., 2025; Yao et al., 2024; Guo et al., 2025). Collectively, these methods place cost information in penalties, constraints, feasibility signals, trajectory conditions, switching criteria, or distributional regularizers. Different from existing approaches, SAVA uses successor-state cumulative cost to determine whether the reward Bellman target is replaced by a fixed penalty, allowing high-cost outcomes to affect earlier action decisions before policy optimization.

Policy Extraction under Safety Constraints. A more direct connection to SAVA arises in offline RL, where policy extraction is determined by the ordering induced by learned values over dataset-supported actions (Kostrikov et al., 2022). In in-sample methods such as IQL, the actor is extracted from learned advantages without explicitly querying out-of-distribution actions (Kostrikov et al., 2022). Recent safe extensions incorporate constraint information into value estimation (Xu et al., 2022; Guo et al., 2025), policy conditioning (Liu et al., 2023b; Zhang et al., 2023; Lin et al., 2023; Guo et al., 2025), or policy selection (Zheng et al., 2024; Chemingui et al., 2025), thereby incorporating constraint information into policy extraction.

However, incorporating cumulative cost into the learning pipeline does not by itself specify how long-horizon cost term should alter the reward value learning that determines the relative preference among dataset-supported actions. Consequently, a learned reward value can still favor an action whose future trajectory leads toward high predicted cost, even when cumulative cost is subsequently used for feasibility guidance (Zheng et al., 2024), posterior safety verification (Zhang et al., 2023), and deployment-time policy switching (Chemingui et al., 2025). SAVA instead uses successor-state cumulative cost to reshape reward Bellman learning before policy optimization.

3 PRELIMINARIES

Offline Safe RL. Safe reinforcement learning is commonly formulated as a constrained Markov decision process (CMDP) (Altman, 2021), $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, r, c, \gamma, \rho_0)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, P is the transition dynamics, r and c are the reward and cost functions, $\gamma \in [0, 1)$ is the discount factor, and ρ_0 is the initial state distribution. For a policy $\pi(a \mid s)$, define

$$J_r(\pi) := \mathbb{E}_{s_0 \sim \rho_0, \pi, P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \quad J_c(\pi) := \mathbb{E}_{s_0 \sim \rho_0, \pi, P}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)\right]. \tag{1}$$

Safe RL seeks a reward-maximizing policy under a cost budget:

$$\max_{\pi} J_r(\pi) \qquad \text{s.t.} \qquad J_c(\pi) \leq \ell, \tag{2}$$

where $\ell \geq 0$ is the CMDP cost budget. The DSRL benchmark denotes the same budget by κ, so the standard benchmark setting uses $\ell = \kappa$.

Offline safe RL learns from a fixed dataset $\mathcal{D}$ collected by a behavior policy π_β, commonly with regularization that limits deviation from the offline behavior distribution.

In-Sample Value Learning. SAVA builds on the in-sample value-learning principle of IQL (Kostrikov et al., 2022), where value estimation and policy extraction are restricted to actions supported by the offline dataset. Given a dataset $\mathcal{D}$, IQL learns a state-action value function Q and a state value function V. The reward critic is optimized using the Bellman regression objective

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}\left[\left(r + \gamma V(s') - Q(s, a)\right)^2\right]. \tag{3}$$

The value function is learned through expectile regression:

$$\mathcal{L}_V = \mathbb{E}_{(s,a) \sim \mathcal{D}}\left[\mathcal{L}_2^\tau(Q(s, a) - V(s))\right], \quad \text{where} \quad \mathcal{L}_2^\tau(u) = |\tau - \mathbf{1}(u < 0)|u^2. \tag{4}$$

The learned advantage $A(s, a) = Q(s, a) - V(s)$ is then used for policy extraction, such as advantage-weighted regression (AWR). SAVA follows this value-learning framework and modifies the reward Bellman target when learning the value functions.

4 METHODS

SAVA incorporates learned cumulative cost into reward Bellman learning through a region-aware penalized target, allowing predicted high-cost outcomes to affect earlier decisions before policy optimization. The method first uses the learned cost value to construct an estimated low-cost and high-cost partition of the state space. Transitions entering the estimated high-cost region receive a penalized reward Bellman target, which induces a region-aware penalized value objective for policy extraction. This penalized value landscape provides the primary policy-extraction signal. We detail this construction in Section 4.1. We further analyze the penalized objective, showing when it favors safe policies over unsafe ones and bounding the reward gap induced by value-estimation and policy-extraction errors in Section 4.2. Finally, we provide the implementation details for practical offline policy learning in Section 4.3.

4.1 Region-Aware Penalized Value Learning

SAVA modifies the reward Bellman target according to the safety status of successor states estimated by the learned cost value. Rather than using cost values only as an additional optimization signal, SAVA first identifies states with predicted cumulative cost above the threshold and then modifies the corresponding reward value updates. This design separates cost-satisfying and cost-violating successor states and provides a direct mechanism for incorporating constraint information into the reward value used for policy extraction. Following Section 3, ℓ denotes the CMDP cost budget, with $\ell = \kappa$ in the standard benchmark setting. Alternative internal threshold instantiations are evaluated only in Section 5.3.

Using the learned cost value $\widehat{V}_c$, we define the estimated low-cost and high-cost regions as

$$\widehat{\mathcal{S}}_{\text{low}}(\ell) = \{s \in \mathcal{S} : \widehat{V}_c(s) \leq \ell\}, \qquad \widehat{\mathcal{S}}_{\text{high}}(\ell) = \mathcal{S} \setminus \widehat{\mathcal{S}}_{\text{low}}(\ell). \tag{5}$$

For a transition (s, a, r, s'), SAVA modifies the reward Bellman target according to the predicted cost of the successor state.

$$\mathcal{T}_\ell^r(s, a, s') = \begin{cases} r(s,a) + \gamma V^r(s'), & s' \in \widehat{\mathcal{S}}_{\text{low}}(\ell), \\ P_{\text{pen}}, & s' \in \widehat{\mathcal{S}}_{\text{high}}(\ell), \end{cases} \tag{6}$$

where $P_{\text{pen}} < 0$ is a fixed penalized reward target. For high-cost successor states, SAVA replaces the standard reward Bellman target with a fixed penalized target instead of only modifying the bootstrap term. Its complete future return is replaced by a fixed negative value.

SAVA learns the penalized reward value functions (Q^r, V^r) through the resulting Bellman error:

$$\mathcal{L}_{Q^r} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(\mathbf{1}[\widehat{V}_c(s') \leq \ell]\big(r(s,a) + \gamma V^r(s')\big) + \mathbf{1}[\widehat{V}_c(s') > \ell]P_{\text{pen}} - Q^r(s,a)\right)^2\right]. \tag{7}$$

The corresponding expectile value loss is

$$\mathcal{L}_{V^r} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\mathcal{L}_2^\tau\big(Q^r(s,a) - V^r(s)\big)\right]. \tag{8}$$

The cost critic is learned separately with in-sample value estimation:

$$\mathcal{L}_{Q^c} = \mathbb{E}_{(s,a,c,s')\sim\mathcal{D}}\left[\left(c(s,a) + \gamma\widehat{V}_c(s') - \widehat{Q}_c(s,a)\right)^2\right], \tag{9}$$

and

$$\mathcal{L}_{V^c} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\mathcal{L}_2^\tau\big(\widehat{V}_c(s) - \widehat{Q}_c(s,a)\big)\right]. \tag{10}$$

The learned long-horizon cost values determine when the standard reward Bellman target is replaced, allowing constraint information to directly influence the reward value used for policy extraction.

4.2 Analysis of Region-Aware Value Learning

We theoretically analyze the effect of the region-aware penalty on the reward objective and the optimality of the extracted policy. Specifically, we explore two questions: (1) whether a sufficiently large penalty coefficient C can eliminate reward preferences that violate safety constraints; and (2) how much performance deviation is introduced between the penalized objective and the optimal safe policy. The proof details of these theoretical results are provided in Appendix A.

Let Π_{safe} denote the set of policies satisfying the safety constraint, and let $\pi^\star$ denote the optimal safe policy, $\pi^\star \in \arg\max_{\pi\in\Pi_{\text{safe}}} J_r(\pi)$. Let the discounted occupancy measure of the high-cost region under policy π be

$$\eta^\pi = \mathbb{E}_\pi\left[\sum_{t=0}^{T-1}\gamma^t\mathbf{1}(s_t \in \mathcal{S}_{\text{high}})\right]. \tag{11}$$

The following proposition gives a sufficient condition on the penalty coefficient under which every safe policy achieves a higher penalized return than any unsafe policy.

Proposition 4.1. *For any safe policy $\pi_s \in \Pi_{\text{safe}}$ and unsafe policy $\pi_u \notin \Pi_{\text{safe}}$, the penalized reward objective* (7) *satisfies*

$$J_{\bar{r}}(\pi_s) > J_{\bar{r}}(\pi_u) \tag{12}$$

when the penalty coefficient satisfies

$$C > \frac{\sum_{t=0}^{T-1} \gamma^t (r_{\max} - r_{\min})}{\ell}. \tag{13}$$

Proposition 4.1 shows that the proposed penalty directly modifies the reward-side optimization landscape such that unsafe policies cannot dominate safe policies by exploiting reward advantages. The proof is provided in Appendix A.2. Thus, the safety constraint is incorporated directly into the reward objective before policy extraction, rather than being addressed solely through post-hoc correction. We further characterize the performance gap between the policy optimized under the penalized objective and the optimal safe policy.

Proposition 4.2. *Suppose that the region-aware objective is pessimistic, i.e., $J_{\bar{r}}(\pi) \le J_r(\pi)$ for any policy π. Let $\pi^\star_C \in \arg\max_{\pi \in \Pi_\theta} J_{\bar{r}}(\pi)$ be the optimal policy under the penalized objective* (7)*, and $\widehat{\pi}$ be the policy extracted from the learned penalized value function. be the policy extracted from the learned penalized value function.If the reward is bounded by $r(s,a) \le r_{\max}$, and if the learned value function and policy extraction introduce errors ϵ_Q and ϵ_{ext}, respectively, then the performance gap satisfies*

$$J_r(\pi^\star) - J_r(\widehat{\pi}) \le (C + r_{\max})\eta^{\pi^\star} + \frac{2\epsilon_Q + \epsilon_{\text{ext}}}{1-\gamma}. \tag{14}$$

Proposition 4.2 characterizes the reward gap introduced by the region-aware penalty. Importantly, the penalty-induced loss depends only on the discounted occupancy of the estimated high-cost region under the optimal safe policy, rather than the entire trajectory. Therefore, when the optimal safe policy avoids the estimated high-cost region, i.e., $\eta^{\pi^\star} = 0$, the proposed objective introduces no additional reward distortion apart from value estimation and extraction errors. The proof is provided in Appendix A.3.

4.3 Practical Critic Estimation and Policy Extraction

Section 4.1 defines the practical penalized reward-side construction, while Sections 4.2 characterize the underlying penalization principle at the policy level. We now describe the finite-data critic estimation and policy extraction used in the practical algorithm.

Multi-step penalty propagation. With a one-step target, a delayed high-cost successor must pass through several recursive function approximations before it affects an earlier decision. To reduce this propagation depth, SAVA uses an h-step return and evaluates the learned cost value at the bootstrap state s_{t+h}. Let $R_t^{(h)} = \sum_{k=0}^{h-1} \gamma^k r_{t+k}$. The practical reward target applies the same replacement principle as Section 4.1:

$$y_{r,t}^{(h)} = \begin{cases} R_t^{(h)} + d_t^{(h)} \gamma^h V^r(s_{t+h}), & \widehat{V}_c(s_{t+h}) \le \ell, \\ P_{\text{pen}}, & \widehat{V}_c(s_{t+h}) > \ell, \end{cases} \tag{15}$$

where $d_t^{(h)}$ indicates whether bootstrapping remains valid at the end of the segment. Thus, a high-cost terminal state replaces the complete multi-step reward target, while a low-cost terminal state retains the ordinary reward return and bootstrap. The multi-step construction reduces the number of recursive approximation points through which delayed penalized consequences must propagate.

Guided policy extraction. To represent potentially multimodal dataset-supported actions, SAVA uses a conditional flow actor with classifier-free guidance (CFG). Let $o = \varnothing$ denote the unconditional branch and $o = o^\star$ the preferred branch. At inference, CFG combines the two vector fields as

$$v_\theta^{\text{cfg}}(u,x,s) = v_\theta(u,x,s,\varnothing) + \omega\left[v_\theta(u,x,s,o^\star) - v_\theta(u,x,s,\varnothing)\right], \tag{16}$$

where ω controls guidance toward preferred action distribution. The guided flow generates an action sequence, of which only the first action is executed before replanning.

For states already classified as high-cost, candidate decoding additionally uses the local recovery residual $A_c^{\text{rec}}(s,a) = \widehat{V}_c(s) - \widehat{Q}_c(s,a)$ to rank actions within the CFG-generated support. Full flow training, conditioning, and decoding details are provided in Appendix C.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks and metrics. We first test whether SAVA improves reward while satisfying the standard benchmark budget, using the internal threshold $\ell = \kappa$. We evaluate SAVA on Safety-Gymnasium (Ray et al., 2019), Bullet-Safety (Gronauer, 2022), and MetaDrive tasks from the DSRL benchmark (Liu et al., 2023a). MetaDrive tasks follow the same normalized cost evaluation protocol, with their benchmark-specific budgets provided by the DSRL task definitions. Performance is measured by normalized return and normalized cost:

$$R = \frac{R_\pi - R_{\min}}{R_{\max} - R_{\min}}, \qquad C = \frac{C_\pi}{\kappa}, \tag{17}$$

where R_π and C_π are policy return and cost, $R_{\max}$ and $R_{\min}$ are dataset return statistics, and κ is the benchmark budget. Following FISOR (Zheng et al., 2024), we compare reward only among methods satisfying the normalized safety criterion $C \leq 1$. For Bullet-Safety and Safety-Gymnasium, all methods are evaluated under the same budgets, $\kappa = 5$ and $\kappa = 10$, respectively. These budgets are more stringent than the $\kappa = 40$ setting adopted in several prior studies (Xu et al., 2022; Liu et al., 2023b; Chemingui et al., 2025).

Baselines. We compare SAVA with BC and five representative OSRL methods: CPQ (Xu et al., 2022), COptiDICE (Lee et al., 2022), CDT (Liu et al., 2023b), FISOR (Zheng et al., 2024), and CAPS(IQL) (Chemingui et al., 2025), covering pessimistic value learning, constrained distribution optimization, sequence modeling, feasibility-guided learning, and policy switching. Implementation sources, hyperparameters, and evaluation details are provided in Appendix D.

5.2 MAIN RESULTS

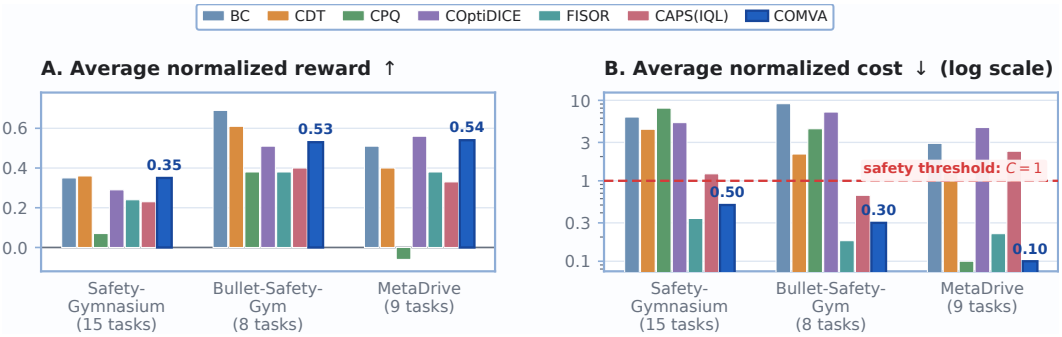


Figure 2: **Benchmark-level summary of the normalized DSRL results.** Bars show the averages normalized reward and cost over Safety-Gymnasium (15 tasks), Bullet-Safety-Gym (8 tasks), and MetaDrive (9 tasks). Lower normalized cost is better, and the dashed line marks the common safety criterion $C = 1$. Detailed per-task results are reported in Appendix F.1.

Main results. SAVA satisfies the safety criterion $C \leq 1$ on all 32 tasks and achieves the highest normalized return on 27 of them. Figure 2 further shows that SAVA achieves the highest benchmark-average reward among methods whose average normalized cost satisfies $C \leq 1$ on all three benchmark suites. Its average normalized costs are 0.50, 0.30, and 0.10 on Safety-Gymnasium, Bullet-Safety-Gym, and MetaDrive, respectively. Several baselines achieve higher average reward on individual suites, but only with average normalized cost above the safety criterion. The complete 32-task results are provided in Appendix F.1, Table 9.

5.3 INTERNAL THRESHOLD CALIBRATION UNDER TERMINAL-ONLY VALUE LEARNING

We study how to instantiate the internal threshold in Section 4.1 when cost values are learned with terminal-only backups. The external budget and metric remain κ and C_π/κ. Some prior protocols

Table 1: **Comparison of termination-and-threshold protocols.** Results are reported as normalized reward and cost under the same external safety criterion. The timeouts + terminals and terminals-only settings use Eq. (18) and Eq. (19), respectively.

Safety-Gymnasium					*Bullet-Safety-Gym*				
	timeouts + terminals		terminals only			timeouts + terminals		terminals only	
Task	reward ↑	cost ↓	reward ↑	cost ↓	Task	reward ↑	cost ↓	reward ↑	cost ↓
CarButton1	0.14	0.48	0.15	0.96	AntRun	0.58	0.39	0.62	0.53
CarButton2	0.09	0.20	0.07	0.25	BallRun	0.30	0.00	0.30	0.00
CarPush1	0.20	0.85	0.29	0.36	CarRun	0.92	0.66	0.93	0.57
CarPush2	0.13	0.68	0.15	0.52	DroneRun	0.53	0.00	0.59	0.26
CarGoal1	0.19	0.55	0.42	0.57	AntCircle	0.23	0.15	0.32	0.23
CarGoal2	0.37	0.49	0.32	0.67	BallCircle	0.33	0.00	0.51	0.49
PointGoal1	0.38	0.28	0.36	0.20	CarCircle	0.38	0.00	0.52	0.11
PointGoal2	0.31	0.48	0.48	0.95	DroneCircle	0.41	0.00	0.52	0.32
AntVel	0.97	0.00	0.92	0.88					
SwimmerVel	0.67	0.00	0.55	0.12					

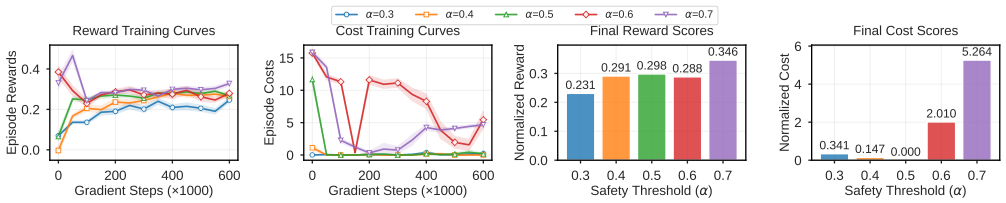


Figure 3: **Effect of the internal α-quantile threshold on *BallRun*.** Larger α expands the estimated low-cost region, while the external budget κ remains unchanged.

(Chemingui et al., 2025; Xu et al., 2022; Liu et al., 2023b) treat both timeouts and genuine terminals as done signals, yielding the T-step threshold

$$\ell = \frac{\kappa(1-\gamma^T)}{(1-\gamma)T}, \tag{18}$$

where T is the episode length. Under terminal-only backups, time limits do not stop bootstrapping, so this finite-horizon rule no longer matches the learned long-horizon cost value.

Data-dependent calibration. Because finite data, function approximation, and bootstrapping can assign small positive values to states associated with low-cost behavior, we define

$$\ell_\alpha = \mathcal{Q}_\alpha\left(\left\{\widehat{V}_c(s) \mid s \sim \mathcal{D}\right\}\right), \tag{19}$$

where $\mathcal{Q}_\alpha$ is the empirical α-quantile. Smaller α yields a stricter partition, whereas larger α expands the estimated low-cost region. This changes only the internal partition and leaves κ unchanged.

Table 1 compares the timeout-and-terminal setting with the terminal-only setting using their corresponding threshold definitions. Both settings satisfy the external safety criterion $C \leq 1$ on all 18 evaluated tasks. The terminal-only setting achieves higher reward on 12 tasks, lower reward on 5 tasks, and the same reward on 1 task. This improvement is particularly consistent on Bullet-Safety-Gym, where reward improves on 7 of 8 tasks and remains unchanged on the remaining task. The terminal-only setting generally incurs higher normalized cost, increasing the average from 0.289 to 0.444.

Figure 3 examines the effect of the quantile level α on BallRun task. Increasing α expands the estimated low-cost region and therefore reduces the frequency of penalized target replacement. Moderate values ($\alpha = 0.3$–0.5) maintain $C \leq 1$, while improving reward from 0.231 to 0.298. In contrast, $\alpha = 0.6$ and 0.7 increase normalized cost to 2.010 and 5.264, respectively, violating the safety criterion. These results illustrate the expected trade-off: an overly small quantile can make the partition unnecessarily restrictive, whereas an overly large quantile admits too many high-cost states into the low-cost region.

Shared quantile level. To test whether a single quantile level can be shared across tasks, we use $\alpha_{\text{fix}} = 0.8$ across all tasks in the 18-task calibration subset. The quantile level is shared, but each

Table 3: **Component and actor ablations (normalized reward / cost).** Gray: unsafe; bold: safe ($C \leq 1$).

Task	No $\overline{V}_r$	No recovery	Gaussian AWR	Ours
BallRun	0.94 / 14.69	**0.29 / 0.09**	**0.08 / 0.05**	**0.30 / 0.00**
AntRun	0.62 / 5.73	0.61 / 1.16	**0.63 / 0.43**	**0.62 / 0.53**
AntCircle	0.58 / 13.00	**0.38 / 0.20**	0.34 / 3.12	**0.32 / 0.23**
DroneRun	0.71 / 13.28	**0.60 / 0.94**	**0.47 / 0.60**	**0.59 / 0.26**
DroneCircle	0.85 / 18.50	0.53 / 1.50	**0.47 / 0.75**	**0.52 / 0.32**
CarGoal1	0.59 / 1.18	**0.35 / 0.48**	**0.33 / 0.62**	**0.42 / 0.57**
CarGoal2	0.36 / 2.41	**0.30 / 0.37**	**0.20 / 0.32**	**0.32 / 0.67**

Table 4: **Performance with critic backup horizon $h_{\text{critic}} = 1$ (normalized reward / cost).** † denotes $C > 1$.

Safety-Gymnasium			*Bullet-Safety-Gym*		
Task	R ↑	C ↓	Task	R ↑	C ↓
CarButton1	0.12	0.70	AntRun	0.65	0.50
CarButton2	0.02	0.81	BallRun	0.11	0.00
CarPush1	0.27	0.59	CarRun	0.92	0.33
CarPush2	0.13	0.29	DroneRun	0.58	0.00
CarGoal1	0.20	0.76	AntCircle	0.21	0.00
CarGoal2	0.14	0.86	BallCircle	0.45	0.00
PointGoal1	0.40	0.24	CarCircle	0.35	0.00
PointGoal2	0.17	0.12	DroneCircle	0.45	0.00
AntVel	0.93	0.89			
SwimmerVel	0.61	$8.46^{\dagger}$			
Safe: $17/18$	$\bar{R} = 0.37$	$\bar{C} = 0.81$			

threshold remains data dependent because it is computed from the corresponding learned cost-value distribution. Table 2 shows that the shared setting satisfies the benchmark criterion on 17 of 18 tasks. This result shows that a shared quantile level remains broadly effective across the evaluated calibration subset.

5.4 Ablation and Analysis

We isolate the region-aware penalized value objective introduced in Section 4.1 from recovery guidance and from the policy-extraction and multi-step choices described in Section 4.3. Additional results are provided in Appendix F.

Table 2: **Performance with shared $\alpha = 0.8$.** Thresholds remain data dependent; † denotes $C > 1$.

Safety-Gymnasium		
Task	reward ↑	cost ↓
CarButton1	0.14	0.87
CarButton2	0.08	0.74
CarPush1	0.21	0.88
CarPush2	0.15	0.67
CarGoal1	0.32	0.74
CarGoal2	0.23	0.31
PointGoal1	0.42	0.52
PointGoal2	0.35	0.41
AntVel	0.91	0.56
SwimmerVel	0.66	$2.81^{\dagger}$
Bullet-Safety-Gym		
Task	reward ↑	cost ↓
AntRun	0.62	0.38
BallRun	0.25	0.42
CarRun	0.94	0.51
DroneRun	0.56	0.00
AntCircle	0.18	0.50
BallCircle	0.37	0.00
CarCircle	0.45	0.00
DroneCircle	0.41	0.00
Safe: $17/18$	$\bar{R} = 0.40$	$\bar{C} = 0.57$

Table 3 provides three complementary controls. Removing $\overline{V}_r$ causes constraint violations on all 7 evaluated tasks even though the flow-based policy extractor is retained, showing that actor expressiveness alone cannot explain SAVA's safety gains. Removing residual recovery retains safety on 5 of the 7 tasks. However, the violations on *AntRun* and *DroneCircle* suggest that recovery improves robustness under local cost-value uncertainty. Finally, replacing the flow actor with a Gaussian AWR actor retains safety on 6 of the 7 tasks, although reward and robustness decrease. Together, these controls identify the region-aware penalized reward value as the primary empirical source of the observed safety alignment among the evaluated variants, while recovery provides additional finite-data robustness and the flow-based actor serves as an expressive policy extractor.

Penalized value landscape. Figure 4 visualizes V_r, V_c, and $\overline{V}_r$ over the same states and shared UMAP coordinates on *BallRun*. High-reward states can also have high cost, whereas $\overline{V}_r$ suppresses these states while preserving reward variation in the estimated low-cost region. Similar low penalized values in estimated high-cost regions also motivate the use of a separate recovery signal for local action ranking.

Critic backup horizon. We further evaluate SAVA with the critic backup horizon reduced to $h_{\text{critic}} = 1$, while fixing the actor sequence horizon to $h_{\text{actor}} = 5$. The residual recovery rule, internal threshold, residual candidate decoding, optimization budget, and evaluation protocol are kept unchanged. Table 4 shows that SAVA with $h_{\text{critic}} = 1$ satisfies the benchmark criterion on 17 of 18 evaluated tasks. The strong safety performance under one-step backups supports treating the

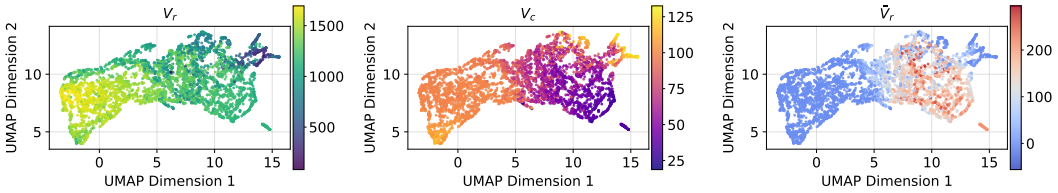


Figure 4: **Penalized value landscape on BallRun.** UMAP visualization of V_r, V_c, and $\overline{V}_r$ on 5,000 states from *BallRun*. All panels use the same states and projection coordinates.

multi-step critic backup as a complementary estimation choice rather than the primary source of the observed safety alignment.

Policy extraction choice. We further isolate the effect of the policy-extraction rule while keeping the region-aware penalized reward objective fixed. In addition to the Gaussian AWR actor in Table 3, we consider *AWR-Flow*, which retains the flow-matching policy parameterization but replaces the CFG-based extraction rule with AWR-style weighting. Implementation details are provided in Appendix F.4.

As shown in Table 5, *AWR-Flow* satisfies the safety criterion on 10 of the 12 evaluated tasks, whereas SAVA satisfies it on all 12. Moreover, SAVA achieves higher normalized reward on 11 of the 12 tasks. Together with the Gaussian AWR control in Table 3, these results indicate that the region-aware penalized reward value provides the primary safety-alignment signal, while the policy-extraction choice affects finite-data robustness and reward performance.

Table 5: **Comparison of policy extraction strategies.** Each entry reports normalized reward / normalized cost.

Task	*AWR-Flow* reward ↑ / cost ↓	SAVA reward ↑ / cost ↓
PointButton1	0.12 / 0.82	0.15 / 0.74
PointButton2	0.20 / 1.04	0.16 / 0.63
PointGoal1	0.29 / 0.86	0.36 / 0.20
PointGoal2	0.37 / 1.43	0.48 / 0.95
PointPush1	0.11 / 0.70	0.19 / 0.36
PointPush2	0.14 / 0.53	0.20 / 0.25
CarButton1	0.07 / 0.25	0.15 / 0.96
CarButton2	-0.05 / 0.06	0.07 / 0.25
CarGoal1	0.28 / 0.36	0.42 / 0.57
CarGoal2	0.22 / 0.67	0.32 / 0.67
CarPush1	0.17 / 0.48	0.29 / 0.36
CarPush2	0.09 / 0.44	0.15 / 0.52

6 Conclusion

We introduced SAVA, an offline safe RL framework that incorporates cumulative cost directly into reward value learning before policy optimization. SAVA uses a partition induced by the learned cost value to assign transitions entering the estimated high-cost region a fixed penalized reward target, allowing long-horizon cost to influence earlier reward value estimates. Under the penalized objective, the analysis shows that sufficiently strong penalization controls constraint violation, while a safe approximate maximizer has a bounded reward gap relative to the optimal safe policy. Under bounded estimation errors, conservative constraint tightening yields satisfaction of the original cost constraint with a controlled reward gap. On DSRL, SAVA achieves the highest returns in 27 of 32 tasks under the common normalized cost criterion. Ablations identify the region-aware penalized reward value as the key component for constraint satisfaction among the evaluated variants, while recovery, flow matching, and multi-step backups play supporting roles in robustness, policy extraction, and critic estimation.

AI use statement

Generative AI tools were used during manuscript preparation to assist with language refinement, improving clarity and readability, drafting and revising limited portions of the text, LaTeX and presentation-related formatting, sourcing information, limited code editing and refinement, and brainstorming. All AI-assisted content and code were reviewed, edited, and verified by the authors. Generative AI tools were not used to formulate the core research problem, develop the proposed method, formulate the theoretical claims or provide critical ingredients for their proofs, design the

experiments, generate or analyze experimental data, or interpret the experimental results. The scientific contributions, methodological decisions, theoretical development, experimental methodology, and conclusions of this work were developed and validated by the authors. The authors take full responsibility for the final content of this work, including any text, code, or artifacts produced with the assistance of generative AI.

Reproducibility statement

We provide detailed information to facilitate the reproducibility of both the theoretical and empirical results. The assumptions and complete proofs supporting the theoretical results are provided in Appendix A. The experimental protocol and evaluation metrics are described in Section 5.1, with benchmark details and hyperparameters in Appendices B and D.3. Appendix C provides the detailed policy-extraction formulation, including the in-sample value ordering, conditional flow-matching objective, guided sampling procedure, and candidate decoding. Appendix D provides the implementation details, pseudocode, and computational costs. Additional experimental results, ablation studies, visualizations, sensitivity analyses, and the complete per-task main-results table are reported in Appendix F; the detailed table appears in Appendix F.1. An anonymous implementation, including the training and evaluation code, is available at `https://anonymous.4open.science/r/osrl_submission-E850`.

References

Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Per-Arne Andersen, Morten Goodwin, and Ole-Christoffer Granmo. Towards safe reinforcement-learning in industrial grid-warehousing. *Information Sciences*, 537:467–484, 2020.

Aluízio FR Araújo and Arthur PS Braga. Reward-penalty reinforcement learning scheme for planning and reactive behaviour. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 1998.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

Yassine Chemingui, Aryan Deshwal, Honghao Wei, Alan Fern, and Jana Doppa. Constraint-adaptive policy switching for offline safe reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Yuhao Ding and Javad Lavaei. Provably efficient primal-dual reinforcement learning for cmdps with non-stationary objectives and constraints. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *ArXiv*, abs/2505.23458, 2025.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2019.

Milan Ganai, Zheng Gong, Chenning Yu, Sylvia Herbert, and Sicun Gao. Iterative reachability estimation for safe reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. In *International Conference on Learning Representations (ICLR)*, 2023.

Ze Gong, Akshat Kumar, and Pradeep Varakantham. Offline safe reinforcement learning using trajectory classification. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

Sven Gronauer. Bullet-safety-gym: A framework for constrained reinforcement learning. Technical report, mediaTUM, 2022.

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A review of safe reinforcement learning: Methods, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:11216–11235, 2024.

Zijian Guo, Weichao Zhou, Shengao Wang, and Wenchao Li. Constraint-conditioned actor-critic for offline safe reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *ArXiv*, abs/2304.10573, 2023.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *ArXiv*, abs/1606.08415, 2023.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv*, abs/1412.6980, 2014.

Prajwal Koirala, Zhanhong Jiang, Soumik Sarkar, and Cody Fleming. Latent safety-constrained policy approach for safe offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Adam Laud and Gerald DeJong. The influence of reward on the speed of reinforcement learning: An analysis of shaping. In *International Conference on Machine Learning (ICML)*, 2003.

Jongmin Lee, Cosmin Paduraru, Daniel J Mankowitz, Nicolas Heess, Doina Precup, Kee-Eung Kim, and Arthur Guez. Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation. *ArXiv*, abs/2204.08957, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

Qian Lin, Bo Tang, Zifan Wu, Chao Yu, Shangqin Mao, Qianlong Xie, Xingxing Wang, and Dong Wang. Safe offline reinforcement learning with real-time budget constraints. In *International Conference on Machine Learning (ICML)*, 2023.

Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, and Ding Zhao. Datasets and benchmarks for offline safe reinforcement learning. *ArXiv*, abs/2306.09303, 2023a.

Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. Constrained decision transformer for offline safe reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023b.

Yixiu Mao, Hongchang Zhang, Chen Chen, Yi Xu, and Xiangyang Ji. Supported value regularization for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Nilaksh, Abhishek Ranjan, Shreenabh Agrawal, Aayush Jain, Pushpak Jagtap, and Shishir Kolathaya. Barrier functions inspired reward shaping for reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Santiago Paternain, Miguel Calvo-Fullana, Luiz F. O. Chamon, and Alejandro Ribeiro. Safe policies for reinforcement learning via primal-dual methods. *IEEE Transactions on Automatic Control*, 68: 1321–1336, 2023.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. Technical report, OpenAI, 2019.

Tianyu Shi, Dong Chen, Kaian Chen, and Zhaojian Li. Offline reinforcement learning for autonomous driving with safety and exploration enhancement. *ArXiv*, abs/2110.07067, 2021.

Garrett Thomas, Yuping Luo, and Tengyu Ma. Safe reinforcement learning by imagining the near future. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Zifan Wu, Bo Tang, Qian Lin, Chao Yu, Shangqin Mao, Qianlong Xie, Xingxing Wang, and Dong Wang. Off-policy primal-dual safe reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Chenjun Xiao, Han Wang, Yangchen Pan, Adam White, and Martha White. The in-sample softmax for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. In *International Conference on Learning Representations (ICLR)*, 2023.

Yihang Yao, Zhepeng Cen, Wenhao Ding, Haohong Lin, Shiqi Liu, Tingnan Zhang, Wenhao Yu, and Ding Zhao. OASIS: Conditional distribution shaping for offline safe reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Donghao Ying, Yunkai Zhang, Yuhao Ding, Alec Koppel, and Javad Lavaei. Scalable primal-dual actor-critic method for safe multi-agent rl with general utilities. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

Qin Zhang, Linrui Zhang, Haoran Xu, Li Shen, Bowen Wang, Yongzhe Chang, Xueqian Wang, Bo Yuan, and Dacheng Tao. Saformer: A conditional sequence modeling approach to offline safe reinforcement learning. *ArXiv*, abs/2301.12203, 2023.

Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. In *International Conference on Learning Representations (ICLR)*, 2024.

A Additional Formulation and Proofs for the Theoretical Results

This appendix provides the proofs of Propositions 4.1 and 4.2 from Section 4.2.

A.1 Additional Problem Formulation

Offline safe RL learns from a fixed dataset

$$\mathcal{D} = \left\{ \left(s_t^{(i)}, a_t^{(i)}, r_t^{(i)}, c_t^{(i)}, s_{t+1}^{(i)} \right)_{t=0}^{T-1} \right\}_{i=0}^{N-1}, \tag{20}$$

containing N trajectories collected by a behavior policy π_β. In addition to the CMDP constraint in Eq. (2), a common behavior-regularized offline formulation is

$$\begin{aligned} \max_{\pi} \quad & J_r(\pi) \\ \text{s.t.} \quad & J_c(\pi) \le \ell, \qquad D(\pi \| \pi_\beta) \le \epsilon, \end{aligned} \tag{21}$$

where $D(\pi \| \pi_\beta)$ regularizes deviation from the offline behavior distribution.

For the theoretical analysis in Section 4, let $\mathcal{S}_{\text{high}}$ denote the high-cost region associated with the penalized objective. The discounted occupancy measure of high-cost region under a policy is defined as

$$\eta^\pi := \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t \mathbf{1}\{s_t \in \mathcal{S}_{\text{high}}\} \right]. \tag{22}$$

We assume that the high-cost region representation is consistent with the policy-level safety constraint in the sense that

$$\eta^{\pi_s} = 0 \quad \forall\, \pi_s \in \Pi_{\text{safe}}, \qquad \eta^{\pi_u} \ge \ell \quad \forall\, \pi_u \notin \Pi_{\text{safe}}. \tag{23}$$

This assumption formalizes the separation used in Proposition 4.1: safe policies do not enter the high-cost region, whereas a policy violating the safety constraint incurs at least ℓ discounted occupancy.

For the corresponding policy-level penalized objective, assume that the reward is bounded as

$$r_{\min} \le r(s, a) \le r_{\max}. \tag{24}$$

Given a penalty magnitude $C > 0$, define the penalized reward as

$$\bar{r}(s_t, a_t) = \begin{cases} r(s_t, a_t), & s_t \notin \mathcal{S}_{\text{high}}, \\ -C, & s_t \in \mathcal{S}_{\text{high}}, \end{cases} \tag{25}$$

with the associated objective

$$J_{\bar{r}}(\pi) := \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t \bar{r}(s_t, a_t) \right]. \tag{26}$$

To compare the penalized optimum with the optimal safe policy within the policy class accessible to offline policy extraction, define

$$\Pi_{\text{safe},\theta} := \{ \pi \in \Pi_\theta : J_c(\pi) \le \ell \}. \tag{27}$$

The optimal safe policy and the optimal policy under the penalized objective are respectively

$$\pi^\star \in \arg \max_{\pi \in \Pi_{\text{safe},\theta}} J_r(\pi), \qquad \pi_C^\star \in \arg \max_{\pi \in \Pi_\theta} J_{\bar{r}}(\pi). \tag{28}$$

A.2 Proof of Proposition 4.1

Proof. For any safe policy $\pi_s \in \Pi_{\text{safe}}$, the accumulated reward is lower bounded by

$$J_r(\pi_s) \ge \sum_{t=0}^{T-1} \gamma^t r_{\min}. \tag{29}$$

Since π_s never enters the high-cost region, its penalized reward satisfies

$$J_{\bar{r}}(\pi_s) = J_r(\pi_s) \geq \sum_{t=0}^{T-1} \gamma^t r_{\min}. \tag{30}$$

For any unsafe policy $\pi_u \notin \Pi_{\text{safe}}$, the total reward before violating the constraint is upper bounded by $\sum_{t=0}^{T-1} \gamma^t r_{\max}$. Moreover, because π_u violates the safety constraint, its expected discounted occupancy measure of the high-cost region satisfies $\eta^{\pi_u} \geq \ell$. Therefore, the penalized reward value of π_u is upper bounded by

$$J_{\bar{r}}(\pi_u) \leq \sum_{t=0}^{T-1} \gamma^t r_{\max} - C\ell. \tag{31}$$

When

$$C > \frac{\sum_{t=0}^{T-1} \gamma^t (r_{\max} - r_{\min})}{\ell}, \tag{32}$$

we have

$$\sum_{t=0}^{T-1} \gamma^t r_{\max} - C\ell < \sum_{t=0}^{T-1} \gamma^t r_{\min}. \tag{33}$$

Combining Eq. 30 and Eq. 31 gives

$$J_{\bar{r}}(\pi_s) > J_{\bar{r}}(\pi_u). \tag{34}$$

Therefore, the penalized reward objective always prefers the safe policy over any unsafe policy when the penalty coefficient satisfies the above condition. □

A.3 Proof of Proposition 4.2

Proof. Let $\pi_C^\star \in \arg\max_{\pi \in \Pi_\theta} J_{\bar{r}}(\pi)$ denote the optimal policy under the true penalized objective, and assume $\pi^\star \in \Pi_\theta$. We decompose

$$\begin{aligned} J_r(\pi^\star) - J_r(\widehat{\pi}) = &\big(J_r(\pi^\star) - J_{\bar{r}}(\pi^\star)\big) + \big(J_{\bar{r}}(\pi^\star) - J_{\bar{r}}(\pi_C^\star)\big) \\ &+ \big(J_{\bar{r}}(\pi_C^\star) - J_{\bar{r}}(\widehat{\pi})\big) + \big(J_{\bar{r}}(\widehat{\pi}) - J_r(\widehat{\pi})\big). \end{aligned} \tag{35}$$

For the region-aware penalized objective, let $\eta^\pi := \mathbb{E}_\pi\Big[\sum_{t\geq 0} \gamma^t \mathbf{1}\{s_t \in \mathcal{S}_{\text{high}}\}\Big]$. Since the original reward is replaced by $-C$ on penalized transitions and $r(s,a) \leq r_{\max}$,

$$\begin{aligned} J_r(\pi^\star) - J_{\bar{r}}(\pi^\star) &= \mathbb{E}_{\pi^\star}\left[\sum_{t\geq 0} \gamma^t \mathbf{1}\{s_t \in \mathcal{S}_{\text{high}}\}\big(r(s_t,a_t) + C\big)\right] \\ &\leq (C + r_{\max})\eta^{\pi^\star}. \end{aligned} \tag{36}$$

Moreover, because $\pi_C^\star$ maximizes $J_{\bar{r}}$ over the same policy class containing $\pi^\star$, $J_{\bar{r}}(\pi^\star) - J_{\bar{r}}(\pi_C^\star) \leq 0$.

Next, assume that the learned penalized action-value function satisfies $\|\widehat{Q}_{\bar{r}} - Q_{\bar{r}}^\star\|_\infty \leq \epsilon_Q$, and that $\widehat{\pi}$ is ϵ_{ext}-greedy with respect to $\widehat{Q}_{\bar{r}}$, i.e.,

$$\max_a \widehat{Q}_{\bar{r}}(s,a) - \mathbb{E}_{a\sim\widehat{\pi}(\cdot|s)}[\widehat{Q}_{\bar{r}}(s,a)] \leq \epsilon_{\text{ext}}, \qquad \forall s, \tag{37}$$

where the maximization is over the admissible actions associated with the same policy class. Extending the classical approximate-greedy performance-loss argument (**?**), for every s,

$$\begin{aligned} V_{\bar{r}}^\star(s) = \max_a Q_{\bar{r}}^\star(s,a) &\leq \max_a \widehat{Q}_{\bar{r}}(s,a) + \epsilon_Q \\ &\leq \mathbb{E}_{a\sim\widehat{\pi}(\cdot|s)}[\widehat{Q}_{\bar{r}}(s,a)] + \epsilon_Q + \epsilon_{\text{ext}} \\ &\leq \mathbb{E}_{a\sim\widehat{\pi}(\cdot|s)}[Q_{\bar{r}}^\star(s,a)] + 2\epsilon_Q + \epsilon_{\text{ext}}. \end{aligned} \tag{38}$$

Using the Bellman equations for $Q_{\bar{r}}^\star$ and $V_{\bar{r}}^{\widehat{\pi}}$, we obtain

$$V_{\bar{r}}^\star(s) - V_{\bar{r}}^{\widehat{\pi}}(s) \leq 2\epsilon_Q + \epsilon_{\text{ext}} + \gamma \mathbb{E}_{s'\sim P,\, a\sim\widehat{\pi}}\Big[V_{\bar{r}}^\star(s') - V_{\bar{r}}^{\widehat{\pi}}(s')\Big]. \tag{39}$$

Taking the supremum over s yields

$$\|V_{\bar{r}}^{\star} - V_{\bar{r}}^{\widehat{\pi}}\|_{\infty} \leq 2\epsilon_Q + \epsilon_{\text{ext}} + \gamma\|V_{\bar{r}}^{\star} - V_{\bar{r}}^{\widehat{\pi}}\|_{\infty}, \tag{40}$$

and therefore

$$J_{\bar{r}}(\pi_C^{\star}) - J_{\bar{r}}(\widehat{\pi}) \leq \frac{2\epsilon_Q + \epsilon_{\text{ext}}}{1-\gamma}. \tag{41}$$

Finally, by the assumed pessimism of the region-aware objective, $J_{\bar{r}}(\widehat{\pi}) \leq J_r(\widehat{\pi})$, so $J_{\bar{r}}(\widehat{\pi}) - J_r(\widehat{\pi}) \leq 0$. Substituting these bounds into Eq. (35) gives

$$J_r(\pi^{\star}) - J_r(\widehat{\pi}) \leq (C + r_{\max})\eta^{\pi^{\star}} + \frac{2\epsilon_Q + \epsilon_{\text{ext}}}{1-\gamma}, \tag{42}$$

which completes the proof. □

B Benchmark Details

B.1 Safety Gymnasium

Safety-Gymnasium (Ray et al., 2019), a MuJoCo-based benchmark for safe RL, evaluates the trade-off between performance and safety through two categories of environments. The first involves obstacle avoidance tasks performed by a *Car* agent across three scenarios (*Goal*, *Button*, and *Push*), each with two difficulty levels (1 and 2). These tasks represent an important class of real-world reinforcement learning problems in which an agent must move through an environment and interact with objects to achieve a goal. The agent receives rewards for task completion and penalties for contacting hazards. These environments are named in the format {*Agent*}{*Task*}{*Difficulty*} (e.g., *CarGoal1*). The second category consists of velocity-constrained environments featuring three agents (*Ant*, *HalfCheetah*, and *Swimmer*), where the objective is to maximize forward movement reward while adhering to safe speed limits. This tests the ability to balance performance with precise control to avoid failures due to overspeed. Environments in this category follow the naming scheme AgentVel (e.g., *AntVelocity*).

B.2 Bullet Safey Gym

Bullet-Safety-Gym (Gronauer, 2022) is a safe RL benchmark based on the PyBullet engine. Unlike Safety-Gymnasium, it uses shorter time horizons to enable faster training, serving as its efficient complementary alternative, with a broader variety of robotic agents, including *Ball*, *Car*, *Drone*, and *Ant*. The task settings remain relatively straightforward, offering two main types: *Circle* and *Run*. Each environment is named concisely using the convention {*Agent*}{*Task*}, (e.g., *AntCircle*).

B.3 MetaDrive

MetaDrive provides autonomous-driving tasks in the DSRL benchmark. These tasks evaluate safe navigation under interactive driving scenarios, where the policy must trade off task completion reward against collision-related costs. We include MetaDrive tasks following the same normalized reward and cost evaluation protocol described in Section 5.1.

B.4 Composition of the Dataset

The datasets used in our evaluations predominantly contain a mix of safe and unsafe trajectories. The safety compliance of policies during evaluation is determined by their adherence to the undiscounted cost-return threshold. We provide further details regarding the dataset composition.

Figure 5 and Figure 6 show the reward and cost returns of trajectories in the dataset, with each point representing one pre-collected trajectory. Most datasets used in our evaluation contain both safe and unsafe trajectories. For Bullet-Safety-Gym, the red dashed line at cost 10. The benchmark evaluation uses $\kappa = 5$, and all reported normalized costs are computed using the external criterion C_π/κ.

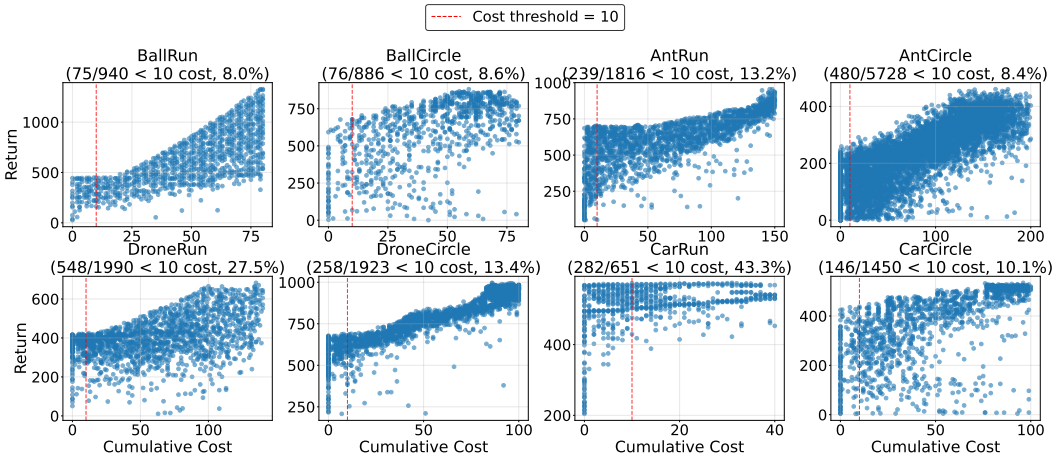


Figure 5: Dataset Composition Overview for Bullet-Safety-Gym Tasks.

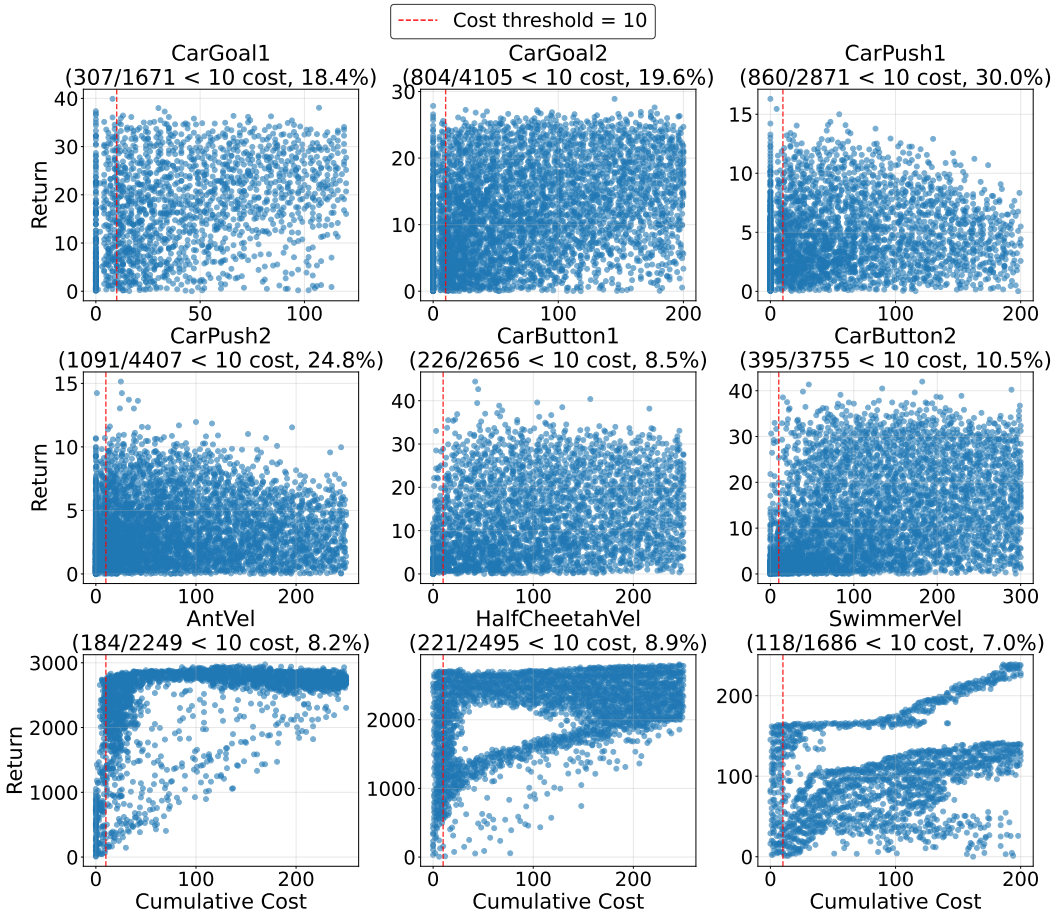


Figure 6: Dataset Composition Overview for Safety-Gymnasium Tasks.

B.5 Baseline Methods

We compare SAVA with behavior cloning (BC) and five representative offline safe RL methods. For completeness, we summarize below the main mechanism and optimization objective of each baseline. We use $\mathcal{D}$ to denote the offline dataset and κ to denote the cost budget when needed.

Behavior Cloning (BC). BC directly fits the behavior distribution represented by the offline dataset without explicitly optimizing reward or cost. For a stochastic policy π_θ, the standard maximum-likelihood objective is

$$\mathcal{L}_{\mathrm{BC}}(\theta) = -\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\log \pi_\theta(a \mid s)\right]. \tag{43}$$

BC therefore serves as a behavioral reference that does not explicitly perform reward maximization or constraint-aware policy improvement.

Constraints Penalized Q-Learning (CPQ). CPQ (Xu et al., 2022) learns separate reward and cost critics and uses the cost critic both to penalize out-of-distribution actions and to restrict reward backups to actions estimated to satisfy the constraint. Its cost critic is trained with

$$\min_{Q_c} \ \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(Q_c(s,a) - \mathcal{T}^\pi Q_c(s,a)\right)^2\right] - \alpha\, \mathbb{E}_{s\sim\mathcal{D},\, a\sim\nu(\cdot|s)}\left[Q_c(s,a)\right], \tag{44}$$

where ν is chosen to emphasize actions detected as out-of-distribution. The additional term increases their estimated cost. The reward critic then uses the constraint-penalized Bellman operator

$$\mathcal{T}_P^\pi Q_r(s,a) = r(s,a) + \gamma \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[\mathbb{I}(Q_c(s',a') \le \kappa)\, Q_r(s',a')\right]. \tag{45}$$

Accordingly, policy improvement favors high-reward actions whose estimated cost satisfies the threshold:

$$\max_{\pi} \ \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi(\cdot|s)}\left[\mathbb{I}(Q_c(s,a) \le \kappa)\, Q_r(s,a)\right]. \tag{46}$$

Thus, CPQ incorporates safety through conservative cost estimation, masked reward backups, and cost-filtered policy improvement.

COptiDICE. COptiDICE (Lee et al., 2022) formulates offline constrained RL directly in the space of discounted stationary state–action distributions rather than optimizing an actor and critic separately. Let $d(s,a)$ denote a candidate stationary distribution and $d_{\mathcal{D}}(s,a)$ the empirical dataset distribution. Its basic constrained formulation is

$$\max_{d} \quad \mathbb{E}_{(s,a)\sim d}\left[r(s,a)\right] - \alpha D_f(d \,\|\, d_{\mathcal{D}}), \tag{47}$$

$$\text{s.t.} \quad \mathbb{E}_{(s,a)\sim d}\left[c(s,a)\right] \le \kappa, \tag{48}$$

together with the Bellman-flow constraints required for d to correspond to a valid policy. Defining the stationary-distribution correction

$$w(s,a) = \frac{d(s,a)}{d_{\mathcal{D}}(s,a)} \tag{49}$$

and

$$e_{\lambda,\nu}(s,a) = r(s,a) - \lambda c(s,a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[\nu(s')] - \nu(s), \tag{50}$$

COptiDICE obtains the closed-form correction

$$w^\star_{\lambda,\nu}(s,a) = \left[(f')^{-1}\left(\frac{e_{\lambda,\nu}(s,a)}{\alpha}\right)\right]_+. \tag{51}$$

The resulting policy is recovered from the optimized stationary distribution as

$$\pi^\star(a \mid s) = \frac{d^\star(s,a)}{\sum_{a'} d^\star(s,a')}. \tag{52}$$

COptiDICE further uses a conservative upper estimate of the cost when enforcing the constraint to reduce deployment-time constraint violations caused by offline estimation error.

Constrained Decision Transformer (CDT). CDT (Liu et al., 2023b) formulates offline safe RL as conditional sequence modeling. For a trajectory, it defines the reward-to-go and cost-to-go as

$$R_t = \sum_{t'=t}^{T} r_{t'}, \qquad C_t = \sum_{t'=t}^{T} c_{t'}. \tag{53}$$

The policy conditions on histories of reward returns, cost returns, states, and previous actions,

$$o_t = \{R_{t-K:t}, C_{t-K:t}, s_{t-K:t}, a_{t-K:t-1}\}, \tag{54}$$

and parameterizes a stochastic Gaussian action distribution

$$\pi_\theta(\cdot \mid o_t) = \mathcal{N}\left(\mu_\theta(o_t), \Sigma_\theta(o_t)\right). \tag{55}$$

CDT is trained by minimizing the negative log-likelihood while encouraging policy entropy:

$$\min_\theta \ \mathbb{E}_{(o,a)\sim\mathcal{D}}\left[-\log \pi_\theta(a \mid o) - \lambda_{\text{ent}} \mathcal{H}(\pi_\theta(\cdot \mid o))\right]. \tag{56}$$

In addition, CDT performs Pareto-frontier-oriented return relabeling to augment reward–cost conditioning pairs. At deployment, varying the target cost return allows the same conditional policy to adapt to different constraint levels.

FISOR. FISOR (Zheng et al., 2024) converts hard safety-constrained offline RL into feasibility identification followed by feasibility-dependent policy optimization. It first learns a feasibility value using a reversed expectile objective:

$$\mathcal{L}_{V_h} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_{\text{rev}}\left(Q_h(s,a) - V_h(s)\right)\right], \tag{57}$$

$$\mathcal{L}_{Q_h} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left((1-\gamma)h(s) + \gamma \max\{h(s), V_h(s')\} - Q_h(s,a)\right)^2\right], \tag{58}$$

where $h(s)$ represents the state-wise safety signal and L_{rev} emphasizes lower feasible values. The learned feasibility value determines the largest estimated feasible region through $V_h(s) \le 0$.

Policy optimization then depends on whether the current state lies inside or outside this region. In a compact form, its surrogate objective favors reward advantage in feasible states and reduction of feasibility violation in infeasible states:

$$\max_\pi \ \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\mathbb{I}(V_h^\star(s) \le 0)\, A_r^\star(s,a) - \mathbb{I}(V_h^\star(s) > 0)\, A_h^\star(s,a)\right], \tag{59}$$

with feasible states additionally restricting policy support to actions satisfying

$$Q_h^\star(s,a) \le 0. \tag{60}$$

FISOR realizes this feasibility-dependent policy extraction with an energy-guided diffusion policy.

CAPS(IQL). Constraint-Adaptive Policy Switching (CAPS) (Chemingui et al., 2025) learns a collection of policies spanning different reward–cost trade-offs and selects among their actions at deployment. In the IQL instantiation, separate reward and cost critics produce

$$A_r(s,a) = Q_r(s,a) - V_r(s), \qquad A_c(s,a) = Q_c(s,a) - V_c(s). \tag{61}$$

For scalarization coefficient λ_k, the corresponding policy is extracted using

$$J_{\pi_k}(\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp(\beta\left(A_r(s,a) - \lambda_k A_c(s,a)\right)) \log \pi_\phi^k(a \mid s)\right]. \tag{62}$$

The policy set contains a reward-maximizing policy, a cost-minimizing policy, and intermediate policies obtained from different λ_k.

At timestep t, CAPS first filters the actions proposed by these policies according to the remaining cost budget:

$$\mathcal{F}_t = \{\pi(s_t) \mid Q_t^c(s_t, \pi(s_t)) + c_{\le t} \le \kappa,\ \pi \in \mathcal{P}\}. \tag{63}$$

Among the feasible candidates, it selects

$$a_t^\star = \arg\max_{a\in\mathcal{F}_t} Q_t^r(s_t, a). \tag{64}$$

If $\mathcal{F}_t = \varnothing$, CAPS instead chooses the candidate with the smallest estimated future cost. Thus, CAPS incorporates the safety constraint primarily through deployment-time filtering and switching among a pre-trained set of policies.

C Policy Extraction over Multimodal Data-Supported Actions

This appendix specifies the policy-extraction objectives used in the practical algorithm. We provide the advantage-weighted policy extraction objective, the conditional flow-matching objective, the binary optimality conditioning rule, and the finite-candidate decoding procedure.

C.1 Policy Extraction

IQL (Kostrikov et al., 2022) avoids explicit out-of-distribution action queries by fitting the action value on dataset transitions and estimating a state value through expectile regression:

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a)\right)^2\right], \tag{65}$$

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\mathcal{L}_2^\tau\left(Q_{\hat{\theta}}(s,a) - V_\psi(s)\right)\right]. \tag{66}$$

A standard advantage-weighted regression (AWR) extractor minimizes

$$\mathcal{L}_\pi(\phi) = -\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\eta_{\text{awr}}\left(Q_{\hat{\theta}}(s,a) - V_\psi(s)\right)\right)\log\pi_\phi(a \mid s)\right], \tag{67}$$

where $\eta_{\text{awr}} > 0$ controls the concentration on high-value dataset actions.

For SAVA, policy extraction uses the region-aware penalized value rather than the vanilla reward value. When the preferred data-supported actions are multimodal, a unimodal actor may allocate probability between supported modes. This is a limitation of the policy family rather than of advantage weighting, motivating the expressive extractor defined next.

C.2 Conditional Flow Matching for Multimodal Actions

SAVA uses a conditional flow to represent non-Gaussian and multimodal distributions over data-supported action sequences. Let $\mathbf{a}_t := a_{t:t+h-1} \in \mathbb{R}^{hd_a}$ denote the h-action sequence used by the actor, and set $x^1 = \mathbf{a}_t$. Given a base sample $x^0 \sim \mathcal{N}(0, I)$ and $u \sim \text{Unif}[0,1]$, the linear conditional path is

$$x^u = (1-u)x^0 + ux^1. \tag{68}$$

For a binary condition $o \in \{0,1\}$, the vector field $v_\theta(u, x^u, s, o)$ predicts the conditional displacement $x^1 - x^0$. The corresponding sample-wise flow-matching loss is

$$\ell_{\text{FM}}\big(\theta; s, \mathbf{a}_t, o, x^0, u\big) := \left\|v_\theta(u, x^u, s, o) - (x^1 - x^0)\right\|_2^2. \tag{69}$$

The associated ordinary differential equation is

$$\frac{d}{du}\phi_u(x^0) = v_\theta\big(u, \phi_u(x^0), s, o\big), \qquad \phi_0(x^0) = x^0. \tag{70}$$

Integrating Eq. (70) from $u = 0$ to $u = 1$ produces an action sequence. Only its first action is executed, after which the policy is replanned from the next observed state.

C.3 Region-Aware Preference Conditioning

The implementation uses a binary condition rather than separate positive, negative, and null labels. The branch $o = 0$ is the unconditional in-distribution branch and is trained on every sampled dataset sequence. The branch $o = 1$ is the preferred-sequence branch and is trained only on sequences selected by a region-dependent binary mask.

For an initial state s and dataset action sequence $\mathbf{a}$, define

$$\begin{aligned} w^+(s,\mathbf{a}) = \mathbb{1}\left[\widehat{V}_c(s) \le \ell\right]\mathbb{1}\left[\overline{Q}_r(s,\mathbf{a}) - \overline{V}_r(s) \ge 0\right] \\ + \mathbb{1}\left[\widehat{V}_c(s) > \ell\right]\mathbb{1}\left[\widehat{Q}_c(s,\mathbf{a}) - \widehat{V}_c(s) \le 0\right]. \end{aligned} \tag{71}$$

Thus, estimated low-cost states retain dataset sequences with nonnegative penalized reward advantage. Estimated high-cost states retain sequences whose predicted cost is no larger than the state-level cost baseline, equivalently those with nonnegative recovery residual $\widehat{V}_c(s) - \widehat{Q}_c(s,\mathbf{a})$.

Algorithm 1 Conditional Flow Training

1: **for** each gradient step **do**
2: Sample $(s, \mathbf{a}) \sim \mathcal{D}$, where $\mathbf{a} = a_{t:t+h-1}$.
3: Compute $w^+(s, \mathbf{a})$ using Eq. (71).
4: Sample (x^0, u) and $(\widetilde{x}^0, \widetilde{u})$ independently.
5: Compute $\mathcal{L}_{\text{pref}}$ with condition $o = 1$.
6: Compute $\mathcal{L}_{\text{base}}$ with condition $o = 0$.
7: Update θ using Eq. (74).
8: **end for**

Algorithm 2 Guided Flow Sampling

1: Sample $x_0 \sim \mathcal{N}(0, I)$.
2: **for** $n = 0, \ldots, N-1$ **do**
3: $u_n \leftarrow n/N$, $\Delta u \leftarrow 1/N$.
4: Evaluate v_θ^{cfg} using Eq. (75).
5: $x_{n+1} \leftarrow x_n + \Delta u\, v_\theta^{\text{cfg}}(u_n, x_n, s)$.
6: **end for**
7: **return** x_N.

The preferred and unconditional branches are trained by

$$\mathcal{L}_{\text{pref}}(\theta) = \mathbb{E}\left[w^+(s, \mathbf{a})\, \ell_{\text{FM}}\big(\theta; s, \mathbf{a}, 1, x^0, u\big)\right], \tag{72}$$

$$\mathcal{L}_{\text{base}}(\theta) = \mathbb{E}\left[\ell_{\text{FM}}\big(\theta; s, \mathbf{a}, 0, \widetilde{x}^0, \widetilde{u}\big)\right], \tag{73}$$

where the two expectations may use independent base samples and interpolation times, as in the implementation. The actor objective is

$$\mathcal{L}_{\text{actor}}(\theta) = \mathcal{L}_{\text{pref}}(\theta) + \beta_{\text{base}} \mathcal{L}_{\text{base}}(\theta), \qquad \beta_{\text{base}} = 0.1. \tag{74}$$

At inference, classifier-free guidance combines the preferred and unconditional branches as

$$v_\theta^{\text{cfg}} = v_\theta(u, x, s, 0) + \omega\left[v_\theta(u, x, s, 1) - v_\theta(u, x, s, 0)\right], \tag{75}$$

where ω controls concentration toward the region-dependent preferred sequence distribution.

C.4 Region-Aware Candidate Decoding

The learned flow resolves the multimodal representation problem, but finite offline data leave state-dependent uncertainty in the estimated safety boundary. SAVA therefore generates K candidate action sequences $\{\mathbf{a}^{(k)}\}_{k=1}^K$ and performs region-dependent decoding on their first actions $a_t^{(k)}$. In an estimated low-cost state, we use the penalized reward advantage

$$A^r_{\text{pen}}(s, a) := \overline{Q}_r(s, a) - \overline{V}_r(s). \tag{76}$$

In an estimated high-cost state, the relevant direction is recovery rather than immediate reward. The residual

$$A^c_{\text{rec}}(s, a) := \widehat{V}_c(s) - \widehat{Q}_c(s, a) \tag{77}$$

is positive when the candidate action is predicted to incur lower future cost than the state-level cost baseline. The region-dependent decoding score can therefore be written as

$$\begin{aligned} G_\ell(s, a) = \mathbf{1}\Big[s \in \widehat{\mathcal{S}}_{\text{low}}(\ell)\Big]\, A^r_{\text{pen}}(s, a) \\ + \mathbf{1}\Big[s \in \widehat{\mathcal{S}}_{\text{high}}(\ell)\Big]\, A^c_{\text{rec}}(s, a), \end{aligned} \tag{78}$$

with

$$k^\star \in \arg\max_{1 \le k \le K} G_\ell\big(s, a_t^{(k)}\big). \tag{79}$$

The controller executes $a_t^{(k^\star)}$ and replans at the next state. The selection in Eqs. (78)–(79) is restricted to the generated candidate support rather than the entire continuous action space. This restriction combines local cost-based ordering with the behavior support represented by the flow.

D Implement Details

Our implementation is based on PyTorch (Paszke et al., 2019). All experiments were conducted on a single NVIDIA RTX 4090 GPU. The pseudocode, computational costs, and hyperparameters are detailed in Appendices D.1–D.3.

D.1 PSEUDOCODE

To provide an intuitive understanding of our method, we present a brief summary of it in this subsection.

Algorithm 3 SAVA

```
Initialize reward value estimators, cost value estimators, and policy extractor
Value learning with short-horizon targets:
for each gradient steps do
    get batch of (s_t, a_{t:t+h}, s_{t+h}) ~ D
    update reward penalized values and cost values by the corresponding Bellman objectives
    update target networks
end for
Policy extraction:
for each gradient steps do
    get batch of (s_t, a_{t:t+h}) ~ D
    update vector field v_θ using Eq. 69
end for
```

For finite-data stability, we use a short multi-step horizon h in the practical value-learning implementation. We estimate the reward penalized value and cost value using short-horizon trajectory segments. Specifically, we utilize transitions of the form $(s_t, a_{t:t+h}, s_{t+h})$, and apply expectile regression, using the same state-value function estimation principle as IQL, to estimate the corresponding value functions. For $\mathcal{L}_{Q_r}$, the target construction incorporates the region-aware safety condition described in Section 4.1, replacing the reward continuation when the successor state is classified into the estimated high-cost region.

Our policy extraction is performed using the penalized reward value functions. We employ a CFG method for policy extraction, based on Frans et al. (2025), with slight adjustments to the optimality criteria. Our actor is designed to generate a sequence of h consecutive actions. During execution, however, only the first action in each sequence is executed, consistent with standard 1-step actors.

D.2 COMPUTATIONAL COSTS

We report the training time and GPU memory utilization for our method (SAVA) and the baselines on the *CarGoal1* task in Table 6. All models were trained for one million gradient steps. All algorithms employed their original hyperparameters as reported in the respective publications.

	Training time	CUDA memory
BC	34min	580MB
CDT	12h10min	4466MB
CPQ	2h24min	638MB
COptiDICE	1h40min	594MB
FISOR	2h34min	728MB
CAPS(IQL)	1h47min	620MB
SAVA(ours)	2h16min	880 MB

Table 6: Computational costs of baselines and our method on *CarGoal1* task.

D.3 HYPERPARAMETERS

To facilitate understanding and analysis of hyperparameter effects, we categorize the hyperparameters into two groups. Table 7 reports the unified hyperparameters used across DSRL experiments, including the learning rate, optimizer, network architecture, flow steps, candidate number, multi-step horizon h, expectile level, and discount factor. These hyperparameters are shared by the standard $\ell = \kappa$ setting and the adaptive-threshold variant unless otherwise specified.

Table 7: Hyperparameters for DSRL experiments.

Hyperparameter	**Value**
Learning rate	0.0003
Optimizer	Adam (Kingma & Ba, 2014)
Gradient steps	600000
Minibatch size	1024
MLP dimensions	$[512, 512, 512, 512]$
Nonlinearity	GELU (Hendrycks & Gimpel, 2023)
Flow steps	32
Flow time sampling distribution	$\mathrm{Unif}([0, 1])$
CFG scale w	2.0
Generated candidate numbers	16
Multi-step horizon h	1 for *HalfCheetahVel* task and 5 for others
Expectile τ	0.9
Discount factor γ	0.99
Target critic soft update	0.005
Reward scale	200.0 for *Car* and *Point* series task in Safey-Gymnasium, 1.0 for others
Cost scale	1.0

The threshold used for learned region construction follows the notation defined in Section 5.1. In the standard benchmark setting, SAVA uses the original CMDP cost budget as the region threshold:

$$\ell = \kappa.$$

This setting is used for the direct benchmark comparison with prior offline safe RL baselines, and all methods are evaluated by the same normalized cost C_π/κ.

For the adaptive-threshold experiments in Section 5.3, we replace the region threshold inside the value-shaping rule by

$$\ell_\alpha = \mathcal{Q}_\alpha \left(\{ \widehat{V}_c(s) : s \sim \mathcal{D} \} \right).$$

Here, α is the quantile level used only to instantiate the adaptive threshold ℓ_α. It does not change the external evaluation budget κ, and it is not part of the policy-level objective analyzed in Section 4.2. Table 8 reports the per-task quantile levels used in this adaptive-threshold variant, together with the penalty coefficient C_{pen}.

The practical coefficient C_{pen} sets the fixed reward-target penalty for transitions entering the estimated high-cost region and is distinct from the policy-level penalty coefficient analyzed in Section 4.2. An excessively large value may make finite-data value estimation unstable under limited model capacity. We therefore use a small set of candidate values and report the selected values in Table 8. Additional sensitivity results in Appendix F show that SAVA remains stable across the evaluated penalty range on *CarGoal1* and *CarGoal2*.

E LIMITATIONS

One limitation of the current implementation is the inference cost of the expressive flow-matching policy extractor. Although it can capture multi-modal action distributions, the iterative ODE integration used for flow sampling introduces additional computational overhead, which may limit real-time deployment. Developing more efficient extraction procedures is an important direction for future work.

Another limitation is that SAVA does not provide a reachability-style hard safety certificate under arbitrary function-approximation error. The theoretical analysis characterizes the policy-level penalization principle under oracle and bounded-error settings, while the practical performance still depends on learned cost values, finite-data coverage, and value-function approximation quality.

Table 8: Per-task quantile levels and penalty coefficients for the adaptive-threshold variant.

Task	**Quantile** α	**Penalty Coefficient** C_{pen}
PointPush1	0.9	10.0
PointPush2	0.9	10.0
PointGoal1	0.9	10.0
PointGoal2	0.9	10.0
PointButton1	0.8	10.0
PointButton2	0.5	10.0
CarPush1	0.9	10.0
CarPush2	0.9	10.0
CarGoal1	0.9	10.0
CarGoal2	0.9	10.0
CarButton1	0.8	10.0
CarButton2	0.5	10.0
AntVelocity	0.95	1.0
HalfCheetahVelocity	0.99	1.0
SwimmerVelocity	0.7	1.0
AntRun	0.7	1.0
BallRun	0.5	1.0
CarRun	0.95	1.0
DroneRun	0.7	1.0
AntCircle	0.7	1.0
BallCircle	0.85	1.0
CarCircle	0.9	1.0
DroneCircle	0.7	1.0
easysparse	0.6	1.0
easymean	0.6	1.0
easydense	0.6	1.0
mediumsparse	0.7	1.0
mediummean	0.7	1.0
mediumdense	0.7	1.0
hardsparse	0.5	1.0
hardmean	0.5	1.0
harddense	0.5	1.0

F Additional Experimental Results

F.1 Detailed Main Results

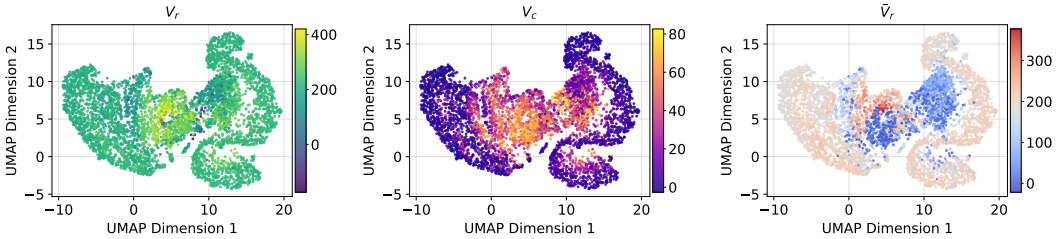
Table 9 reports the complete per-task normalized reward and cost results summarized in Figure 2.

F.2 Additional Visualization Results

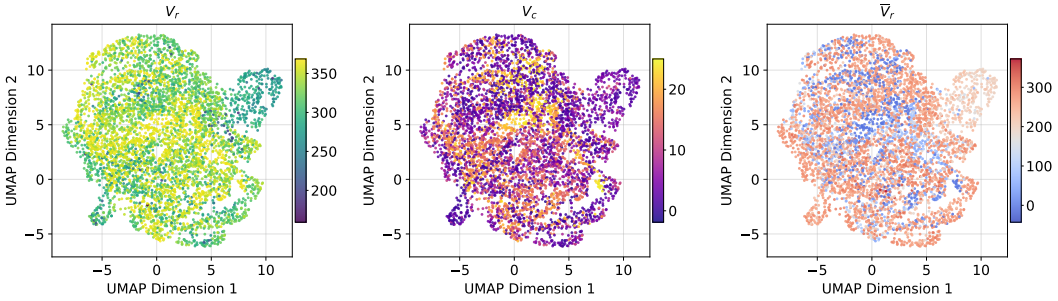
We provide additional UMAP visualizations of V_r, V_c, and $\bar{V}^r$ in Figure 7.

F.3 Per-Task Analysis of Shared Quantile Calibration

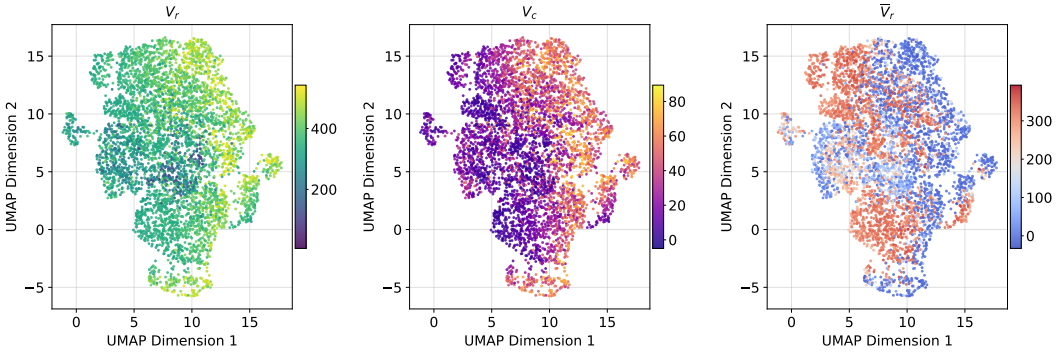
The main text evaluates whether a shared quantile level can replace task-specific threshold selection. Here we provide per-task comparisons between the shared configuration and the task-specific configuration. Specifically, the shared setting uses $\alpha_{\text{fix}} = 0.8$ for all tasks, while the task-specific setting uses the quantile levels reported in Appendix D.3. In both cases, the internal threshold is computed from the corresponding learned cost-value distribution, and all variants are evaluated using the same external normalized cost criterion C_π/κ.



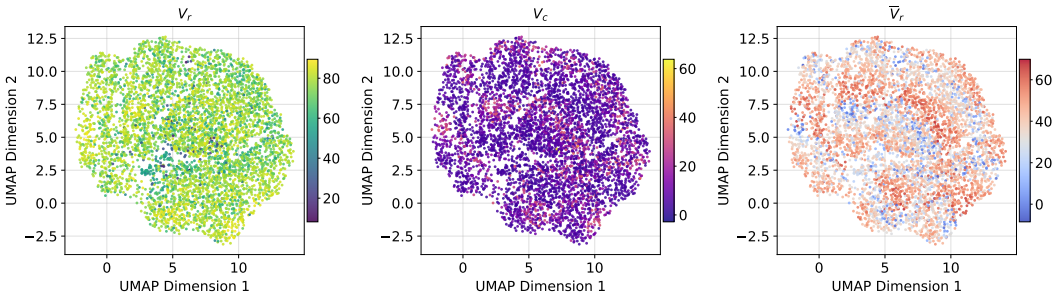
(a) UMAP visualization in the *DroneRun* task.



(b) UMAP visualization in the *DroneCircle* task.



(c) UMAP visualization in the *AntRun* task.



(d) UMAP visualization in the *AntCircle* task.

Figure 7: Comparative visualization of learned state representations using UMAP dimensionality reduction across four distinct locomotion tasks: *DroneRun*, *DroneCircle*, *AntRun*, and *AntCircle*.

Table 9: **Normalized DSRL (Liu et al., 2023a) benchmark results.** ↑ means the higher the better. ↓ means the lower the better. Each value is averaged over 20 evaluation episodes and 3 random seeds. Gray: Unsafe agents. **Bold**: Safe agents whose normalized cost is smaller than 1. **Red**: Safe agents with the highest reward. **Blue**: Safe agents with the second highest reward.

Task	BC		CDT		CPQ		COptiDICE		FISOR		CAPS(IQL)		SAVA(ours)	
	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
PointButton1	0.10	3.36	0.41	9.72	0.26	8.06	0.08	4.59	**0.14**	**0.33**	-0.01	1.23	**0.15**	**0.74**
PointButton2	0.29	8.20	0.23	5.56	0.34	11.29	0.18	5.74	**0.13**	**0.65**	0.13	2.91	**0.16**	**0.63**
PointPush1	0.20	3.09	0.21	2.02	0.01	1.51	0.13	2.91	**0.23**	**0.32**	0.11	1.01	**0.19**	**0.36**
PointPush2	0.18	4.30	0.17	2.23	0.08	17.28	0.04	4.59	**0.12**	**0.64**	0.09	1.87	**0.20**	**0.25**
PointGoal1	0.64	4.44	0.52	2.69	**0.52**	**0.50**	0.53	4.75	**0.16**	**0.30**	**0.21**	**0.96**	**0.36**	**0.20**
PointGoal2	0.52	8.16	0.30	2.64	0.03	6.38	0.42	6.55	**0.10**	**0.07**	0.23	1.33	**0.48**	**0.95**
CarButton1	0.05	5.56	0.14	7.87	0.09	16.48	-0.13	3.52	**-0.01**	**0.00**	-0.07	1.27	**0.15**	**0.96**
CarButton2	-0.06	6.85	0.23	11.88	0.12	38.18	-0.15	4.36	**-0.10**	**0.36**	-0.08	1.36	**0.07**	**0.25**
CarPush1	0.19	1.56	0.19	1.71	**-0.07**	**0.91**	0.19	1.30	**0.26**	**0.57**	**0.17**	**0.45**	**0.29**	**0.36**
CarPush2	0.11	3.71	0.15	4.11	-0.37	10.69	0.10	3.80	**0.14**	**0.88**	0.04	1.61	**0.15**	**0.52**
CarGoal1	0.40	2.46	0.56	3.05	0.29	2.65	0.38	2.46	**0.36**	**0.42**	**0.25**	**0.95**	**0.42**	**0.57**
CarGoal2	0.28	4.40	0.14	2.52	0.06	4.37	0.28	4.97	**0.12**	**0.62**	0.14	2.73	**0.32**	**0.67**
AntVel	0.98	12.05	0.91	4.52	**-1.01**	**0.00**	0.99	5.99	**0.90**	**0.00**	**0.85**	**0.13**	**0.92**	**0.88**
HalfCheetahVel	0.98	22.66	0.59	4.60	0.68	1.37	**0.67**	**0.00**	**0.93**	**0.00**	**0.90**	**0.19**	**0.95**	**0.13**
SwimmerVel	0.45	2.17	**0.62**	**0.37**	**0.01**	**0.45**	0.65	23.70	**0.11**	**0.00**	**0.53**	**0.34**	**0.55**	**0.12**
SafetyGym Average	0.35	6.20	0.36	4.37	0.07	8.00	0.29	5.28	**0.24**	**0.34**	0.23	1.22	**0.35**	**0.50**
AntRun	0.73	9.16	0.72	1.62	**0.01**	**0.00**	0.59	2.03	**0.45**	**0.65**	0.49	2.71	**0.62**	**0.53**
BallRun	0.46	7.30	**0.29**	**0.44**	0.41	5.06	0.43	10.19	**0.19**	**0.00**	**0.10**	**0.16**	**0.30**	**0.00**
CarRun	0.95	1.21	0.98	1.25	**0.92**	**0.04**	**0.84**	**0.00**	**0.76**	**0.08**	**0.96**	**0.37**	**0.89**	**0.42**
DroneRun	0.55	1.44	**0.59**	**0.41**	0.93	15.45	0.71	14.13	**0.34**	**0.63**	0.19	2.01	**0.59**	**0.26**
AntCircle	0.68	16.84	0.41	6.70	0.02	1.53	0.18	13.25	**0.19**	**0.00**	**0.30**	**0.00**	**0.32**	**0.23**
BallCircle	0.70	8.81	0.59	2.16	0.64	4.78	0.70	8.92	**0.34**	**0.00**	**0.38**	**0.00**	**0.51**	**0.49**
CarCircle	0.58	11.26	0.72	2.22	0.24	1.61	0.44	7.88	**0.29**	**0.00**	**0.42**	**0.01**	**0.52**	**0.11**
DroneCircle	0.90	16.64	0.54	2.42	-0.17	7.04	**0.21**	**0.79**	**0.47**	**0.11**	**0.32**	**0.00**	**0.52**	**0.32**
BulletGym Average	0.69	9.08	0.61	2.15	0.38	4.43	0.51	7.15	**0.38**	**0.18**	**0.40**	**0.66**	**0.53**	**0.30**
easysparse	**0.10**	**0.29**	**0.28**	**0.00**	**-0.06**	**0.10**	0.81	8.16	**0.37**	**0.16**	0.77	9.74	**0.42**	**0.07**
easymean	0.59	3.83	**0.32**	**0.00**	**-0.06**	**0.00**	0.55	5.85	**0.40**	**0.49**	**0.00**	**0.21**	**0.38**	**0.02**
easydense	0.12	1.25	0.65	1.76	**-0.06**	**0.16**	0.52	5.12	**0.36**	**0.26**	0.31	1.93	**0.38**	**0.05**
mediumsparse	0.96	4.14	**-0.06**	**0.00**	**-0.08**	**0.10**	0.52	3.08	**0.55**	**0.50**	**0.36**	**0.57**	**0.86**	**0.21**
mediummean	0.89	4.04	0.95	3.33	**-0.08**	**0.16**	0.92	4.59	**0.36**	**0.03**	**0.38**	**0.48**	**0.84**	**0.28**
mediumdense	0.59	2.40	0.93	4.21	**-0.08**	**0.10**	0.85	4.65	**0.49**	**0.02**	**0.22**	**0.16**	**0.82**	**0.15**
hardsparse	0.46	3.60	0.35	1.52	**-0.04**	**0.16**	0.38	4.35	**0.33**	**0.10**	**0.17**	**0.76**	**0.40**	**0.03**
hardmean	0.46	3.42	**0.11**	**0.11**	**-0.05**	**0.16**	0.28	3.30	**0.30**	**0.02**	0.30	3.62	**0.38**	**0.03**
harddense	0.40	3.29	**0.10**	**0.02**	**-0.04**	**0.00**	0.19	2.22	**0.29**	**0.37**	0.50	3.51	**0.37**	**0.02**
MetaDrive Average	0.51	2.92	0.40	1.22	**-0.06**	**0.10**	0.56	4.59	**0.38**	**0.22**	0.33	2.33	**0.54**	**0.10**

Table 10 shows that a single shared quantile level achieves comparable reward–safety tradeoffs across the evaluated tasks. Although task-specific quantile levels can provide additional empirical adjustment on individual tasks, the shared configuration preserves the effectiveness of the data-dependent threshold construction without requiring per-task threshold selection.

F.4 Policy-Extraction Ablation Details

This section provides implementation details for the alternative policy extractors evaluated in Section 5.4.

Gaussian AWR actor. For the Gaussian AWR control in Section 5.4, we use a three-layer MLP Gaussian actor trained with the weighted behavior regression objective:

$$\mathcal{L}_\pi(\phi) = -\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w_{\text{awr}}(s,a)\log\pi_\phi(a|s)\right], \tag{80}$$

where

$$\begin{aligned} w_{\text{awr}}(s,a) =& \mathbf{1}\left[s\in\widehat{\mathcal{S}}_{\text{low}}\right]\exp\left(\eta_r\left(\overline{Q}_r(s,a)-\overline{V}_r(s)\right)\right) \\ &+\mathbf{1}\left[s\in\widehat{\mathcal{S}}_{\text{high}}\right]\exp\left(\eta_c\left(\widehat{V}_c(s)-\widehat{Q}_c(s,a)\right)\right). \end{aligned} \tag{81}$$

Here, $\eta_r, \eta_c > 0$ are temperature parameters. We set $\eta_r = \eta_c = 6.0$ in all experiments. The first term favors higher-value actions according to the region-aware penalized reward advantage, while the second term provides local cost-residual ranking near the estimated high-cost region.

AWR-Flow. For the AWR-Flow control in Section 5.4, we retain the flow-matching policy parameterization but replace the CFG-based extraction rule with the same AWR-style weighting used above.

Table 10: **Comparison of task-specific and shared quantile levels under terminal-only value learning.** Each entry reports normalized reward / normalized cost. The shared setting uses $\alpha_{\text{fix}} = 0.8$ for every task in the calibration subset, whereas the task-specific setting uses the quantile levels reported in Appendix F.3.

Task	task-specific α reward ↑ / cost ↓	$\alpha_{\text{fix}} = 0.8$ reward ↑ / cost ↓
Safety-Gymnasium		
CarButton1	0.15 / 0.96	0.14 / 0.87
CarButton2	0.07 / 0.25	0.08 / 0.74
CarPush1	0.29 / 0.36	0.21 / 0.88
CarPush2	0.15 / 0.52	0.15 / 0.67
CarGoal1	0.42 / 0.57	0.32 / 0.74
CarGoal2	0.32 / 0.67	0.23 / 0.31
PointGoal1	0.36 / 0.20	0.42 / 0.52
PointGoal2	0.48 / 0.95	0.35 / 0.41
AntVel	0.92 / 0.88	0.91 / 0.56
SwimmerVel	0.55 / 0.12	0.66 / 2.81
Bullet-Safety-Gym		
AntRun	0.62 / 0.53	0.62 / 0.38
BallRun	0.30 / 0.00	0.25 / 0.42
CarRun	0.93 / 0.57	0.94 / 0.51
DroneRun	0.59 / 0.26	0.56 / 0.00
AntCircle	0.32 / 0.23	0.18 / 0.50
BallCircle	0.51 / 0.49	0.37 / 0.00
CarCircle	0.52 / 0.11	0.45 / 0.00
DroneCircle	0.52 / 0.32	0.41 / 0.00

The resulting weighted flow-matching objective is

$$\mathcal{L}_v(\theta) = \mathbb{E}_{(s_t, a_{t:t+h})\sim\mathcal{D}}\left[w_{\text{awr}}(s_t, a_t)\left\|v_\theta(a^i_{t:t+h}, i, s_t) - \left(a_{t:t+h} - a^0_{t:t+h}\right)\right\|^2\right]. \tag{82}$$

All other components of the region-aware penalized value objective are kept unchanged. The corresponding empirical comparison is reported in Section 5.4.

F.5 Additional Ablation on Extraction Implementation Choices

In the main experiments, we use 32 flow steps for action generation and sample 16 candidates for candidate-based residual decoding. These choices provide a practical tradeoff between extraction expressiveness and computational cost. To examine the sensitivity of the policy extraction stage to these implementation choices, we evaluate simplified variants using either 8 flow steps or a single candidate, i.e., without residual candidate decoding.

Table 11 compares these variants with the default CFG-based SAVA extractor while retaining the same region-aware penalized reward objective. The simplified extraction settings preserve the overall safety behavior, although reduced sampling capacity can affect reward performance and finite-data robustness. Together with the Gaussian AWR control in Section 5.4, these results show that the safety alignment induced by SAVA is not specific to the default CFG-based extractor on the evaluated subset.

F.6 Additional Ablation on Penalty Coefficient

We further evaluate the sensitivity of SAVA to the practical penalty coefficient C_{pen}, with $P_{\text{pen}} = -C_{\text{pen}}$. This coefficient controls the fixed reward-target penalty applied when a transition enters the estimated high-cost region. Although a larger penalty can strengthen the separation between low-cost and high-cost regions in the learned reward value, an excessively large value may increase the difficulty of value approximation under finite data and finite model capacity.

To examine this effect, we vary C_{pen} from 1 to 15 on the *CarGoal1* and *CarGoal2* tasks. Table 12 shows that SAVA remains safe over the evaluated coefficient range on both *CarGoal1* and *CarGoal2*, with moderate reward variation.

Table 11: Reward–safety performance and action-selection latency under reduced flow-policy sampling settings. The latency includes flow generation, candidate scoring, and final candidate selection, and is measured on an NVIDIA RTX 4090 with batch size 1.

	8 steps + 16 candidates		32 steps + 1 candidate		32 steps + 16 candidates	
Task	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
PointGoal1	0.35	0.28	0.45	0.57	0.36	0.20
PointGoal2	0.31	0.27	0.39	1.54	0.48	0.95
CarGoal1	0.43	0.59	0.38	0.80	0.42	0.57
CarGoal2	0.23	0.30	0.19	0.41	0.32	0.67
End-to-end action-selection latency (ms/action)						
PointGoal1 latency	0.89		3.01		2.73	
CarGoal1 latency	0.83		2.71		2.65	

	$C_{\text{pen}} = 1$		$C_{\text{pen}} = 5$		$C_{\text{pen}} = 10$		$C_{\text{pen}} = 15$	
Task	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
CarGoal1	0.41	0.78	0.43	0.79	0.42	0.57	0.45	0.26
CarGoal2	0.35	0.49	0.32	0.28	0.32	0.67	0.27	0.25

Table 12: Sensitivity to the practical penalty coefficient C_{pen} on *CarGoal1* and *CarGoal2*. Each entry reports normalized reward / normalized cost.