PRIMAL-ONLY SAFE POLICY IMPROVEMENT AT TEST TIME

Anonymous authors
Paper under double-blind review

ABSTRACT

Safe reinforcement learning typically synthesizes a cost constraint into policy parameters during training, and the deployed policy does not re-check the constraint at the state being visited. We ask whether the constraint can instead be imposed at test time on sampled candidate actions. An action constructed at decision time can be checked against the current cost critic at the visited state, without waiting for a policy or a dual variable to update. We propose safe test-time extraction of policies (STEP), which draws candidates from a behavior prior and refines each with a primal-only switch. Specifically, a candidate follows the reward gradient while its estimated cost is below the threshold and receives a cost correction otherwise. The gate evaluates each candidate as a single deviation followed by the behavior policy, and safety under repeated STEP execution is assessed empirically. On 28 DSRL tasks, the average cost of STEP over 8 seeds stays within the budget on 26, and it attains the highest return among agents within the budget on 16. In online fine-tuning on five tasks, STEP maintains or slightly improves return while reducing cost, without an initial performance drop.

1 INTRODUCTION

A fundamental design choice in safe reinforcement learning (RL) is where the constraint enters the algorithm. In general, a policy requires maximizing return under a separate constraint within a constrained Markov decision process (CMDP). The common approach is to augment the constraint into the policy objective and train against it. For example, Lagrangian methods (Xu et al., 2022; Lee et al., 2022a), feasibility-weighted behavioral cloning (BC, Zheng et al., 2024), and cost return-to-go conditioned sequence models (Liu et al., 2023b; Lin et al., 2023) all 'compile' the constraint into the policy parameters during training. At test time, however, such a policy is a forward pass that does not re-check the constraint at the state being visited.

Crucially, in *online learning* settings, such methods can continue to refine the policy as the agent collects new data and performs further optimization. In offline settings, however, the policy is learned from a fixed dataset, and limited data coverage can lead to approximation and generalization errors that persist at deployment. Namely, the constraint is only as reliable as the data that compiled it.

Can these errors be mitigated by performing policy improvement directly at deployment, for example by greedily selecting actions according to a learned critic? Indeed, a classical perspective from dynamic programming is that online greedy action selection with respect to a value function (*i.e.*, akin to model predictive control) constitutes a policy-improvement step, without directly updating the policy parameters (Silver et al., 2017; Bertsekas, 2024).

Driven by this motivation, we propose STEP, an offline RL algorithm that checks the estimated cost constraint on each action at deployment. We train a behavior prior via flow matching (Lipman et al., 2022) and reward and cost critics via SARSA (Sutton et al., 1998). At each decision, STEP draws candidate actions from the prior, refines each with a primal-only switch that follows the reward gradient when the estimated cost is below the bar and applies a cost correction otherwise. Then, it admits candidates through a gate that accounts for ensemble disagreement. We show that refinement raises the true cost of an in-support proposal by at most a Lipschitz term in the refinement radius and, when the best feasible sample has slack, improves its estimated reward while remaining admitted. On 28 DSRL scenarios (Liu et al., 2023a), STEP satisfies the cost budget on 26 and attains the highest return among safe agents on 16. Since the objectives are unchanged from offline to online, the same operator applies to online fine-tuning, where STEP improves return and lowers cost without the initial drop observed in parametric baselines.

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.

2 Related Works

RL in CMDPs. Safe RL has been formulated in various ways (Garcia & Fernández, 2015), and among these, the CMDP (Altman, 2021) is the most widely adopted. Solving CMDPs using deep RL methods has been extensively studied in the literature (Achiam et al., 2017; Dalal et al., 2018; Eysenbach et al., 2018; As et al., 2022; Sootla et al., 2022; Huang et al., 2024), where a common algorithmic choice for solving the constrained optimization problem is by using primal-dual optimization techniques. Concretely, they relax the constraint with a multiplier and optimize the resulting saddle point through a dual variable adapted online (Tessler et al., 2019; Stooke et al., 2020; As et al., 2022; Huang et al., 2024). Such methods satisfy the constraint only at convergence, and the multiplier introduces its own dynamics whose speed relative to the actor governs whether the policy violates the budget during learning. Primal-only methods remove the dual variable altogether and switch between the reward and the cost objective depending on which constraint is currently active. This replaces a scalar that must be tuned with a single-loop algorithm that is simple yet theoretically grounded (Polyak, 1967). In the context of CMDPs, Xu et al. (2021) show that this approach is highly effective within a policy gradients scheme that updates the policy parameters. In this work, we focus instead of using this technique within online action selection.

Test-time search in RL. Recently, several works in RL with generative modeling have focused on test-time methods that use a value function to select or refine actions from a reference policy at deployment. For example, sampling-based action selection (Chen et al., 2023; Park et al., 2024), optimality control (Kumar et al., 2019b; Frans et al., 2025), value gradient update to sampled reference action (Xu et al., 2026), and gradient guide to action denoising process (Zhou et al., 2026). These methods primarily study reward-driven action selection or refinement. On the safety side, Srinivasan et al. (2020) reject actions with high predicted failure risk during policy adaptation, and Dalal et al. (2018) append a safety layer that projects each action onto a linearized single-step cost constraint in closed form. Both only filter a single action from a trained policy, either discarding it or correcting it once against an immediate cost without regard to return. In contrast, we refine multiple behavioral candidates at deployment by switching between reward ascent and cost correction on a cumulative cost critic, so that each candidate gains return while remaining within the estimated budget.

Safe offline-to-online RL. After offline training, online fine-tuning faces two challenges. First, conservative value estimates should be recalibrated under the online objective, which can cause an initial performance drop (Lee et al., 2022b; Nakamoto et al., 2023). Second, policies that remain close to the offline data support may collect little novel experience, causing improvement to plateau (Mark et al., 2022; Ball et al., 2023). Safety constraints intensify this trade-off, i.e., departing from the dataset is necessary for improvement but may lead to unsafe exploration. Recent approaches address this problem through safe policy expansion (Cao et al., 2025), or prior-guided safe exploration (Wendl et al., 2026; Vignola et al., 2026). In particular, model-based methods use epistemic dynamics uncertainty to regulate exploration and gradually expand the certified region as uncertainty decreases.

3 Background

Problem setting. We consider safe offline RL, modeled as a constrained Markov decision process (CMDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, r, c, \gamma, \gamma_c, B \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is a continuous action space, $T(s' \mid s, a)$ is the transition kernel, $r(s, a)$ is the reward, $c(s, a) \geq 0$ is a cost per-step, $\gamma, \gamma_c \in [0, 1)$ are the reward and cost discounts, and B is an episode cost budget. A policy is feasible if $J_c(\pi) = \mathbb{E}_\pi[\sum_t \gamma_c^t c(s_t, a_t)] \leq B$, and the goal is to maximize $J_r(\pi) = \mathbb{E}_\pi[\sum_t \gamma^t r(s_t, a_t)]$ over feasible policies. The agent learns from a fixed dataset $\mathcal{D} = \{(s, a, r, c, s')\}$ collected by a behavior policy $\mu_\beta(a \mid s)$. Offline RL algorithms often use off-policy techniques to approximate action values $Q_r^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a]$ and $Q_c^\pi(s, a)$, and extract a policy from them. We write Q_r and Q_c for the learned critics; trained by SARSA, they approximate Q_r^μ and Q_c^μ, where $\mu \equiv \mu_\beta$. To avoid out-of-distribution (OOD) actions, we extract the policy by maximizing Q_r while staying close to μ_β, subject to a cost constraint, as follows:

$$\begin{aligned} \pi^* \in \operatorname*{argmax}_\pi \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi(\cdot|s)}\big[Q_r(s, a)\big] \quad \text{subject to} \\ \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi(\cdot|s)}\big[Q_c(s, a)\big] \leq d \;\text{ and }\; \mathbb{E}_{s\sim\mathcal{D}}\big[D\big(\pi(\cdot \mid s) \| \mu_\beta(\cdot \mid s)\big)\big] \leq \delta, \end{aligned} \tag{1}$$

where $D(\cdot\|\cdot)$ is a divergence measure (*e.g.*, forward (Nair et al., 2021) or backward (Wu et al., 2019) KL-divergence or ℓ_2-norm), d is a task-induced cost budget, and δ is a threshold.

Behavior policy modeling via flow matching. We model μ_β with a conditional flow policy μ_w (Lipman et al., 2022). A velocity field $v_w(s, x, u)$ carries a Gaussian sample z to a data action a along the interpolation $x_u = (1-u)z + ua$, and is trained by regressing the displacement as follows:

$$\mathcal{L}_{\text{prior}}(w) = \mathbb{E}_{(s,a)\sim\mathcal{D},z\sim\mathcal{N}(0,I),u\sim\text{Unif}[0,1]}\left[\left\|v_w(s, x_u, u) - (a - z)\right\|_2^2\right]. \tag{2}$$

The learned v_w defines $\mu_w(\cdot \mid s)$: a sample is obtained by integrating v_w from $u = 0$ to $u = 1$ with a fixed number of Euler steps and clipping to $\mathcal{A}$. Namely, μ_w serves as a behavior prior.

Safe policy extraction. The objective in (1) is most often solved by Lagrangian relaxation (Fisher, 1981). The cost constraint becomes a penalty $-\lambda Q_c$ in the actor objective, and λ is raised by dual ascent whenever the estimated cost exceeds d (Xu et al., 2022; Lee et al., 2022a). The result is a parametric actor whose constraint holds only at the fixed point of the multiplier dynamics. Another approach keeps the behavior prior and checks the constraint by selection. Given N candidates $a_j \sim \mu_w(\cdot \mid s)$, where $j \in [N]$, constrained best-of-N returns the following:

$$a^* \in \text{argmax}_{a_j : Q_c(s,a_j) \le d}\, Q_r(s, a_j). \tag{3}$$

Since every candidate is sampled from μ_w, a^* stays within the support of the behavior prior, so the divergence constraint holds implicitly without regularization.

Test-time policy improvement. Test-time generalization is an important issue in offline RL, a field where improving data coverage is challenging. A simple approach is to steer the policy guided by the learned value function on test-time states (Park et al., 2024). Specifically, a learned value function can improve actions at the state being visited. A basic reward update is as follows:

$$a^{(k+1)} \leftarrow a^{(k)} + \alpha \nabla_a Q_r(s, a^{(k)}), \qquad k = 0, \dots, K-1, \tag{4}$$

where $a^{(0)} \sim \mu_w(\cdot \mid s)$ is a sampled reference action, α is a step size, and K is the number of refinement steps. The critic and prior remain fixed during refinement.

3.1 Assumptions

This subsection states the assumptions used throughout the paper. For a density level $\rho > 0$, the in-support action set at s is $A_\rho(s) = \{a : p_{\mathcal{D}}(a \mid s) \ge \rho\}$.

Assumption 3.1 (Regularity). *Rewards are bounded, $0 \le c(s, a) \le c_{\max}$, the action set $\mathcal{A}$ is closed and convex, episodes have at most H steps, and terminal states are absorbing.*

Convexity makes $\Pi_{\mathcal{A}}$, the projection onto action space $\mathcal{A}$, non-expansive. Namely, clipping a refined action back into $\mathcal{A}$ cannot increase its distance to any point of $\mathcal{A}$, and a box action space satisfies it.

Assumption 3.2 (On-support critic accuracy). *There exist $\epsilon_r, \epsilon_c \ge 0$ such that $|Q_r(s, a) - Q_r^\mu(s, a)| \le \epsilon_r$ and $|Q_c(s, a) - Q_c^\mu(s, a)| \le \epsilon_c$ for every $a \in A_\rho(s)$.*

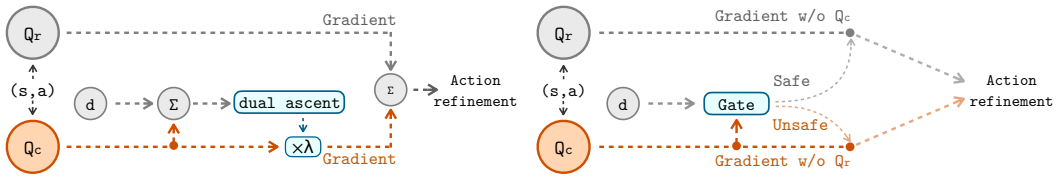
Each bound below starts from a critic value, and a critic trained on $\mathcal{D}$ can be expected to be accurate only where there is coverage. Therefore, we require accuracy only on $A_\rho(s)$, as is common in analyses of behavior-regularized offline RL (Kumar et al., 2019a).

Assumption 3.3 (Prior coverage). *The initial candidates are independent draws from $\mu_w(\cdot \mid s)$ and lie in $\mathcal{A}$. There exists $\zeta \in [0, 1)$ such that $\Pr_{a\sim\mu_w(\cdot|s)}\left(a \notin A_\rho(s)\right) \le \zeta$.*

The guarantees rely on initial samples lying where Assumption 3.2 applies, and ζ is the probability that the prior proposes an OOD action. A generative model fit to $\mathcal{D}$ by flow matching targets exactly $p_{\mathcal{D}}(a \mid s)$, so ζ measures how well μ_w fits $p_{\mathcal{D}}$; we do not estimate it.

Assumption 3.4 (Action-Lipschitz values). *For every s, the maps $a \mapsto Q_c^\mu(s, a)$ and $a \mapsto Q_r^\mu(s, a)$ are L_c- and L_r-Lipschitz on $\mathcal{A}$.*

It is necessary to relate a covered action to a refined one, and it prices a displacement of ϵ at $L_c\epsilon$ in cost. Continuity of a fixed policy's value in the action holds under additional conditions, *e.g.*, Pirotta et al. (2015). We treat it as an assumption.



(a) Naïve safe RL approach (primal-dual) (b) Our solution (primal-only switching)

Figure 1: **The idea.** (*a*) A primal-dual requires nested optimization loops, wherein the inner loop adapts the dual value so as to enforce constraint satisfaction, while the outer loop performs gradient descent on the Lagrangian (see (5)) (*b*) Instead, primal-only requires only a single optimization loop. Owing to the switching "gate", the candidate follows reward ascent while admissible and cost descent as soon as it is not.

4 Safe Test-time Extraction of Policies (STEP)

This section introduces STEP, which improves sampled actions under a cost constraint at test time.

4.1 Main Idea

Our hypothesis is that test-time refinement can improve the reward-cost trade-off by using current cost estimates before an action is executed. It allows candidate-specific correction at the visited state and can incorporate critic uncertainty into action selection.

Why not a primal-dual? Perhaps the most natural extension of (4) for safe RL replaces the reward objective with $Q_r(s,a) - \lambda(Q_c(s,a) - d)$. The corresponding primal-dual updates are as follows:

$$\begin{aligned} a_j^{(k+1)} &= a_j^{(k)} + \alpha\big[\nabla_a Q_r(s, a_j^{(k)}) - \lambda_j^{(k)} \nabla_a Q_c(s, a_j^{(k)})\big], \\ \lambda_j^{(k+1)} &= \Big[\lambda_j^{(k)} + \beta\big(Q_c(s, a_j^{(k)}) - d\big)\Big]_+, \end{aligned} \tag{5}$$

where $\beta > 0$ is the dual step size and $[x]_+ = \max(x, 0)$. This approach requires updating both the candidate actions and their multipliers. If sufficient gradient steps are taken, a primal-dual approach can find the optimal policy (Ding et al., 2020). Otherwise, the multipliers may not adjust sufficiently. The choice of multiplier also poses a trade-off. Candidate-specific multipliers λ_j each need their own dual ascent and increase computation, whereas a shared or fixed multiplier may fail to enforce the constraint across candidates with different local reward-cost trade-offs (Narasimhan et al., 2020).

Solution. Our main idea is to choose the refinement objective from each candidate's estimated feasibility. We follow the reward gradient when the cost criterion is met and apply cost correction otherwise. Violating candidates receive cost correction without waiting for a multiplier to adapt. We also bound displacement from the initial behavioral samples to limit extrapolation during refinement.

4.2 Primal-only Safe Policy Improvement at Test-time

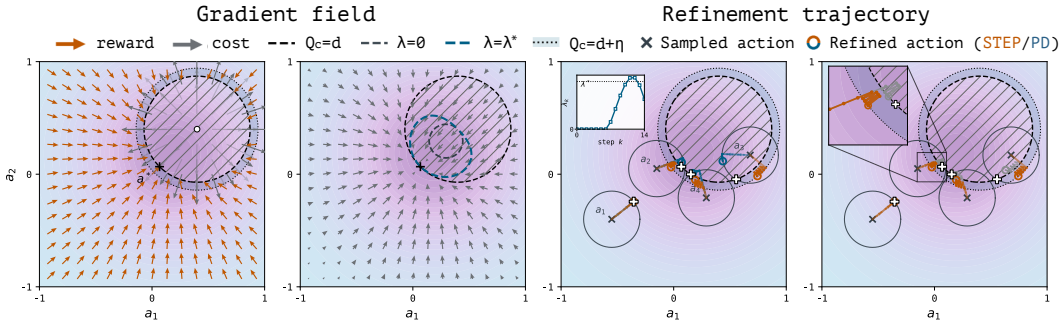
At a visited state s, STEP draws N candidates $a_j^{(0)} \sim \mu_w(\cdot \mid s)$, where $j \in [N]$ as in (3), and refines each for K steps. The subscript j indexes candidates and the superscript (k) indexes refinement steps.

Primal-only action refinement. For each candidate, we seek an action with a higher reward value that satisfies the estimated cost constraint. We limit its displacement from the initial sample by a radius $\epsilon > 0$. The resulting local problem is as follows:

$$\max_{a \in \mathcal{A}} Q_r(s,a) \quad \text{s.t.} \quad Q_c(s,a) \le d, \qquad \|a - a_j^{(0)}\|_2 \le \epsilon. \tag{6}$$

The proposed solution approximately solves this problem by choosing the update objective from the current cost estimate. The switch is run against the bar $d + \eta$, tightened by a slack $\eta \le 0$; $\eta = 0$ recovers the untightened rule. The primal-only update rule of STEP is as follows:

$$a_j^{(k+1)} \leftarrow a_j^{(k)} + \begin{cases} \alpha \dfrac{\nabla_a Q_r(s, a_j^{(k)})}{\|\nabla_a Q_r(s, a_j^{(k)})\|_2}, & Q_c(s, a_j^{(k)}) \le d + \eta, \\ -\dfrac{(Q_c(s, a_j^{(k)}) - d - \eta)}{\|\nabla_a Q_c(s, a_j^{(k)})\|_2^2} \nabla_a Q_c(s, a_j^{(k)}), & Q_c(s, a_j^{(k)}) > d + \eta. \end{cases} \tag{7}$$



(a) Primal-only switching (b) Primal-dual (c) STEP vs. Primal-dual (d) Slack vs. No slack

Figure 2: **Toy example.** This shows the gradient field and test-time refinement trajectory of primal-dual (PD) and STEP. (*a*) Field of (7); cost arrows are truncated where $\nabla_a Q_c$ vanishes. (*b*) Primal-dual field at $\lambda = \lambda^*$, with the stalling curves for $\lambda = 0$ and $\lambda = \lambda^*$. (*c*) $K = 14$ steps from four samples, STEP vs. PD. (*d*) This shows how slack η influences STEP, with $\eta = 0$ against $\eta < 0$; inset zooms on a_2.

For a nonzero cost gradient, the cost step is the minimum-norm correction that places the linearized cost at $d + \eta$. Therefore, a violating candidate receives cost correction without adapting a multiplier. For $\eta < 0$ the correction lands in the interior of the estimated feasible set rather than on its boundary, and the reward branch then operates with a margin of $|\eta|$ below d used in (9). We define the corresponding update to be zero when its gradient vanishes.

Each refined action is projected back to the neighborhood of its initial sample. With $\mathcal{B}_j = \{a : \|a - a_j^{(0)}\|_2 \le \epsilon\}$, the accepted update is as follows:

$$a_j^{(k+1)} \leftarrow \Pi_{\mathcal{B}_j}\big(a_j^{(k+1)}\big), \qquad k = 0, \dots, K-1. \tag{8}$$

Safe action selection. Selection admits candidates against d and accounts for disagreement among an ensemble of E cost critics $\{Q_{c,e}\}_{e=1}^{E}$, whose mean is Q_c. Let $\sigma_c(s,a)$ denote their standard deviation at (s,a). We define the pessimistic cost and the feasible set as follows:

$$\underbrace{U_c(s,a) = Q_c(s,a) + \kappa\sigma_c(s,a)}_{\texttt{pessimistic cost}} \qquad \underbrace{\mathcal{F}_K(s) = \{a \in \mathcal{C}_K(s) : U_c(s,a) \le d\}}_{\texttt{feasible candidate set}}, \tag{9}$$

where $\kappa \ge 0$ and $\mathcal{C}_K(s) = \{a_j^{(0)}, a_j^{(K)} : j \in [N]\}$. We retain the original samples so that refinement cannot remove a proposal that already passes the cost check. STEP selects the highest-value action, or the action with the lowest pessimistic cost when no candidate passes, as follows:

$$a^*(s) \in \begin{cases} \operatorname{argmax}_{a \in \mathcal{F}_K(s)} Q_r(s,a), & \mathcal{F}_K(s) \neq \varnothing, \\ \operatorname{argmin}_{a \in \mathcal{C}_K(s)} U_c(s,a), & \mathcal{F}_K(s) = \varnothing. \end{cases} \tag{10}$$

With $K = 0$, this rule reduces to constrained best-of-N selection.

Didactic example. Figure 2 uses a toy example whose reward peak lies inside the hazard, so the constraint binds at the optimum. The *gradient fields* show how each rule encodes the constraint. In Figure 2 (*a*), the switch field points toward the bar from both sides, so the boundary itself attracts candidates. In Figure 2 (*b*), the primal-dual field is smooth, and where it stalls depends on λ; it stalls on the boundary only at $\lambda = \lambda^*$. In short, the switch enforces the constraint through its structure, whereas primal-dual enforces it only once λ is right. The *refinement trajectories* show what this costs at test time. Since λ starts at 0 and grows only after a violation, primal-dual iterates climb the reward into the hazard first and turn back only once λ has caught up, which may not happen within K steps (Figure 2 (*c*)). The switch needs no such warm-up: once a candidate crosses the bar, the cost branch solves the linearized constraint and returns it in a single step. The slack $\eta < 0$ further leaves a margin below d, so the next reward step does not immediately cross the boundary again (Figure 2 (*d*)). With a small refinement budget, reacting at the first violation matters more than converging eventually.

4.3 Connection to constrained best-of-N sampling

Projection onto $\mathcal{B}_j$ keeps every refined action within ϵ of the behavioral sample it came from. The executed action always lies in $\bigcup_{j\in[N]} \mathcal{B}_j$. Namely, the divergence constraint in (1) becomes a hard proximity constraint around behavioral proposals, whose radius is chosen at deployment rather than compiled into the policy. Refined actions may leave the data support, and the resulting deviation is accounted for by the $O(\epsilon)$ term below. Setting $K = 0$ recovers constrained best-of-N. The two remarks below make this precise at a visited state s. They use Assumptions 3.1–3.4 and learned critics Q_r and U_c that are $\hat{L}_r$- and L_U-Lipschitz, and they hold on the event that all N proposals lie in $A_\rho(s)$. By Assumption 3.3, this event has probability at least $(1-\zeta)^N$. Both remarks bound single-deviation values Q^μ. Appendix B extends them to repeated execution via the performance difference lemma (Kakade & Langford, 2002), under additional assumptions.

Remark 4.1. Let $a = a_j^{(K)} \in \mathcal{F}_K(s)$ be any admitted candidate, including one that the cost branch pulled back from an infeasible origin $a_j^{(0)}$. Its true cost exceeds the budget only by the critic error on the data and by how far refinement moved it, as follows:

$$\begin{aligned} Q_c^\mu(s,a) &\le Q_c^\mu\big(s, a_j^{(0)}\big) + L_c\,\epsilon \\ &\le Q_c\big(s, a_j^{(0)}\big) + \epsilon_c + L_c\,\epsilon \\ &\le U_c(s,a) + \epsilon_c + (L_c + L_U)\,\epsilon \\ &\le d + \underbrace{\epsilon_c}_{\texttt{critic error}} + \underbrace{(L_c + L_U)\,\epsilon}_{\texttt{price of refinement}} \end{aligned} \tag{11}$$

Remark 4.2. Because the unrefined samples stay in the pool, $\mathcal{F}_0(s) \subseteq \mathcal{F}_K(s)$ and refinement only adds options to the final selection. Suppose $\mathcal{F}_0(s) \neq \varnothing$ and let a_b be the constrained best-of-N pick. Let $a_j^{(0)}$ be the sample from which $a^*(s)$ was refined. Then,

$$\begin{aligned} Q_r^\mu\big(s, a^*(s)\big) &\ge Q_r\big(s, a_j^{(0)}\big) - \epsilon_r - L_r\,\epsilon \\ &\ge Q_r\big(s, a^*(s)\big) - \epsilon_r - (L_r + \hat{L}_r)\,\epsilon \\ &\ge Q_r(s, a_b) - \epsilon_r - (L_r + \hat{L}_r)\,\epsilon \\ &\ge Q_r^\mu(s, a_b) - \underbrace{\big(2\epsilon_r + (L_r + \hat{L}_r)\,\epsilon\big)}_{\texttt{worst-case loss vs. best-of-N}} \end{aligned} \tag{12}$$

Why is it useful? The proximity constraint keeps behavior-regularized extraction without training against it, so the conservatism of STEP becomes a deployment knob. Remark 4.1 separates STEP from rejection (Srinivasan et al., 2020) and one-shot projection (Dalal et al., 2018). An infeasible sample need not be discarded, because the bound for pulling it back into the admitted set differs from that of a behavioral sample only by a term linear in ϵ. Remark 4.2 shows that refinement never lowers the estimated value of best-of-N and costs at most the same two error terms in true value. The gain is local. If a_b has cost slack $\xi = d - U_c(s, a_b)$, the reward branch keeps it below the bar for any $K\alpha \le \min\{\epsilon, (\xi + \eta)/L_U\}$. When Q_r is smooth near a_b, each of these steps raises the estimated value. This gain carries over to the true value only while the refined action stays in $A_\rho(s)$. Finally, both bounds depend on ϵ but not on K, since K only sets how much of the radius refinement uses.

4.4 Algorithm summary

We summarize STEP in Algorithm 1. Training consists of two decoupled objectives: (*i*) the flow-matching loss $\mathcal{L}_{\text{prior}}$ in (2) fits μ_w, from which Algorithm 2 draws samples by Euler integration of v_w, and (*ii*) Q_r and Q_c are trained by SARSA. At deployment, STEP draws N candidates by Algorithm 2, refines each for K steps by Algorithm 3, and then selects one by (10). Since the pool retains the unrefined samples, $K = 0$ recovers the constrained best-of-N, and $K > 0$ can only enlarge the admitted set. The knobs $(K, \epsilon, \kappa, \eta)$ and d act only at deployment, so none requires retraining. See implementation details and hyperparameters in Appendix C and E.

Algorithm 1 STEP

Require: $\mathcal{D}$, discount factors γ, γ_c, ensemble sizes E
while not converged **do**
  Sample $(s, a, r, c, s', a') \sim \mathcal{D}$
  # BEHAVIOR PRIOR VIA FLOW MATCHING
  $z \sim \mathcal{N}(0, I)$, $u \sim \mathrm{Unif}[0, 1]$
  $x_u \leftarrow (1-u)\, z + u\, a$
  $\mathcal{L}_{\text{prior}} \leftarrow \left\| v_w(s, x_u, u) - (a - z) \right\|_2^2$
  # VALUE LEARNING
  Train Q_r and Q_c by SARSA
# DEPLOYMENT
while episode not done **do**
  $\{a_j^{(0)}\}_{j=1}^N \leftarrow \textsc{Euler}(s, v_w, N)$ ▷ Alg. 2
  $\{a_j^{(K)}\} \leftarrow \textsc{Refine}(s, Q_r, Q_c, \{a_j^{(0)}\}, K)$ ▷ Alg. 3
  Get feasible set $\mathcal{F}_K$ by (9)
  Get safe best-of-N a^* from $\mathcal{F}_K$ by (10)
  Roll out a^* and observe (s', r, c)
  Set $s \leftarrow s'$

Algorithm 2 ODE solver

Require: M
function $\textsc{Euler}(s, v_w, N)$
  $a \leftarrow z \sim \mathcal{N}(0, I)$
  for $m = 0, \ldots, M-1$ **do**
    $a \leftarrow a + \frac{1}{M}\, v_w\left(s, a, \frac{m}{M}\right)$
  return a

Algorithm 3 Test-time action refinement

Require: η, d
function $\textsc{Refine}(s, Q_r, Q_c, \{a_j\}, K)$
  $a_j^{(0)} \leftarrow a_j$
  for $k = 0, \ldots, K-1$ **do**
    Get $a_j^{(k+1)}$ from $a_j^{(k)}$ by (7) with condition $d + \eta$
    Project $a_j^{(k+1)}$ onto $\mathcal{B}_j$ by (8)
  return $\{a_j^{(K)}\}$

5 EXPERIMENTS

We evaluate the proposed solution on three offline safe RL testbeds, spanning 28 offline datasets from the DSRL benchmark (Liu et al., 2023a). We first introduce the experimental setup, *including* selected testbeds, the baselines, and the training & evaluation protocol.

Task and dataset. We use 28 offline datasets from the DSRL benchmark (Liu et al., 2023a), *including* three testbeds: `Safety-Gymnasium` (Ji et al., 2023) (11 scenarios), `Bullet-Safety-Gym` (Gronauer, 2022) (8 scenarios), and `MetaDrive` (Li et al., 2022) (9 scenarios). Detailed task descriptions and cost definitions are given in Appendix D.

Baselines. We organize baselines into three families by how the safety constraint is imposed. As in *imitation learning*, `BC` clones the transitions of the data support, while `Safe-BC` (Liu et al., 2023a) clones only the trajectories whose episodic cost stays within the budget. Both variations are trained by flow-matching. For *soft constraints*, `BEAR-L` (Kumar et al., 2019a) augments BEAR with a Lagrangian multiplier on the cost critic; `CPQ` (Xu et al., 2022) treats unsafe and out-of-distribution actions alike and penalizes their values; and `COptiDICE` (Lee et al., 2022a) performs constrained stationary-distribution correction in the space of state-action visitations. For *hard constraints*, `FISOR` (Zheng et al., 2024) converts the constraint into a Hamilton-Jacobi reachability problem and extracts a diffusion policy restricted to the identified feasible set; and `SafeFQL` (Tayal et al., 2026) builds a flow-matching policy filtered by a learned feasibility score via the reach-avoid value.

We consider 1M offline gradient steps, state normalization by dataset statistics, and dataset rewards rescaled to a common episodic range. We evaluate every 100K gradient steps over 20 episodes and report the policy at the final step, averaged over 8 seeds. Following DSRL (Liu et al., 2023a), we report normalized return and normalized cost, the latter divided by the task's cost limit. A value of at most 1.0 indicates that the average cost over evaluation episodes and seeds is within the budget. We set the cost limit to 10 on the more difficult `Safety-Gymnasium` and to 5 on other benchmarks.

5.1 EXPERIMENTAL RESULTS

Table 1 shows that `STEP` satisfies the budget on 26 of 28 tasks and achieves the highest return among safe agents on 16. It is also the only method whose average cost stays below the budget on all three benchmarks. The baselines fail in one of two ways. `BC`, `BEAR-L`, and `COptiDICE` obtain high returns by violating the budget on most tasks. `COptiDICE` has the highest average return 0.57 at an average cost of 11.7. `SafeFQL` also violates the budget on most tasks, without a return advantage over `STEP` on average. `Safe-BC` and `CPQ` stay within the budget on almost all of `MetaDrive` and much of `Bullet-Safety-Gym` but give up return, *e.g.*, `CPQ` averages 0.01 on `MetaDrive`. `FISOR` is the

Table 1: **Results on safe RL tasks.** Normalized reward (R) and normalized cost (C) on the DSRL benchmark, averaged over 8 seeds. A normalized cost above 1.0 means the budget is violated: gray denotes unsafe agents and normal font implies safe agents, while **orange** marks the highest reward among the safe agents in each task.

	Task	BC		Safe-BC		BEAR-L		CPQ		COptiDICE		SafeFQL		FISOR		STEP	
		R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)
Safety-Gymnasium	CarButton1	0.15	7.28	0.04	1.83	0.01	4.81	0.43	35.9	0.09	4.93	0.50	18.9	-0.09	0.61	0.01	0.92
	CarButton2	0.13	7.78	-0.02	2.54	-0.12	1.70	0.42	38.8	-0.06	4.42	0.55	20.5	-0.03	0.53	-0.01	1.08
	CarPush1	0.22	2.25	0.15	1.03	0.41	2.80	0.10	6.40	0.22	2.09	0.42	1.96	0.24	0.85	0.27	0.74
	CarPush2	0.10	3.14	0.02	1.10	0.38	7.66	0.17	14.7	0.12	4.94	0.28	6.03	0.07	0.54	0.03	0.61
	CarGoal1	0.32	2.33	0.19	0.74	0.73	5.60	0.69	4.47	0.38	2.16	0.76	6.30	0.42	1.22	0.18	0.56
	CarGoal2	0.25	3.98	0.11	1.45	0.70	10.9	0.88	17.8	0.23	3.29	0.77	13.8	0.14	0.62	0.09	0.89
	CarCircle1	0.71	16.4	0.44	6.47	0.17	11.8	0.02	1.12	0.68	18.5	0.52	21.0	0.66	7.11	0.15	0.80
	CarCircle2	0.64	17.9	0.49	11.0	0.18	4.50	0.29	1.77	0.72	23.0	0.66	16.3	0.58	6.73	0.09	0.96
	AntVel	0.96	16.4	0.80	1.53	0.46	0.89	-0.60	0.01	0.98	13.2	0.94	0.00	0.89	0.00	0.85	0.32
	HalfCheetahVel	0.90	12.6	0.69	0.65	0.93	0.82	0.36	0.69	1.00	43.9	0.99	0.00	0.88	0.00	0.95	0.14
	SwimmerVel	0.42	7.80	0.45	1.92	0.28	4.27	0.09	6.99	0.63	22.5	-0.05	0.47	-0.03	0.48	0.47	0.12
	Average	**0.44**	**8.89**	**0.31**	**2.75**	**0.38**	**5.07**	**0.26**	**11.7**	**0.45**	**13.0**	**0.58**	**9.57**	**0.34**	**1.70**	**0.28**	**0.65**
Bullet-Safety-Gym	BallRun	0.47	9.98	0.21	2.21	0.05	5.09	0.87	17.0	0.82	14.3	0.08	4.45	0.13	3.09	0.25	1.05
	BallCircle	0.71	9.35	0.29	0.38	0.66	5.94	0.64	0.03	0.77	10.8	0.34	0.00	0.33	0.00	0.26	0.97
	CarRun	0.85	2.84	0.77	0.38	0.96	4.38	0.89	4.20	0.96	5.18	0.69	0.00	0.36	0.32	0.77	0.89
	CarCircle	0.29	8.41	0.17	0.69	0.52	1.98	0.49	0.23	0.56	9.50	0.55	5.13	0.35	0.11	0.07	0.72
	DroneRun	0.60	11.6	0.52	1.00	0.38	8.80	0.37	10.0	0.60	12.1	0.56	12.8	0.35	3.43	0.54	0.58
	DroneCircle	0.72	10.1	0.46	0.95	0.52	1.09	-0.26	0.73	0.64	14.5	0.02	2.28	0.50	0.06	0.42	0.74
	AntRun	0.77	15.0	0.59	1.65	0.63	6.02	0.10	0.04	0.83	21.3	0.31	0.04	0.43	0.73	0.59	0.98
	AntCircle	0.52	16.1	0.36	6.48	0.30	3.60	0.00	0.00	0.51	12.3	0.56	1.31	0.22	0.03	0.46	0.16
	Average	**0.62**	**10.4**	**0.42**	**1.72**	**0.50**	**4.61**	**0.39**	**4.03**	**0.71**	**12.5**	**0.39**	**3.25**	**0.33**	**0.97**	**0.42**	**0.76**
MetaDrive	easysparse	0.67	6.55	0.35	0.00	0.34	1.06	-0.06	0.23	0.92	17.8	0.27	3.66	0.56	0.44	0.53	0.94
	easymean	0.74	8.16	0.38	0.00	0.49	3.07	-0.03	0.33	0.74	14.4	0.33	4.27	0.51	0.17	0.55	0.83
	easydense	0.61	5.94	0.39	0.00	0.37	1.02	-0.06	0.25	0.52	10.5	0.14	2.13	0.55	0.49	0.60	0.83
	mediumsparse	0.81	4.06	0.63	0.02	0.81	1.60	0.01	0.62	0.82	8.90	0.22	2.31	0.77	0.38	0.79	0.87
	mediummean	0.79	4.19	0.52	0.02	0.84	2.12	0.01	0.43	0.74	8.17	0.04	0.41	0.70	0.66	0.66	0.84
	mediumdense	0.79	4.01	0.59	0.04	0.85	2.71	-0.01	0.42	0.82	8.72	0.27	0.96	0.76	0.30	0.78	0.71
	hardsparse	0.49	5.36	0.35	0.05	0.37	0.89	0.07	1.02	0.26	5.95	0.12	1.57	0.42	0.74	0.43	0.84
	hardmean	0.46	5.12	0.33	0.03	0.37	1.00	0.08	0.86	0.29	6.63	0.05	0.42	0.38	0.69	0.41	0.95
	harddense	0.40	3.55	0.33	0.06	0.34	1.13	0.08	0.90	0.21	4.87	0.17	1.04	0.39	0.68	0.39	0.79
	Average	**0.64**	**5.22**	**0.43**	**0.03**	**0.53**	**1.62**	**0.01**	**0.56**	**0.59**	**9.55**	**0.18**	**1.86**	**0.56**	**0.51**	**0.57**	**0.84**
All (28 tasks)		0.55	8.15	0.38	1.58	0.46	3.83	0.22	5.93	0.57	11.7	0.39	5.29	0.41	1.11	0.41	0.74

See Appendix F and E for the full table with standard deviation and per-task selected hyperparameters.

strongest baseline and is safe on 23 tasks, but it exceeds the budget on CarGoal1, CarCircle1/2, BallRun, and DroneRun, and STEP is safe on all of these except BallRun.

5.2 CAN STEP BE FINE-TUNED WITH ONLINE EXPLORATION VIA ACTION REFINEMENT?

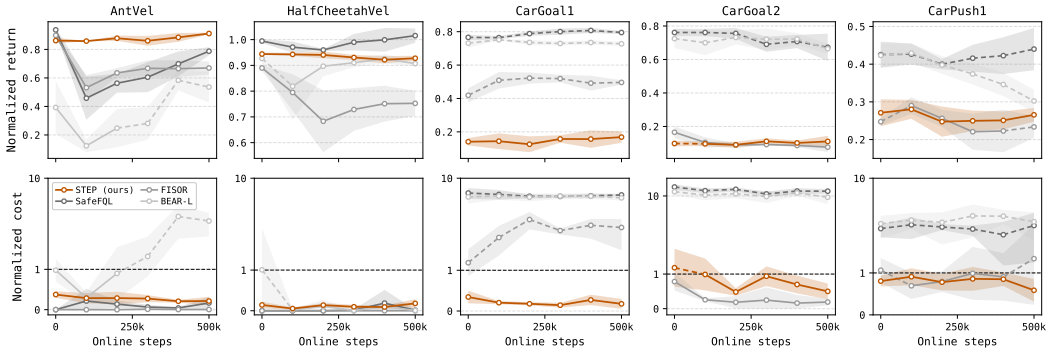
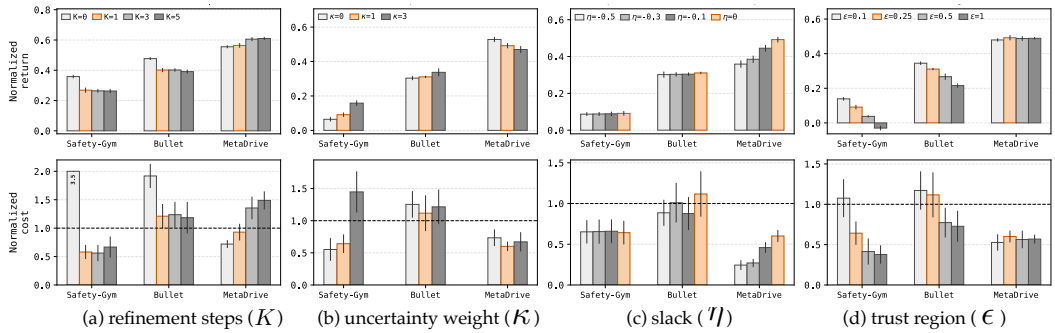


Figure 3: **Online fine-tuning results over** 8 **seeds.** This visualizes the online fine-tuned performance in five scenarios of Safety-Gymnasium. The dotted line implies that the policy fails to meet the constraint.

Figure 3 shows 500K online fine-tuning steps on two velocity and three navigation tasks. STEP improves or holds return and lowers cost on all five tasks, with no drop at the start; the one task that begins above the budget is brought under. The baselines show the two failure modes: where they are safe (*i.e.*, velocity), SafeFQL and FISOR lose much of their offline return in the first online steps and recover only partially. Where they start unsafe (*i.e.*, car navigation), SafeFQL and BEAR-L stay unsafe, and FISOR trades, gaining return on one task by violating further and becoming safe on another by giving up return. On the other hand, STEP shows neither. We speculate that STEP's

stability to its conservatism lies in test-time knobs rather than in the training objective, so nothing is recalibrated. In other words, each update can gradually expand the set of allowed options.

5.3 Ablation studies



(a) refinement steps (K) (b) uncertainty weight (κ) (c) slack (η) (d) trust region (ϵ)

Figure 4: **`STEP` only needs K to be tuned in general.** The four plots show how different values of K, κ, η, ϵ affect safe offline RL performance. Each plot is aggregated over all scenarios per task.

Figure 4 sweeps each knob of `STEP` with the others at their per-task values.

In general, best-of-N often suffices, and one step helps with the rest. With $K = 0$, `STEP` reduces to constrained best-of-N, a sample-based approximation of the closed-form solution of (1), restricted to admitted samples (Nair et al., 2021). Like the closed form, it can only reselect what the prior proposes. When the prior already proposes feasible high-value actions, as on `MetaDrive`, $K = 0$ is already safe, and each additional step raises return and cost together as the reward branch takes over. On `Safety-Gymnasium` and `Bullet-Safety-Gym`, $K = 0$ has the highest return but violates the budget, since the prior is fit to unfiltered data, and a single step alleviates the cost without training a separate safe prior. This suggests a simple rule: use $K = 1$ where the prior is unsafe, and raise K for return only as far as the remaining cost slack allows.

κ has little effect in general. $\kappa = 0$ and $\kappa = 1$ are indistinguishable on all benchmarks. With a large κ, the fallback $\operatorname{argmin} U_c$ favors low-variance actions over low-cost ones, so the executed action drifts toward the behavior policy. Therefore, we use $\kappa = 1$ by default and adjust it on a few tasks.

The slack η matters only where the prior is already safe. η has no effect on `Safety-Gymnasium` or `Bullet-Safety-Gym`, but induces a monotonic reward-cost tradeoff on `MetaDrive`. This is consistent with its role as a margin below d in (7): the margin affects candidate selection only when candidates fall below the resulting bar, as they do in `MetaDrive`.

The trust region ϵ is the safety dial. Increasing ϵ from 0.1 to 1 monotonically lowers cost on `Safety-Gymnasium` and `Bullet-Safety-Gym` at a loss in return, and has no effect on `MetaDrive`. Remark 4.1 prices refinement linearly in ϵ in the worst case, but a larger radius also lets the cost branch pull violating candidates further; where the prior is unsafe, the latter dominates.

6 Conclusion

In this work, we show that `STEP`, a novel model-free safe RL method, satisfies the cost budget more often than parametric methods at comparable average return, and fine-tunes online without the side effect. Taken together, these results support a positive answer to the question of Section 1: the constraint can be imposed at test time on sampled candidate actions rather than in the policy objective, and the policy that results has access at deployment to what a parametric one does not.

The absence of a model also fixes the horizon of the check. The constraint is verified one action at a time, at the visited state. It remains open whether the same switch can act over a planning horizon, where a model would let it correct a short rollout rather than a single action. The lookahead policy of Sikchi et al. (2022), which plans over a learned model and closes the horizon with a model-free value, may provide clues. We leave this extension to future work. See more discussion about limitations and future directions, *including* heuristic hyperparameter tuning and inference time, in Appendix A.

AI USE STATEMENT

We used AI assistants to polish the writing and check grammar, and to check and refine our mathematical proofs. We also used them for engineering tasks, *including* writing job submission scripts, organizing the result tables and computing averaged scores, and drafting a JAX implementation of a baseline from its PyTorch implementation. The leading author reviewed and verified all AI-assisted content and code, and takes full responsibility for the paper.

ETHICS STATEMENT

This work presents foundational machine learning research and does not involve human subjects, sensitive personal data, or high-risk dual-use applications. The authors have evaluated the potential ethical implications of this work and anticipate no foreseeable negative societal impacts, safety risks, or ethical concerns beyond those standard to general computational research.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we include detailed descriptions of the implementation of our algorithm in Appendix C, experimental setup in Appendix D, and hyperparameters in Appendix E. We will release code upon publication.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *ICML*, 2017.

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

Yarden As, Ilnura Usmanova, Sebastian Curi, and Andreas Krause. Constrained policy optimization via bayesian world models. In *ICLR*, 2022.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *ICML*, 2023.

Dimitri P. Bertsekas. Model predictive control and reinforcement learning: A unified framework based on dynamic programming, 2024.

Chenyang Cao, Yucheng Xin, Silang Wu, Longxiang He, Zichen Yan, Junbo Tan, and Xueqian Wang. Fosp: Fine-tuning offline safe policy through world models. In *ICLR*, 2025.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *ICLR*, 2023.

Xiaoyuan Cheng, Wenxuan Yuan, Boyang Li, Yuanchao Xu, Yiming Yang, Hao Liang, Bei Peng, Robert Loftin, Zhuo Sun, and Yukun Hu. How does the lagrangian guide safe reinforcement learning through diffusion models? *arXiv preprint arXiv:2602.02924*, 2026.

Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, 2000.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Dongsheng Ding, Kaiqing Zhang, Tamer Basar, and Mihailo Jovanovic. Natural policy gradient primal-dual method for constrained markov decision processes. *NeurIPS*, 2020.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *ICLR*, 2018.

Marshall L Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, 27(1):1–18, 1981.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025.

Javier Garcia and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 2015.

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

Weidong Huang, Jiaming Ji, Chunhe Xia, Borong Zhang, and Yaodong Yang. Safedreamer: Safe reinforcement learning with world models. In *ICLR*, 2024.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety gymnasium: A unified safe reinforcement learning benchmark. In *NeurIPS*, 2023.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *ICLR*, 2022.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *NeurIPS*, 2019a.

Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019b.

Jongmin Lee, Wonseok Jeon, Byungjun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *ICML*, 2021.

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. In *ICLR*, 2022a.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *CoRL*, 2022b.

Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 45(3):3461–3475, 2022.

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 *ICML*, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, et al. Datasets and benchmarks for offline safe reinforcement learning. *arXiv preprint arXiv: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 *ICML*, 2023b.

Max Sobol Mark, Ali Ghadirzadeh, Xi Chen, and Chelsea Finn. Fine-tuning offline policies with optimistic action selection. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets, 2021.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *NeurIPS*, 2023.

Harikrishna Narasimhan, Andrew Cotter, Yichen Zhou, Serena Wang, and Wenshuo Guo. Approximate heavily-constrained learning with lagrange multiplier models. *NeurIPS*, 2020.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? In *NeurIPS*, 2024.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *ICML*, 2025.

Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Policy gradient in lipschitz markov decision processes. *Machine Learning*, 100(2):255–283, 2015.

Boris Polyak and Anatoli Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Boris T Polyak. A general method for solving extremal problems. In *Soviet Mathematics Doklady*, 1967.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019.

Stephane Ross and J Andrew Bagnell. Agnostic system identification for model-based reinforcement learning. *arXiv preprint arXiv:1203.1007*, 2012.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *CoRL*, 2022.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 2017.

Aivar Sootla, Alexander I Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H Mguni, Jun Wang, and Haitham Ammar. Saute rl: Almost surely safe reinforcement learning using state augmentation. In *ICML*, 2022.

Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *ICML*, 2020.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Mumuksh Tayal, Manan Tayal, and Ravi Prakash. Safe flow q-learning: Offline safe reinforcement learning with reachability-based flow policies. In *RLC*, 2026.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *ICLR*, 2019.

Luca Vignola, Bruce D Lee, Manish Prajapat, Manuel Wendl, Melanie Zeilinger, Andreas Krause, and Yarden As. Sampling-based safe reinforcement learning. *arXiv preprint arXiv:2605.19469*, 2026.

Manuel Wendl, Yarden As, Manish Prajapat, Anton Pollak, Stelian Coros, and Andreas Krause. Safe exploration via policy priors. In *ICLR*, 2026.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized Q-learning for safe offline reinforcement learning. In *AAAI*, 2022.

Haoran Xu, Kaiwen Hu, Somayeh Sojoudi, and Amy Zhang. Reinforcement learning via value gradient flow. In *ICLR*, 2026.

Tengyu Xu, Yingbin Liang, and Guanghui Lan. Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *ICML*, 2021.

Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. In *ICLR*, 2024.

Zhiyuan Zhou, Andy Peng, Charles Xu, Qiyang Li, Tobias Springenberg, Kevin Frans, and Sergey Levine. Test-time gradient guidance of flow policies in reinforcement learning. *arXiv preprint arXiv:2606.11087*, 2026.

Appendix

CONTENTS

A Limitations and Future Directions **15**

B Additional Mathematics **15**

C Implementation Details **15**

D Experimental Details **16**

D.1 Tasks and datasets 16

D.2 Methods 16

E Hyperparameters **17**

F Additional Experimental Results **19**

F.1 Full table with standard deviation 19

F.2 Design choice of value function 20

F.3 Design choice of gradient update 21

F.4 Inference time 21

F.5 Hyperparameter comparison on FISOR 22

A Limitations and Future Directions

In addition to the limitations discussed in Section 6, one limitation of STEP is its sensitivity to the deployment knobs. While separating the behavior prior, the safety constraint, and value maximization is empirically effective, the benefit of refinement depends on the number of steps K and other knobs (Figure 4). Namely, the current solution tunes the hyperparameter as a heuristic; a remaining open question is how to design a stopping rule. We believe this may be improved by adapting the trust region to how well the critics agree, as classical trust-region methods adapt their radius to how well the local model predicts the objective (Conn et al., 2000).

Additionally, STEP is cheaper than diffusion policies with many denoising steps (*e.g.*, about 4.9 ms reported by Cheng et al. (2026)) and comparable to FISOR (Appendix F.4). However, it remains slower than one-step policies such as SafeFQL, since each decision integrates the flow prior and differentiates the critics. Amortizing the refinement by distilling refined actions into a policy would reduce this cost at the price of its test-time nature.

B Additional Mathematics

Proposition B.1 (Repeated execution)**.** *Suppose the conditions of Remark 4.1 hold at every state visited by $\pi_{\texttt{STEP}}$, and let G_t be the event that all N proposals at s_t lie in $A_\rho(s_t)$ and $\mathcal{F}_K(s_t) \neq \varnothing$. With $V_c^\mu(s) = \mathbb{E}_{a\sim\mu(\cdot|s)} Q_c^\mu(s,a)$,*

$$J_c(\pi_{\texttt{STEP}}) \leq J_c(\mu) + \sum_{t<H} \gamma_c^t \, \mathbb{E}\Big[\big(d - V_c^\mu(s_t) + \epsilon_c + (L_c + L_U)\epsilon\big)\mathbb{1}\{G_t\} + c_{\max} H \, \mathbb{1}\{\neg G_t\}\Big],$$

where $\Pr(\neg G_t \mid s_t) \leq N\zeta + \Pr(\mathcal{F}_K(s_t) = \varnothing)$.

Proof. The performance difference lemma (Kakade & Langford, 2002) gives $J_c(\pi) - J_c(\mu) = \sum_{t<H} \gamma_c^t \mathbb{E}_{s_t\sim\pi}[Q_c^\mu(s_t,a_t) - V_c^\mu(s_t)]$. On G_t, Remark 4.1 bounds $Q_c^\mu(s_t,a_t)$ by $d + \epsilon_c + (L_c + L_U)\epsilon$. Otherwise, $Q_c^\mu(s_t,a_t) - V_c^\mu(s_t) \leq Q_c^\mu(s_t,a_t) \leq c_{\max} H$ by Assumption 3.1. The probability bound follows from Assumption 3.3 and a union bound over the N draws. □

The first term is the cost slack that an admitted action may spend at each visited state. It is non-positive when $d \leq V_c^\mu(s_t)$ along the trajectory, *e.g.*, for hard constraints with $d = 0$ on states where μ incurs no cost. Otherwise, admitted actions can accumulate this slack over an episode, so the per-state gate alone does not certify $J_c(\pi_{\texttt{STEP}}) \leq B$. The bound is also relative to $J_c(\mu)$, which may exceed B when $\mathcal{D}$ contains unsafe trajectories, and it requires Assumptions 3.2–3.3 on states visited by STEP, which we neither certify nor estimate.

C Implementation Details

Flow matching for behavior prior. We learn a velocity field via flow matching in (2). We fix the number of flow steps to 10 when sampling from the learned model. The trained behavior prior $\mu_w(\cdot \mid s)$ serves as the reference distribution. In the proposed solution, we sample N action particles $a_j^{(0)} \sim \mu_w(\cdot \mid s)$, $j \in [N]$, as candidates at a given state s.

SARSA value learning. We train Q_r and Q_c with SARSA targets (Sutton et al., 1998), rather than TD targets. Bootstrapping through a threshold-selected action can intensify cost underestimation. Namely, admitting $a^*(s')$ only when $Q_c(s', a^*(s')) \leq d$ bounds the target by $c + \gamma_c d$, even when the selected continuation is unsupported by data. Therefore, for each transition (s, a, r, c, s', a'), we use $y_r = r + \gamma Q_r(s', a')$, and $y_c = c + \gamma_c Q_c(s', a')$. These targets use observed state-action pairs and evaluate the data policy μ_β, targeting Q_r^μ and Q_c^μ. Consequently, even with accurate evaluation, passing the cost check certifies only a single deviation from μ_β, followed by its continuation (see Proposition B.1 for repeated execution).

Action projection. All methods use a box action space $\mathcal{A} = [-1, 1]^m$ with action dimension m, so $\Pi_\mathcal{A}$ is elementwise clipping. For STEP, the refinement in (7) applies two projections in sequence,

first onto the trust region and then onto the action space, as follows:

$$a_j^{(k+1)} \leftarrow \Pi_{\mathcal{A}}\Big(\Pi_{\mathcal{B}_j}\big(a_j^{(k+1)}\big)\Big), \qquad k = 0, \dots, K-1. \tag{13}$$

The order matters. Since $a_j^{(0)} \in \mathcal{A}$ and $\mathcal{A}$ is convex, $\Pi_{\mathcal{A}}$ is non-expansive toward $a_j^{(0)}$. Hence, it cannot move a point out of $\mathcal{B}_j$, so the composed iterate satisfies both constraints. The reverse order does not have this property, as projecting onto the ball can leave $\mathcal{A}$.

Online fine-tuning. Every online run starts after the 1M-step offline training and performs 500K online steps. The replay buffer is balanced (Ross & Bagnell, 2012): each minibatch of 256 transitions is drawn half from the offline dataset and half from the online buffer. We do not consider additional ideas for exploration, including objective changing and exploration noise. All baselines are simply set to stochastic action selection of the actor network.

Network architectures. To fairly compare with baselines, every method is built on the same backbone: all actor, critic, and value heads are four-layer MLPs of width 512, i.e., $[512, 512, 512, 512]$, trained with batch size 256, discount factor $\gamma, \gamma_c = 0.99$, and Polyak averaging $\tau = 0.005$ (Polyak & Juditsky, 1992). All methods also consume identically normalized data: observations are standardized with the dataset statistics and rewards are rescaled with the same normalization, so no method sees a different input or return scale. The critic uses layer normalization, and the actor does not.

However, `FISOR` is the single exception. Under the same structure, its feasibility collapses, so we report it in the architecture of its own publication: $[256, 256]$ critics, a width-256 diffusion actor with $[128, 128]$ conditioning, and batch size 2048. We provide a comparison between `FISOR` with our architecture and with paper reported one (Zheng et al., 2024) in Appendix F.5.

D Experimental Details

D.1 Tasks and datasets

Safety-Gymnasium. A robot navigates a cluttered arena while a cost signal fires whenever it enters a hazard or contacts a fragile object (Ray et al., 2019). We test on eight `Car` navigation tasks, namely `Button1/2`, `Push1/2`, `Goal1/2`, and `Circle1/2`, which differ in the goal semantics and in the density of obstacles, together with three velocity-constrained locomotion tasks (*i.e.*, `Ant`, `HalfCheetah`, `Swimmer`) in which the agent is rewarded for moving fast but penalized for exceeding a speed threshold, yielding 11 configurations.

Bullet-Safety-Gym. Four embodiments (*i.e.*, `Ball`, `Car`, `Drone`, `Ant`) are each paired with two objectives, `Run` and `Circle`, giving 8 configurations (Gronauer, 2022). Because the safe set is held fixed, this testbed isolates how well a safety mechanism tolerates increasing control difficulty, *i.e.*, from a fully actuated ball to an underactuated quadrotor and a high-DoF quadruped.

MetaDrive. An autonomous vehicle drives through road networks with surrounding traffic, and cost is incurred by collisions and out-of-road events (Li et al., 2022). We use nine scenarios formed by crossing three levels of road difficulty (`easy`, `medium`, `hard`) with three traffic densities (`sparse`, `mean`, `dense`), which decouples the geometric difficulty from the density of the hazards.

D.2 Methods

BC and Safe-BC. Both clone the behavior policy with a conditional flow-matching field, the same prior `STEP` builds on, so the gap to `STEP` isolates the selection rule rather than the policy class. The field is a $[512, 512, 512, 512]$-sized MLP, and actions are produced with 10 Euler steps. `Safe-BC` differs only in that a transition enters the loss only if its episode finished within the task's cost budget, applied as a per-sample mask with the loss normalized by the number of feasible rows so the effective learning rate does not scale with how safe the dataset is.

CPQ. For `CPQ` (Xu et al., 2022), we re-implement a PyTorch-based open-source implementation in JAX. The policy is a tanh-squashed Gaussian actor, the reward and cost critics are two-head ensembles, and the behavior model is a conditional VAE whose latent dimension is twice the action dimension. All networks are $[512, 512, 512, 512]$-sized MLPs, with layer normalization enabled in the critics and disabled in the actor.

BEAR-L. `BEAR-L` is BEAR (Kumar et al., 2019a), whose actor is held inside the data support by an MMD penalty against a VAE behavior model, with a cost critic whose multiplier is driven by a PID Lagrangian. For this, we re-implement a PyTorch-based open-source implementation in JAX. We keep the reference values, *including* Gaussian kernel $\sigma = 50$, 10 MMD samples, support threshold 0.05, clipped Q mixing $\lambda = 0.75$, PID gains $(0.1, 0.003, 0.001)$.

COptiDICE. For `COptiDICE` (Lee et al., 2022a), we re-implement a PyTorch-based open-source implementation in JAX. This lifts OptiDICE (Lee et al., 2021) to the constrained problem and optimizes a stationary-distribution correction subject to a cost constraint instead of learning a policy directly (*i.e.*, soft-χ^2 divergence, divergence coefficient 0.5, cost upper bound 0.01).

SafeFQL. `SafeFQL` (Tayal et al., 2026) is flow Q-learning (FQL) (Park et al., 2025) with `FISOR`'s feasibility machinery. For this, we use a JAX-based open-source implementation. The behavior teacher is a conditional flow-matching vector field integrated with 10 Euler steps during training, and the deployed policy is the one-step student distilled from it. The distillation term enters the actor loss with coefficient 1. The reward critic is a two-head ensemble with an expectile value at $\tau_r = 0.9$, and the cost side uses the safe-expectile variant at $\tau_c = 0.9$ on top of an HJ reach-avoid cost critic with the dense margin label scaled by 25. All networks are $[512, 512, 512, 512]$-sized MLPs with layer normalization enabled in the critics and value and disabled in the actor.

FISOR. Unlike the other baselines, `FISOR` (Zheng et al., 2024) is not held to the same hyperparameter setup. We simply follow the reported hyperparameters by Zheng et al. (2024). The actor is a LayerNorm-ResNet diffusion policy with 3 blocks of width 256, dropout 0.1 and $[128, 128]$ conditioning, trained with batch size 2048; the critics and value are $[256, 256]$ two-head with LayerNorm disabled. DDPM uses $T = 5$ denoising steps, with 16 candidates sampled per observation at evaluation and the lowest-cost one executed. Both expectiles are 0.9, the HJ cost critic uses label scale 25 and upper bound 150, and the advantage temperatures are 3 for reward and 5 for cost.

We make this exception because the same hyperparameter setup understates the method: `FISOR` with our setup stays within budget on only $3/11$ `Safety-Gymnasium` tasks instead of $8/11$ with the reported hyperparameters. See Appendix F.5.

STEP. See Appendix C.

E HYPERPARAMETERS

Table 2 lists the optimization hyperparameters shared by all methods. Table 3 lists the constants of `STEP` fixed across all tasks. Table 4 lists the deployment knobs selected per task.

Table 2: Hyperparameters for RL optimization

Hyperparameter	Default	Hyperparameter	Default
Reward critic heads	2	Cost critic heads	4
Learning rate	0.0003	Batch size	256
Flow steps	10	Target update rate	0.005
γ	0.99	γ_c	0.99
d	$B\,(1-\gamma_c^H)/\big((1-\gamma_c)H\big)$	Reward normalization	$r \cdot H/(r_{\max} - r_{\min})$

d follows prior works (Liu et al., 2023a; Zheng et al., 2024).

Table 3: Hyperparameters for STEP

Hyperparameter	Default
N number of particles	10
ϵ trust region	0.25
α reward step size	0.05
η switch slack	-0.1
κ uncertainty weight	1.0
K refinement steps	$\{0, 1, 3, 5\}$
Reward value aggregation	min (selection), mean (backup)
Cost value aggregation	mean

Table 4: Deployment knobs of STEP per task.

Task	K	ϵ	κ	Task	K	ϵ	κ
AntVel	1	0.25	1	BallRun	1	0.5	1
CarButton1	0	0.25	1	CarCircle	0	0.25	1
CarButton2	0	0.25	0	CarRun	5	0.25	1
CarCircle1	3	0.25	3	DroneCircle	5	0.25	1
CarCircle2	1	0.5	1	DroneRun	0	0.25	1
CarGoal1	0	0.25	3	easydense	0	0.25	1
CarGoal2	0	0.25	1	easymean	1	0.25	1
CarPush1	3	0.25	1	easysparse	1	0.25	1
CarPush2	0	0.25	1	harddense	0	0.25	1
HalfCheetahVel	1	0.25	1	hardmean	5	0.25	1
SwimmerVel	0	0.25	1	hardsparse	5	0.25	1
AntCircle	0	0.25	1	mediumdense	0	0.25	0
AntRun	1	0.25	1	mediummean	0	0.25	0
BallCircle	1	0.5	1	mediumsparse	1	0.25	0

F Additional Experimental Results

F.1 Full table with standard deviation

Table 5: **Full results of Table 1 on safe RL tasks with standard deviations.**

	Task	BC		Safe-BC		BEAR-L		CPQ		COptiDICE		SafeFQL		FISOR		STEP	
		R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)	R (↑)	C (↓)
Safety-Gymnasium	CarButton1	0.15 ±0.05	7.28 ±1.75	0.04 ±0.03	1.83 ±1.03	0.01 ±0.25	4.81 ±5.00	0.43 ±0.05	35.9 ±4.8	0.09 ±0.05	4.93 ±1.50	0.50 ±0.04	18.9 ±1.1	−0.09 ±0.05	0.61 ±0.34	0.01 ±0.01	0.92 ±0.32
	CarButton2	0.13 ±0.03	7.78 ±1.79	−0.02 ±0.02	2.54 ±2.07	−0.12 ±0.09	1.70 ±1.15	0.42 ±0.07	38.8 ±6.4	−0.06 ±0.10	4.42 ±1.65	0.55 ±0.02	20.5 ±2.5	−0.03 ±0.03	0.53 ±0.32	−0.01 ±0.01	1.08 ±0.61
	CarPush1	0.22 ±0.03	2.25 ±0.59	0.15 ±0.02	1.03 ±0.62	0.41 ±0.04	2.80 ±1.01	0.10 ±0.02	6.40 ±3.25	0.22 ±0.02	2.09 ±0.83	0.42 ±0.03	1.96 ±0.55	0.24 ±0.02	0.85 ±0.31	0.27 ±0.03	0.74 ±0.26
	CarPush2	0.10 ±0.02	3.14 ±1.11	0.02 ±0.02	1.10 ±0.57	0.38 ±0.05	7.66 ±1.35	0.17 ±0.03	14.7 ±5.9	0.12 ±0.02	4.94 ±1.95	0.28 ±0.03	6.03 ±1.84	0.07 ±0.03	0.54 ±0.32	0.03 ±0.01	0.61 ±0.35
	CarGoal1	0.32 ±0.05	2.33 ±0.54	0.19 ±0.03	0.74 ±0.39	0.73 ±0.02	5.60 ±0.88	0.69 ±0.26	4.47 ±1.91	0.38 ±0.07	2.16 ±0.56	0.76 ±0.02	6.30 ±0.82	0.42 ±0.04	1.22 ±0.27	0.18 ±0.04	0.56 ±0.19
	CarGoal2	0.25 ±0.05	3.98 ±0.66	0.11 ±0.03	1.45 ±0.57	0.70 ±0.06	10.9 ±1.8	0.88 ±0.01	17.8 ±1.7	0.23 ±0.05	3.29 ±0.90	0.77 ±0.03	13.8 ±1.1	0.14 ±0.03	0.62 ±0.27	0.09 ±0.01	0.89 ±0.50
	CarCircle1	0.71 ±0.03	16.4 ±1.2	0.44 ±0.03	6.47 ±1.04	0.17 ±0.21	11.8 ±6.4	0.02 ±0.19	1.12 ±1.61	0.68 ±0.02	18.5 ±1.3	0.52 ±0.10	21.0 ±2.8	0.66 ±0.02	7.11 ±1.15	0.15 ±0.04	0.80 ±0.50
	CarCircle2	0.64 ±0.05	17.9 ±2.9	0.49 ±0.07	11.0 ±3.6	0.18 ±0.15	4.50 ±4.00	0.29 ±0.16	1.77 ±2.00	0.72 ±0.05	23.0 ±3.2	0.66 ±0.03	16.3 ±3.7	0.58 ±0.03	6.73 ±2.93	0.09 ±0.05	0.96 ±1.51
	AntVel	0.96 ±0.02	16.4 ±3.8	0.80 ±0.05	1.53 ±0.40	0.46 ±0.20	0.89 ±0.40	−0.60 ±0.23	0.01 ±0.01	0.98 ±0.01	13.2 ±2.6	0.94 ±0.01	0.00 ±0.00	0.89 ±0.01	0.00 ±0.00	0.85 ±0.03	0.32 ±0.06
	HalfCheetahVel	0.90 ±0.02	12.6 ±2.5	0.69 ±0.08	0.65 ±0.35	0.93 ±0.02	0.82 ±0.80	0.36 ±0.41	0.69 ±0.78	1.00 ±0.03	43.9 ±9.6	0.99 ±0.01	0.00 ±0.01	0.88 ±0.02	0.00 ±0.00	0.95 ±0.01	0.14 ±0.08
	SwimmerVel	0.42 ±0.04	7.80 ±1.35	0.45 ±0.08	1.92 ±2.87	0.28 ±0.18	4.27 ±4.66	0.09 ±0.17	6.99 ±15.4	0.63 ±0.08	22.5 ±4.5	−0.05 ±0.01	0.47 ±1.11	−0.03 ±0.04	0.48 ±0.96	0.47 ±0.06	0.12 ±0.12
	Average	**0.44**	**8.89**	**0.31**	**2.75**	**0.38**	**5.07**	**0.26**	**11.7**	**0.45**	**13.0**	**0.58**	**9.57**	**0.34**	**1.70**	**0.28**	**0.65**
Bullet-Safety-Gym	BallRun	0.47 ±0.05	9.98 ±0.63	0.21 ±0.04	2.21 ±1.35	0.05 ±0.25	5.09 ±6.23	0.87 ±0.41	17.0 ±4.6	0.82 ±0.03	14.3 ±0.5	0.08 ±0.06	4.45 ±4.48	0.13 ±0.08	3.09 ±2.61	0.25 ±0.02	1.05 ±0.43
	BallCircle	0.71 ±0.04	9.35 ±0.72	0.29 ±0.04	0.38 ±0.21	0.66 ±0.05	5.94 ±2.20	0.64 ±0.03	0.03 ±0.09	0.77 ±0.03	10.8 ±0.8	0.34 ±0.03	0.00 ±0.01	0.33 ±0.03	0.00 ±0.00	0.26 ±0.03	0.97 ±0.28
	CarRun	0.85 ±0.03	2.84 ±1.20	0.77 ±0.04	0.38 ±0.13	0.96 ±0.03	4.38 ±2.57	0.89 ±0.06	4.20 ±3.03	0.96 ±0.03	5.18 ±0.77	0.69 ±0.03	0.00 ±0.00	0.36 ±0.15	0.32 ±0.74	0.77 ±0.06	0.89 ±0.17
	CarCircle	0.29 ±0.04	8.41 ±1.17	0.17 ±0.03	0.69 ±0.76	0.52 ±0.09	1.98 ±1.69	0.49 ±0.16	0.23 ±0.47	0.56 ±0.05	9.50 ±0.87	0.55 ±0.06	5.13 ±4.69	0.35 ±0.03	0.11 ±0.30	0.07 ±0.01	0.72 ±0.61
	DroneRun	0.60 ±0.05	11.6 ±1.7	0.52 ±0.08	1.00 ±0.83	0.38 ±0.22	8.80 ±5.89	0.37 ±0.09	10.0 ±9.7	0.60 ±0.02	12.1 ±1.9	0.56 ±0.14	12.8 ±5.1	0.35 ±0.08	3.43 ±2.77	0.54 ±0.03	0.58 ±0.23
	DroneCircle	0.72 ±0.07	10.1 ±2.5	0.46 ±0.04	0.95 ±0.32	0.52 ±0.02	1.09 ±0.62	−0.26 ±0.01	0.73 ±0.52	0.64 ±0.04	14.5 ±0.9	0.02 ±0.07	2.28 ±1.39	0.50 ±0.01	0.06 ±0.06	0.42 ±0.02	0.74 ±0.23
	AntRun	0.77 ±0.03	15.0 ±3.1	0.59 ±0.04	1.65 ±0.30	0.63 ±0.11	6.02 ±4.00	0.10 ±0.03	0.04 ±0.08	0.83 ±0.05	21.3 ±3.8	0.31 ±0.02	0.04 ±0.04	0.43 ±0.07	0.73 ±0.58	0.59 ±0.01	0.98 ±0.23
	AntCircle	0.52 ±0.05	16.1 ±0.9	0.36 ±0.04	6.48 ±1.91	0.30 ±0.08	3.60 ±2.22	0.00 ±0.00	0.00 ±0.00	0.51 ±0.06	12.3 ±2.8	0.56 ±0.06	1.31 ±0.36	0.22 ±0.02	0.03 ±0.06	0.46 ±0.04	0.16 ±0.06
	Average	**0.62**	**10.4**	**0.42**	**1.72**	**0.50**	**4.61**	**0.39**	**4.03**	**0.71**	**12.5**	**0.39**	**3.25**	**0.33**	**0.97**	**0.42**	**0.76**
MetaDrive	easysparse	0.67 ±0.07	6.55 ±1.60	0.35 ±0.02	0.00 ±0.00	0.34 ±0.14	1.06 ±0.61	−0.06 ±0.00	0.23 ±0.10	0.92 ±0.10	17.8 ±1.5	0.27 ±0.31	3.66 ±3.86	0.56 ±0.02	0.44 ±0.66	0.53 ±0.05	0.94 ±0.36
	easymean	0.74 ±0.08	8.16 ±2.51	0.38 ±0.06	0.00 ±0.00	0.49 ±0.11	3.07 ±1.51	−0.03 ±0.09	0.33 ±0.30	0.74 ±0.10	14.4 ±1.5	0.33 ±0.26	4.27 ±3.29	0.51 ±0.02	0.17 ±0.29	0.55 ±0.03	0.83 ±0.25
	easydense	0.61 ±0.11	5.94 ±1.99	0.39 ±0.07	0.00 ±0.00	0.37 ±0.11	1.02 ±0.73	−0.06 ±0.00	0.25 ±0.06	0.52 ±0.06	10.5 ±0.8	0.14 ±0.22	2.13 ±2.96	0.55 ±0.02	0.49 ±0.47	0.60 ±0.02	0.83 ±0.32
	mediumsparse	0.81 ±0.08	4.06 ±1.71	0.63 ±0.05	0.02 ±0.01	0.81 ±0.12	1.60 ±0.71	0.01 ±0.07	0.62 ±0.30	0.82 ±0.13	8.90 ±1.35	0.22 ±0.27	2.31 ±2.42	0.77 ±0.02	0.38 ±0.35	0.79 ±0.05	0.87 ±0.49
	mediummean	0.79 ±0.07	4.19 ±1.40	0.52 ±0.07	0.02 ±0.01	0.84 ±0.07	2.12 ±0.98	0.01 ±0.07	0.43 ±0.32	0.74 ±0.05	8.17 ±0.63	0.04 ±0.09	0.41 ±0.21	0.70 ±0.02	0.66 ±0.23	0.66 ±0.04	0.84 ±0.34
	mediumdense	0.79 ±0.05	4.01 ±0.85	0.59 ±0.04	0.04 ±0.02	0.85 ±0.10	2.71 ±0.85	−0.01 ±0.06	0.42 ±0.20	0.82 ±0.13	8.72 ±1.28	0.27 ±0.20	0.96 ±1.07	0.76 ±0.03	0.30 ±0.27	0.78 ±0.06	0.71 ±0.36
	hardsparse	0.49 ±0.05	5.36 ±1.35	0.35 ±0.03	0.05 ±0.02	0.37 ±0.08	0.89 ±1.03	0.07 ±0.09	1.02 ±0.72	0.26 ±0.03	5.95 ±0.55	0.12 ±0.10	1.57 ±1.45	0.42 ±0.01	0.74 ±0.21	0.43 ±0.02	0.84 ±0.31
	hardmean	0.46 ±0.03	5.12 ±1.00	0.33 ±0.03	0.03 ±0.02	0.37 ±0.05	1.00 ±0.52	0.08 ±0.11	0.86 ±0.49	0.29 ±0.06	6.63 ±1.01	0.05 ±0.05	0.42 ±0.41	0.38 ±0.03	0.69 ±0.42	0.41 ±0.02	0.95 ±0.28
	harddense	0.40 ±0.07	3.55 ±0.74	0.33 ±0.04	0.06 ±0.01	0.34 ±0.17	1.13 ±1.04	0.08 ±0.08	0.90 ±0.36	0.21 ±0.05	4.87 ±1.13	0.17 ±0.11	1.04 ±1.56	0.39 ±0.03	0.68 ±0.32	0.39 ±0.03	0.79 ±0.32
	Average	**0.64**	**5.22**	**0.43**	**0.03**	**0.53**	**1.62**	**0.01**	**0.56**	**0.59**	**9.55**	**0.18**	**1.86**	**0.56**	**0.51**	**0.57**	**0.84**
All (28 tasks)		**0.55**	**8.15**	**0.38**	**1.58**	**0.46**	**3.83**	**0.22**	**5.93**	**0.57**	**11.7**	**0.39**	**5.29**	**0.41**	**1.11**	**0.41**	**0.74**

F.2 Design choice of value function

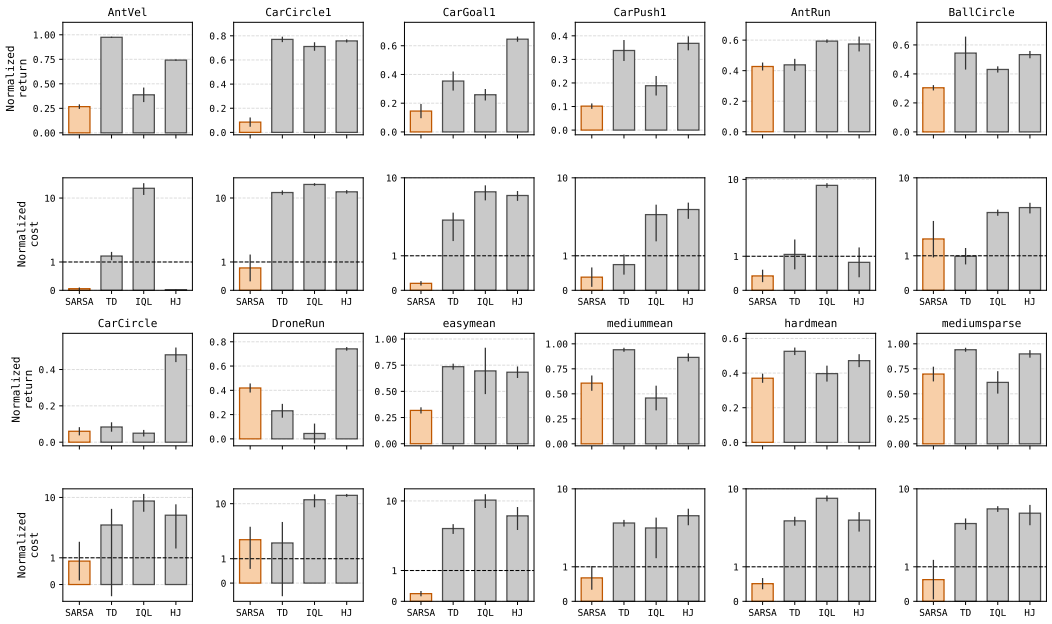


Figure 5: **Ablation study for the value function choice.**

The cost critic guides both action refinement in (7) and filtering in (9). We compare four cost backups on 12 tasks with all other components fixed. `SARSA` uses the recorded next action, whereas `TD` uses the action selected by `STEP`. `IQL` (Kostrikov et al., 2022) uses a low expectile over dataset actions. `HJ` adapts the reachability backup of Zheng et al. (2024) to the recorded next action, replacing cumulative cost with a max-based violation measure:

$$
\begin{aligned}
y^{\mathrm{SARSA}} &= c + \gamma_c \bar{Q}_c(s', a'), \\
y^{\mathrm{TD}} &= c + \gamma_c \bar{Q}_c\big(s', a^*(s')\big), \\
y^{\mathrm{IQL}} &= c + \gamma_c V_c(s'), \qquad V_c = \operatorname*{argmin}_V \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\ell^2_{\tau_{\mathrm{IQL}}}\big(\bar{Q}_c(s,a) - V(s)\big)\Big], \\
y^{\mathrm{HJ}} &= (1-\gamma_c)h(s) + \gamma_c \max\big\{h(s), \bar{Q}_h(s', a')\big\},
\end{aligned}
$$

where $\bar{Q}_c$ and $\bar{Q}_h$ denote target networks, $\ell^2_\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2$ is the expectile loss, and $\tau_{\mathrm{IQL}} < 0.5$, so that V_c approximates the cost of the lowest-cost action in the data support. We use $\tau_{\mathrm{IQL}} = 0.1$ in Figure 5 and the same constraint-violation signal $h(s)$, following `FISOR` and `SafeFQL`.

Figure 5 shows that `SARSA` satisfies the budget most consistently; the alternatives often achieve higher returns with more violations. We attribute this pattern to differences in their cost targets. `TD` selects the next action using the same learned cost estimates that drive its updates, which can reinforce the underestimation. `IQL` favors low-cost actions in the dataset, potentially understating the cost of following `STEP`. `HJ` does not accumulate repeated violations and thus does not directly represent the episode budget; it meets the budget only on `AntVel` and `AntRun` in this comparison. `SARSA` instead estimates cumulative cost under the behavior policy using recorded next actions, providing a target grounded in the data and aligned with the budget's cumulative-cost objective.

F.3 Design choice of gradient update

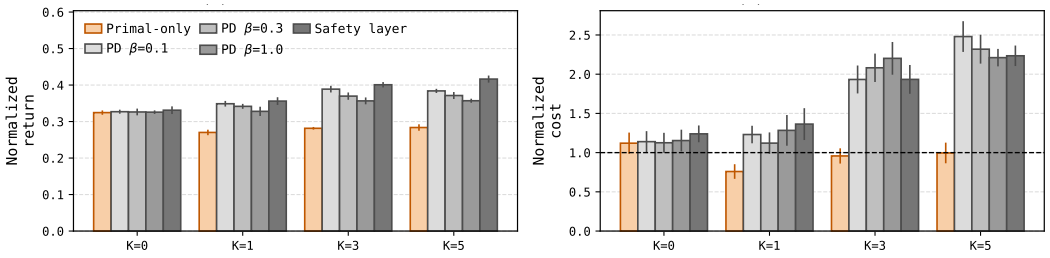


Figure 6: **Ablation study for the gradient update choice.** Primal only is the proposed solution, PD is primal-dual optimization, and safety layer comes from Dalal et al. (2018).

Figure 6 compares refinement rules, averaged over 28 tasks and 8 seeds. Besides the primal-dual update (5) with dual step size β, we consider the safety layer of Dalal et al. (2018), the closest prior rule to STEP. Like STEP, it corrects an action at deployment without retraining, and its correction is the same closed-form minimum-norm step onto a linearized constraint as our cost branch. It differs in what it linearizes and when: it applies the step once, after K reward-ascent steps, to a one-step cost model $c(s,a) \approx c_w(s) + g_w(s)^\top a$, rather than to Q_c at every step.

By construction, primal-only and PD reduce to constrained best-of-N for $K = 0$. PD raises return with K but exceeds the budget for every β, with cost up to 2.5, because $K \leq 5$ steps are too few for the multiplier to grow before reward ascent dominates. The safety layer attains the highest return but exceeds the budget at every K, with cost from 1.2 to 2.2, as meeting a per-step allowance does not account for the cost an action incurs later in the episode. The primal-only switch instead first pulls violating candidates below the bar, then spends the remaining slack on return, reaching but not exceeding the budget at $K = 5$.

F.4 Inference time

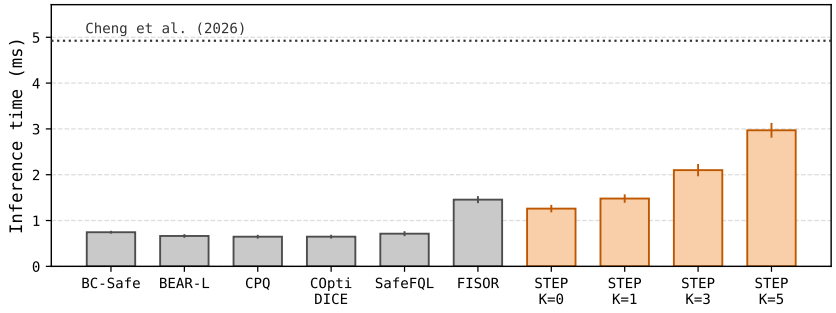


Figure 7: **Inference time comparison.**

Figure 7 reports the wall-clock time per action. Baselines that generate a single action take 0.65–0.75 ms. STEP with $K = 0$ takes about 1.3 ms because it samples and scores N candidates, which is close to FISOR at 1.5 ms. Each refinement step adds about 0.35 ms, so $K = 5$ takes about 3 ms. For reference, the diffusion-based method of Cheng et al. (2026) reports approximately 4.9 ms per action (dotted line in Figure 7), which we did not measure on our hardware.

F.5 HYPERPARAMETER COMPARISON ON FISOR

Table 6: **Ablation on FISOR.** We compare two hyperparameter variants of FISOR.

Safety-Gymnasium					Bullet-Safety-Gym					MetaDrive				
Task	R(↑)	C(↓)	R(↑)	C(↓)	Task	R(↑)	C(↓)	R(↑)	C(↓)	Task	R(↑)	C(↓)	R(↑)	C(↓)
CarButton1	−0.09 ±0.05	0.61 ±0.34	−0.02 ±0.02	1.21 ±0.41	BallRun	0.13 ±0.08	3.09 ±2.61	0.03 ±0.15	5.28 ±4.32	easysparse	0.56 ±0.02	0.44 ±0.66	0.37 ±0.06	0.29 ±0.10
CarButton2	−0.03 ±0.03	0.53 ±0.32	−0.05 ±0.04	1.40 ±0.42	BallCircle	0.33 ±0.03	0.00 ±0.00	0.14 ±0.03	0.70 ±0.86	easymean	0.51 ±0.02	0.17 ±0.29	0.28 ±0.13	0.09 ±0.08
CarPush1	0.24 ±0.02	0.85 ±0.31	0.16 ±0.03	1.99 ±0.76	CarRun	0.36 ±0.15	0.32 ±0.74	−0.12 ±0.20	2.16 ±4.84	easydense	0.55 ±0.02	0.49 ±0.47	0.32 ±0.04	0.15 ±0.12
CarPush2	0.07 ±0.03	0.54 ±0.32	0.04 ±0.05	2.15 ±1.52	CarCircle	0.35 ±0.03	0.11 ±0.30	0.10 ±0.04	0.59 ±0.85	mediumsparse	0.77 ±0.02	0.38 ±0.35	0.46 ±0.22	0.76 ±0.75
CarGoal1	0.42 ±0.04	1.22 ±0.27	0.24 ±0.11	2.00 ±1.20	DroneRun	0.35 ±0.08	3.43 ±2.77	0.28 ±0.13	7.21 ±3.48	mediummean	0.70 ±0.02	0.66 ±0.23	0.39 ±0.19	0.71 ±0.84
CarGoal2	0.14 ±0.03	0.62 ±0.27	0.11 ±0.03	2.07 ±1.35	DroneCircle	0.50 ±0.01	0.06 ±0.06	0.17 ±0.06	1.69 ±0.55	mediumdense	0.76 ±0.03	0.30 ±0.27	0.39 ±0.26	0.91 ±1.81
CarCircle1	0.66 ±0.02	7.11 ±1.15	0.69 ±0.03	12.16 ±2.32	AntRun	0.43 ±0.07	0.73 ±0.58	0.38 ±0.08	0.10 ±0.09	hardsparse	0.42 ±0.01	0.74 ±0.21	0.27 ±0.03	0.09 ±0.14
CarCircle2	0.58 ±0.03	6.73 ±2.93	0.61 ±0.04	14.46 ±3.48	AntCircle	0.22 ±0.02	0.03 ±0.06	0.06 ±0.02	0.00 ±0.00	hardmean	0.38 ±0.03	0.69 ±0.42	0.28 ±0.03	0.15 ±0.19
AntVel	0.89 ±0.01	0.00 ±0.00	0.62 ±0.03	0.00 ±0.00						harddense	0.39 ±0.03	0.68 ±0.32	0.34 ±0.03	0.34 ±0.38
HalfCheetahVel	0.88 ±0.02	0.00 ±0.00	0.20 ±0.05	0.00 ±0.00										
SwimmerVel	−0.03 ±0.04	0.48 ±0.96	−0.04 ±0.03	0.18 ±0.38										
Average	0.34	1.70	0.23	3.42	**Average**	0.33	0.97	0.13	2.21	**Average**	0.56	0.51	0.34	0.39