Conservatism Controllable Compositional Guidance for Offline Safe Reinforcement Learning

Ruiqi Xue[1,2] , Shenghe Hu[1,2], Xin Gao[1,2], Jing-Wen Yang[3]*
[1] National Key Laboratory for Novel Software Technology, Nanjing University
[2] School of Artificial Intelligence, Nanjing University
Nanjing, China
[3] Tencent Game AI Center
Shenzhen, China
xuerq@lamda.nju.edu.cn
{hush,231830150}@smail.nju.edu.cn
jingwenyang@tencent.com

Abstract

Learning constraint-satisfying policies from offline data without risky online interaction is crucial for reliable deployment of reinforcement learning (RL) agents. Representative methods leverage Implicit Q-Learning and advantage-weighted regression to learn value functions and balance the reward–safety trade-off, demonstrating solid safety performance in safety-critical hard constraint scenarios. However, these methods typically couple reward and safety optimization during offline training and rely on a predefined unified factor to balance the reward–safety trade-off (conservatism). Which ignores the fact that different tasks, due to their distinct reward and cost functions, may require different degrees of conservatism to satisfy safety constraints, resulting in sub-optimal performance. To address this limitation, we propose Co3G, a compositional generative model with test-time controllable conservatism for offline safe RL. During training, Co3G decouples the optimization objectives by separately employing reward-optimal guidance and safety-optimal guidance to induce high-reward and high-safety actions. During deployment, Co3G combines the reward and safety objectives via compositional generation, and further provides three mechanisms—manual tuning, rejection sampling, and RL—to balance the reward and safety guidance scales during deployment, thereby achieving a superior reward–safety trade-off. Extensive experiments on the OSRL benchmark show that Co3G delivers strong safety performance under stringent safety constraints while retaining flexible control over conservatism, significantly outperforming previous hard-constraint and soft-constraint baselines.

1 Introduction

Safe reinforcement learning (RL), which focuses on learning policies that comply with predefined safety constraints, plays a critical role in real-world applications, including autonomous driving [23], robotic control [50], and large language model value alignment [28]. However, a key limitation of conventional online RL lies in its dependence on trial-and-error exploration, which is inherently risky and may hinder its deployment in safety-critical scenarios [33]. To address these challenges, offline safe RL has emerged as a promising framework, aiming to learn safe policies solely from pre-collected datasets while avoiding potentially hazardous online interactions [39, 52].

*Corresponding Author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Numerous methods have been proposed for offline safe policy learning. Earlier approaches seek to integrate conservative value estimation with safe RL to mitigate the extrapolation error arising in offline settings [21, 22, 46]. Although achieving some progress, their safety performance are still unsatisfactory. Consequently, greater attention has been directed toward more stable behavior cloning (BC)-based methods. Methods such as CDT [38] introduce Decision Transformer (DT) [8] to perform safe decision-making conditioned on the constraint budget, whereas methods such as CAPS [7] and C2IQL [54] leverage advantage-weighted regression (AWR) in Implicit Q-Learning (IQL) [30] to extract safe policies, achieving stronger performance in soft-constraint settings and ideally allowing policy conservatism to be adjusted according to different budgets. Nevertheless, such controllability is often effective only when the budget is sufficiently large; in more stringent hard-constraint scenarios, they still struggle to satisfy safety requirements.

To achieve safer decision-making under hard constraints, methods such as FISOR [52] integrate Hamilton–Jacobi (HJ) reachability analysis [4] with IQL, outperforming previous approaches in terms of average safety performance. However, these methods typically rely on a predefined unified factor to balance the reward–safety trade-off (conservatism) across all tasks, ignoring the fact that different tasks, owing to their distinct reward and cost functions, may require different levels of conservatism to satisfy safety constraints. Consequently, such methods may be overly aggressive on some challenging tasks, failing to meet the stringent safety requirements, while being overly conservative on others, potentially causing task failure. This naturally raises a key question: **can the idea of test-time conservatism adjustment, which has proven useful in soft-constraint methods, be introduced to hard-constraint algorithms to enable a more fine-grained reward–safety trade-off?**

To address the aforementioned challenge, we propose **Co**nservatism **Co**ntrollable **Co**mpositional **G**uidance (Co3G), a novel compositional generative model with test-time controllability over conservatism. Specifically, Co3G first trains decoupled reward-optimal and cost-optimal conditional generative models to separately generate high-reward and safe actions. During deployment, Co3G then performs compositional guided generation using reward and cost guidance scales to enable both safe and high-quality decision-making. In addition, Co3G provides three mechanisms—manual tuning, rejection sampling, and RL—for adjusting the reward and safety guidance scales, thereby achieving a better reward–safety trade-off. Extensive experiments on the safety-critical Safety Gymnasium tasks of the OSRL benchmark [39] show that Co3G delivers strong safety performance under stringent safety constraints while retaining flexible control over conservatism, significantly outperforming both previous hard-constraint and soft-constraint baselines.

2 Preliminaries

2.1 Offline Safe RL

In this work, we consider safe RL under hard constraints, which can be modeled as a hard-constrained CMDP, represented by the tuple $\langle S, A, r, h, c, P, \gamma \rangle$. Here, S and A denote the state and action spaces, respectively; $r : S \times A \to [-R_{\text{max}}, R_{\text{max}}]$ is the reward function; $h : S \to [h_{\text{min}}, h_{\text{max}}]$ is the constraint violation function; $c : S \to 0, 1$ is the cost function; $P : S \times A \times S \to [0, 1]$ defines the transition dynamics; and $\gamma \in (0, 1)$ is the discount factor. In general, $c(s) = \mathbb{I}(h(s) > 0)$, meaning that a state is unsafe if $h(s) > 0$ and safe otherwise. A policy $\pi : S \to \Delta(A)$ maps each state to a distribution over actions. Given a policy π, the expected discounted reward return and cost return are defined as $R(\pi) = \mathbb{E}_{\tau \sim P_\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$ and $C(\pi) = \mathbb{E}_{\tau \sim P_\pi} \left[\sum_{t=0}^{\infty} \gamma^t c(s_t)\right]$, where $\tau = (s_0, a_0, s_1, a_1, \dots) \sim P_\pi$ is a trajectory induced by π and the environment dynamics P. Accordingly, the objective in a hard-constrained CMDP is to find a policy that maximizes reward return while incurring zero cost return.

In the offline setting, we are given an offline dataset $\mathcal{D}$ generated by the behavior policy $\hat{\pi}$. Now, the goal is to learn a safe policy purely from this dataset. To avoid extrapolation error in offline RL, the optimization objective is formulated as:

$$\max_{\pi} \; R(\pi), \; s.t. \, C(\pi) \leq 0 \; ; \; D(\pi || \hat{\pi}) \leq \epsilon, \tag{1}$$

where $D(\pi||\hat{\pi})$ is a divergence term (e.g., KL divergence $D_{\text{KL}}(\pi||\hat{\pi})$) used to prevent distribution shift. Meanwhile, in most practical applications, a continuous constraint violation function h is often unavailable. Accordingly, in this work we define h as also a binary function: $h(s) = h_{\text{min}} \leq 0$ if $c(s) = 0$, and $h(s) = h_{\text{max}} > 0$ otherwise.

2.2 HJ Reachability Analysis

HJ reachability analysis [4] is one of the most widely used approaches for solving hard-constrained safe RL. It typically assumes a deterministic policy π and defines the feasible set accordingly.

Definition 2.1 (Feasible set [49])**.** The feasible set of a specific policy π can be defined as

$$S_f^\pi := \{s \in S | h(s_t^\pi | s_0 = s) \leq 0, \forall t \in \mathbb{N}\}. \tag{2}$$

The largest feasible set S_f^* is a subset of S composed of states from which there exists at least one policy that keeps the system satisfying the constraint, i.e.,

$$S_f^* := \{s \in S | \exists \pi, h(s_t^\pi | s_0 = s) \leq 0, \forall t \in \mathbb{N}\}. \tag{3}$$

Above, $h(s_t^\pi | s_0 = s)$ specifies the state constraint sequence $\{h(s_0^\pi), h(s_1^\pi), \ldots, | s_0 = s, \pi\}$.

Definition 2.2 (Optimal feasible value function [49])**.** The optimal feasible state-value function V_h^*, and the optimal feasible action-value function Q_h^* are defined as

$$\begin{aligned} V_h^*(s) &:= \min_\pi V_h^\pi(s) := \min_\pi \max_{t \in \mathbb{N}} h(s_t), s_0 = s, a_t \sim \pi(\cdot | s_t), \\ Q_h^*(s,a) &:= \min_\pi Q_h^\pi(s,a) := \min_\pi \max_{t \in \mathbb{N}} h(s_t), s_0 = s, a_0 = a, a_t \sim \pi(\cdot | s_t), \end{aligned} \tag{4}$$

where V_h^π represents the maximum constraint violations in the trajectory induced by policy π. Thus, the largest feasible set can be written as $S_f^* := \{s | V_h^*(s) \leq 0\}$.

Based on Definition 2.1, once S_f^* is obtained, the feasibility of states in $\mathcal{D}$ can be determined. Furthermore, Definition 2.2 indicates that S_f^* can be obtained by computing the optimal feasible value function $V_h^*(s)$. Thus, the feasible Bellman operator $\mathcal{B}^*$ [12] is proposed:

$$\mathcal{B}^* Q_h(s,a) := (1-\gamma) h(s) + \gamma \max\{h(s), V_h^*(s')\},\ V_h^*(s') = \min_{a'} Q_h(s', a'). \tag{5}$$

Based on the above formulation, FISOR [52] replaces the cost value function in Equation (1) with the feasible value function, and reformulates it equivalently into an optimal advantage guided form:

$$\begin{aligned} \textbf{Feasible:} \max_\pi \mathbb{E}_{a \sim \pi} \left[A_r^*(s,a) \cdot \mathbb{I}_{V_h^*(s) \leq 0} \right] \quad & \textbf{Infeasible:} \max_\pi \mathbb{E}_{a \sim \pi} \left[-A_h^*(s,a) \cdot \mathbb{I}_{V_h^*(s) > 0} \right] \\ \text{s.t.} \int_{\{a | Q_h^*(s,a) \leq 0\}} \pi(a|s) da = 1, \forall s \in \mathcal{S}_f^* \quad & \text{s.t.} \int_a \pi(\cdot|s) da = 1 \\ D_{\mathrm{KL}}(\pi || \hat{\pi}) \leq \epsilon \quad & D_{\mathrm{KL}}(\pi || \hat{\pi}) \leq \epsilon \end{aligned}, \tag{6}$$

where $A^*(s,a) = Q^*(s,a) - V^*(s)$ are optimal advantage functions, $\mathbb{I}$ is the indicator function. For the above optimization object, it is then proved that the closed-form solution is

$$\pi^*(a|s) \propto w(s,a) \hat{\pi}(a|s),\ w(s,a) = \begin{cases} \exp(\alpha_1 A_r^*(s,a)) \cdot \mathbb{I}_{Q_h^*(s,a) \leq 0} & V_h^*(s) \leq 0 \\ \exp(-\alpha_2 A_h^*(s,a)) & V_h^*(s) > 0 \end{cases}, \tag{7}$$

where α_1 and α_2 are two positive values. Although the aforementioned HJ reachability analysis is originally formulated for deterministic policies, the derivation from Equation (1) to Equation (6) replaces the value function with the policy-independent optimal value function. As a result, both Equation (6) and Equation (7) naturally extend to the stochastic-policy setting.

3 Method

This section gives the detailed Co3G, a novel offline safe policy learning algorithm. Section 3.1 presents the reward–safety score decomposition of Equation (7) and the corresponding theoretical analysis, Section 3.2 introduces three approaches for adjusting the reward and safety guidance scales to control conservatism, whereas Section 3.3 presents a practical flow-matching optimization procedure for the decomposed policy.

3.1 Decoupled Reward-Safety Compositional Score Matching

The reason why conventional IQL-based methods struggle to achieve desirable policy conservatism is that the objectives of reward maximization and safety satisfaction are coupled during training. As a result, the level of conservatism is effectively fixed once training is completed. Therefore, to enable deployment-phase adjustment of conservatism, a natural idea is to decouple the reward and safety objectives during training. To this end, we introduce the compositional score matching [37], which enables decoupled score learning during training while compositional generation during deployment.

First, under feasible states, the optimal policy in Equation (7) can be written as

$$\pi^*(a|s) \propto \exp(\alpha_1 A_r^*(s,a)) \cdot \mathbb{I}_{Q_h^*(s,a)\leq 0} \cdot \hat{\pi}(a|s) \propto (\frac{f_1(A_r^*(s,a))}{Z_1(s)})^{\alpha_1} \cdot (\frac{f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0})}{Z_3(s)})^{\alpha_3} \cdot \hat{\pi}(a|s), \tag{8}$$

where $Z_1(s)$ and $Z_3(s)$ are normalization terms independent of the action, f_1 is the exponential function, f_3 is the clipped linear function $\min(\max(x,0),1)$, and α_3 can be any positive scalar, since $f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0}) = \mathbb{I}_{Q_h^*(s,a)\leq 0}$ and raising an indicator function to any positive power leaves it unchanged. We then define the optimality variable as a binary random variable $o \in \{\emptyset, 0, 1\}$ as in [13]. In particular, we consider a reward optimality variable o_r and an indicator optimality variable o_I, whose corresponding likelihoods are defined by f_1 and f_3:

$$p(o_r|s,a) = \frac{f_1(A_r^*(s,a))}{Z_1(s)},\ p(o_I|s,a) = \frac{f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0})}{Z_3(s)}. \tag{9}$$

Similarly, the optimal policy for infeasible states can be written as

$$\pi^*(a|s) \propto \exp(-\alpha_2 A_h^*(s,a)) \cdot \hat{\pi}(a|s) \propto (\frac{f_2(-A_h^*(s,a))}{Z_2(s)})^{\alpha_2} \cdot \hat{\pi}(a|s), \tag{10}$$

where f_2 is the exponential function and $Z_2(s)$ is the normalization term. The likelihood of feasible optimality variable o_h can be defined as $p(o_h|s,a) = \frac{f_2(-A_h^*(s,a))}{Z_2(s)}$. After rewriting Equation (6) in terms of the above optimality variables, we can decouple the reward and safety objectives through the log operator, leading to the following compositional score matching objective:

Theorem 3.1. *Let $o_{I,h} = \mathbb{I}_{V_h^*(s)\leq 0} o_I + \mathbb{I}_{V_h^*(s)>0} o_h$. Then, there exist positive scalars w_r^* and w_c^* such that the score of the optimal policy in Equation* (7) *satisfies*

$$\begin{aligned}\nabla_a \log \pi^*(a|s) = \nabla_a \log \hat{\pi}(a|s) + w_r^* \mathbb{I}_{V_h^*(s)\leq 0}(\nabla_a \log \hat{\pi}(a|s,o_r) - \nabla_a \log \hat{\pi}(a|s)) \\ + w_c^*(\nabla_a \log \hat{\pi}(a|s,o_{I,h}) - \nabla_a \log \hat{\pi}(a|s)).\end{aligned} \tag{11}$$

Theorem 3.1 decouples the reward and safety optimization, such that training only requires learning the reward-optimal policy $\hat{\pi}(a|s,o_r)$ and the safety-optimal policy $\hat{\pi}(a|s,o_{I,h})$ separately, while deployment is carried out via compositional generation with guidance scales w_r and w_c.

3.2 Conservatism Control with Guidance Scale Adjustment

The previous section derives the compositional score matching form of the optimal policy for feasibility-guided safe RL, which decouples the reward and safety optimization objectives. However, at deployment time, it still requires appropriate guidance scales to achieve a desired reward–safety balance. Therefore, we present three practical strategies for setting the guidance scales.

Manual. The most straightforward strategy is to adjust the guidance scales according to actual deployment performance. Specifically, we begin with an initial guidance-scale pair $(w_{r,0}, w_{c,0})$ that is sufficiently conservative. If the resulting policy satisfies the safety constraints, we moderately increase the reward guidance scale while keeping the cost guidance scale fixed. Otherwise, we increase the cost guidance scale while keeping the reward guidance scale fixed. By repeating this procedure, we ultimately select the guidance-scale pair that yields the best reward–safety trade-off.

Rejection Sampling (RS). **Manual** requires hand-designed guidance scales, whereas **RS** offers an automatic alternative. Specifically, given the optimal value functions Q_r and Q_h learned during training, for each state s, we first sample M^2 candidate guidance-scale pairs $\{(w_{r,i}, w_{c,i})\}_{i=1}^{M^2}$, and

for each pair generate N noise samples $\{z_{i,j}\}_{j=1}^{N}$. These inputs are then used to produce $M^2 \times N$ candidate actions. Among the actions satisfying $Q_h \leq \delta_h$ (if exists), where δ_h is a hyperparameter, we finally select the one with the highest Q_r as the output. Otherwise, we select the action with lowest Q_h as the output.

Reinforcement Learning (RL). In **RS**, safe actions are selected using the optimal feasible value function Q_h learned during training. However, since Q_h is obtained via expectile regression and is therefore policy-independent, it cannot accurately evaluate safety violations induced by the policy itself. To overcome this limitation, we further propose a RL-based approach for more adaptive guidance-scale generation. Specifically, the previously defined score-matching policy can be formulated as $\pi(a \mid s, w_r, w_c, z)$ (z is noise). We then introduce a latent policy that takes the state s as input and produces the guidance scales and generation noise, namely $\pi_l(w_r, w_c, z \mid s)$. The outputs of this latent policy are subsequently passed to π to generate the final action. Since our score matching policy can be regarded as a batch-constrained base model [14], the offline issue can be alleviated to some extent when training π_l, which allows us to incorporate online algorithms for its optimization. In this paper, we adapt the RESPO [15] algorithm to make it suitable for training π_l.

Specifically, similar to Wagenmaker et al. [44], we first train two aggregated policy value networks

$$\begin{aligned}\mathcal{L}_{Q_r^A} &= \mathbb{E}_{(s,a,s',r)\sim\mathcal{D},(w_r',w_c',z')\sim\pi_l(\cdot|s'),a'\sim\pi(\cdot|s',w_r',w_c',z')}\left[r+\gamma Q_r^A(s',a')-Q_r^A(s,a)\right],\\ \mathcal{L}_{Q_c^A} &= \mathbb{E}_{(s,a,s',c)\sim\mathcal{D},(w_r',w_c',z')\sim\pi_l(\cdot|s'),a'\sim\pi(\cdot|s',w_r',w_c',z')}\left[c+\gamma Q_c^A(s',a')-Q_c^A(s,a)\right],\end{aligned} \tag{12}$$

where Q_c^A is updated directly using the cost c, rather than h. The two value networks are then distilled into two latent value networks

$$\begin{aligned}\mathcal{L}_{Q_r^l} &= \mathbb{E}_{s\sim\mathcal{D},w_r,w_c\sim U(0,b_g),z\sim\mathcal{N}(0,1),a\sim\pi(\cdot|s,w_r,w_c,z)}\left[Q_r^l(s,w_r,w_c,z)-Q_r^A(s,a)\right],\\ \mathcal{L}_{Q_c^l} &= \mathbb{E}_{s\sim\mathcal{D},w_r,w_c\sim U(0,b_g),z\sim\mathcal{N}(0,1),a\sim\pi(\cdot|s,w_r,w_c,z)}\left[Q_c^l(s,w_r,w_c,z)-Q_c^A(s,a)\right],\end{aligned} \tag{13}$$

where U denotes the uniform distribution, b_g is a hyperparameter that defines the bound of w_r, w_c produced by π_l, and z output by π_l is also clipped to $[-b_l, b_l]$, where b_l is another hyperparameter. Finally, let $e = (w_r, w_c, z)$, π_l and the Lagrange multiplier are optimized by

$$\min_{\pi_l}\max_{\lambda}\mathcal{L}_{\pi_l,\lambda} = \min_{\pi_l}\max_{\lambda}\left(\mathbb{E}_{s\sim\mathcal{D},e\sim\pi_l(\cdot|s)}\left[\left[-Q_r^l(s,e)+\lambda\cdot Q_C^l(s,e)\cdot(1-p(s))+Q_C^l(s,e)\cdot p(s)\right]\right]\right), \tag{14}$$

where $p(s) = \frac{V_h(s)-h_{\min}}{h_{\max}-h_{\min}}$ (V_h is the learned optimal feasible value function) acts as a feasible indicator bounded in $[0, 1]$, and is used to smooth the policy optimization process, consistent with RESPO.

3.3 Practical Algorithm

During the training of π, we first train the optimal reward value function via expectile regression

$$\begin{aligned}\mathcal{L}_{V_r} &= \mathbb{E}_{(s,a)\sim\mathcal{D}}[\mathcal{L}^\tau(Q_r(s,a)-V_r(s))],\\ \mathcal{L}_{Q_r} &= \mathbb{E}_{(s,a,s',r)\sim\mathcal{D}}[(r+\gamma V_r(s')-Q_r(s,a))^2],\end{aligned} \tag{15}$$

where $\mathcal{L}^\tau(u) = |\tau - \mathbb{I}(u<0)|u^2$, and train the optimal feasible value function via

$$\begin{aligned}\mathcal{L}_{V_h} &= \mathbb{E}_{(s,a)\sim\mathcal{D}}[\mathcal{L}_{\text{rev}}^\tau(Q_h(s,a)-V_h(s))],\\ \mathcal{L}_{Q_h} &= \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[((1-\gamma)h(s)+\gamma\max(h(s),V_h(s'))-Q_h(s,a))^2],\end{aligned} \tag{16}$$

where $\mathcal{L}_{\text{rev}}^\tau(u) = |\tau - \mathbb{I}(u>0)|u^2$, τ is a hyperparameter. Next, we instantiate optimality criterias as

$$o_r = \mathbb{I}_{A_r^*(s,a)\geq 0},\ o_h = \mathbb{I}_{A_h^*(s,a)\leq 0},\ o_I = \mathbb{I}_{Q_h^*(s,a)\leq 0}, \tag{17}$$

and change the exponential function f_1, f_2 to the clipped linear function $\max(x, 0)$ as in Frans et al. [13]. Meanwhile, we also replace the score-matching learning objective for π with flow matching, which exhibits similar properties while offering greater training stability [13, 16, 36]. Thus, the policy π is modeled by two velocity fields v_r, v_c trained via the following loss

$$\mathcal{L}_v = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[||v_r(a_t,t,s,o_r)-(a-a_0)||^2+||v_c(a_t,t,s,o_{I,h})-(a-a_0)||^2\right], \tag{18}$$

where $a_t = (1-t)a_0 + ta$, $o_r, o_{h,I} \in \{\emptyset, 0, 1\}$, t is sampled uniformly between $[0, 1]$, and $a_0 \sim \mathcal{N}(0, 1)$ is Gaussian noise, just the z output by π_l. Theoretical proofs and more implementation details can be found in Appendix A and B.

3.4 Theoretical Results

Previously, we introduce methods for adjusting the guidance scales to obtain a better reward–safety trade-off. Here, we further provide the performance improvement theorem to justify their rationale:

Proposition 3.2. *For any infeasible state s, let $0 \leq w_{c,1} < w_{c,2}$, and let π_1, π_2, and $\hat{\pi}$ denote three policies. If the score functions of the policies satisfy $\nabla_a \log \pi_i(a|s) = \nabla_a \log \hat{\pi}(a|s) + w_{c,i}(\nabla_a \log \hat{\pi}(a|s, o_h) - \nabla_a \log \hat{\pi}(a|s))$, $i = 1, 2$, we have*

$$\mathbb{E}_{\pi_2}[A_h^*(s, a)] \leq \mathbb{E}_{\pi_1}[A_h^*(s, a)]. \tag{19}$$

Proposition 3.3. *For any feasible state s, let $w_c \geq 0$ and $0 \leq w_{r,1} < w_{r,2}$, and let π_1, π_2, and $\hat{\pi}$ denote three policies. Suppose that their score functions satisfy $\nabla_a \log \pi_i(a|s) = \nabla_a \log \hat{\pi}(a|s) + w_{r,i}(\nabla_a \log \hat{\pi}(a|s, o_r) - \nabla_a \log \hat{\pi}(a|s)) + w_c(\nabla_a \log \hat{\pi}(a|s, o_I) - \nabla_a \log \hat{\pi}(a|s))$, $i = 1, 2$, we have*

$$\mathbb{E}_{\pi_2}[A_r^*(s, a)] \geq \mathbb{E}_{\pi_1}[A_r^*(s, a)]. \tag{20}$$

Proposition 3.4. *For any feasible state s, let $w_r \geq 0$ and $w_c > 0$, and let π and $\hat{\pi}$ denote two policies. Suppose that their score functions satisfy $\nabla_a \log \pi(a|s) = \nabla_a \log \hat{\pi}(a|s) + w_r(\nabla_a \log \hat{\pi}(a|s, o_r) - \nabla_a \log \hat{\pi}(a|s)) + w_c(\nabla_a \log \hat{\pi}(a|s, o_I) - \nabla_a \log \hat{\pi}(a|s))$. If $\hat{\pi}$ is good enough, it follows that*

$$\mathbb{E}_{\pi}[\mathbb{I}_{Q_h^*(s,a) \leq 0}] = 1. \tag{21}$$

Theorem 3.5. *In the optimization objective 6, when the KL constraint threshold ϵ is sufficiently large, let the policy π satisfy $\nabla_a \log \pi(a|s) = \nabla_a \log \hat{\pi}(a|s) + w_r \mathbb{I}_{V_h^*(s) \leq 0}(\nabla_a \log \hat{\pi}(a|s, o_r) - \nabla_a \log \hat{\pi}(a|s)) + w_c(\nabla_a \log \hat{\pi}(a|s, o_{I,h}) - \nabla_a \log \hat{\pi}(a|s))$, where $\hat{\pi}$ is a given policy and $w_r \geq 0, w_c > 0$. Then, increasing w_r while fixing w_c, or increasing w_c while fixing w_r, produces a new policy π' that is no worse or better than π for the optimization objective 6.*

4 Experiments

In this section, we present our experimental analysis conducted on 21 Safety-Gymnasium [27] tasks from the OSRL [39] dataset to answer the following questions: (1) Can Co3G outperform other baselines across various tasks (Section 4.2)? (2) Does Co3G possess the ability to flexibly adjust its conservatism (Section 4.3)? (3) What's the impact of different components and hyperparameters on Co3G 's performance (Section 4.4)? More results and details can be seen in Appendix C and D.

4.1 Baselines and Tasks

To evaluate the reward–safety trade-off achieved by Co3G, we compare three variants of our method—**Co3G Manual**, **Co3G RS**, and **Co3G RL**—with a diverse set of baselines. We first consider representative soft-constraint methods, including **CDT** [38] as a typical DT-based baseline, **CCAC** [22] as a representative conservative-value-estimation-based baseline, **CAPS** (IQL) [7] as a strong IQL-based baseline, and **TREBI** [35] as a performative diffusion-based baseline. We then include **LSPC**-O [29] and **FISOR** [52], two important methods developed for the more stringent hard-constraint setting, to enable a more thorough comparison of safety performance. Finally, we introduce **BC-Safe**, a behavior cloning baseline based on safe data filtering, to verify the ability of our method to exploit more diverse data and thereby achieve superior performance. For the results, reward returns are normalized using the offline dataset's minimum and maximum, while costs are scaled by 10 as in previous works [29, 52]. Outcomes with normalized cost ≤ 1 are considered safe, with higher rewards indicating better performance among safe algorithms.

Our experiments use 16 navigation and 5 velocity tasks from Safety-Gymnasium. The navigation tasks involve two robot types (Point and Car) across four scenarios—Button, Circle, Goal, and Push—with two tasks per scenario. The velocity tasks include five robots: Ant, HalfCheetah, Hopper, Swimmer, and Walker2d. In addition, for visualization experiments, we further select the AntCircle task from Bullet Safety Gym in OSRL for evaluation and visualization.

4.2 Competitive Results

We first present the overall performance of Co3G variants and all baselines across Safety-Gymnasium tasks, summarized in Table 1. First, CCAC shows the poorest safety performance, suggesting that

Table 1: Overall normalized results. Each value is averaged over 20 evaluation episodes, and 3 random seeds. **Bold**: Safe agents. Gray: Unsafe agents. **Blue**: Safe agent with the highest reward.

Task	CDT		CCAC		CAPS		TREBI		BC-Safe		LSPC		FISOR		Co3G Manual		Co3G RS		Co3G RL	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
CarButton1	0.19	9.40	0.42	38.57	0.03	1.64	0.07	3.75	0.03	2.32	-0.01	2.54	**-0.03**	**0.64**	**0.02**	**1.00**	**-0.04**	**0.39**	**-0.01**	**0.30**
CarButton2	0.22	11.98	0.41	42.94	-0.05	1.56	**-0.03**	**0.97**	0.00	2.28	-0.07	2.05	**-0.02**	**0.80**	**-0.02**	**0.83**	**-0.03**	**0.49**	**-0.03**	**0.42**
CarCircle1	0.51	7.45	0.05	22.88	0.54	7.18	**0.15**	**0.03**	0.10	1.87	**0.34**	**0.86**	0.67	10.15	**0.34**	**0.97**	**0.35**	**0.77**	**0.12**	**0.32**
CarCircle2	0.62	12.67	0.52	8.62	0.55	8.80	**0.16**	**0.00**	0.25	4.39	**0.26**	**0.06**	0.58	10.17	**0.26**	**0.88**	**0.34**	**0.11**	**0.21**	**0.34**
CarGoal1	0.56	3.30	0.79	5.84	**0.34**	**0.87**	0.41	1.16	**0.22**	**0.50**	0.33	1.12	0.42	1.38	**0.30**	**0.82**	**0.31**	**0.24**	**0.13**	**0.25**
CarGoal2	0.40	6.38	0.88	17.61	0.13	1.72	0.13	1.16	0.15	2.14	**0.11**	**0.79**	**0.05**	**0.31**	**0.16**	**0.71**	**0.08**	**0.44**	**0.02**	**0.18**
CarPush1	0.27	2.06	-0.03	2.63	**0.15**	**0.73**	0.26	1.03	**0.18**	**0.75**	**0.14**	**0.84**	0.28	1.15	**0.27**	**0.87**	**0.33**	**0.65**	**0.19**	**0.51**
CarPush2	0.15	4.28	0.26	17.33	0.02	1.36	0.12	2.65	**0.04**	**0.54**	0.05	3.32	**0.12**	**0.98**	**0.07**	**0.76**	**0.18**	**0.76**	**-0.03**	**0.66**
PointButton1	0.53	11.40	0.56	17.11	0.04	1.35	0.01	2.10	0.06	2.54	0.05	2.52	0.07	1.53	**-0.04**	**0.59**	0.12	2.39	**-0.04**	**0.76**
PointButton2	0.42	10.09	0.55	18.14	0.15	3.12	0.02	1.75	0.09	2.94	0.28	6.35	0.12	3.59	0.06	1.52	0.20	4.34	0.01	1.35
PointCircle1	**0.54**	**0.75**	0.58	6.41	**0.42**	**0.25**	**0.33**	**0.03**	0.31	1.15	0.46	12.68	0.49	12.35	**0.41**	**0.76**	-0.08	22.42	0.17	1.02
PointCircle2	0.60	1.79	0.49	2.41	**0.49**	**0.60**	**0.20**	**0.03**	**0.24**	**0.83**	0.65	15.45	0.75	18.31	0.59	3.40	0.32	14.95	**0.33**	**0.65**
PointGoal1	0.70	3.99	0.75	5.07	0.29	1.02	**0.28**	**0.65**	**0.20**	**0.68**	0.33	1.73	0.67	4.86	0.22	1.23	0.57	3.95	**0.11**	**0.20**
PointGoal2	0.47	5.23	0.65	15.63	0.24	1.43	0.20	1.49	0.28	2.51	0.30	3.09	0.23	1.93	0.24	2.25	0.33	3.19	**0.13**	**0.77**
PointPush1	0.25	3.23	0.08	1.93	0.13	1.16	0.07	1.17	0.09	1.16	0.19	1.54	0.26	2.70	**0.15**	**0.88**	**0.32**	**0.76**	**0.17**	**0.42**
PointPush2	0.19	4.01	0.15	4.19	0.13	1.37	0.02	1.14	0.10	2.76	0.18	4.01	0.23	3.28	0.14	1.19	0.28	3.43	**0.09**	**0.59**
AntVelocityV1	**0.95**	**0.70**	**-1.01**	**0.00**	**0.85**	**0.11**	**0.31**	**0.00**	**0.95**	**0.77**	**0.97**	**0.62**	**0.89**	**0.00**	**0.98**	**0.25**	**0.74**	**0.02**	**0.94**	**0.00**
HalfCheetahVelocityV1	**0.97**	**0.49**	**0.95**	**0.88**	**0.89**	**0.21**	**0.87**	**0.23**	**0.88**	**0.57**	0.99	15.57	**0.89**	**0.00**	**1.00**	**0.03**	**0.97**	**0.00**	**1.01**	**0.00**
HopperVelocityV1	**0.64**	**0.75**	0.15	2.67	**0.46**	**0.91**	0.04	1.24	**0.08**	**0.67**	0.33	4.63	0.16	1.18	**0.66**	**0.81**	**0.15**	**0.14**	**0.09**	**0.20**
SwimmerVelocityV1	0.66	1.51	0.02	1.76	**0.48**	**0.43**	0.42	1.31	0.40	3.65	0.50	4.87	**-0.03**	**0.02**	**0.46**	**0.17**	**-0.04**	**0.10**	**0.37**	**0.33**
Walker2dVelocityV1	**0.78**	**0.60**	0.62	9.73	**0.79**	**0.06**	0.38	4.48	**0.79**	**0.01**	**0.79**	**0.35**	0.57	1.72	**0.79**	**0.34**	**0.26**	**0.84**	**0.58**	**0.43**
Average	0.50	4.86	0.37	11.54	0.34	1.71	0.21	1.26	0.26	1.67	0.34	4.05	0.35	3.67	**0.34**	**0.97**	0.27	2.88	**0.22**	**0.46**
Safe Percent	5/21		2/21		9/21		8/21		9/21		6/21		7/21		16/21		14/21		19/21	

conservative value estimation is less stable in safe RL. CDT, trained via DT-style behavior cloning, improves stability but remains suboptimal in safety. CAPS combines IQL with rejection sampling for action selection and achieves better average performance; however, its constraint-threshold mechanism provides limited control over conservatism, so safety constraints are fully satisfied in only a few environments. TREBI clearly improves average safety performance through diffusion-based planning, but at the cost of a considerable reduction in reward. Moreover, the number of environments in which it actually satisfies the constraints remains limited. Among hard-constraint methods, LSPC decouples reward and safety learning, but its reward-maximization objective still substantially affects safety, limiting overall safety performance. BC-Safe achieves the best safety among baselines by using safety-filtered data, yet lacks flexibility and underutilizes available data. FISOR incorporates HJ-reachability analysis and improves safety, but its fixed conservatism during training leads to suboptimal reward–safety trade-offs in most environments. In contrast, our method achieves markedly stronger performance while using the same value functions as FISOR. Co3G Manual adjusts conservatism at test time based on guidance-scale theory, substantially improving safety while maintaining rewards comparable to CAPS, LSPC, and FISOR. Co3G RS uses identical hyperparameters across tasks; despite limited conservatism adjustment, it satisfies safety constraints in many tasks. Co3G RL further applies RL to impose fine-grained, policy-aware safety constraints, greatly improving safety and satisfying constraints in twice as many environments as the best baseline.

4.3 Conservatism Controllability Analysis

First, we conducted visualization experiments in the AntCircle environment. Figure 1(a) shows how the trajectories are affected by the guidance scales. As the reward guidance scale gradually increases from 0.3, the trajectories expand outward and increasingly follow the prescribed circular arc. In contrast, as the cost guidance scale increases from 0.0, the trajectories gradually contract until they remain entirely within the safety bound. Figure 1(b) illustrates the variation in advantage under different guidance scales. For A_r^*, we observe that the distribution under $w_r = 1.0$ is clearly better than that under $w_r = 0.0$. Moreover, although the middle region of the distribution under $(1.0, 2.0)$ is less dense than that under $(1.0, 0.0)$, it has more mass in the high-A_r^* region, indicating that increasing w_c does not necessarily degrade reward performance. A similar conclusion can be drawn from $-A_h^*$: the distribution under $w_c = 2.0$ is clearly better than that under $w_c = 0.0$. Taken together, these results show that $(1.0, 2.0)$ outperforms $(0.0, 0.0)$ in terms of both reward and safety, which to some extent validates our theoretical claim that increasing the guidance scales can improve the policy. Finally, Figure 1(c) and (d) present the changes in reward and safety performance more explicitly.

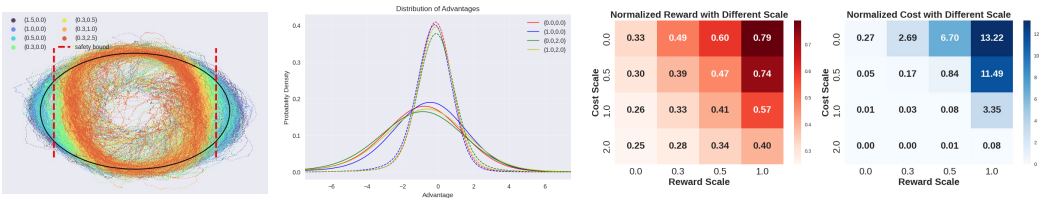


Figure 1: (a) Trajectories in AntCircle under varying (w_r, w_c) settings. (b) Distributions of action advantages under different (w_r, w_c) settings; solid lines correspond to A_r^* and dashed lines to $-A_h^*$. (c) Reward performance of the policy in AntCircle across different (w_r, w_c). (d) Cost performance of the policy in AntCircle across different (w_r, w_c).

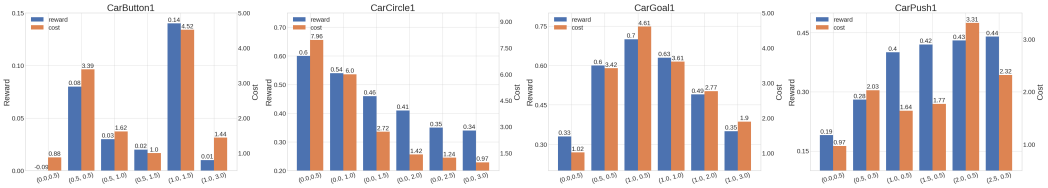


Figure 2: Performance of the policy in more tasks across different (w_r, w_c).

When w_c is fixed, increasing w_r strictly improves the reward; when w_r is fixed, increasing w_c strictly reduces the cost. Furthermore, the lower-left half of the matrix shows that along the diagonal from the upper-left to the lower-right, increasing (w_r, w_c) jointly generally leads to higher reward and lower cost, further confirming that the guidance scale can promote policy improvement within a limited range. We further evaluate the impact of the guidance scales across a broader set of tasks, as shown in Figure 2. The results on CarButton1 suggest a feasible adjustment trajectory starting from $(0.0, 0.5)$. The remaining three respectively illustrate three adjustment strategies: progressively increasing w_c, first increasing w_r and then w_c, and progressively increasing w_r. Overall, the conclusions remain consistent with our previous observations: increasing w_r significantly improves reward performance, while increasing w_c leads to substantial gains in safety performance.

4.4 Ablation and Sensitivity Studies

We conduct ablation studies to assess the key components of Co3G. First, we introduce **Diff.**, which replaces flow matching with diffusion score matching as the policy. As shown in Figure 3(a), diffusion and flow matching perform comparably under Manual (Man.) and RS variants, but Diff. shows degraded safety under RL, indicating that the more stable training of flow matching benefits latent policy learning. Next, we evaluate **Single Condition (SC)** and **FISOR RL**. SC multiplies o_r and $o_{I,h}$ and uses a single guidance scale, while FISOR RL steers the noise in FISOR. As shown in Figure 3(b), all SC variants suffer substantial safety degradation compared with Co3G, highlighting the importance of compositional generation. FISOR RL further exhibits a pronounced safety drop relative to Diff. RL, despite both relying on diffusion, underscoring the importance of base model capability for latent RL. Finally, we compare latent RL variants in Figure 3(c). **TD3** replaces the objective in Equation (6) with a Q^π-guided form, **Lagrange** removes the feasible indicator from RESPO, while both use the feasible policy network Q_h^π instead of Q_c^π. **Org Q** builds on Lagrange but directly distills the Q_h and Q_r used in compositional flow matching, rather than Q^A. **W/o Noise** does not train the noise and instead follows FISOR by using rejection sampling to select the noise with the smallest Q_h. Overall, Co3G achieves the best reward–safety trade-off, while Org Q shows a substantial safety drop, confirming the necessity of policy-aware value functions for latent RL.

Next, we examine the sensitivity of the algorithm to its hyperparameters. Figure 4(a) shows the performance of Co3G RS under varying δ_h. As expected, increasing δ_h slightly reduces policy conservatism, but its effect remains minor within the range $[-0.8, 0.0]$. Figure 4(b) presents the performance of Co3G RS across different values of M. For any given M, both w_r and w_c are sampled uniformly within the range $[0.0, 3.0]$. For example, when $M = 2$, the lists of w_r and w_c are $[0.0, 3.0]$, resulting in four possible combinations. The results indicate negligible sensitivity to M, suggesting

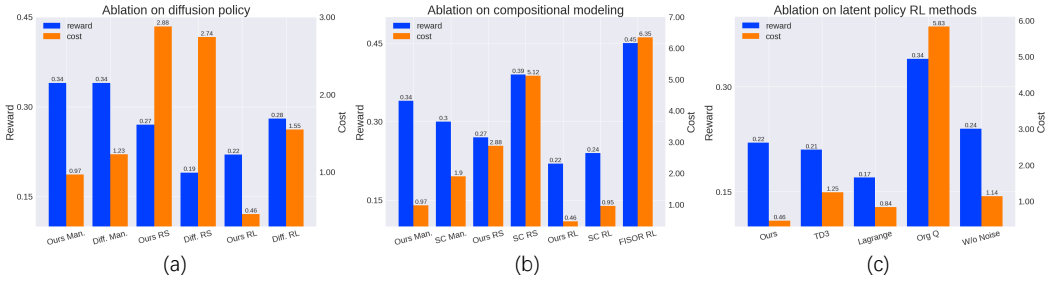


Figure 3: Ablation study results.

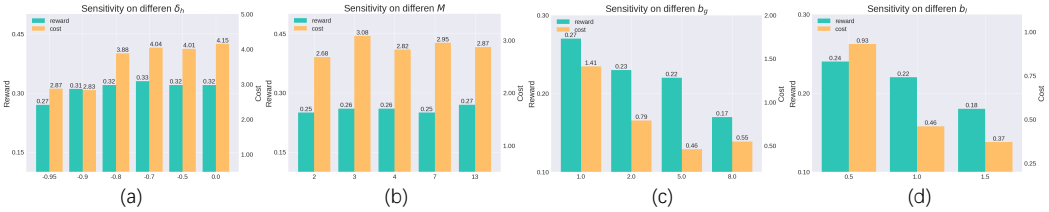


Figure 4: Sensitivity study results.

that the policy is robust to this hyperparameter. Finally, Figure 4(c) and (d) similarly show that when b_g or b_l is small, the search space of Co3G RL is constrained, weakening the effect of the constraints and producing more aggressive policies. While increasing b_g or b_l enhances safety performance. However, it also induces more conservative behavior and a higher probability of out-of-distribution actions, leading to a slight reduction in overall reward.

5 Related Work

Safe RL. Safe RL ensures policy deployment under safety constraints in addition to reward maximization, which is typically formulated as a Constrained Markov Decision Process (CMDP) [3] and addressed through constrained optimization techniques [17, 20]. Lagrangian-based methods [42] and trust-region methods [1] are among the most widely used approaches for online safe policy learning. For hard constraints, HJ reachability analysis [4] has also been employed to learn cost functions and impose more stringent safety guarantees [15, 49]. Nevertheless, both types of methods still rely on unsafe real-world interactions, which restricts their applicability in practice. To address this issue, recent studies have increasingly focused on offline safe RL, which learns safe policies from pre-collected datasets and thereby avoids unsafe exploration. Some methods incorporate conservative value estimation into cost value function learning to alleviate cost value underestimation [22, 46], while others adopt BC-based policy learning frameworks, such as DT [38, 48], diffusion planning [35], DICE [31], and IQL [7, 29, 52, 54], to better mitigate extrapolation errors [14] in safe RL. Beyond single-task settings, another line of work explores transferable safe policies that can be applied across a broader range of tasks [18, 6].

Generative models in RL. Driven by the strong capacity of generative models to capture multimodal distributions, together with their rapid advances in domains such as natural language processing, a growing body of work has sought to apply them to RL [34]. Among these approaches, Transformers are introduced early on to RL and used to perform decision-making via sequence modeling [8, 25, 32, 47]. Subsequently, diffusion models have found broader applications in RL, serving either as generative planners [2, 26] or as policies for offline [9, 45] and online learning [5, 10, 40]. Building upon diffusion models, compositional diffusion has achieved strong performance in multi-condition image generation via score decomposition [37], and has consequently been adopted as world models in embodied intelligence [51, 53]. Besides, flow matching has recently received growing attention in RL and embodied intelligence [11, 13, 24, 41], due to its simpler and more stable generation process relative to diffusion models.

6 Final Remarks

In this work, we propose Co3G, a novel compositional generative model that enables fine-grained adjustment of the test-time reward–safety trade-off for learning safe policies offline. Co3G first decouples the training of reward- and safety-optimal policies, and then, at deployment, combines these objectives through compositional generation using guidance scales. Additionally, Co3G introduces three methods for setting the guidance scale: manual, RS, and RL. Extensive experiments on the OSRL benchmark demonstrate that Co3G achieves strong safety performance while enabling effective control over the reward–safety trade-off. Looking forward, extending Co3G to embodied intelligence and vision language action models represents a promising research direction. In addition, leveraging Co3G 's decoupled training of reward and safety, studying its multi-task learning, transfer, and generalization capabilities when combining a shared reward velocity model with different safety velocity models constitutes another important avenue for future work.

Acknowledgements

This work was supported by the Natural Science Foundation of Jiangsu under Grants BK20243039, BK20241199, the "111 Center" (No. B26023), and Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118). We thank the reviewers for their support and helpful discussions on improving the paper.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. Pmlr, 2017.

[2] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023.

[3] Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

[4] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253, 2017.

[5] Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

[6] Mohamad Fares El Hajj Chehade, Amrit Singh Bedi, Amy Zhang, and Hao Zhu. Tram: Test-time risk adaptation with mixture of agents. *arXiv preprint arXiv:2408.08812*, 2024.

[7] Yassine Chemingui, Aryan Deshwal, Honghao Wei, Alan Fern, and Jana Doppa. Constraint-adaptive policy switching for offline safe reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 15722–15730, 2025.

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: reinforcement learning via sequence modeling. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 15084–15097, 2021.

[9] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

[10] Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 53945–53968, 2024.

[11] Nicolas Espinosa-Dice, Yiyi Zhang, Yiding Chen, Bradley Guo, Owen Oertell, Gokul Swamy, Kiante Brantley, and Wen Sun. Scaling offline rl via efficient and expressive shortcut models. *arXiv preprint arXiv:2505.22866*, 2025.

[12] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation*, pages 8550–8556, 2019.

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

[14] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

[15] Milan Ganai, Zheng Gong, Chenning Yu, Sylvia Herbert, and Sicun Gao. Iterative reachability estimation for safe reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 69764–69797, 2023.

[16] Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P Murphy, and Tim Salimans. Diffusion meets flow matching: Two sides of the same coin. 2024. *URL https://diffusionflow. github. io*, 2024.

[17] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[18] Michael Gimelfarb, André Barreto, Scott Sanner, and Chi-Guhn Lee. Risk-aware transfer in reinforcement learning using successor features. *Advances in Neural Information Processing Systems*, 34:17298–17310, 2021.

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

[20] 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*, 2024.

[21] Jiayi Guan, Guang Chen, Jiaming Ji, Long Yang, Ao Zhou, Zhijun Li, and Changjun Jiang. Voce: variational optimization with conservative estimation for offline safe reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 33758–33780, 2023.

[22] Zijian Guo, Weichao Zhou, Shengao Wang, and Wenchao Li. Constraint-conditioned actor-critic for offline safe reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[23] Wenhui Huang, Haochen Liu, Zhiyu Huang, and Chen Lv. Safety-aware human-in-the-loop reinforcement learning with shared control for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 25(11):16181–16192, 2024.

[24] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[25] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 1273–1286, 2021.

[26] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915, 2022.

[27] Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Juntao Dai, and Yaodong Yang. Safety-gymnasium: a unified safe reinforcemei learning benchmark. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 18964–18993, 2023.

[28] Geon-Hyeong Kim, Yu Jin Kim, Byoungjip Kim, Honglak Lee, Kyunghoon Bae, Youngsoo Jang, and Moontae Lee. Safedpo: A simple approach to direct preference optimization with enhanced safety. *arXiv preprint arXiv:2505.20065*, 2025.

[29] Prajwal Koirala, Zhanhong Jiang, Soumik Sarkar, and Cody Fleming. Latent safety-constrained policy approach for safe offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[31] 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 *International Conference on Learning Representations*, 2022.

[32] Kuang-Huei Lee, Ofir Nachum, Sherry Yang, Lisa Lee, C Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. In *Advances in Neural Information Processing Systems*, 2022.

[33] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[34] Yinchuan Li, Xinyu Shao, Jianping Zhang, Haozhi Wang, Leo Maxime Brunswic, Kaiwen Zhou, Jiqian Dong, Kaiyang Guo, Xiu Li, Zhitang Chen, et al. Generative models in decision making: A survey. *arXiv preprint arXiv:2502.17100*, 2025.

[35] 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*, pages 21127–21152. PMLR, 2023.

[36] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

[37] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European conference on computer vision*, pages 423–439. Springer, 2022.

[38] 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*, pages 21611–21630, 2023.

[39] 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. *Journal of Data-centric Machine Learning Research*, 2024.

[40] Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. *arXiv preprint arXiv:2502.00361*, 2025.

[41] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

[42] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143, 2020.

[43] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033, 2012.

[44] Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Conference on Robot Learning*, pages 258–282. PMLR, 2025.

[45] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[46] Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8753–8760, 2022.

[47] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *International Conference on Machine Learning*, pages 24631–24645, 2022.

[48] Ruiqi Xue, Ziqian Zhang, Lihe Li, Cong Guan, Lei Yuan, and Yang Yu. Adaptable safe policy learning from multi-task data with constraint prioritized decision transformer. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[49] Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International conference on machine learning*, pages 25636–25655, 2022.

[50] Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Josef Dai, Yuanpei Chen, and Yaodong Yang. Safevla: Towards safety alignment of vision-language-action model via constrained learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[51] Hongxin Zhang, Zeyuan Wang, Qiushi Lyu, Zheyuan Zhang, Sunli Chen, Tianmin Shu, Behzad Dariush, Kwonjoon Lee, Yilun Du, and Chuang Gan. Combo: Compositional world models for embodied multi-agent cooperation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[52] 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 *The Twelfth International Conference on Learning Representations*, 2024.

[53] Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. *arXiv preprint arXiv:2404.12377*, 2024.

[54] LIU Zifan, Xinran Li, and Jun Zhang. C2iql: Constraint-conditioned implicit q-learning for safe offline reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide a discussion on limitations in Section F.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical result and proofs are provided in Section A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: Detailed algorithms and hyperparameters are provided in Section B.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The data and code are provided in the supplementary material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed algorithms and hyperparameters are provided in Section B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The standard deviation of the main experiment is provided in Section D.7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The time complexity analysis is provided in Section D.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We provide a discussion on societal impacts in Section E.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets used in the paper are properly credited, and the license and terms of use are explicitly mentioned and properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

A Mathematical Proofs

A.1 Proof of Theorem 3.1

Proof. First, according to the definition of the optimality variable in the main text, under a feasible state, π^* can be expressed as

$$\pi^*(a|s) \propto p(o_r|s,a)^{\alpha_1} p(o_I|s,a)^{\alpha_3} \hat{\pi}(a|s). \tag{22}$$

Since the gradient is taken only with respect to the action, normalization terms that do not depend on a are discarded when taking the gradient of the log. Therefore, taking the log on both sides and computing the gradient yields

$$\nabla_a \log \pi^*(a|s) = \nabla_a \log \hat{\pi}(a|s) + \alpha_1 \nabla_a \log p(o_r|s,a) + \alpha_3 \nabla_a \log p(o_I|s,a). \tag{23}$$

By applying Bayes' theorem, we obtain

$$p(o|s,a) = \frac{p(a|s,o)}{p(a|s)} p(o|s) = \frac{\hat{\pi}(a|s,o)}{\hat{\pi}(a|s)} p(o|s), \tag{24}$$

where $p(o \mid s)$ is independent of a, and therefore, substituting it in yields

$$\begin{aligned} \nabla_a \log \pi^*(a|s) = \nabla_a \log \hat{\pi}(a|s) &+ \alpha_1 (\nabla_a \log \pi_\beta(a|s,o_r) - \nabla_a \log \hat{\pi}(a|s)) \\ &+ \alpha_3 (\nabla_a \log \pi_\beta(a|s,o_I) - \nabla_a \log \hat{\pi}(a|s)). \end{aligned} \tag{25}$$

Similarly, under infeasible states, we have

$$\nabla_a \log \pi^*(a|s) = \nabla_a \log \hat{\pi}(a|s) + \alpha_2 (\nabla_a \log \pi_\beta(a|s,o_h) - \nabla_a \log \hat{\pi}(a|s)). \tag{26}$$

Let $w_r^* = \alpha_1$. Additionally, since α_3 can take any positive value, set $w_c^* = \alpha_2 = \alpha_3$. By combining the results from both feasible and infeasible states, we then obtain

$$\begin{aligned} \nabla_a \log \pi^*(a|s) = \nabla_a \log \hat{\pi}(a|s) &+ w_r^* \mathbb{I}_{V_h^*(s) \le 0} (\nabla_a \log \pi_\beta(a|s,o_r) - \nabla_a \log \hat{\pi}(a|s)) \\ &+ w_c^* \mathbb{I}_{V_h^*(s) \le 0} (\nabla_a \log \pi_\beta(a|s,o_I) - \nabla_a \log \hat{\pi}(a|s)) \\ &+ w_c^* (1 - \mathbb{I}_{V_h^*(s) \le 0}) (\nabla_a \log \pi_\beta(a|s,o_h) - \nabla_a \log \hat{\pi}(a|s)). \end{aligned} \tag{27}$$

Finally, since o_I and o_h do not occur simultaneously, substituting $o_{I,h} = \mathbb{I}_{V_h^*(s) \le 0} o_I + \mathbb{I}_{V_h^*(s) > 0} o_h$ yields

$$\begin{aligned} \nabla_a \log \pi^*(a|s) = \nabla_a \log \hat{\pi}(a|s) &+ w_r^* \mathbb{I}_{V_h^*(s) \le 0} (\nabla_a \log \hat{\pi}(a|s,o_r) - \nabla_a \log \hat{\pi}(a|s)) \\ &+ w_c^* (\nabla_a \log \hat{\pi}(a|s,o_{I,h}) - \nabla_a \log \hat{\pi}(a|s)). \end{aligned} \tag{28}$$

□

A.2 Proof of Proposition 3.2 and 3.3

Lemma A.1. *Let $\pi(a|s) \propto f(A(s,a))^x, \hat{\pi}(a|s)$, where f is a bounded, measurable, non-negative, and non-decreasing function, x is any positive number, and A is any integrable function. If $Z(s) = \mathbb{E}_{\hat{\pi}}[f(A(s,a))^x] > 0$, then we have*

$$\mathbb{E}_{a \sim \pi}[A(s,a)] \ge \mathbb{E}_{a \sim \hat{\pi}}[A(s,a)]. \tag{29}$$

Proof. First, since f is a bounded, measurable, non-negative, and non-decreasing function and $x > 0$, it follows that $g = f^x$ is also bounded, measurable, non-negative, and non-decreasing. Moreover, because $Z(s) = \mathbb{E}_{\hat{\pi}}[f(A(s,a))^x] > 0$, we have

$$\pi(a|s) = \frac{g(A(s,a))}{Z(s)} \hat{\pi}(a|s). \tag{30}$$

Therefore, we have

$$\mathbb{E}_{\pi}[A(s,a)] = \frac{\mathbb{E}_{\hat{\pi}}[A(s,a) g(A(s,a))]}{\mathbb{E}_{\hat{\pi}}[g(A(s,a))]}. \tag{31}$$

Thus

$$\mathbb{E}_{\pi}[A] - \mathbb{E}_{\hat{\pi}}[A] = \frac{\mathbb{E}_{\hat{\pi}}[A g(A)] - \mathbb{E}_{\hat{\pi}}[g(A)] \mathbb{E}_{\hat{\pi}}[A]}{\mathbb{E}_{\hat{\pi}}[g(A)]} = \frac{Cov_{\hat{\pi}}(A, g(A))}{\mathbb{E}_{\hat{\pi}}[g(A)]}. \tag{32}$$

We next show that $\mathrm{Cov}_{\hat{\pi}}(A, g(A)) \geq 0$. Let A_1 and A_2 be two independent and identically distributed random variables (i.i.d.), then (omitting the subscript $\hat{\pi}$)

$$\begin{aligned} &\mathbb{E}[(A_1 - A_2)(g(A_1) - g(A_2))] \\ =&\mathbb{E}[A_1 g(A_1)] + \mathbb{E}[A_2 g(A_2)] - \mathbb{E}[A_1 g(A_2)] - \mathbb{E}[A_2 g(A_1)] \\ =&2\mathbb{E}[Ag(A)] - 2\mathbb{E}[A]\mathbb{E}[g(A)] \\ =&2Cov(A, g(A)). \end{aligned} \tag{33}$$

Since g is a non-decreasing function, it follows that for any u, v

$$(u - v)(g(u) - g(v)) \geq 0. \tag{34}$$

Therefore, the following holds almost everywhere

$$(A_1 - A_2)(g(A_1) - g(A_2)) \geq 0. \tag{35}$$

Which means

$$Cov_{\hat{\pi}}(A, g(A)) \geq 0. \tag{36}$$

Finally, we obtain

$$\mathbb{E}_{a\sim\pi}[A(s, a)] \geq \mathbb{E}_{a\sim\hat{\pi}}[A(s, a)]. \tag{37}$$

□

Based on Lemma A.1, we proceed to prove Proposition 3.2.

Proof. First, following the proof of Theorem 3.1, we have (multiplying or dividing by any factor independent of a preserves the proportionality)

$$\pi_i(a|s) \propto f(-A_h^*(s, a))^{w_{c,i}} \hat{\pi}(a|s), i = 1, 2, \tag{38}$$

where f it the exponential function. Since h is a bounded function and the policy trajectory length is finite, A_h^* is also bounded. Let $\max |A_h^*| = A_{h,max}$. Define f' as a clipped exponential function, that is when $x \leq A_{h,max}$, $f'(x) = \exp(x)$, otherwise when $x > A_{h,max}$, $f'(x) = \exp(A_{h,max})$. It follows that f' is a bounded, measurable, non-negative, and non-decreasing function, and satisfies

$$\pi_i(a|s) \propto f'(-A_h^*(s, a))^{w_{c,i}} \hat{\pi}(a|s), i = 1, 2. \tag{39}$$

Therefore, π_1 and π_2 can be expressed as

$$\pi_i(a|s) = \frac{f'(-A_h^*(s, a))^{w_{c,i}} \hat{\pi}(a|s)}{Z_i(s)}, i = 1, 2, \tag{40}$$

where $Z_i(s)$ are normalization functions. Therefore, we have

$$\pi_2(a|s) = f'(-A_h^*(s, a))^{w_{c,2} - w_{c,1}} \frac{Z_1(s)}{Z_2(s)} \pi_1(a|s). \tag{41}$$

Since, for a given s, $\frac{Z_1(s)}{Z_2(s)}$ can be treated as a constant factor and $w_{c,2} - w_{c,1} > 0$, by setting $A = -A_h^*$, $f^x = \frac{Z_1(s)}{Z_2(s)} f'^{w_{c,2} - w_{c,1}}$ in Lemma 1, we have

$$\mathbb{E}_{\pi_2}[-A_h^*(s, a)] \geq \mathbb{E}_{\pi_1}[-A_h^*(s, a)], \tag{42}$$

that is

$$\mathbb{E}_{\pi_2}[A_h^*(s, a)] \leq \mathbb{E}_{\pi_1}[A_h^*(s, a)]. \tag{43}$$

In fact, within this proof, f is not required to be exponential; any measurable, non-negative, and non-decreasing function suffices. For example, a linear function defined over non-negative inputs also satisfies the conditions. □

The proof of Proposition 3.3 follows analogously to Proposition 3.2 and is therefore omitted here.

A.3 Proof of Proposition 3.4

Lemma A.2. *For feasible states, if*

$$\pi(a|s) \propto (\frac{f_1(A_r^*(s,a))}{Z_1(s)})^{\alpha_1} \cdot (\frac{f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0})}{Z_3(s)})^{\alpha_3} \cdot \hat{\pi}(a|s), \tag{44}$$

where $f_3 = \min(\max(x,0),1)$ *and* $\alpha_3 > 0$*, then* $\mathbb{E}_\pi[\mathbb{I}_{Q_h^*(s,a)\leq 0}] = 1$ *or* $\mathbb{E}_{\hat{\pi}}[(\frac{f_1(A_r^*(s,a))}{Z_1(s)})^{\alpha_1}\mathbb{I}_{Q_h^*(s,a)\leq 0}] = 0$.

Proof. Let $U(a) = \left(\frac{f_1(A_r^*(s,a))}{Z_1(s)}\right)^{\alpha_1}$. Since $f_3 = \min(\max(x,0),1)$, we have

$$f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0}) = \mathbb{I}_{Q_h^*(s,a)\leq 0}. \tag{45}$$

Moreover, for $\alpha_3 > 0$,

$$f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0})^{\alpha_3} = \mathbb{I}^{\alpha_3}_{Q_h^*(s,a)\leq 0} = \mathbb{I}_{Q_h^*(s,a)\leq 0} \tag{46}$$

Let $Y(a) = \mathbb{I}_{Q_h^*(s,a)\leq 0}$, then we have

$$\pi(a|s) \propto \frac{1}{Z_3(s)^{\alpha_3}} U(a)Y(a)\hat{\pi}(a|s). \tag{47}$$

Absorbing terms that are independent of a into the normalization factor, we have

$$\pi(a|s) = \frac{U(a)Y(a)\hat{\pi}(a|s)}{\mathbb{E}_{\hat{\pi}}[U(a)Y(a)]}, \tag{48}$$

If $\mathbb{E}_{\hat{\pi}}[U(a)Y(a)] \neq 0$, we obtain

$$\mathbb{E}_\pi[Y(a)] = \frac{\int Y(a)U(a)Y(a)\hat{\pi}(a|s)}{\int U(a)Y(a)\hat{\pi}(a|s)} = \frac{\int U(a)Y^2(a)\hat{\pi}(a|s)}{\int U(a)Y(a)\hat{\pi}(a|s)} = \frac{\int U(a)Y(a)\hat{\pi}(a|s)}{\int U(a)Y(a)\hat{\pi}(a|s)} = 1. \tag{49}$$

□

Based on Lemma A.2, we proceed to prove Proposition 3.4.

Proof. First, following the proof of Theorem 3.1, we have

$$\pi(a|s) \propto (\frac{f_1(A_r^*(s,a))}{Z_1(s)})^{w_r} \cdot (\frac{f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0})}{Z_3(s)})^{w_c} \cdot \pi_\beta(a|s), \tag{50}$$

where $f_3 = \min(\max(x,0),1)$. Since $\hat{\pi}$ is sufficiently good, for any feasible state it has a non-zero probability of selecting actions with not low A_r^* (when f_1 is the exponential function, any A_r^* can be all right since $f_1(A_r^*) > 0$ always holds) and $Q_h^*(s,a) \leq 0$, i.e.,

$$\mathbb{E}_{\hat{\pi}}[(\frac{f_1(A_r^*(s,a))}{Z_1(s)})^{\alpha_1}\mathbb{I}_{Q_h^*(s,a)\leq 0}] > 0. \tag{51}$$

Moreover, since $w_c > 0$, applying Lemma 2 yields

$$\mathbb{E}_\pi[\mathbb{I}_{Q_h^*(s,a)\leq 0}] = 1. \tag{52}$$

□

A.4 Proof of Theorem 3.5

Proof. When w_c is fixed, increasing w_r:

- For infeasible states, w_r has no effect, and π' remains unchanged from π.

- For any feasible state, if $\mathbb{E}_{\hat{\pi}}[(\frac{f_1(A_r^*(s,a))}{Z_1(s)})^{\alpha_1}\mathbb{I}_{Q_h^*(s,a)\leq 0}] = 0$ then neither π nor π' can satisfy the constraint $\mathbb{E}_{\pi}[\mathbb{I}_{Q_h^*(s,a)\leq 0}] = 1$ in that state (i.e., the behavioral policy cannot find any action that meets the requirement), and both are therefore infeasible solutions. Otherwise, by Proposition 3.4, we have $\mathbb{E}_{\pi}[\mathbb{I}_{Q_h^*(s,a)\leq 0}] = \mathbb{E}_{\pi'}[\mathbb{I}_{Q_h^*(s,a)\leq 0}] = 1$ and, according to Proposition 3.3, $\mathbb{E}_{\pi'}[A_r^*(s,a)] \geq \mathbb{E}_{\pi}[A_r^*(s,a)]$.

In summary, in this case, π' is an equal or superior solution to π with respect to Equation (6).

When w_r is fixed, increasing w_c:

- For infeasible states, according to Proposition 3.2, we have $\mathbb{E}_{\pi'}[A_h^*(s,a)] \leq \mathbb{E}_{\pi}[A_h^*(s,a)]$.
- For feasible states, following the proof of Theorem 3.1, we have

$$\pi(a|s) \propto (\frac{f_1(A_r^*(s,a))}{Z_1(s)})^{w_r} \cdot (\frac{f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0})}{Z_3(s)})^{w_c} \cdot \hat{\pi}(a|s). \tag{53}$$

  Let's denote the increased value of w_c as $w_c' > w_c$, then we have

$$\pi'(a|s) \propto (\frac{f_1(A_r^*(s,a))}{Z_1(s)})^{w_r} \cdot (\frac{f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0})}{Z_3(s)})^{w_c'} \cdot \hat{\pi}(a|s). \tag{54}$$

  If $\mathbb{E}_{\hat{\pi}}[(\frac{f_1(A_r^*(s,a))}{Z_1(s)})^{\alpha_1}\mathbb{I}_{Q_h^*(s,a)\leq 0}] = 0$ then neither π nor π' can satisfy the constraint $\mathbb{E}_{\pi}[\mathbb{I}_{Q_h^*(s,a)\leq 0}] = 1$ in that state, and both are therefore infeasible solutions. Otherwise, if $w_c = 0$, then π cannot satisfy the constraint $\mathbb{E}_{\pi}[\mathbb{I}_{Q_h^*(s,a)\leq 0}] = 1$ and is therefore an infeasible solution. In contrast, since $w_c' > 0$, π' is a feasible solution by Proposition 3.4 and is thus superior to π. Otherwise, because $f_3 = \min(\max(x, 0), 1)$, we have

$$f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0})^{w_c} = f_3(\mathbb{I}_{Q_h^*(s,a)\leq 0})^{w_c'} = \mathbb{I}_{Q_h^*(s,a)\leq 0}. \tag{55}$$

  Since $Z_3(s)$ is independent of a, it follows that $\pi(a|s) = \pi'(a|s)$ in this case.

In summary, in this case, π' is also an equal or superior solution to π with respect to Equation (6). □

B Implementation Details

In this section, we will offer the implementation details of Co3G.

B.1 Compositional Flow Matching Policy Learning

For the compositional flow matching policy π, in practice we model it using three networks: v_u, v_r, and v_c, where v_u is the unconditional velocity network, and v_r and v_c are the reward-optimal and safety-optimal velocity networks, respectively. All networks are implemented as MLPs. During training, the optimality variables are not fed directly as inputs to the networks; instead, they are used as binary weights. The specific training objectives are as follows:

$$\begin{aligned}\mathcal{L}_v = \mathbb{E}_{(s,a)\sim\mathcal{D}}[&||v_u(a_t,t,s) - (a - a_0)||^2 + o_r \cdot ||v_r(a_t,t,s) - (a - a_0)||^2 \\ &+ o_{I,h} \cdot ||v_c(a_t,t,s) - (a - a_0)||^2].\end{aligned} \tag{56}$$

During compositional generation, velocities are generated as follows

$$v = v_u(a_t,t,s) + w_r(v_r(a_t,t,s) - v_u(a_t,t,s)) + w_c(v_c(a_t,t,s) - v_u(a_t,t,s)). \tag{57}$$

The specific algorithms for training and sampling the flow matching policy are presented in Algorithms 1 and 2, respectively.

Algorithm 1 `Co3G` **Training**

Input: offline dataset $\mathcal{D}$.
Initialize: optimal feasible value networks Q_h, V_h, optimal reward networks Q_r, V_r, velocity networks v_u, v_r, v_c.
for step in policy training steps **do**
  Sample $(s, a, s', r, h) \sim \mathcal{D}$.
  Update V_h, Q_h with Equation (16).
  Update V_r, Q_r with Equation (15).
  Sample $a_0 \sim \mathcal{N}(0, 1),\ t \sim U(0, 1)$.
  Compute $o_r, o_{I,h}$ with Equation (17) and $a_t = (1-t)a_0 + ta$.
  Update v_u, v_r, v_c with Equation (56).
end for
Return $V_h, Q_h, V_r, Q_r, v_u, v_r, v_c$.

Algorithm 2 `Co3G` **Sampling**

Input: State s, velocity networks v_u, v_r, v_c, guidance scales w_r, w_c, sampling step T.
Sample $a \sim \mathcal{N}(0, 1)$, set $t \leftarrow 0$.
for $n \in [0, \dots, T-1]$ **do**
  Compute $v = v_u(a, t, s) + w_r(v_r(a, t, s) - v_u(a, t, s)) + w_c(v_c(a, t, s) - v_u(a, t, s))$.
  Compute $a \leftarrow a + (n/T)v$.
  Compute $t \leftarrow t + (n/T)$.
end for
Return a.

B.2 Guidance Scale Adjustment

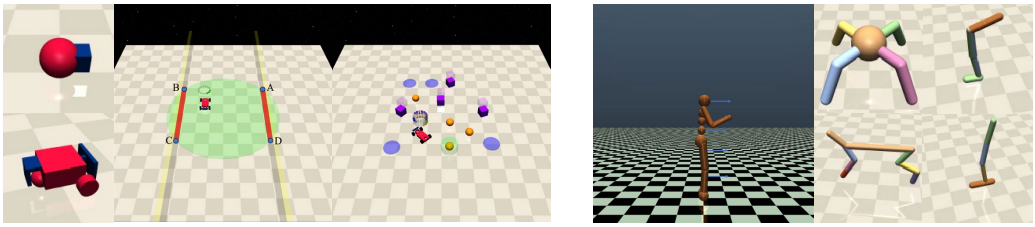
In Co3G Manual, both w_r and w_c are selected from $[0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0]$. In fact, using finer intervals for w_r, such as 0.3, could potentially yield improved results. However, we consider the current settings sufficient, and readers are welcome to experiment with alternative choices.

In Co3G RS, the candidate values for w_r and w_c are drawn from the list $[0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0]$, yielding 13 possible values for each. All pairwise combinations result in 169 candidate pairs. Since rejection sampling is performed in parallel over batches, increasing the number of candidate pairs has negligible impact on computation time.

In Co3G RL, the scale generation networks and policy value networks are also implemented with MLPs, while the Lagrange multipliers are updated using a PID controller with parameters $(k_P, k_I, k_D) = (0.1, 0.003, 0.001)$. The algorithm for learning π_l can be seen in Algorithm 3.

Algorithm 3 `Co3G` **RL**

Input: offline dataset $\mathcal{D}$, base flow matching velocity networks v_u, v_r, v_c, optimal feasible value network V_h.
Initialize: Aggregated policy value networks Q_r^A, Q_c^A, latent value networks Q_r^l, Q_c^l, velocity networks, latent policy π_l, Lagrange multiplier λ.
for step in latent policy training steps **do**
  Sample $(s, a, s', r, c) \sim \mathcal{D}$.
  Update Q_r^A, Q_c^A with Equation (12).
  Update Q_r^l, Q_c^l with Equation (13).
  Update π_l, λ with Equation (14).
end for
Return π_l.



(a) Navigation Tasks (b) Velocity Tasks

Figure 5: Tasks used in this paper. (a) Navigation Tasks based on Point and Car robots. (b) Velocity Tasks based on Ant, HalfCheetah, Hopper, Walker2d, and Swimmer robots.

B.3 Hyperparameters

The training of Co3G involves the selection of hyperparameters. To ensure reproducibility, this section outlines the specific hyperparameters used in our experiments, as shown in Table 2. Co3G is implemented based on OSRL [2] and FISOR [3] code bases. All other parameters not mentioned shared with FISOR remain the same, most of the hyperparameters follow FISOR's defaults, except for training steps, which is set to $2e6$ for Co3G and FISOR in our experiments. For Co3G RL, any policy- and value-learning hyperparameters not explicitly mentioned—such as batch size, learning rate, and network size—are kept consistent with those used in flow matching policy learning as well as the default hyperparameters in FISOR.

Table 2: Hyperparameter choices of Co3G.

	Hyperparameter	Value
Compositional Flow Matching	flow matching sampling step T	5
Guidance Scale Adjustment	RS sampling threshold δ_h	-0.95
	RS sampling number M	13
	RL guidance bound b_g	5.0
	RL noise bound b_l	1.0
	RL training steps	$2e6$

C Detailed Description of the Tasks and Baselines

C.1 Tasks

All tasks used in this paper are derived from the Safety-Gymnasium's Navigation Tasks and Velocity Tasks. In the Navigation Tasks, there are two different types of robots: Point and Car, which we need to control to navigate through the environment and earn rewards by reaching target points, pressing the correct buttons, or moving in designated directions. Different tasks also have varying costs, such as avoiding collisions with specific targets, preventing incorrect button presses, and staying within designated boundaries.

The Velocity Tasks are built on traditional MuJoCo [43] simulations, requiring robots such as Ant, HalfCheetah, Swimmer, and Walker2d to move, where higher speeds result in higher rewards. However, each robot has specific safety velocity thresholds for different tasks, and exceeding these thresholds leads to unsafe states. For detailed descriptions of each task, refer to the original Safety-Gymnasium paper [27].

C.2 Baselines

We provide a more detailed introduction to the baselines in this section.

[2] https://github.com/liuzuxin/OSRL
[3] https://github.com/ZhengYinan-AIR/FISOR

- **CDT** is the previous advanced algorithm under sequence modeling for offline safe RL. After incorporating CTG into DT, CDT seeks to address the conflict between safety constraints and reward maximization. To tackle this issue, CDT proposes a data augmentation approach, where reward returns for certain safe but low-reward trajectories in the offline dataset are re-labeled with higher values.
- **CCAC** is one of the advanced algorithms for soft-constraint offline safe RL. It employs conservative estimation to obtain pessimistic value estimates and incorporates the constraint condition into the policy input, enabling the policy to handle multiple constraint thresholds.
- **FISOR** is one of the advanced offline safe RL algorithm based on hard-constraint modeling. Similar to other algorithms under hard-constraints [49, 15], it first divides the state space into feasible and infeasible regions. It then uses IQL [30] to learn feasible value functions for offline feasible region identification. Next, FISOR sets distinct learning objectives for the feasible and infeasible regions. In the feasible region, it aims to maximize the reward while ensuring feasibility, while in the infeasible region, it focuses on minimizing constraint violations. Finally, FISOR employs a diffusion model to represent and learn the policy.
- **LSPC**-O, similar to FISOR, performs hard-constraint safe policy learning. It first trains a CVAE using the feasibility advantage as a weighting factor, such that the CVAE generates only safe actions. Subsequently, optimization is conducted in the latent space of the CVAE with the objective of maximizing reward, aiming to achieve a balance between reward optimization and safety preservation.
- **CAPS**(IQL) is also an advanced algorithm under the soft-constraint setting. Unlike methods that explicitly incorporate constraint thresholds during training, CAPS does not use the constraint threshold in its training process. Instead, similar to FISOR, it adopts IQL-style expectile regression for value function learning and employs weighted behavior cloning for policy learning. To enable decision-making under different constraint thresholds, CAPS utilizes a multi-head policy that outputs multiple candidate actions, from which an action satisfying the specified constraint threshold and achieving high reward is selected.

Table 3: Time complexity analysis.

	FISOR	**Co3G**	**Co3G Manual**	**Co3G RS**	**Co3G RL (latent training)**
Training	1.23h	1.67h	/	/	3.62h
Deployment	10ms/step	/	26ms/step	33ms/step	45ms/step

D More Experimental Results

D.1 Time Complexity Analysis

To better understand the practical utility of Co3G, we evaluate its training time overhead in this section. Using the CarButton environment as an example, we report the training time (without evaluation) and deployment time measured on a single NVIDIA 4090 GPU under an otherwise idle setting (i.e., with no additional CPU/GPU workload), as summarized in Table 3. As shown in the table, although Co3G uses two additional neural networks compared with FISOR, the overall extra training overhead is modest, remaining within half an hour. For Co3G RL, since each training step involves full action generation with the flow matching policy, it indeed requires additional training time compared with both FISOR and Co3G. Nevertheless, the training time remains within 4 hours, which is still acceptable. During deployment, Co3G requires inference through three networks. As a result, the inference time of the Manual variant is almost three times that of FISOR. For RS, the increased batch size leads to a slight increase in inference time, while RL further introduces an additional network and thus incurs another increase. Overall, however, the per-step inference time remains below 50 ms for all variants, which is also within an acceptable range.

D.2 Failure Reason Analysis

In the main text, we present the optimal policy equivalence result in Theorem 3.1. However, Co3G Manual does not satisfy the safety constraints in all environments. We attribute these failures primarily

Table 4: Performance with more conservative feasible value function.

Task	Co3G Manual		FISOR	
	r↑	c↓	r↑	c↓
PointButton2	**0.06**	**0.96**	**0.03**	**0.23**
PointCircle2	**0.42**	**0.42**	0.48	1.42
PointGoal1	**0.32**	**0.75**	**0.26**	**0.20**
PointGoal2	**0.20**	**0.96**	**0.05**	**0.00**
PointPush2	**0.14**	**0.94**	**0.09**	**0.06**

to inaccurate learning of the optimal feasible value function, which is overly underestimated in some environments. To investigate this, we reduce the expectile τ to 0.8 and evaluate Co3G Manual and FISOR under this setting. The results are shown in Table 4. We can observe that, with a more conservative optimal feasible value function, Co3G Manual satisfies the safety constraints in all environments where it previously failed. Moreover, while achieving safety in more environments than FISOR, it also obtains substantially higher rewards across all safe environments. These results further demonstrate the strong capability of Co3G Manual to finely control the reward–safety trade-off.

Table 5: Unconstrained reward performance. **Blue**: The best performing agent.

Task	Co3G Manual	Co3G RS	CCAC	CAPS	CDT
	r↑	r↑	r↑	r↑	r↑
CarButton1	0.53	**0.56**	0.43	0.08	0.21
CarButton2	**0.60**	**0.60**	0.45	0.02	0.31
CarCircle1	**0.75**	0.48	0.14	0.71	0.63
CarCircle2	**0.72**	0.57	0.71	0.69	0.69
CarGoal1	0.75	**0.81**	0.27	0.41	0.68
CarGoal2	0.81	**0.86**	0.82	0.24	0.47
CarPush1	**0.44**	0.41	-0.34	0.22	0.21
CarPush2	**0.46**	0.35	0.09	0.10	0.17
PointButton1	0.66	**0.71**	0.65	0.14	0.47
PointButton2	**0.64**	0.60	0.60	0.29	0.40
PointCircle1	**0.83**	-0.05	0.25	0.76	0.58
PointCircle2	**0.82**	0.43	0.15	0.68	0.65
PointGoal1	**0.76**	0.74	0.10	0.46	0.67
PointGoal2	**0.84**	0.83	0.69	0.52	0.58
PointPush1	0.45	**0.47**	0.02	0.19	0.30
PointPush2	0.40	**0.42**	-0.14	0.22	0.18
AntVel	**1.06**	0.99	-1.01	0.99	0.97
HalfCheetahVel	1.41	**1.53**	1.09	0.99	0.96
HopperVel	**0.71**	0.37	0.01	0.23	0.55
SwimmerVel	**0.51**	0.47	-0.01	0.25	0.45
Walker2dVel	0.79	0.11	0.08	**0.93**	0.64
Average	**0.71**	0.58	0.24	0.44	0.51

D.3 Zero-shot Adaptation to Unconstrained Settings

In some scenarios, safety constraints may be removed, leaving task completion, i.e., reward maximization, as the sole objective. To examine this setting, we evaluate Co3G against other soft-constraint methods with adjustable conservatism. In our experiments, CCAC and CDT use 300 as the input Cost Return, which typically corresponds to the maximum safety-violation level in the offline dataset. CAPS, similar to Co3G RS, selects actions by maximizing the reward value function. The results are

shown in Table 5. We can observe that Co3G outperforms the soft-constraint baselines on most tasks and in terms of average performance, further demonstrating its flexible and strong capability to adjust the reward–safety trade-off.

Table 6: Comparison pure offline with offline+online.

Task	Pure Offline		With Online Rollouts	
	r↑	c↓	r↑	c↓
CarButton1	**-0.01**	**0.30**	**0.00**	**0.36**
CarButton2	**-0.03**	**0.42**	**-0.03**	**0.58**
CarCircle1	**0.12**	**0.32**	**0.14**	**0.00**
CarCircle2	**0.21**	**0.34**	**0.18**	**0.02**
CarGoal1	**0.13**	**0.25**	**0.18**	**0.66**
CarGoal2	**0.02**	**0.18**	**0.02**	**0.90**
CarPush1	**0.19**	**0.51**	**0.21**	**0.71**
CarPush2	**-0.03**	**0.66**	**0.01**	**0.60**
PointButton1	**-0.04**	**0.76**	**-0.01**	**0.74**
PointButton2	0.01	1.35	**0.02**	**0.90**
PointCircle1	0.17	1.02	0.04	1.85
PointCircle2	**0.33**	**0.65**	**0.28**	**0.35**
PointGoal1	**0.11**	**0.20**	**0.11**	**0.17**
PointGoal2	**0.13**	**0.77**	**0.14**	**0.81**
PointPush1	**0.17**	**0.42**	**0.12**	**0.26**
PointPush2	**0.09**	**0.59**	**0.12**	**0.76**
AntVelocityV1	**0.94**	**0.00**	**0.93**	**0.01**
HalfCheetahVelocityV1	**1.01**	**0.00**	**1.02**	**0.00**
HopperVelocityV1	**0.09**	**0.20**	**0.10**	**0.32**
SwimmerVelocityV1	**0.37**	**0.33**	**0.38**	**0.20**
Walker2dVelocityV1	**0.58**	**0.43**	**0.61**	**0.22**
Average	**0.22**	**0.46**	**0.22**	**0.50**
Safe Percent	19/21		20/21	

D.4 Adding Some Online Rollouts for Co3G RL

Since Co3G RL adopts an online algorithm, it naturally fits the online training paradigm. We therefore conduct experiments on the use of online rollouts. In this experiment, we collect one online trajectory every 20,000 training steps, resulting in 100 additional online trajectories that are used together with the offline data for training. The results are shown in Table 6. We can observe that incorporating a small number of online trajectories indeed improves policy performance: the resulting policy satisfies the safety constraints in more environments and achieves slightly higher overall reward than the purely offline setting. However, likely because the amount of online data remains limited, the overall improvement is modest. Increasing the rollout frequency may lead to better results, but would also incur additional training cost.

D.5 Extending Co3G to Soft Constraint Settings

Our compositional generation formulation is not limited to the hard-constraint setting; it can also be extended to the soft-constraint setting. Specifically, under soft constraints, the optimal feasible value functions Q_h and V_h are replaced with cost value functions Q_c and V_c learned via expectile regression. Action generation is then performed via rejection sampling based on a given constraint threshold, following the same value-function learning and action-selection procedure as CAPS. We therefore conduct experiments in the soft-constraint setting under three constraint thresholds, $[20, 40, 80]$. The results are shown in Table 7. We observe that Co3G achieves the best average safety performance. Moreover, it satisfies the safety constraints in the largest number of environments while

Table 7: Soft constraint setting results.

Task	Co3G		CDT		CAPS		CCAC	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
CarButton1	**0.03**	**0.98**	0.16	3.07	**0.02**	**0.51**	0.43	10.81
CarButton2	**0.10**	**0.92**	0.25	3.55	**-0.09**	**0.33**	0.38	12.31
CarCircle1	**0.36**	**0.33**	0.56	2.77	0.59	2.28	0.15	5.90
CarCircle2	**0.41**	**0.92**	0.64	3.66	0.58	2.73	0.52	1.06
CarGoal1	**0.33**	**0.36**	0.67	1.23	**0.31**	**0.48**	0.80	1.94
CarGoal2	**0.30**	**0.95**	0.50	2.19	**0.14**	**0.41**	0.87	4.83
CarPush1	**0.24**	**0.15**	**0.29**	**0.68**	**0.20**	**0.35**	**0.10**	**0.68**
CarPush2	**0.11**	**0.90**	0.19	1.95	**0.07**	**0.54**	0.24	5.35
PointButton1	0.18	1.44	0.52	2.95	**0.08**	**0.76**	0.60	4.33
PointButton2	**0.18**	**0.92**	0.45	3.11	0.15	1.03	0.57	4.96
PointCircle1	0.56	1.35	**0.58**	**0.78**	**0.54**	**0.56**	0.64	2.64
PointCircle2	**0.35**	**0.30**	0.62	1.04	**0.54**	**0.85**	0.49	1.50
PointGoal1	**0.34**	**0.35**	0.71	1.08	**0.34**	**0.29**	0.73	1.45
PointGoal2	**0.33**	**0.73**	0.54	2.04	**0.36**	**0.66**	0.70	4.56
PointPush1	**0.22**	**0.19**	**0.27**	**0.95**	**0.15**	**0.32**	**0.17**	**0.60**
PointPush2	**0.21**	**0.42**	0.20	1.38	**0.13**	**0.49**	0.13	1.83
AntVelocityV1	**0.89**	**0.14**	**0.99**	**0.62**	**0.95**	**0.41**	**-1.01**	**0.00**
HalfCheetahVelocityV1	**0.90**	**0.09**	**0.97**	**0.15**	**0.95**	**0.56**	**0.96**	**0.96**
HopperVelocityV1	**0.24**	**0.66**	**0.76**	**0.72**	**0.58**	**0.67**	**0.10**	**0.91**
SwimmerVelocityV1	0.27	1.59	**0.65**	**0.91**	**0.48**	**0.41**	**0.03**	**0.61**
Walker2dVelocityV1	**0.74**	**0.01**	**0.78**	**0.26**	**0.80**	**0.44**	0.43	2.64
Average	**0.35**	**0.65**	0.54	1.67	**0.38**	**0.72**	0.38	3.33
Safe Percent	18/21		8/21		18/21		6/21	

also obtaining the best return in the largest number of environments, demonstrating the effectiveness of compositional generation. However, its overall advantage in the soft-constraint setting is less pronounced than in the hard-constraint setting.

D.6 BulletGym Results

To provide a more comprehensive evaluation of our method, we conduct additional experiments on the BulletGym benchmark [19], with results reported in Table 8. Since the safety constraints in BulletGym are comparatively less demanding, we set $\delta_h = 0.0$ for Co3G RS and $b_g = 2.0$ for Co3G RL to reduce policy conservatism. Co3G RS achieves rewards comparable to FISOR, but exhibits weaker safety performance in some environments, particularly DroneRun, likely due to inaccurate estimates of Q_h. In contrast, Co3G Manual and Co3G RL achieve rewards comparable to TREBI with substantially better safety performance, while also obtaining considerably higher rewards than FISOR. These results further demonstrate Co3G's flexibility in controlling the reward–safety trade-off across domains. They also suggest that appropriately reducing the conservatism of Q_c enables Co3G RL to achieve strong reward performance without producing overly conservative policies.

D.7 Standard Deviation of the Main Results

In this section, we present the standard deviation of Co3G and various baselines computed with three different random seeds, as shown in Table 9. The results demonstrate that variants of Co3G achieve lower standard deviation compared to other baselines, indicating their stability.

D.8 Detailed ablation results

In this section, we provide the detailed results of the ablation studies, as shown in Table 10.

Table 8: Normalized results on BulletGym tasks. **Bold**: Safe agents. Gray: Unsafe agents. **Blue**: Safe agent with the highest reward.

Task	TREBI		FISOR		Co3G Manual		Co3G RS		Co3G RL	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
AntRun	0.66	2.65	**0.40**	**0.21**	**0.71**	**0.64**	**0.30**	**0.00**	**0.56**	**0.29**
BallRun	0.29	1.84	**0.18**	**0.04**	**0.21**	**0.50**	**0.17**	**0.83**	**0.31**	**0.97**
CarRun	**0.96**	**0.78**	**0.73**	**0.06**	**0.92**	**0.12**	**0.65**	**0.45**	**0.97**	**0.00**
DroneRun	0.51	1.07	**0.22**	**0.91**	**0.53**	**0.65**	0.43	2.21	0.50	2.02
AntCircle	0.49	1.38	**0.20**	**0.00**	**0.42**	**0.13**	**0.34**	**0.12**	**0.42**	**0.00**
BallCircle	0.64	1.16	**0.31**	**0.00**	**0.56**	**0.61**	**0.45**	**0.00**	**0.66**	**0.20**
CarCircle	**0.49**	**0.92**	**0.40**	**0.05**	**0.45**	**0.83**	**0.47**	**0.00**	**0.66**	**0.00**
DroneCircle	0.55	1.28	**0.48**	**0.04**	**0.58**	**0.36**	**0.15**	**0.77**	**0.50**	**0.09**
Average	0.57	1.39	**0.37**	**0.16**	**0.55**	**0.48**	**0.37**	**0.55**	**0.57**	**0.45**
Safe Percent	2/8		8/8		8/8		7/8		7/8	

Table 9: Detailed standard deviation results of the main experiment. All results are computed using three different random seeds.

Task	CDT		CCAC		CAPS		BC-Safe		LSPC		FISOR		Co3G Manual		Co3G RS		Co3G RL	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
CarButton1	0.06	2.06	0.05	1.32	0.03	0.96	0.01	1.42	0.04	1.62	0.02	0.31	0.03	0.38	0.01	0.05	0.02	0.30
CarButton2	0.08	6.91	0.06	4.04	0.05	1.43	0.01	0.58	0.05	0.61	0.02	0.55	0.01	0.23	0.03	0.23	0.01	0.42
CarCircle1	0.05	1.28	0.46	7.89	0.02	0.62	0.05	1.17	0.06	0.89	0.02	1.14	0.01	0.17	0.03	0.57	0.01	0.32
CarCircle2	0.02	1.25	0.04	5.10	0.03	1.61	0.06	2.18	0.04	0.10	0.00	1.27	0.01	0.52	0.02	0.09	0.03	0.34
CarGoal1	0.05	0.91	0.03	1.22	0.02	0.45	0.03	0.28	0.10	0.34	0.04	0.57	0.04	0.19	0.03	0.12	0.05	0.25
CarGoal2	0.08	0.46	0.08	1.97	0.05	0.76	0.06	0.30	0.03	0.31	0.01	0.24	0.01	0.22	0.02	0.31	0.01	0.18
CarPush1	0.02	0.22	0.36	1.02	0.03	0.30	0.01	0.26	0.03	0.72	0.04	0.48	0.03	0.14	0.03	0.13	0.03	0.51
CarPush2	0.01	1.48	0.05	7.26	0.04	0.43	0.07	0.31	0.06	1.73	0.03	0.68	0.00	0.56	0.01	0.06	0.08	0.66
PointButton1	0.02	1.70	0.08	1.76	0.02	0.79	0.03	0.81	0.03	0.78	0.02	1.15	0.01	0.23	0.03	0.29	0.01	0.76
PointButton2	0.05	0.20	0.03	1.05	0.05	1.27	0.03	0.61	0.06	2.24	0.02	1.00	0.01	0.17	0.04	1.04	0.02	1.35
PointCircle1	0.01	0.64	0.06	3.82	0.04	0.15	0.06	0.10	0.26	10.85	0.07	4.48	0.01	0.45	0.03	0.67	0.13	1.02
PointCircle2	0.00	0.39	0.09	0.94	0.04	0.24	0.07	0.70	0.21	13.63	0.03	3.35	0.01	0.13	0.04	4.36	0.04	0.65
PointGoal1	0.01	1.09	0.01	0.16	0.03	0.25	0.05	0.19	0.09	0.33	0.01	1.64	0.08	0.12	0.01	0.29	0.02	0.20
PointGoal2	0.06	1.32	0.05	4.26	0.04	0.75	0.03	0.59	0.11	0.34	0.09	1.52	0.02	0.62	0.00	0.99	0.06	0.77
PointPush1	0.03	0.99	0.02	0.73	0.02	0.55	0.05	0.59	0.08	0.32	0.01	1.30	0.01	0.55	0.04	0.09	0.02	0.42
PointPush2	0.04	1.98	0.12	2.92	0.01	0.30	0.02	0.65	0.04	2.48	0.03	2.30	0.01	0.16	0.05	1.49	0.00	0.59
AntVelocityV1	0.02	0.05	0.00	0.00	0.01	0.02	0.02	0.59	0.00	0.22	0.01	0.00	0.00	0.03	0.04	0.02	0.01	0.00
HalfCheetahVelocityV1	0.01	0.11	0.04	0.11	0.01	0.09	0.03	0.63	0.03	19.88	0.01	0.00	0.01	0.03	0.00	0.00	0.00	0.00
HopperVelocityV1	0.23	0.37	0.14	4.17	0.13	0.36	0.05	0.20	0.18	3.93	0.04	1.00	0.02	0.08	0.03	0.08	0.04	0.20
SwimmerVelocityV1	0.00	0.23	0.17	1.53	0.04	0.18	0.09	3.36	0.09	5.57	0.01	0.02	0.08	0.15	0.01	0.07	0.14	0.33
Walker2dVelocityV1	0.04	1.83	0.15	2.87	0.00	0.06	0.01	0.01	0.01	0.25	0.11	0.77	0.01	0.23	0.07	0.10	0.18	0.43
Average	0.04	1.21	0.10	2.58	0.03	0.55	0.04	0.74	0.08	3.20	0.03	1.13	0.02	0.26	0.03	0.53	0.04	0.46

D.9 Detailed Sensitivity Analysis Results

Finally, we provide the detailed results of sensitivity studies, as shown in Table 11, 12 and 13.

E Impact Statement

The goal of this work is to advance offline safe RL, thereby enabling the deployment of learned policies in a broader range of real-world applications. For example, the vast majority of current Vision-Language-Action (VLA) models in embodied intelligence are based on flow matching policies. Therefore, our method could serve as a plug-and-play approach for safe fine-tuning of VLAs, potentially contributing further to advancements in robotics. The work presented in this paper does not introduce any additional ethical concerns beyond those commonly associated with RL research. Therefore, no special discussion of ethical issues is required.

Table 10: Detailed ablation results.

Task	Diff. Manual		Diff. RS		Diff. RL		SC Manual		SC RS		SC RL		FISOR RL		TD3		Lagrange		Org Q		W/o Noise	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
CarButton1	**-0.02**	**0.67**	**-0.06**	**0.39**	0.02	1.06	-0.03	2.40	0.02	1.53	-0.01	1.21	0.06	1.96	**-0.02**	**0.79**	**-0.01**	**0.18**	0.01	1.15	**-0.04**	**0.33**
CarButton2	**0.01**	**0.81**	**-0.04**	**0.44**	0.00	1.27	-0.13	1.39	0.02	1.38	**-0.08**	**0.82**	0.04	4.17	**-0.03**	**0.85**	**-0.03**	**0.73**	0.01	3.34	**-0.05**	**0.48**
CarCircle1	**0.31**	**0.97**	0.61	6.88	**0.13**	**0.62**	0.01	6.44	0.53	8.65	0.19	1.05	0.74	17.12	**0.18**	**0.91**	**0.14**	**0.82**	0.39	9.28	**0.18**	**0.47**
CarCircle2	**0.42**	**0.87**	**0.27**	**0.07**	**0.24**	**0.56**	0.11	1.77	0.48	2.04	**0.28**	**0.17**	0.69	19.53	**0.19**	**0.10**	**0.21**	**0.48**	0.40	3.19	**0.24**	**0.44**
CarGoal1	**0.36**	**0.74**	**0.18**	**0.14**	0.25	1.59	0.31	1.24	0.52	1.49	**0.25**	**0.92**	0.63	3.45	**0.12**	**0.58**	**0.12**	**0.03**	0.61	3.94	**0.20**	**0.48**
CarGoal2	**0.15**	**0.69**	**0.02**	**0.00**	0.09	1.63	0.14	1.64	0.29	2.36	0.09	1.34	0.40	6.36	0.05	1.11	**-0.02**	**0.35**	0.15	4.20	**0.05**	**0.74**
CarPush1	**0.33**	**0.82**	**0.18**	**0.64**	**0.26**	**0.76**	0.43	1.04	0.38	1.10	0.16	1.25	0.36	1.96	**0.16**	**0.83**	0.17	2.13	0.43	3.05	**0.20**	**0.98**
CarPush2	**0.14**	**0.88**	**0.11**	**0.12**	0.10	2.40	0.10	1.17	0.34	2.12	0.00	1.05	0.28	4.71	0.07	1.89	**0.00**	**0.55**	0.43	6.43	**0.07**	**0.74**
PointButton1	**0.00**	**0.91**	0.02	2.31	**0.01**	**0.73**	0.07	2.29	0.26	5.64	0.02	1.33	0.24	5.01	0.01	1.10	-0.01	1.13	0.19	5.26	-0.02	1.29
PointButton2	**0.00**	**0.90**	**0.04**	**0.71**	0.15	4.17	0.22	4.83	0.46	8.63	0.16	3.55	0.33	8.20	**0.07**	**0.90**	**0.03**	**0.84**	0.35	10.37	0.09	1.88
PointCircle1	0.53	3.63	-0.11	13.45	0.44	3.30	0.51	3.16	-0.05	22.70	0.09	1.14	0.68	12.46	0.09	13.31	0.06	5.88	0.13	22.08	0.21	8.38
PointCircle2	0.62	7.27	0.24	21.50	0.44	3.05	**0.63**	**0.20**	0.55	23.65	0.29	1.18	0.76	17.90	**0.19**	**0.75**	**0.12**	**0.76**	0.54	19.03	0.38	1.74
PointGoal1	0.27	1.01	0.36	6.31	0.25	1.21	0.34	1.56	0.69	6.33	**0.21**	**0.35**	0.71	4.32	**0.06**	**0.13**	**0.07**	**0.26**	0.68	8.31	**0.23**	**0.89**
PointGoal2	**0.23**	**0.91**	0.13	1.32	0.27	2.87	0.34	4.15	0.63	9.67	0.19	1.09	0.66	10.99	**0.10**	**0.89**	**0.12**	**0.75**	0.55	10.85	0.20	1.02
PointPush1	**0.14**	**0.69**	**0.17**	**0.63**	0.27	2.56	0.17	1.58	0.37	1.54	**0.12**	**0.47**	0.37	3.22	**0.11**	**0.60**	**0.11**	**0.53**	0.37	3.05	**0.14**	**0.99**
PointPush2	0.09	1.70	0.17	1.92	0.15	2.65	0.11	3.16	0.33	6.93	0.09	1.32	0.29	6.38	**0.06**	**0.70**	**0.07**	**0.94**	0.31	5.69	0.12	1.90
AntVelocityV1	**0.98**	**0.15**	**0.64**	**0.03**	**0.92**	**0.01**	**0.98**	**0.24**	**0.93**	**0.04**	**0.94**	**0.04**	**0.93**	**0.03**	**0.92**	**0.01**	**0.87**	**0.01**	**0.48**	**0.51**	**0.91**	**0.00**
HalfCheetahVelocityV1	**1.00**	**0.06**	**0.95**	**0.00**	**1.00**	**0.02**	**1.00**	**0.80**	**0.99**	**0.00**	**0.97**	**0.00**	**0.90**	**0.00**	**1.01**	**0.00**	**0.98**	**0.08**	**0.96**	**0.77**	**1.00**	**0.00**
HopperVelocityV1	**0.48**	**0.81**	**0.03**	**0.14**	**0.08**	**0.24**	**0.01**	**0.54**	**0.11**	**0.31**	0.26	1.08	**0.07**	**0.68**	**0.03**	**0.08**	**0.08**	**0.08**	0.06	1.10	**0.05**	**0.07**
SwimmerVelocityV1	**0.33**	**0.50**	**-0.05**	**0.04**	**0.24**	**1.00**	**0.09**	**0.18**	**-0.05**	**0.03**	**0.01**	**0.11**	0.05	1.72	**0.32**	**0.46**	**0.08**	**0.42**	**-0.01**	**0.44**	**0.33**	**0.30**
Walker2dVelocityV1	**0.78**	**0.74**	**0.13**	**0.53**	**0.65**	**0.92**	**0.79**	**0.21**	0.33	1.29	**0.72**	**0.55**	0.32	3.16	**0.68**	**0.35**	**0.43**	**0.54**	**0.08**	**0.46**	**0.58**	**0.75**
Average	0.34	1.23	0.19	2.74	0.28	1.55	0.30	1.90	0.39	5.12	**0.24**	**0.95**	0.45	6.35	0.21	1.25	**0.17**	**0.84**	0.34	5.83	0.24	1.14
Safe Percent	17/21		14/21		9/21		6/21		4/21		9/21		3/21		17/21		18/21		4/21		15/21	

Table 11: Detailed sensitivity analysis results on δ_h.

Task	-0.95		-0.9		-0.8		-0.7		-0.5		0.0	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
CarButton1	**-0.04**	**0.39**	**-0.02**	**0.63**	0.02	1.06	**0.02**	**1.00**	**0.02**	**1.00**	0.02	1.23
CarButton2	**-0.03**	**0.49**	**-0.01**	**0.62**	0.01	1.14	0.00	1.26	0.01	1.60	0.01	1.38
CarCircle1	**0.35**	**0.77**	0.37	1.57	0.47	3.64	0.46	4.33	0.47	4.61	0.47	5.21
CarCircle2	**0.34**	**0.11**	**0.39**	**0.24**	**0.45**	**0.78**	**0.45**	**0.85**	**0.45**	**0.92**	**0.45**	**0.80**
CarGoal1	**0.31**	**0.24**	**0.38**	**0.57**	0.47	1.21	0.51	1.16	0.51	1.33	0.51	1.38
CarGoal2	**0.08**	**0.44**	**0.08**	**0.29**	**0.09**	**0.40**	**0.12**	**0.48**	**0.11**	**0.44**	**0.12**	**0.65**
CarPush1	**0.33**	**0.65**	0.32	1.05	0.36	1.45	0.39	1.37	0.38	1.37	0.37	1.49
CarPush2	**0.18**	**0.76**	0.25	1.63	0.25	2.20	0.30	2.27	0.28	2.07	0.26	2.26
PointButton1	0.12	2.39	0.12	2.85	0.13	3.34	0.16	3.15	0.16	3.42	0.16	3.75
PointButton2	0.20	4.34	0.19	4.23	0.24	5.51	0.24	5.80	0.26	5.98	0.27	5.39
PointCircle1	-0.08	22.42	-0.05	21.21	-0.06	22.38	-0.06	23.36	-0.09	22.21	-0.08	22.92
PointCircle2	0.32	14.95	0.53	10.67	0.36	18.01	0.36	18.37	0.35	19.00	0.35	18.64
PointGoal1	0.57	3.95	0.59	4.81	0.58	6.88	0.60	7.16	0.57	7.31	0.58	8.30
PointGoal2	0.33	3.19	0.37	2.82	0.46	6.16	0.47	6.17	0.46	5.99	0.44	5.93
PointPush1	**0.32**	**0.76**	0.32	2.04	0.37	1.55	0.38	2.23	0.39	1.59	0.40	1.91
PointPush2	0.28	3.43	0.27	2.74	0.33	3.54	0.36	4.33	0.34	4.56	0.37	4.20
AntVelocityV1	**0.74**	**0.02**	**0.85**	**0.00**	**0.88**	**0.00**	**0.85**	**0.02**	**0.86**	**0.03**	**0.86**	**0.02**
HalfCheetahVelocityV1	**0.97**	**0.00**	**0.97**	**0.00**	**0.96**	**0.00**	**0.98**	**0.00**	**0.98**	**0.00**	**0.99**	**0.00**
HopperVelocityV1	**0.15**	**0.14**	**0.24**	**0.40**	0.18	1.56	**0.12**	**0.72**	**0.12**	**0.78**	**0.12**	**0.91**
SwimmerVelocityV1	**-0.04**	**0.10**	**-0.03**	**0.21**	**-0.02**	**0.05**	**-0.04**	**0.03**	**-0.05**	**0.02**	**-0.04**	**0.01**
Walker2dVelocityV1	**0.26**	**0.84**	**0.17**	**0.78**	**0.14**	**0.62**	**0.16**	**0.70**	**0.17**	**0.07**	**0.16**	**0.65**
Average	0.27	2.88	0.31	2.83	0.32	3.88	0.33	4.04	0.32	4.01	0.32	4.15
Safe Percent	14/21		10/21		6/21		8/21		8/21		7/21	

F Limitations

Although this work effectively addresses the reward–safety trade-off in offline safe RL, it still has several limitations. First, while we theoretically demonstrate the effectiveness of tuning (w_r, w_c) for policy improvement, all results rely on an implicit assumption: that the optimal value function is accurate. In practice, this assumption is difficult to satisfy, which is a primary reason why Co3G does not achieve optimal safety performance across all tasks. Therefore, learning more accurate optimal value functions remains one of the most important directions for future research in offline safe RL. Second, Co3G Manual still requires a small amount of online evaluation to select near-optimal guidance scales. A natural extension would be to reduce this overhead by performing guidance scale search via model-based policy evaluation or in simulators, rather than deploying directly in real environments. Additionally, we already provide RS and RL as two methods for setting guidance scales without extra cost. Further optimizing these approaches to achieve an improved reward–safety trade-off represents another promising direction for future work.

Table 12: Detailed sensitivity analysis results on M.

Task	2		3		4		7		13	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
CarButton1	**-0.01**	**0.48**	**-0.03**	**0.79**	**-0.06**	**0.70**	**-0.07**	**0.55**	**-0.04**	**0.39**
CarButton2	**-0.05**	**0.79**	-0.04	1.10	**-0.04**	**0.73**	**-0.03**	**0.56**	**-0.03**	**0.49**
CarCircle1	**0.27**	**0.22**	**0.26**	**0.99**	**0.31**	**0.32**	**0.29**	**0.18**	**0.35**	**0.77**
CarCircle2	**0.29**	**0.29**	**0.33**	**0.11**	**0.31**	**0.12**	**0.31**	**0.11**	**0.34**	**0.11**
CarGoal1	**0.28**	**0.16**	**0.32**	**0.39**	**0.27**	**0.37**	**0.31**	**0.29**	**0.31**	**0.24**
CarGoal2	**0.07**	**0.12**	**0.06**	**0.21**	**0.09**	**0.20**	**0.09**	**0.12**	**0.08**	**0.44**
CarPush1	0.34	1.32	0.32	1.83	**0.31**	**0.83**	0.28	1.07	**0.33**	**0.65**
CarPush2	0.23	2.05	0.22	1.54	0.25	1.35	0.19	1.36	**0.18**	**0.76**
PointButton1	0.07	1.94	0.10	2.37	0.11	2.26	0.08	3.12	0.12	2.39
PointButton2	0.19	3.46	0.20	4.11	0.18	4.42	0.17	2.97	0.20	4.34
PointCircle1	-0.05	20.25	-0.06	21.76	-0.10	16.88	-0.12	19.35	-0.08	22.42
PointCircle2	0.34	12.21	0.32	15.33	0.27	15.07	0.25	18.19	0.32	14.95
PointGoal1	0.51	5.06	0.52	5.27	0.54	5.85	0.51	4.91	0.57	3.95
PointGoal2	0.33	4.01	0.37	2.70	0.34	2.91	0.31	2.49	0.33	3.19
PointPush1	**0.29**	**0.84**	0.29	1.10	0.33	1.53	0.32	1.54	**0.32**	**0.76**
PointPush2	0.26	2.05	0.26	3.33	0.28	3.95	0.29	3.46	0.28	3.43
AntVelocityV1	**0.58**	**0.03**	**0.68**	**0.02**	**0.71**	**0.00**	**0.69**	**0.02**	**0.74**	**0.02**
HalfCheetahVelocityV1	**0.96**	**0.00**	**0.97**	**0.00**	**0.97**	**0.00**	**0.97**	**0.00**	**0.97**	**0.00**
HopperVelocityV1	**0.15**	**0.07**	**0.14**	**0.15**	**0.16**	**0.14**	**0.18**	**0.27**	**0.15**	**0.14**
SwimmerVelocityV1	**-0.05**	**0.22**	**-0.06**	**0.66**	**-0.05**	**0.62**	**-0.05**	**0.48**	**-0.04**	**0.10**
Walker2dVelocityV1	**0.27**	**0.77**	**0.25**	**0.85**	**0.23**	**0.90**	**0.33**	**0.88**	**0.26**	**0.84**
Average	0.25	2.68	0.26	3.08	0.26	2.82	0.25	2.95	0.27	2.88
Safe Percent	12/21		10/21		12/21		11/21		14/21	

Table 13: Detailed sensitivity analysis results on b_g and b_l.

Task	b_g=5.0/b_l=1.0		b_g=1.0		b_g=2.0		b_g=8.0		b_l=0.5		b_l=1.5	
	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓	r↑	c↓
CarButton1	**-0.01**	**0.30**	**-0.07**	**0.46**	**-0.06**	**0.40**	**-0.01**	**0.62**	**0.01**	**0.70**	**-0.02**	**0.28**
CarButton2	**-0.03**	**0.42**	**-0.07**	**0.33**	**-0.20**	**0.57**	**-0.02**	**0.73**	-0.03	1.34	**-0.06**	**0.84**
CarCircle1	**0.12**	**0.32**	0.54	6.24	0.36	1.97	**0.06**	**0.21**	**0.19**	**0.36**	**0.14**	**0.28**
CarCircle2	**0.21**	**0.34**	0.55	10.91	0.45	4.15	**0.15**	**0.20**	**0.21**	**0.33**	**0.19**	**0.10**
CarGoal1	**0.13**	**0.25**	**0.17**	**0.85**	**0.15**	**0.64**	**0.07**	**0.29**	**0.18**	**0.60**	**0.08**	**0.24**
CarGoal2	**0.02**	**0.18**	**-0.01**	**0.55**	**0.04**	**0.62**	**0.02**	**0.62**	**0.03**	**0.76**	**0.03**	**0.80**
CarPush1	**0.19**	**0.51**	**0.23**	**0.77**	**0.21**	**0.82**	**0.14**	**0.57**	**0.21**	**0.62**	**0.15**	**0.26**
CarPush2	**-0.03**	**0.66**	**0.03**	**0.99**	**-0.02**	**0.77**	**-0.01**	**0.50**	-0.01	1.09	**-0.02**	**0.25**
PointButton1	**-0.04**	**0.76**	0.04	1.40	**0.02**	**0.98**	**-0.02**	**0.76**	0.02	1.68	**-0.01**	**0.33**
PointButton2	0.01	1.35	0.12	2.17	0.09	1.88	0.02	1.02	0.07	1.59	**0.01**	**0.90**
PointCircle1	0.17	1.02	**0.17**	**0.32**	**0.16**	**0.21**	0.05	2.61	0.13	3.73	0.06	1.07
PointCircle2	**0.33**	**0.65**	**0.38**	**0.39**	**0.38**	**0.70**	**0.28**	**0.28**	**0.35**	**0.76**	**0.30**	**0.05**
PointGoal1	**0.11**	**0.20**	**0.22**	**0.67**	**0.09**	**0.35**	**0.06**	**0.07**	**0.15**	**0.35**	**0.06**	**0.36**
PointGoal2	**0.13**	**0.77**	**0.11**	**0.97**	**0.15**	**0.56**	**0.12**	**0.80**	0.23	1.67	**0.07**	**0.16**
PointPush1	**0.17**	**0.42**	**0.05**	**0.62**	**0.07**	**0.38**	**0.08**	**0.27**	**0.15**	**0.73**	**0.07**	**0.37**
PointPush2	**0.09**	**0.59**	**0.07**	**0.79**	**0.03**	**0.98**	**0.05**	**0.57**	0.11	2.35	**0.08**	**0.64**
AntVelocityV1	**0.94**	**0.00**	**0.95**	**0.00**	**0.94**	**0.00**	**0.91**	**0.00**	**0.89**	**0.01**	**0.92**	**0.00**
HalfCheetahVelocityV1	**1.01**	**0.00**	**0.97**	**0.00**	**1.02**	**0.00**	**1.00**	**0.00**	**1.01**	**0.00**	**1.01**	**0.00**
HopperVelocityV1	**0.09**	**0.20**	**0.36**	**0.98**	**0.15**	**0.26**	**0.05**	**0.30**	**0.11**	**0.11**	**0.10**	**0.16**
SwimmerVelocityV1	**0.37**	**0.33**	**0.13**	**0.12**	**0.17**	**0.03**	**0.26**	**0.50**	**0.32**	**0.35**	**0.25**	**0.30**
Walker2dVelocityV1	**0.58**	**0.43**	**0.78**	**0.14**	**0.64**	**0.26**	**0.38**	**0.57**	**0.71**	**0.53**	**0.38**	**0.39**
Average	**0.22**	**0.46**	0.27	1.41	**0.23**	**0.79**	**0.17**	**0.55**	**0.24**	**0.93**	**0.18**	**0.37**
Safe Percent	19/21		17/21		18/21		19/21		14/21		20/21	