Utility-Constrained Policy Optimization

Mehrdad Moghimi
York University
Toronto, Canada
moghimi@yorku.ca

Bernardo Ávila Pires
Google DeepMind
London, United Kingdom
bavilapires@google.com

Abstract

Constrained MDPs (CMDPs) are a widely adopted framework for incorporating safety into RL agents; however, the framework does not support risk-sensitive constraints. This can be problematic: For example, CMDPs allow for optimal solutions that, in order to satisfy the risk-neutral constraints, mix infrequent catastrophic behaviors and frequent, overly conservative ones. Moreover, prior empirical results suggest that enforcing stricter, risk-sensitive constraints can improve performance even under risk-neutral evaluation. The natural framework to incorporate risk-sensitive constraints is utility-constrained MDPs (UCMDPs), but no practical solutions for this problem existed. In this work, we introduce a simple yet powerful methodology for UCMDPs and constrained RL. Besides allowing for risk-sensitive constraints, our framework does not require us to fix constraint limits in advance of training the agent, provided that a sensible range is known. This increases policy flexibility and, in practice, allows for adjustments to these limits at no extra training cost. Besides benefiting from the generality of the framework, our agent shows strong performance in practice, consistently matching or outperforming existing baselines in several Safety Gymnasium benchmark tasks.

1 Introduction

An important part of building goal-oriented intelligent agents is to express and enforce constraints on their behavior, because behavioral requirements are present in many domains, in the form of operational constraints, safety requirements, regulations, user preferences, and so forth.

Reinforcement learning (RL) [Sutton and Barto, 2018] is a popular framework for designing goal-oriented agents, and the standard way to incorporate constraints into RL is through constrained Markov decision processes (CMDPs) [Altman, 1999]. A common RL problem is to find a policy that maximizes the expected (possibly discounted) sum of rewards, and the CMDP framework incorporates constraints on the expected (possibly discounted) sums of other reward-like signals observed along with the standard rewards.

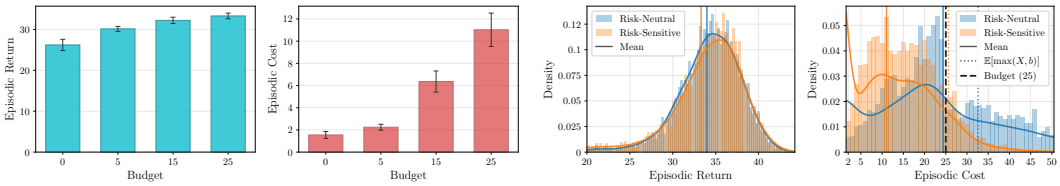
Despite being widely adopted as the basis for many constrained RL methods (for example, [Ji et al., 2023, 2024]), a substantial limitation of the CMDP framework is that constraints are risk-neutral, that is, they are only enforced in expectation. Because of risk neutrality, catastrophic constraint violations are acceptable as long as they are infrequent enough, however this can lead to unreliable (yet optimal) policies—for example, policies that satisfy constraints some of the time [Ray et al., 2019], or overly-conservative policies [Achiam et al., 2017].

We are not the first ones to consider this limitation. Kadota et al. [2006] extended CMDPs to include utility-based objectives and constraints, enabling risk-sensitive formulations, but without a clear practical solution, as optimal policies are history-dependent and often intractable to learn. Achiam et al. [2017] proposed to satisfy a conservative upper-bound on the constraint; Sun et al. [2021] proposed to terminate episodes whenever a constraint is violated; Huang et al. [2023] ensured that

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

constraints were satisfied frequently before optimizing the objective; Sootla et al. [2022a] added a penalty for constraint violation to the RL rewards, with increasing weights throughout training (in a barrier-like fashion); Sootla et al. [2022b] also added a similar penalty and used a curriculum on the constraint limits for learning constraint-satisfying policies (instead of the barrier-like violation penalty); Jiang et al. [2024] added a penalty relative to the constraint violation, and Sikchi et al. [2022] used model-based planning with a worst-case cost constraint during action selection. Somewhat surprisingly, these efforts to address risk-neutral CMDP constraints have led to improved performance in constrained RL benchmarks, *where constraints are risk-neutral*. This suggests that there are further performance gains to be made if we can improve how risk-sensitive constraints are satisfied, and we argue that we can accomplish this via a suitable problem formulation and a solution method for it.

In this work, we modify the utility-constrained MDPs introduced by Kadota et al. [2006] to allow for practical solutions, and introduce a practical method for UCMDPs. Our proposal builds on a framework introduced by Ávila Pires et al. [2025] that allows for solving unconstrained MDPs with expected-utility objectives via dynamic programming. A convenient side-effect of our modification is that constraint limits are inputs to policies along with the initial state of the MDP, so they can be set *after* training agents, similar to what Sauté RL affords [Sootla et al., 2022a]. The solution method we introduce is called *utility-constrained policies* (UCP), a Lagrangian deep RL agent for UCMDPs, and we demonstrate its effectiveness on the Safety Gymnasium benchmark [Ji et al., 2023]. Figure 1 illustrates some of our agent's capabilities in the *Safety Navigation CarGoal* (level 1) task.



(a) Episodic return (left) and cost (right) for different test-time cost budgets. For each seed, results are averaged over 1000 episodes; we then report the mean and 95% Student's t confidence intervals across five seeds.

(b) Empirical densities of episodic returns (left) and costs (right) for UCP with risk-sensitive vs. risk-neutral constraints. Solid lines show means; dotted lines show expected excess cost; dashed line indicates the budget.

Figure 1: Illustration of UCP in the *Safety Navigation* CarGoal1 task.

Figure 1a shows the episodic return and the cost incurred by UCP, as a function of different cost budgets. We use the same policy with different input values for the cost budgets (that is, we do not retrain the agent). We see that the performance of the agent increases with the cost budget, as one would expect from an active constraint. More importantly, we observe the marginal gains in return as the budget increases, which can guide the choice of an appropriate budget. Not only does the agent exhibit strong performance in the risk-neutral sense[1], but it also reduces the tail of the distribution above the budget. Figure 1b illustrates the benefits of risk-sensitive constraints. The plot shows the empirical distribution of the episodic return and cost for UCP (which has risk-sensitive constraints) and a variant of UCP with risk-neutral constraints (similar to CMDPs). The risk-neutral UCP closely satisfies the constraint (the mean cost is slightly below the budget), often exceeding the cost budget, and sometimes incurring more than twice the cost, whereas the risk-sensitive UCP, which only penalizes the cost above the budget, has a smaller tail of exceeding the budget (the expected excess cost is almost zero, i.e. $\mathbb{E}[\max(X, b)] \approx b$), and a relatively small reduction in episodic return.

Contributions.

- We introduce a simple modification to UCMDPs to support optimal Markov policies and practical solutions. This modification allows constraint limits to be set *after* computing a solution—they are inputs to policies along with the initial state of the MDP.
- We introduce *utility-constrained policies* (UCP), a Lagrangian algorithm for solving UCMDPs.
- We conduct an extensive evaluation in Safety Gymnasium, demonstrating UCP's strong performance: UCP matches (overlapping 95% confidence intervals) or outperforms the baselines in all

[1]To our knowledge, the best prior methods satisfying the expected cost budget of 25 are the model-based RCEPETS [Liu et al., 2020, Chua et al., 2018], achieving an average return of 29.08 [Ji et al., 2024, Table 3(e)], and the model-free early-terminated TRPO [Sun et al., 2021], achieving 22.09 [Ji et al., 2024, Table 3(b)]. See Section 5.1 for additional results.

six *Safety Velocity* tasks considered, and outperforms, by a large margin, all the baselines in all sixteen *Safety Navigation* tasks considered.
- We empirically demonstrate the effects of addressing the two CMDP limitations: UCP reduces the expectation of costs above the budget relative to its risk-neutral variant, and it can perform subject to different cost budgets without the need to retrain.

2 Background

Basic notation. We denote the real numbers by $\mathbb{R}$, natural numbers by $\mathbb{N}$, and $\mathbb{N}_0 \doteq \{0\} \cup \mathbb{N}$. $\mathbb{I}\{\cdot\}$ denotes the indicator function and $(x)_+ \doteq \max\{x, 0\}$. For a set $\mathcal{X}$, $\Delta(\mathcal{X})$ denotes the set of probability distributions over $\mathcal{X}$. For a random variable $X \sim \nu$, we write $\mathrm{df}(X) = \nu$.

Markov decision processes. In the standard RL problem, the environment is formalized as a Markov decision process (MDP) [Puterman, 2014], defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition kernel $\mathscr{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and a reward kernel $\mathscr{R} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$. In this work, we consider a more general MDP formulation, where we replace the reward signal with vector-valued *cumulants* [Sutton et al., 2011, Sutton and Barto, 2018]. In this formulation, we have $\mathcal{Z} \doteq \mathbb{R}^m$ (for some $m \in \mathbb{N}$) and we use, instead of the reward kernel, a *cumulant kernel* $\mathscr{R} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{Z})$. We assume that cumulants have uniformly bounded first moment, and we denote the set of such distributions by $\mathcal{P}^1(\mathcal{Z})$. We also assume that $\mathcal{A}$ is compact.

At each time step t, the agent observes state S_t, selects action A_t, and observes a reward vector $R_{t+1} \in \mathcal{Z}$ and next state S_{t+1}. The history up to timestep t is the sequence $S_0, A_0, S_1, R_1, \ldots, S_t, R_t$, and a policy is a mapping from histories to distributions over actions. We denote the set of history-based policies by Π_{H}. A *Markov policy* is a mapping $\mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{A})$.

Given a discount factor $\gamma \in (0, 1)$, the *discounted return* of a policy π, is the sum of discounted rewards observed by following π starting from S_t: $G_t^\pi \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. With $\gamma < 1$ and the assumption on bounded first moments, the return distributions are also in $\mathcal{P}^1(\mathcal{Z})$. We will occasionally use double-indexing for the returns, writing $G_t = (G_{t,1}, \ldots, G_{t,m})$.

In standard RL, we have $\mathcal{Z} = \mathbb{R}$ ($m = 1$) and the cumulant has the semantics of a reward; the problem is to find a policy that maximizes expected return, $\sup_{\pi \in \Pi_{\mathrm{H}}} \mathbb{E}[G_0^\pi]$, with $\mathcal{Z} = \mathbb{R}$. Under our assumptions, it can be shown that this problem can be solved with Markov policies (the supremum over Markov policies is equal to the supremum over history-based policies).

Distributional RL. Distributional RL [Morimura et al., 2010, Bellemare et al., 2023] extends value-related classic RL results to return distributions, and provides methods to evaluate policies in terms of their (per-state, per-action) return distributions (whereas classic policy evaluation concerns scalar value functions). On the practical side, a number of methods have been proposed for evaluating policies in a distributional sense, using function approximators, notably, categorical methods [Bellemare et al., 2017], quantile regression [Dabney et al., 2018], and particle-based methods [Freirich et al., 2019, Wiltzer et al., 2024].

Return distribution optimization. The problem of return distribution optimization (RDO) was studied by Marthe et al. [2023], Ávila Pires et al. [2025] as a family of RL-like objectives that we wish to optimize with dynamic programming. These objectives are expressed in terms of functionals of distributions (besides the expectation), and a special case of RDO is to optimize the *expected utility* objective: $\sup_{\pi \in \Pi_{\mathrm{H}}} \mathbb{E}[u(G_0^\pi)]$, where $u : \mathcal{Z} \to \mathbb{R}$ is a *utility*. This problem cannot be solved with Markov policies except for a small class of utilities (identity and exponential functions) [Marthe et al., 2023]. Informally, the state does not contain enough information for making correct decisions, and some kind of memory is required.

Ávila Pires et al. [2025] proposed to address this shortcoming by leveraging a summary of the history that they called *stock*, here denoted $Z_t \in \mathcal{Z}$ (and defined below), and which is incorporated into the objective being optimized and into the states, yielding *stock-augmented* RDO. Henceforth, we only consider all RDO problems to be stock-augmented.

We call the special case of RDO with expected utilities as *utility optimization* (UO):

$$\sup_{\pi \in \Pi_{\mathrm{H}}} \mathbb{E}[u(Z_0 + G_0^\pi)], \tag{1}$$

where Z_0 is an *initial stock* from the initial stock-augmented state (S_0, Z_0). The stock Z_t is defined recursively for $t \in \mathbb{N}_0$ as:

$$Z_{t+1} \doteq \gamma^{-1} (Z_t + R_{t+1}) . \tag{2}$$

Ávila Pires et al. [2025] showed that Equation (1) can be solved with policies that are Markov with respect to stock-augmented states when $x \mapsto u(x) - u(0)$ is positively homogeneous[2] and Lipschitz—a broader class than in the case without stock augmentation.

Constrained Markov decision processes. Informally, the constrained Markov decision process (CMDP) [Altman, 1999] problem is to optimize expected return subject to constraints on expected costs. Normally, these quantities are denoted separately, but it is convenient for us to combine the reward and costs into a vector of cumulants where, in the case of CMDPs (but not in general), the first coordinate corresponds to the reward, and the others to costs. Formulated this way, the CMDP problem is:

$$\sup_{\pi \in \Pi_{\mathrm{H}}} \mathbb{E}[G^{\pi}_{0,1}] \quad \text{subject to} \quad \mathbb{E}[G^{\pi}_{0,i}] \leq v_i \quad (2 \leq i \leq m),$$

where $\mathcal{Z} = \mathbb{R}^m$ and $v \in \mathcal{Z}$ is a vector of *constraint limits* (v_1 is irrelevant). From Altman [Theorem 2.1, 1999, p. 26], it can be shown that an optimal Markov policy exists for the constrained RL problem.

There are multiple solution methods for CMDPs; in this work, we focus on the *Lagrangian approach*, which consists of solving the *dual problem*

$$\inf_{\lambda \geq 0} \sup_{\pi \in \Pi_{\mathrm{H}}} \left(\mathbb{E}[G^{\pi}_{0,1}] - \sum_{i=2}^{m} \lambda_i \left(\mathbb{E}[G^{\pi}_{0,i}] - v_i \right) \right) . \tag{3}$$

For fixed λ, the inner optimization can be cast as an RL problem with a *scalarized reward* $R_{t,1} - \sum_{i=2}^{m} \lambda_i R_{t,i}$ and solved with standard methods.

Utility-constrained Markov decision processes. Kadota et al. [2006] generalized the CMDP formulation so that the objective and constraints can be utilities of the return distribution, namely, the UCMDP problem:

$$\sup_{\pi \in \Pi_{\mathrm{H}}} \mathbb{E}[f(G^{\pi}_{0})] \quad \text{subject to} \quad \mathbb{E}[g(G^{\pi}_{0})] \leq 0, \tag{4}$$

where $f : \mathcal{Z} \to \mathbb{R}$ is objective function, $g : \mathcal{Z} \to \mathbb{R}^n$ is the constraint function (for some $n \in \mathbb{N}$, with constraint limits folded into g). Kadota et al. [2006] studied the existence of a constrained optimal history-based policy, and the result of Ávila Pires et al. [2025] for the unconstrained case suggests that, without stock augmentation, we may not be able to tackle the UCMDP problem using Markov policies, except for a limited family of objective and constraint utilities.

3 Related Work

CMDPs. The challenge of learning optimal policies while adhering to safety or operational limits has been typically addressed using CMDPs, with different solution strategies.

Primal-dual methods are the most common approach to solve CMDPs, converting the constrained problem into an unconstrained one by penalizing constraint violations using adaptive Lagrange multipliers. Simple Lagrangian-based algorithms have been observed to perform well empirically, matching or exceeding more sophisticated approaches in earlier benchmark evaluations [Ray et al., 2019]. Several works have improved on the basic primal-dual methodology, in the context of constrained RL: RCPO uses a multi-timescale approach to guarantee convergence to a feasible fixed point [Tessler et al., 2019], while PID Lagrangian employs control-theoretic methods for more stable multiplier updates with less Lagrange multiplier oscillation [Stooke et al., 2020].

The so-called primal approaches, in contrast, do not introduce Lagrange multipliers and instead operate directly on the constrained optimization problem. These methods improve the reward while using constraint functions to remain within the feasible region. The seminal work in this category is CPO, which extends trust-region methods by ensuring each policy update satisfies constraint

[2]We say a function $f : \mathcal{Z} \to \mathbb{R}$ is *positively homogeneous* if there exists $t \in \mathbb{R}$ such that for all $\alpha > 0$ and $x \in \mathcal{Z}$, we have $f(\alpha x) = \alpha^t f(x)$.

upper-bounds [Achiam et al., 2017]. Subsequent primal methods have explored different update mechanisms: PCPO alternates between reward-maximizing and projection steps [Yang et al., 2020], FOCOPS simplifies the update by first solving in a non-parametric space [Zhang et al., 2020], and CRPO alternates between reward-maximizing and constraint-satisfying updates based on constraint status, all without using dual variables [Xu et al., 2021].

Beyond these optimization-based approaches, other methods have been explored. Lyapunov-based methods, inspired by control theory, define a function estimating the potential for future constraint violation and guarantee cumulative constraint satisfaction by ensuring the policy does not increase its expected value [Chow et al., 2018b]. Another line of work reformulates constraints, for instance by converting cumulative constraints into instantaneous, state-based constraints using Backward Value Functions [Satija et al., 2020], or by casting safe RL as a probabilistic inference problem and extending MPO [Abdolmaleki et al., 2018] to the constrained setting [Liu et al., 2022].

Beyond CMDPs. As mentioned, CMDPs suffer from the limitation of risk-neutral constraints. Kadota et al. [2006] introduced UCMDPs and studied the existence of optimal policies. These were history-based policies, which can be intractable to learn and are, we argue, impractical. A number of works have attempted to remedy the risk-neutral constraints in CMDPs, without explicitly adopting the UCMDP formulation. However, in these works, state augmentation has proved a crucial component for performance. Early work by Xu and Mannor [2011] demonstrated that chance-constrained MDPs could be solved by augmenting the state with accumulated reward, and Chow et al. [2018a] extended this idea to static Conditional Value-at-Risk (CVaR) constraints by tracking cumulative constraint cost. Building on these ideas, Sauté RL [Sootla et al., 2022a] and Simmer [Sootla et al., 2022b] augmented the state with discounted stock to guide decision-making, leading to improved constraint satisfaction and safer policies, while Sun et al. [2021] used the undiscounted stock to implement an early termination mechanism that enforces constraints. Budgeted MDPs [Boutilier and Lu, 2016, Carrara et al., 2019] also condition the policy on a budget, but the budget limits expected cost and is assigned by a dynamic program or selected by the policy, whereas our stock adds only realized cost. Appendix A contains a more detailed review of related work, comparing the design of existing methods and ours.

Distributional RL methods, which learn the full distribution of returns rather than just expectations [Morimura et al., 2010, Bellemare et al., 2017, Dabney et al., 2018], offer another avenue to go beyond risk-neutral constraints. These techniques enable a range of risk-aware constraints: constraining quantiles [Jung et al., 2022] or spectral risk measures [Kim et al., 2024] of the cost distribution, constraining the cost-to-go CVaR [Yang et al., 2021, 2023], and leveraging Extreme Value Theory to target high-impact tail events that expectation-based methods often miss [Gao et al., 2025]. In this work, we leverage distributional estimates to build the policy improvement criterion for our agent.

4 Methodology

Given the practical effectiveness of stock augmentation for constrained problems and its effectiveness for unconstrained UO problems, we adopt the stock-augmented formulation of the UCMDP problem (*cf.* Equation (4)):

$$\sup_{\pi\in\Pi_{\mathrm{H}}} \mathbb{E}[f(Z_0 + G_0^\pi)] \quad \text{subject to} \quad \mathbb{E}[g(Z_0 + G_0^\pi)] \leq 0, \tag{5}$$

where $f, g : \mathcal{Z} \to \mathbb{R}$ are objective and constraint utilities, respectively[3]. We propose to solve this problem with a primal-dual actor-critic method that we call *utility-constrained policies* (UCP). UCP optimizes the following objective (*cf.* Equation (3)) for $z \in \mathcal{Z}$:

$$\inf_{\lambda_z\geq 0} \sup_{\pi\in\Pi_{\mathrm{H}}} \Big(\mathbb{E}[f(Z_0 + G_0^\pi)] - \lambda_z \,\mathbb{E}[g(Z_0 + G_0^\pi)]\Big) \quad (Z_0 = z). \tag{6}$$

For a fixed λ_z, the dual problem in Equation (6) reduces to UO with $u(x) = f(x) - \lambda_z g(x)$. We repurpose SAC [Haarnoja et al., 2018] for solving UO following the rationale presented by Ávila Pires

[3] In general, g will have a vector-valued co-domain, but we restrict our presentation to the scalar case for simplicity (and because the extension to the vector-valued case is straightforward).

et al. [2025, Section 6], which amounts to modifying SAC so that policy improvement maximizes an estimate of the UO objective, rather than an estimate of the RL objective (Q-values).

The actor in SAC optimizes the RL objective via the reparameterization trick:

$$L^{\mathrm{RL}}_{\mathrm{actor}}(s,\theta) = Q_\psi(s, A_\theta) - \alpha \log \pi_\theta(A_\theta|s) \qquad (A_\theta \doteq \tanh(\mu_\theta(s) + \sigma_\theta(s)\,\varepsilon))$$

where Q_ψ is the critic's action-value function estimator with parameters ψ, θ are the actor parameters, and ε is sampled from a standard Normal distribution. The critic parameters in SAC are learned through least squares regression, similar to fitted Q iteration [Ernst et al., 2005].

For UCP, we replace Q_ψ with our estimate of the UO objective:

$$L^{\mathrm{UO}}_{\mathrm{actor}}(s,z,\theta) = \mathbb{E}[u(z + G_\psi(s,z,A_\theta))] - \alpha \log \pi_\theta(A_\theta|s,z) \;\; (A_\theta \doteq \tanh(\mu_\theta(s,z) + \sigma_\theta(s,z)\,\varepsilon))$$

where the $\mathrm{df}(G_\psi(s,z,a))$ is given by a distributional critic estimating return distributions through quantile regression [Dabney et al., 2018].

We also need to optimize λ_z in order to solve the dual problem in Equation (6). To do so, we generate episodes periodically throughout training starting from (S_0, z), observe the n independent samples of the G^π_0, denoted by $(G^{(1)}_0, \ldots, G^{(n)}_0)$, and update the Lagrange multiplier via $\lambda_z \leftarrow \left(\lambda_z + \eta_\lambda \left(\frac{1}{n}\sum_{i=1}^n g\left(z + G^{(i)}_0\right) - \epsilon\right)\right)_+$ where η_λ is a learning rate and $\epsilon > 0$ is a small tolerance[4]. In practice, we employ a function approximator to maintain λ_z for $z \in \mathcal{Z}$ (see Appendix C).

In our formulation, $-Z_{0,\geq 2}$ acts as the effective constraint limits, and a desirable property is to be able to change this value to obtain different constrained behavior from the same optimal UCMDP policy, without needing to retrain it (similar to the generalization across safety budgets discussed by Sootla et al. [2022a, p. 7]). Therefore, during training, we sample the initial stock Z_0 uniformly at random from a range of stock values that we wish the agent to learn to cope with. Note that the dependency of λ_z on z in Equation (6) is important to retain this flexibility to constraint limits. In some settings, this randomization may also introduce additional, helpful diversity to the training data [Ávila Pires et al., 2025]—it may increase coverage of the augmented-state space.

Besides sampling Z_0 at the beginning of each episode, we diverge from Ávila Pires et al. [2025] and decouple the discounting used in the agent's (future-looking) returns from the one used in the agent's (past-looking) stock updates. More specifically, we use an *undiscounted stock*: $\bar{Z}_{t+1} = \bar{Z}_t + R_{t+1}$. (We write $\bar{Z}_t$ to distinguish it from discounted stock and more clearly indicate where this stock update is used, but we do not modify the notation for $\bar{Z}_0$ since $\bar{Z}_0 = Z_0$.) We have found out that, empirically, this leads to better performance than using the discounted stock from Equation (2) (see the comparison in Section 5.3). We attribute this improvement to two factors. First, in standard constrained RL, constraint limits are defined over the undiscounted cumulative cost of an episode. Using an undiscounted stock directly aligns the agent's state with this actual evaluation metric, allowing the stock to act as a tracker of the remaining budget regardless of how much time has passed. Second, the reverse-discounted update exponentially inflates early costs over time. This creates a larger range of stock values that the agent must learn to control for, which can make early training too hard. By keeping the stock undiscounted, we prevent this expansion of the augmented state space, leading to an easier learning problem.

5 Experiments

In this section, we show that our practical approach to UCMDPs benefits from the expressivity of utility constraints, while attaining strong benchmark performance, in some cases with substantial performance gains. We evaluate UCP in the Safety Gymnasium benchmark [Ji et al., 2023], a suite of tasks where the agent must maximize expected return, while satisfying constraints on a scalar *cost* signal (Ji et al. [2023] stipulate a cost limit of 25 in their evaluations). In this paper, we consider only the single-agent tasks of the benchmark. With the cumulants in "(reward, cost)" form (thus,

[4]The constraint $\mathbb{E}[g(z + G_0)] \leq 0$ may not be strictly feasible [Boyd and Vandenberghe, 2004, Section 5.2.3, p. 226] (for example, when $g(z) = (z_1)_+$, as we consider for our experiments), making the optimization more challenging. In addition, imperfect return distribution estimates further complicate the problem. Introducing a tolerance mitigates both issues and has the practical benefit that λ_z decreases when the constraint is satisfied (unlike the case $\epsilon = 0$). In Appendix F we present a brief empirical study of the sensitivity of UCP to the choice of ϵ.

$\mathcal{Z} = \mathbb{R}^2$), we frame the constrained RL problem in the benchmark as a UCMDP (see Equation (5)) with $f(x) = x_1$ (maximize expected return), $g(x) = (x_2)_+$ (subject to not exceeding the cost limit), and $Z_{0,2} = -b$ for evaluation, where b is some budget that we can select at evaluation time to maximize performance, as long as the expected cost does not exceed the benchmark's limit of 25. In this case, the constraint in Equation (5) becomes $\mathbb{E}[(G_{0,c} - b)_+] \leq 0$. It targets cost realizations above b, weighted by their excess, while leaving already-safe behavior below b unrestricted. In terms of theoretical guarantees, we know from the results by Ávila Pires et al. [2025] that, with the above choices of f and g, the utility in the inner optimization of the dual problem (Equation (6)) can be optimized via dynamic programming, regardless of λ, since it is Lipschitz and positively homogeneous.

In a constrained RL setting, there are two important criteria for comparing policy performance: Whether they satisfy the constraint, and, among those that do, which has the best performance in terms of the objective. In Safety Gymnasium, we first assess whether the cost is less than 25 in expectation, and then compare expected returns.

We compare UCP to existing baselines in Section 5.1, where it matches or outperforms them on nearly all benchmark tasks. We also empirically demonstrate two key capabilities of stock-augmented UCMDPs: satisfying different constraint limits without retraining and better controlling the tails of cost distributions (Section 5.3). Implementation and hyperparameter tuning details are provided in Appendix C, and the code is available at `https://github.com/MehrdadMoghimi/UCP`.

5.1 Benchmark Performance

Our comparison considers several baselines from previous work (reproduced using OmniSafe [Ji et al., 2024]), in the *Safety Velocity* and *Safety Navigation* task groups.[5] These baselines combine "base" RL methods—PPO [Schulman et al., 2017], SAC [Haarnoja et al., 2018] and TD3 [Fujimoto et al., 2018]—with techniques for converting RL methods into constrained RL methods—Lagrangian optimization [Ray et al., 2019, Section 5.2] (abbreviated Lag), PID-Lagrangian [Stooke et al., 2020], Sauté RL [Sootla et al., 2022a] and early terminated MDPs [Sun et al., 2021] (abbreviated ET).

In UCP, the initial cost stock ($Z_{0,2}$, the negative of the cost budget) is sampled uniformly at random from $[-30, 0]$, and the cost budget for evaluation is set after training. We explore setting different cost budgets in Section 5.3, but for the plots in this section, we have selected the best-performing z_0 per task from $\{0, -5, -15, -25\}$, provided that the resulting policy has expected cost below 25 (with 95% confidence over seeds). We allowed ourselves this choice because we expect that, in practice, UCP constraint limits will be adjusted after training.

We report the estimates of expected (undiscounted) return and expected (undiscounted) cost for UCP and the baselines, in *Safety Velocity* (Figure 2) and *Safety Navigation* (Figure 3) tasks. UCP matches or outperforms the other methods in terms of expected returns, while satisfying the cost constraint throughout. Interestingly, in most of the *Safety Velocity* tasks, the cost has little bearing on

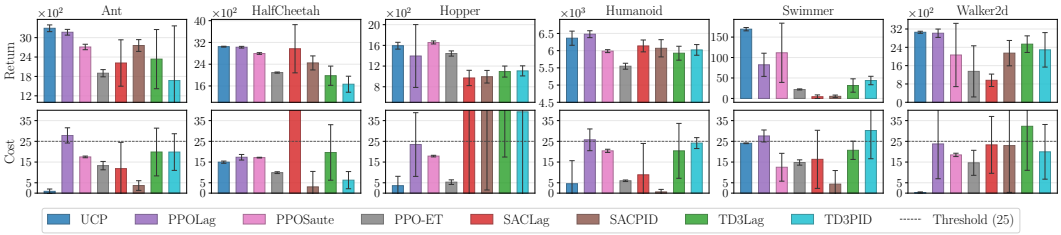


Figure 2: Comparison of UCP and OmniSafe baselines in the *Safety Velocity* tasks.

the performance of UCP—for example, in Ant and Walker2d, UCP outperforms the other methods without incurring almost any cost. While there are relatively minor performance gains compared to the best baseline in these tasks, by evaluating UCP with different budgets, we find that modifying the

[5] For ease of comparison, the results in this section can be found in table format (Table 12) in Appendix H. All intervals are 95% Student's t confidence intervals over 5 training runs. For each seed, we average the performance of 1000 episodes. Following common practice, we report *undiscounted* episodic returns and costs.

cost budget has little *marginal utility* for most of the safety velocity tasks, that is, modifying the cost budget does not significantly change the objective (we explore this in Section 5.3).

In the *Safety Navigation* tasks, we observe substantially larger gains. Moreover, in these tasks, we

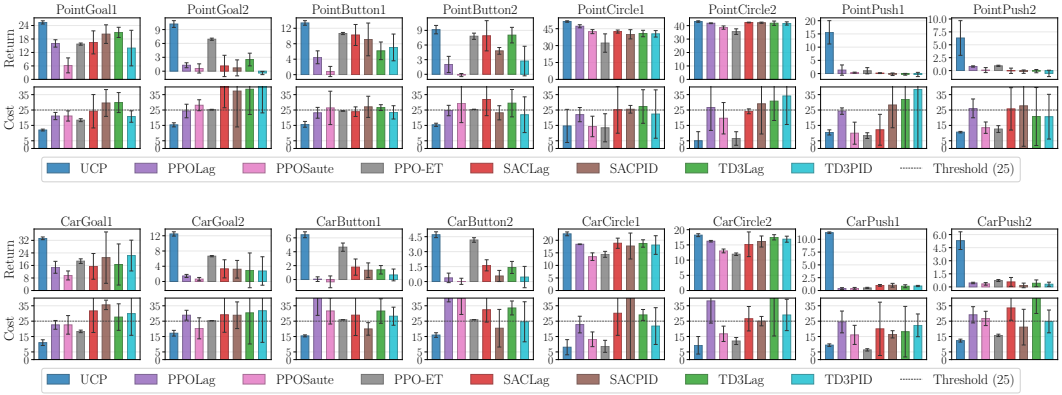


Figure 3: Comparison of UCP and OmniSafe baselines in the *Safety Navigation* tasks.

observe that selecting the cost budget clearly has marginal utility (see Table 13 in Appendix H). Accordingly, we report results at the commonly used budget of 25. While our extensive hyperparameter tuning improved the OmniSafe baselines in several cases (such as the Button tasks), UCP still inherently benefits from certain architectural choices (e.g., distributional critics). To transparently isolate the performance gains of our core proposed components from these standard RL techniques, we conduct a detailed ablation study in Section 5.3.

5.2 Multiple Constraints

Section 5.1 follows the standard Safety Gymnasium evaluation in which the environments return a single scalar cost per step [Ji et al., 2023]. Underneath, this scalar aggregates multiple cost components, such as hazards, vase contact, and vase velocity in CarGoal2. Since UCMDPs permit vector-valued costs, we also evaluate MC-UCP (multi-constraint UCP), with one stock coordinate, one distributional cost critic, and one Lagrange multiplier per cost component. We train MC-UCP for the same number of steps as UCP and evaluate it at different initial cost stocks, which reveals the reward–cost trade-off of a task more clearly. For example, as Table 1 shows for CarGoal2, relaxing only hazards raises hazard cost from 2.66 to 5.89, and relaxing only vase velocity raises that cost from 1.67 to 5.16, whereas relaxing only vase contact, which is rare, barely changes its cost. Relaxing several coordinates jointly also yields a large gain in return. With $z_{0,(2,3,4)} = (-15, -15, -15)$, MC-UCP attains return 14.15 at a total cost of 12.45, whereas UCP attains 12.5 at a total cost of 17.2 (Table 12). Appendix J presents the full results, including CarButton2 and CarPush2.

Table 1: MC-UCP on CarGoal2 at selected initial cost stocks. Total cost is the benchmark's scalar cost, which is not directly constrained. Intervals are 95% Student's t intervals across five seeds.

$(z_{0,2}, z_{0,3}, z_{0,4})$	Return	Total cost	Hazards	Vases Contact	Vases Velocity
$(0, 0, 0)$	-0.33 ± 0.57	4.30 ± 2.18	2.66 ± 1.74	0.34 ± 0.16	1.67 ± 0.47
$(-25, 0, 0)$	2.36 ± 1.30	7.40 ± 0.99	5.89 ± 0.82	0.29 ± 0.13	1.52 ± 0.23
$(0, -25, 0)$	2.17 ± 1.65	3.38 ± 1.27	1.38 ± 0.39	0.38 ± 0.22	1.99 ± 0.89
$(0, 0, -25)$	6.58 ± 2.05	7.53 ± 0.88	2.36 ± 0.69	0.87 ± 0.17	5.16 ± 0.28
$(-25, 0, -25)$	13.30 ± 1.60	19.25 ± 2.01	12.57 ± 0.60	0.98 ± 0.39	6.76 ± 1.67
$(-5, -5, -5)$	5.24 ± 1.16	2.82 ± 0.32	1.42 ± 0.43	0.22 ± 0.03	1.41 ± 0.13
$(-15, -15, -15)$	14.15 ± 0.42	12.45 ± 1.69	6.55 ± 0.58	0.82 ± 0.21	5.92 ± 1.41
$(-25, -25, -25)$	16.36 ± 0.68	27.85 ± 2.39	13.72 ± 0.83	2.19 ± 0.46	14.25 ± 2.45

5.3 Ablation Study

To better understand how each component of our method contributes to its empirical performance, we conduct a comprehensive ablation study with the components summarized in Table 2.

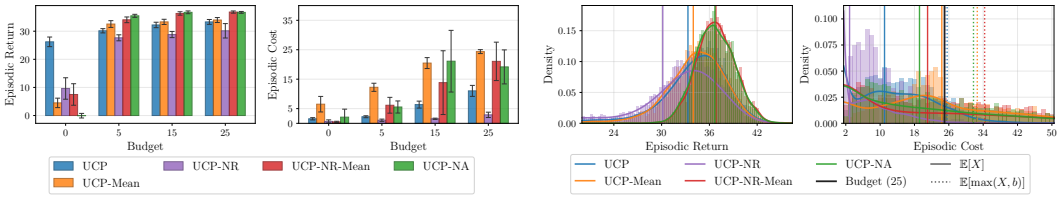
Table 2: Ablation components for all evaluated variants.

Method	Utility $g(x)$	Stock Augmentation	Stock Randomization	Undiscounted Stock
UCP	$(x_2)_+$	✓	✓	✓
UCP-DS	$(x_2)_+$	✓	✓	×
UCP-NR	$(x_2)_+$	✓	×	✓
UCP-Mean	x_2	✓	✓	✓
UCP-NR-Mean	x_2	✓	×	✓
UCP-NA	x_2	×	N/A	N/A

We evaluate the following variants: UCP-NA (No Augmentation), a Lagrangian RL baseline for CMDPs; UCP-NR-Mean (No Randomization, Mean Utility), which adds stock augmentation to study the effect of the larger state space; UCP-Mean, which further adds initial stock randomization while retaining a risk-neutral constraint; UCP-NR (No Randomization), which uses a risk-sensitive constraint but omits stock randomization, highlighting the role of randomization in generalization; UCP-DS (Discounted Stock), which combines risk sensitivity and randomization but uses discounted instead of undiscounted stock to assess this design choice; and UCP, with all components.

We report the results for all methods and tasks across four budget levels, as a way to understand the impact of the budget selection in task performance. UCP-NR, UCP-NR-Mean, and UCP-NA are trained separately for each budget level, whereas UCP-Mean and UCP are trained once with the same number of timesteps and evaluated across all four budgets. UCP-DS performed substantially worse than all other variants; therefore, its results are reported separately in Appendix H, Table 18. For brevity, we discuss CarGoal1 results here, with full results across all environments presented in Appendix D.

By comparing UCP-NA and UCP-NR-Mean in Figure 4a, we observe that stock augmentation does not degrade performance and can even improve it, particularly under the tightest budget of 0. Next,



(a) Mean episodic return and cost across four cost budget levels (0, 5, 15, 25).

(b) Empirical distributions of episodic return and cost at cost budget of 25 (5 seeds, 1000 episodes).

Figure 4: Ablation study on CarGoal1: risk-sensitive utility reduces budget violations, and initial stock randomization enables cross-budget generalization without retraining.

comparing UCP and UCP-Mean to their non-randomized counterparts (UCP-NR and UCP-NR-Mean) shows that randomizing the initial stock enables generalization across budget levels without retraining, albeit with different effects. Relative to UCP-NR, UCP achieves a higher return at the cost of slightly increased constraint violations. This is expected, as UCP-NR is trained separately for each budget level and can learn more precise value estimates to satisfy the constraint. However, since the cost under UCP remains below the allocated budget within the 95% confidence interval, UCP is the preferred approach overall. In contrast, UCP-Mean does not provide a similar benefit: It does not improve return and, in fact, incurs a higher cost than UCP-NR-Mean. The worse performance of UCP-NR-Mean and UCP-Mean relative to UCP in some cases may be surprising, considering that UCP is optimizing for stricter constraints than the benchmark's. However, this is in line with findings from previous work, where stricter constraints improved performance in the risk-neutral sense (see

Section 1). Taken together, these results indicate that UCP achieves the best balance between return and cost among the considered variants. Importantly, this behavior is not specific to CarGoal1 and, as discussed in Appendix D, is consistently observed across other environments. Finally, at the tightest budget of 0, both UCP and UCP-NR achieve high return with near-zero cost, highlighting the effectiveness of the risk-sensitive utility. Notably, UCP attains significantly higher return without a proportional increase in cost, which suggests that the combination of initial stock randomization and risk-sensitive utility can provide an effective setup for strong performance under tight constraints.

To further investigate the effect of these components, Figure 4b shows the distribution of return and cost across all evaluation trajectories (5 seeds, 1000 episodes each) at budget 25. The distributions clearly show that the risk-sensitive utility reduces the probability of exceeding the budget. More precisely, Table 3 shows that UCP-NR achieves a near-zero probability of budget violation, albeit with a reduction in return. UCP also reduces the probability of exceeding the budget, but we believe that the exploration from initial stock randomization mitigates the associated performance drop.

Table 3: Evaluation statistics on CarGoal1 with cost budget of 25. The third column shows the estimated conditional expectation of the undiscounted cost given that it (strictly) exceeds the budget, and the fourth to the estimated probability of the undiscounted cost exceeding the budget.

Method	Return	Cost	Cond. Expected Cost	Freq. of exceeding Budget
UCP	33.3 ± 0.2	11.0 ± 0.3	31.4	0.094
UCP-Mean	34.0 ± 0.1	24.4 ± 0.6	46.0	0.364
UCP-NR	30.1 ± 0.2	2.9 ± 0.2	33.3	0.007
UCP-NR-Mean	36.8 ± 0.1	21.1 ± 0.8	53.6	0.326
UCP-NA	36.7 ± 0.1	19.2 ± 0.6	48.1	0.292

6 Conclusion

We introduced stock augmentation to unlock practical solutions for UCMDPs and to overcome limitations of risk-neutral constraints in CMDPs. As a bonus, constraint limits are included in the initial augmented state, allowing policies to be evaluated with different limits without retraining. To solve stock-augmented UCMDPs, we proposed UCP, a Lagrangian actor-critic method with an outer optimization similar to TRPO-Lagrangian and PPO-Lagrangian [Ray et al., 2019], and an inner optimization consisting of three simple modifications to SAC [Haarnoja et al., 2018, 2017]: i) input undiscounted stock ($\bar{z}_t$) to actor and critic; ii) use a distributional critic (here, quantile regression [Dabney et al., 2018]) to estimate the cumulant distribution; and iii) train the actor to maximize the critic's expected-utility estimate instead of a Q-value estimate.

Despite strong empirical results, a key limitation is the lack of theoretical analysis. For instance, is there a principled reason to use undiscounted rather than discounted stock? What optimality guarantees can be established for Lagrangian dynamic programming in UCMDPs, and can they inform improvements to UCP? Beyond theory, further improvements to UCP may reduce approximation and statistical error, yielding more precise constraint satisfaction. Our implementation is just one instance of a broader class of utility-constrained agents; the SAC-based changes can extend to other RL methods (e.g., PPO [Schulman et al., 2017], MPO [Abdolmaleki et al., 2018]) and other distributional methods for the critic [Bellemare et al., 2017, Wiltzer et al., 2024, Zhang et al., 2021]. The outer optimization could also be improved, for example, with PID-Lagrangian [Stooke et al., 2020] updates.

Acknowledgments and Disclosure of Funding

We thank Mark Rowland for his review and feedback on this work. We thank the Digital Research Alliance of Canada (alliancecan.ca) for providing the computational resources used in this work. We thank the NeurIPS reviewers, area chair and program committee for their review and feedback on this work.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a Posteriori Policy Optimisation. In *International Conference on Learning Representations*, 2018.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained Policy Optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

Eitan Altman. *Constrained Markov Decision Processes*. Chapman & Hall/CRC, 1999.

Bernardo Ávila Pires, Mark Rowland, Diana Borsa, Zhaohan Daniel Guo, Khimya Khetarpal, André Barreto, David Abel, Rémi Munos, and Will Dabney. Optimizing Return Distributions with Distributional Dynamic Programming. *Journal of Machine Learning Research*, 26(185):1–90, 2025.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023.

Craig Boutilier and Tyler Lu. Budget Allocation using Weakly Coupled, Constrained Markov Decision Processes. In *UAI*, 2016.

Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Nicolas Carrara, Edouard Leurent, Romain Laroche, Tanguy Urvoy, Odalric-Ambrym Maillard, and Olivier Pietquin. Budgeted Reinforcement Learning in Continuous State Space. *Advances in Neural Information Processing Systems*, 32, 2019.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *Journal of Machine Learning Research*, 18 (167):1–51, 2018a.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A Lyapunov-Based Approach to Safe Reinforcement Learning. *Advances in Neural Information Processing Systems*, 31, 2018b.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. *Advances in Neural Information Processing Systems*, 31, 2018.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional Reinforcement Learning with Quantile Regression. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6, 2005.

Dror Freirich, Tzahi Shimkin, Ron Meir, and Aviv Tamar. Distributional Multivariate Policy Evaluation and Exploration with the Bellman GAN. In *International Conference on Machine Learning*, pages 1983–1992. PMLR, 2019.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

Shiqing Gao, Yihang Zhou, Shuai Shao, Haoyu Luo, Yiheng Bing, Jiaxin Ding, Luoyi Fu, and Xinbing Wang. Extreme Value Policy Optimization for Safe Reinforcement Learning. In *International Conference on Machine Learning*, 2025.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement Learning with Deep Energy-Based Policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Weidong Huang, Jiaming Ji, Chunhe Xia, Borong Zhang, and Yaodong Yang. SafeDreamer: Safe Reinforcement Learning with World Models. In *International Conference on Learning Representations*, 2023.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety Gymnasium: A Unified Safe Reinforcement Learning Benchmark. *Advances in Neural Information Processing Systems*, 36:18964–18993, 2023.

Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang. Omnisafe: An Infrastructure for Accelerating Safe Reinforcement Learning Research. *Journal of Machine Learning Research*, 25(285):1–6, 2024.

Hao Jiang, Tien Mai, Pradeep Varakantham, and Huy Hoang. Reward Penalties on Augmented States for Solving Richly Constrained RL Effectively. In *AAAI Conference on Artificial Intelligence*, volume 38, pages 19867–19875, 2024.

Whiyoung Jung, Myungsik Cho, Jongeui Park, and Youngchul Sung. Quantile Constrained Reinforcement Learning: A Reinforcement Learning Framework Constraining Outage Probability. *Advances in Neural Information Processing Systems*, 35:6437–6449, 2022.

Yoshinobu Kadota, Masami Kurano, and Masami Yasuda. Discounted Markov Decision Processes with Utility Constraints. *Computers & Mathematics with Applications*, 51(2):279–284, 2006.

Dohyeong Kim, Taehyun Cho, Seungyub Han, Hojun Chung, Kyungjae Lee, and Songhwai Oh. Spectral-Risk Safe Reinforcement Learning with Convergence Guarantees. *Advances in Neural Information Processing Systems*, 37:92465–92498, 2024.

Guopeng Li, Matthijs T.J. Spaan, and Julian F.P. Kooij. Off-Policy Safe Reinforcement Learning with Cost-Constrained Optimistic Exploration. In *International Conference on Learning Representations*, 2026.

Zuxin Liu, Hongyi Zhou, Baiming Chen, Sicheng Zhong, Martial Hebert, and Ding Zhao. Constrained Model-Based Reinforcement Learning with Robust Cross-Entropy Method. *arXiv preprint arXiv:2010.07968*, 2020.

Zuxin Liu, Zhepeng Cen, Vladislav Isenbaev, Wei Liu, Steven Wu, Bo Li, and Ding Zhao. Constrained Variational Policy Optimization for Safe Reinforcement Learning. In *International Conference on Machine Learning*, pages 13644–13668. PMLR, 2022.

Alexandre Marthe, Aurélien Garivier, and Claire Vernade. Beyond Average Return in Markov Decision Processes. *Advances in Neural Information Processing Systems*, 36:56488–56507, 2023.

James McCarthy, Radu Marinescu, Elizabeth Daly, and Ivana Dusparic. Optimistic Exploration for Risk-Averse Constrained Reinforcement Learning. In *European Conference on Artificial Intelligence*, pages 1977–1984. IOS Press, 2025.

Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Nonparametric Return Distribution Approximation for Reinforcement Learning. In *International Conference on Machine Learning*, pages 799–806, 2010.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

Harsh Satija, Philip Amortila, and Joelle Pineau. Constrained Markov Decision Processes via Backward Value Functions. In *International Conference on Machine Learning*, pages 8502–8511. PMLR, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, Better, Faster: Human-Level Atari with Human-Level Efficiency. In *International Conference on Machine Learning*, pages 30365–30380. PMLR, 2023.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning Off-Policy with Online Planning. In *Conference on Robot Learning*, pages 1622–1633. PMLR, 2022.

Aivar Sootla, Alexander I. Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H. Mguni, Jun Wang, and Haitham Ammar. Sauté RL: Almost Surely Safe Reinforcement Learning using State Augmentation. In *International Conference on Machine Learning*, pages 20423–20443. PMLR, 2022a.

Aivar Sootla, Alexander I Cowen-Rivers, Jun Wang, and Haitham Bou Ammar. Effects of Safety State Augmentation on Safe Exploration. In *Advances in Neural Information Processing Systems*, pages 34464–34477, 2022b.

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. PMLR, 2020.

Hao Sun, Ziping Xu, Meng Fang, Zhenghao Peng, Jiadong Guo, Bo Dai, and Bolei Zhou. Safe Exploration by Solving Early Terminated MDP. *arXiv preprint arXiv:2107.04200*, 2021.

Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. Second Edition., 2018.

Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. Horde: A Scalable Real-Time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction. In *International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768, 2011.

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward Constrained Policy Optimization. In *International Conference on Learning Representations*, 2019.

Harley Wiltzer, Jesse Farebrother, Arthur Gretton, and Mark Rowland. Foundations of Multivariate Distributional Reinforcement Learning. *Advances in Neural Information Processing Systems*, 37: 101297–101336, 2024.

Zifan Wu, Bo Tang, Qian Lin, Chao Yu, Shangqin Mao, Qianlong Xie, Xingxing Wang, and Dong Wang. Off-Policy Primal-Dual Safe Reinforcement Learning. In *International Conference on Learning Representations*, 2024.

Huan Xu and Shie Mannor. Probabilistic Goal Markov Decision Processes. In *International Joint Conference on Artificial Intelligence*, volume 22, page 2046. Citeseer, 2011.

Tengyu Xu, Yingbin Liang, and Guanghui Lan. CRPO: A New Approach for Safe Reinforcement Learning with Convergence Guarantee. In *International Conference on Machine Learning*, pages 11480–11491. PMLR, 2021.

Qisong Yang, Thiago D. Simão, Simon H. Tindemans, and Matthijs T.J. Spaan. WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 10639–10646, 2021.

Qisong Yang, Thiago D Simão, Simon H Tindemans, and Matthijs TJ Spaan. Safety-constrained reinforcement learning with a distributional safety critic: Q. yang et al. *Machine Learning*, 112(3): 859–887, 2023.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. Projection-Based Constrained Policy Optimization. In *International Conference on Learning Representations*, 2020.

Pushi Zhang, Xiaoyu Chen, Li Zhao, Wei Xiong, Tao Qin, and Tie-Yan Liu. Distributional Reinforcement Learning for Multi-Dimensional Reward Functions. *Advances in Neural Information Processing Systems*, 34:1519–1529, 2021.

Yiming Zhang, Quan Vuong, and Keith Ross. First Order Constrained Optimization in Policy Space. *Advances in Neural Information Processing Systems*, 33:15338–15349, 2020.

A Comparison with State-Augmented Constrained RL

Augmenting the state space with a variable that tracks the accumulated safety cost has been explored in recent constrained RL literature. In this section, we review two prominent prior works, Sauté RL [Sootla et al., 2022a] and the reward penalty method by Jiang et al. [2024], to highlight the theoretical and practical distinctions of our stock-augmented distributional approach.

Recall our notation for scalar reward and cost: The agent observes a task reward $R_{t,1}$ and a safety cost $R_{t,2}$. We aim to keep the cumulative cost below a threshold b. To achieve this, we augment the state with an undiscounted stock variable $\bar{z}_t \in \mathbb{R}$ related to the cost, initialized as $z_0 = -b$. The boundary for constraint violation is thus reached when $\bar{z}_t > 0$.

A.1 Sauté RL

Sauté RL [Sootla et al., 2022a] proposes to address the constrained RL problem by focusing on *almost sure* (probability 1) safety constraints. The core mechanism of Sauté RL relies on reshaping the immediate task reward based on the state of the (discounted) stock variable z_t.

Sauté RL defines a reshaped immediate reward $\tilde{R}_t$ as follows:

$$\tilde{R}_t(s_t, z_t, a_t) = \begin{cases} R_{t,1} & \text{if } z_{t+1} \leq 0, \\ -n & \text{if } z_{t+1} > 0, \end{cases} \tag{7}$$

where n is a large constant, with the goal of approximating the case with infinite penalty.

A fundamental limitation of this approach is that the almost-sure safety constraint can be problematic for dynamic programming (and likely for established deep RL methods). Ávila Pires et al. [2025, Appendix C.2] show that dynamic programming cannot optimize indicator utilities in the infinite-horizon discounted case. On the practical side, as noted by Jiang et al. [2024], the reward drops to a flat, large (n) penalty upon violation, regardless of the severity of the breach. That is, all violations are equally bad, and the agent may receive a limited policy-improvement signal from violations.

It is also worth noting that Sootla et al. [2022a] use separate discount factors for reward and cost. Because they merge these signals into a single modified reward, this effectively introduces one discount factor for the stock and another for accumulating the modified reward. Our view differs: while our formulation could also allow separate discount factors per signal, we instead use a single discount factor for all signals. As discussed in Section 4, our focus is on how, for a single signal, the agent should discount the backward-looking stock versus the forward-looking return estimate.

A.2 Reward Penalties on Augmented States

To overcome the limitations of Sauté RL, Jiang et al. [2024] proposed a formulation that also uses state augmentation but employs a more structured reward penalty mechanism for "soft" constraints.

Translated to our notation, assuming the stock variable tracks the undiscounted cost, Jiang et al. [2024] apply a scaled Lagrangian penalty directly to the immediate reward, but *only* when the trajectory becomes infeasible. Their modified reward is:

$$\tilde{R}_t(s_t, z_t, a_t) = \begin{cases} R_{t,1} & \text{if } z_t \leq 0 \text{ and } z_{t+1} \leq 0, \\ R_{t,1} - \frac{\lambda}{\gamma^t}(z_{t+1} - z_0) & \text{if } z_t \leq 0 \text{ and } z_{t+1} > 0, \\ R_{t,1} - \frac{\lambda}{\gamma^t} R_{t,2} & \text{if } z_t > 0, \end{cases} \tag{8}$$

where $\lambda > 0$ is a penalty parameter.

It is important to note that the term $(z_{t+1} - z_0)$ in the middle condition corresponds to the total accumulated cost from the beginning of the episode. Consequently, the entirety of the historical accumulated cost is applied as a single-step penalty exactly at the transition where the safety threshold is breached. Because this formulation adds the trajectory's cumulative cost history into the reward of a single time step, the immediate reward function exhibits a sharp discontinuity at the boundary $z_t = 0$, which the standard value function must subsequently approximate.

Despite this discontinuity, this formulation provides a more principled penalty structure than Sauté RL. By scaling the penalty linearly with the actual cost incurred (rather than applying an arbitrary

constant scalar $-n$), Jiang et al. [2024] ensure that the objective function preserves the relative ordering of trajectories based on the magnitude of their constraint violations. This scaling, they claim, enables the optimization of "soft" constraints, such as expected cost and CVaR, without restricting the agent to strictly zero-violation policies like Sauté RL.

A.3 UCP

All stock-augmented methods, including Sauté RL and our own UCP, share the structural advantage of being able to generalize across different safety budgets by sampling z_0 during training. However, prior methods rely on manually designing heuristics to reshape the *immediate scalar reward* based on the current stock z_t. Our methodology offers a more principled alternative using distributional RL.

Instead of using a modified step-by-step task reward, we maintain separate, unmodified signals for reward and cost. Our penalty is applied over the *distribution* of the entire future cost return using the expected utility $\mathbb{E}\left[g(Z_0 + G_0^\pi)\right]$. This distinction provides two key algorithmic advantages:

1. In the frameworks of Sootla et al. [2022a] and Jiang et al. [2024], the non-linear safety penalty is injected directly into the environment's immediate reward. In contrast, our approach keeps the original step-by-step reward and cost signals. We use a distributional critic to model the true distribution of cumulative costs, and apply the utility function over the full distribution during the policy update.
2. Because prior methods combine the task reward and cost penalty into a single scalar $\tilde{R}_t$, the standard critic entangles two different signals. Thus, changing the penalty weight invalidates the learned value function. By separating the reward and cost critics, and handling the trade-off dynamically via the Lagrange multiplier λ strictly at the policy optimization step, our method decouples the task gradient from the safety gradient and eliminates the need for piecewise reward engineering.

We emphasize that we augment the state with the *undiscounted stock*, which we have found, empirically, to be a better choice than augmenting with the discounted stock (see Section 4).

B Risk-neutral versus risk-sensitive constraints

In this section, we provide an illustrative example to motivate why we may want to frame a problem as a stock-augmented UCMDP, instead of a CMDP. Consider the MDP in Figure 5, where action a_0 transitions to the terminal state with reward 0, and action a_1 stays in state s_0, with a reward of 1. Suppose we wish to maximize return, subject to it being at most 1.

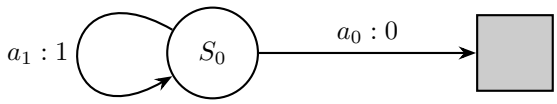


Figure 5: Simple example MDP for illustrating differences between the CMDP and UCMDP frameworks.

In CMDPs, we would maximize expected return subject to $\mathbb{E}[G_0^\pi] \leq 1$. The solution to this problem is a stochastic policy taking action a_1 with probability $(1+\gamma)^{-1}$. This policy mixes overshooting and undershooting the return, but achieves an expected return of exactly one.

This risk-neutral behavior, though optimal, can be undesirable: Undershooting the constraint comes at a cost in terms of the return, and overshooting it can be catastrophic. Yet the policy is optimal, so we cannot do better unless we change the problem formulation in some way.

In the example MDP in Figure 5, a history-based policy can deliver returns of exactly one with probability one. A stock-augmented Markov policy can also do so. Such a policy would be optimal in the stock-augmented CMDP problem, but so would the risk-neutral optimal policy of the original CMDP problem. We can use the UCMDP framework to remedy that.

Consider the (stock-augmented) UCMDP problem with $f(z) = z$ and $g(z) = (z)_+$: The objective is to maximize expected return, subject to $\mathbb{E}[(Z_0 + G_0^\pi)_+] \leq 0$. In this case, in (s_0, z) it is optimal to choose a_1 for $z \leq -1$ and a_0 otherwise. To satisfy the constraint on the return being at most 1 with an optimal policy, we can set $Z_0 = -1$, as the optimal policy at $(s_0, -1)$ is to choose a_1, transition to $(s_0, 0)$, and then choose a_0. This policy obtains a return of exactly one, with probability one, when starting from $(s_0, -1)$. Moreover, the optimal stochastic CMDP policy described above is no longer optimal for this problem.

C Implementation Details

Our implementation is extends the implementation of the SAC algorithm in LeanRL [Huang et al., 2022] with our proposed components.

C.1 Pseudocode

Algorithm 1 contains the pseudocode for UCP.

In the general UCMDP formulation (Equation (5)), the undiscounted stock $\bar{Z}_t \in \mathcal{Z}$ with $\mathcal{Z} \subseteq \mathbb{R}^m$ tracks all cumulants simultaneously. In the case of Safety Gymnasium, where $m = 2$ and f is the *first-coordinate projection function* (i.e. $f(z) = z_1$, so the objective is the expected return), the reward stock $z_{0,1}$ enters the objective only as an additive constant $z_{0,1} + G_{0,1}$ whose expectation shifts the objective by $z_{0,1}$; this does not affect the optimal policy. Consequently, we only augment the state with the *cost* stock $\bar{Z}_{t,2}$, since only $g(z_{0,2} + G_{0,2})$ depends non-trivially on the initial stock (having noted that we use the undiscounted stock $\bar{Z}_t$ instead of Z_t for empirical reasons). We implement stock augmentation by expanding the agent's observation space as described in Ávila Pires et al. [2025].

The cumulant distributions are estimated via quantile regression [Dabney et al., 2018], which estimates the marginal distributions of the total discounted cumulants (reward and cost in the case of Safety Gymnasium), and estimating the marginals is enough for estimating the expected utility objective in our case. Quantile regression uses N evenly-spaced cumulative probabilities $\tau_i = \frac{2i-1}{2N}$ for $i \in \{1, \ldots, N\}$, and the critic network is trained by minimizing the quantile regression loss across all pairwise combinations of quantiles and n-step cumulant estimates. For an n-step trajectory formed as $S_t, R_{t+1}, \ldots, R_{t+n}, (S_{t+n}, \bar{Z}_{t+n})$, we build n-step TD-error estimates as $\delta_{ij} \doteq \left(\sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n Y'_j - Y_i\right)_{i=1,j=1}^{N,N}$, where $(Y_i)_{i=1}^N$ are the critic network outputs for the augmented state-action triple $(S_t, \bar{Z}_t, A_t)$ and $(Y'_j)_{j=1}^N$ are the target critic network outputs for the augmented state-action triple $(S_{t+n}, \bar{Z}_{t+n}, A_{t+n})$. We then minimize the quantile regression loss

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \rho_{\tau_i}(\delta_{ij}) = \sum_{i=1}^{N} \sum_{j=1}^{N} |\delta_{ij}| \, |\tau_i - \mathbf{1}_{\{\delta_{ij} < 0\}}|$$

with respect to the critic network weights. Note that the target bootstrap values $(Y'_j)_{j=1}^N$ for the returns come from a double-critic, to mitigate overestimation bias: Inspired by Soft Actor-Critic [Haarnoja et al., 2018], whenever selecting the target quantiles, we use all the quantiles from the critic network for which the mean over quantiles is smaller. For the cost quantiles, we use a single critic. We use target networks for the actor and the critics, and we update them via Polyak averaging [Schwarzer et al., 2023].

Safety constraints are enforced by jointly optimizing the policy and a Lagrangian multiplier. The Lagrange multiplier, $\lambda(z_0)$, dynamically adjusts the penalty based on the initial cost stock (or budget) sampled at the beginning of each training episode. We use function approximation to maintain the estimates of the Lagrange multipliers λ_z for different stock values, via a K-bin discretization over $[z_{\min}, z_{\max}]$. In Appendix E we present a brief empirical study of the sensitivity of UCP to the choice of K.

C.2 Hyperparameters

The experiments use a shared set of core hyperparameters across all tasks, with specific adjustments made to the discount factor, n-step return, and critic learning rates depending on the task group

Algorithm 1 Utility-Constrained Policies (UCP)

Require: Utilities $f, g : \mathbb{R}^2 \to \mathbb{R}$; Quantile levels $\tau_i = \frac{2i-1}{2N}$, $i \in [N]$; Learning rates $\eta_\theta, \eta_\psi, \eta_\phi, \eta_\lambda$; Regularization weight β; n-step horizon n; Discount factor γ; Batch size B; Polyak coefficient τ_{ema}; K discretization levels for λ over $[z_{\min}, z_{\max}]$; Tolerance $\varepsilon \geq 0$.

Initialize Actor π_θ; Twin reward critics $G^{(1)}_\psi$, $G^{(2)}_\psi$; Cost critic G^c_ϕ; Target parameters $\bar{\psi} \leftarrow \psi$ and $\bar{\phi} \leftarrow \phi$

Initialize Discretized Lagrange Multipliers $\{\lambda_k\}_{k=1}^K$, each $\lambda_k \leftarrow \lambda_0 > 0$ *// index* $k(z_0) = \left\lfloor \frac{z_0 - z_{\min}}{z_{\max} - z_{\min}} (K-1) \right\rfloor + 1$

Initialize Replay buffer $\mathcal{B} \leftarrow \emptyset$; Cost trajectory buffer $\mathcal{B}_c \leftarrow \emptyset$ *// Fixed-size FIFO*

for each environment step $t = 0, 1, 2, \ldots$ **do**

  At episode start: sample $z_0 \sim \text{Uniform}\left(\left\{z_{\min}, z_{\min} + \frac{z_{\max}-z_{\min}}{K-1}, \ldots, z_{\max}\right\}\right)$ *// Initial stock randomization*

  $A_t \sim \pi_\theta(\cdot \mid S_t, \bar{Z}_t)$; $(S_{t+1}, R_{t+1}, C_{t+1}) \sim P(\cdot \mid S_t, A_t)$

  $\bar{Z}_{t+1} \leftarrow \bar{Z}_t + (R_{t+1},\ C_{t+1})$ *// undiscounted stock update*

  $\mathcal{B} \leftarrow \mathcal{B} \cup \{(S_t, \bar{Z}_t, A_t, R_{t+1}, C_{t+1}, S_{t+1}, \bar{Z}_{t+1}, z_{0,2}, \text{done}_t)\}$ *// $z_{0,2}$: episode-start cost stock, stored for multiplier lookup*

  if episode done **then**

    $\mathcal{B}_c \leftarrow \mathcal{B}_c \cup \{(z_{0,2}, G^c_0)\}$ *// G^c_0: episodic cost return*

  end if

  if $|\mathcal{B}| \geq B$ **then**

    Sample $\{(s_i, \bar{z}_i, a_i, r^{(n)}_i, c^{(n)}_i, s'_i, \bar{z}'_i, z_{0,i}, \text{done}_i)\}_{i=1}^B$ from $\mathcal{B}$ *// $r^{(n)}_i, c^{(n)}_i = \sum_{l=0}^{n-1} \gamma^l (R_{t+l+1}, C_{t+l+1})$*

    Distributional Critic Updates

    $\tilde{a}'_i \sim \bar{\pi}_\theta(\cdot \mid s'_i, \bar{z}'_i) \quad \forall i \in \{1, \ldots, B\}$ *// sample next actions for all transitions in batch*

    $j^*_i = \arg\min_{j \in \{1,2\}} \mathbb{E}_\tau \left[f\left(\bar{z}'_{i,1} + G^{(j)}_{\bar{\psi}}(s'_i, \bar{z}'_i, \tilde{a}'_i) \right) \right]$ *// select target critic by lower expected utility*

    $Y_i \leftarrow r^{(n)}_i + \big(1 - \text{done}_i\big)\gamma^n \left(G^{(j^*_i)}_{\bar{\psi}}(s'_i, \bar{z}'_i, \tilde{a}'_i) - \alpha \log \bar{\pi}_\theta(\tilde{a}'_i \mid s'_i, \bar{z}'_i) \right)$ *// distributional Bellman target*

    for $j \in \{1, 2\}$ **do** *// twin reward critics loss*

      $\mathcal{L}^{(j)}_Q(\psi) \leftarrow \frac{1}{BN^2} \sum_{i=1}^B \sum_{j=1}^N \sum_{m=1}^N \rho_{\tau_j} \left(Y_{i,m} - G^{(j)}_{\bar{\psi}}(s_i, \bar{z}_i, a_i)_{\tau_j} \right)$ *// all-pairs quantile loss*

    end for

    $\psi \leftarrow \psi - \eta_\psi \nabla_\psi \left(\mathcal{L}^{(1)}_Q + \mathcal{L}^{(2)}_Q \right)$

    $Y^c_i \leftarrow c^{(n)}_i + \big(1 - \text{done}_i\big)\gamma^n G^c_{\bar{\phi}}(s'_i, \bar{z}'_i, \tilde{a}'_i)$ *// cost target uses same action $\tilde{a}'_i$*

    $\mathcal{L}_C(\phi) \leftarrow \frac{1}{BN^2} \sum_{i=1}^B \sum_{j=1}^N \sum_{m=1}^N \rho_{\tau_j} \left(Y^c_{i,m} - G^c_{\bar{\phi}}(s_i, z_i, a_i)_{\tau_j} \right)$

    $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}_C(\phi)$

    Actor Update (SAC + Lagrangian with UO objective)

    $\tilde{a}_i \sim \pi_\theta(\cdot \mid s_i, \bar{z}_i)$ *// reparameterization: $\tilde{a}_i = \tanh\left(\mu_\theta(s_i, \bar{z}_i) + \sigma_\theta(s_i, \bar{z}_i) \odot \varepsilon\right)$, $\varepsilon \sim \mathcal{N}(0, I)$*

    $\lambda_i \leftarrow \lambda_{k(z_{0,i})}$

    $U_i \leftarrow \mathbb{E}_\tau \left[f\left(\bar{z}_{i,1} + G^{(j^*_i)}_{\bar{\psi}}(s_i, \bar{z}_i, \tilde{a}_i) \right) \right]$ *// j^*_i from current-state critic selection*

    $U^c_i \leftarrow \mathbb{E}_\tau \left[g\left(\bar{z}_{i,2} + G^c_{\bar{\phi}}(s_i, \bar{z}, \tilde{a}_i) \right) \right]$ *// $\bar{z}_{i,2}$ added to each quantile*

    $\mathcal{L}_{\text{reg},i} \leftarrow \beta \left(\|\mu_\theta(s_i, \bar{z}_i)\|^2 + \|\log \sigma_\theta(s_i, \bar{z}_i)\|^2 \right)$

    $\mathcal{L}_\pi(\theta) \leftarrow \frac{1}{B} \sum_{i=1}^B \frac{-U_i + \mathcal{L}_{\text{reg},i} + \lambda_i U^c_i}{1 + \lambda_i}$

    $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}_\pi(\theta)$

    Discretized Lagrange Multiplier Update

    if $|\mathcal{B}_c| > 0$ **then**

      for $(z^{(j)}_{0,2},\ G^{c,(j)}_0) \in \mathcal{B}_c$ **do**

        $\nu^{(j)} \leftarrow g\left(z^{(j)}_{0,2} + G^{c,(j)}_0 \right) - \varepsilon$ *// constraint violation; $\mathbb{E}[g(z_{0,2} + G^c_0)] \leq 0$ is the constraint*

      end for

      $\mathcal{L}_\lambda \leftarrow -\frac{1}{|\mathcal{B}_c|} \sum_j \lambda_{k(z^{(j)}_{0,2})}\, \nu^{(j)}$

      $\{\lambda_k\} \leftarrow \{\lambda_k\} - \eta_\lambda \nabla_{\{\lambda_k\}} \mathcal{L}_\lambda$; $\lambda_k \leftarrow \text{clip}(\lambda_k, \lambda_{\min}, \lambda_{\max})\ \forall k$

    end if

    Polyak Target Updates

    $\bar{\psi} \leftarrow (1 - \tau_{\text{ema}})\bar{\psi} + \tau_{\text{ema}}\psi$

    $\bar{\phi} \leftarrow (1 - \tau_{\text{ema}})\bar{\phi} + \tau_{\text{ema}}\phi$

  end if

end for

(*Safety Velocity* or *Safety Navigation*). These values are summarized in Table 4. Similar to the OmniSafe benchmark, *Safety Navigation* and Humanoid (from *Safety Velocity*) are trained for 3 million steps, while the remaining *Safety Velocity* tasks use 1 million steps. Also, Swimmer (from *Safety Velocity*) uses a discount factor of 0.995, while other tasks from *Safety Velocity* use 0.99. For both our method and the OmniSafe baselines, we performed a hyperparameter search over the discount factor $0.99, 0.995, 0.999$ and critic learning rates $3 \times 10^{-4}, 1 \times 10^{-4}, 3 \times 10^{-5}$, and reported results with the best hyperparameters for each baseline, for each task group. Despite this search, we found the default hyperparameters in OmniSafe to generally yield the best results. The notable exception was the cost normalization setting for off-policy methods; we found that enabling it (`cost_normalize=True`) improved the results for the Push and Button tasks with both Car and Point agents. Consequently, the baseline results for these specific tasks are reported with cost normalization enabled, while all other baseline configurations remain at their defaults.

Finally, our implementation supports n-step returns, for which we observed that larger values (e.g., $n = 10$) improved performance on *Safety Navigation* tasks. OmniSafe does not have n-step returns implemented, so their baselines do not use it. This may be contributing to the baseline performances observed in *Safety Navigation*, though, for reference and comparison, one can use UCP-NA as an RL baseline *with* n-step returns (see the discussion in Appendix H).

Table 4: Algorithm Hyperparameters.

Hyperparameter	***Safety Navigation***	***Safety Velocity***
Policy Learning Rate	3×10^{-4}	
Batch Size	256	
Replay Buffer Size	10^6	
Target Smoothing Coef. Actor (τ_p)	0.05	
Target Smoothing Coef. Critic (τ_q, τ_c)	0.005	
Number of Quantiles (N)	100	
Network Hidden Sizes	$[256, 256]$	
Constraint Violation Tolerance (ϵ)	1×10^{-6}	
Discount Factor* (γ)	0.999	0.99
N-Step Return (n)	10	1
Critic Learning Rate	3×10^{-5}	1×10^{-4}
Training steps*	3×10^6	1×10^6

It is worth noting that some prior works (e.g., Wu et al. [2024], Liu et al. [2022], McCarthy et al. [2025], Li et al. [2026]) modify the simulation time step in Safety Gymnasium agents or disable random layout generation in the environments. In contrast, we keep all Safety Gymnasium settings unchanged. Since such modifications can substantially affect the achievable return and cost, our results are not directly comparable to those works.

C.3 Compute Resources

All experiments were conducted on the Digital Research Alliance of Canada servers. OmniSafe experiments were performed on CPU nodes with 4 CPU cores and 4 GB RAM; each run took up to 2 days, particularly for Safety Navigation tasks with 3M steps. Experiments for UCP and its variants were run on GPU nodes, each using 1/8 of an A100 GPU (5 GB VRAM), along with 2 CPU cores and 15 GB RAM. Depending on the number of training steps, each run took approximately 12–36 hours.

D Component Ablations

Despite our efforts to ensure a fair comparison between UCP and baseline algorithms, by using the same hyperparameters where possible and performing comparable tuning across methods, certain differences (e.g., the use of a distributional value function or n-step returns) can still significantly influence the results. To better isolate the contribution of each proposed component, we therefore conduct an ablation study. In Section 5.3, we introduced the different variants of our method; here,

we present their performance across all environments. We observe that several patterns discussed in Section 5.3 consistently reappear in other environments.

Before comparing algorithms, it is helpful to highlight the differences between *Safety Velocity* and *Safety Navigation* tasks. In *Safety Velocity* tasks, returns across different budget levels are generally similar. In some cases (e.g., Ant, Hopper, and Walker2d), higher budgets even lead to lower returns. By visualizing the learned policies, we observe that agents tend to quickly expend their budget by operating at unsafe velocities early on, and then switch to safer behavior. This is reflected by costs closely matching the budget levels in most tasks (except Humanoid). We hypothesize that the combination of higher initial velocities and the subsequent transition to safe behavior introduces instabilities that can reduce overall return.

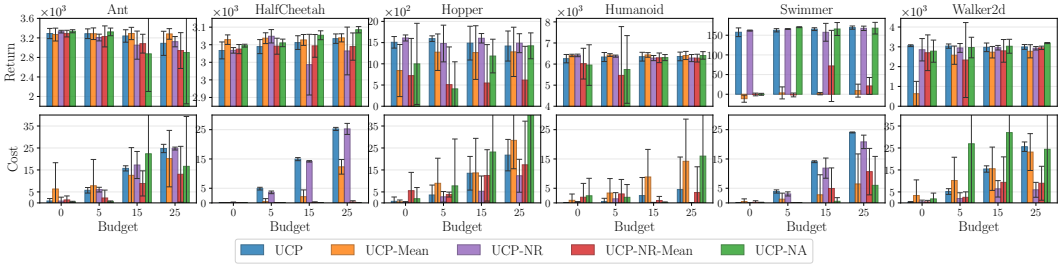


Figure 6: Ablation results on the *Safety Velocity* tasks for UCP, UCP-Mean, UCP-NR, UCP-NR-Mean, and UCP-NA

In contrast, *Safety Navigation* tasks exhibit a different pattern. Here, we observe a strong correlation between budget and return (except in Circle tasks). Agents incur cost while exploring the environment and interacting with costly regions or objects; higher budgets make the agent less sensitive to these penalties, allowing more direct paths to the goal and resulting in higher returns.

Comparing UCP and UCP-Mean with UCP-NR and UCP-NR-Mean, respectively, we find that randomizing the initial stock enables effective generalization across tasks and budget levels without retraining. There are, however, some nuanced differences. In *Safety Navigation* tasks, UCP matches or exceeds the return of UCP-NR across all environments and budget levels, although UCP-NR tends to be more conservative in terms of cost. Importantly, UCP still satisfies the budget constraint at the 95% confidence level, which makes it a preferable approach overall.

Examining UCP-Mean and UCP-NR-Mean across environments and budget levels, we observe that UCP-NR-Mean achieves average costs closer to the allocated budget, whereas UCP-Mean exceeds the budget in several cases. This suggests that stock randomization in UCP-Mean can negatively affect its performance in this setting.

Comparing all four main variants, UCP achieves the best balance between return and cost. For the remaining variants, we observe similar performance between UCP-NA and UCP-NR-Mean across environments and budget levels, indicating that state augmentation does not harm performance and can even provide improvements in some cases. In contrast, UCP-DS generally exhibits higher costs or lower returns compared to UCP across environments and budget levels, empirically supporting our discussion in Section 4.

Finally, the results across tasks and budget levels highlight the advantages of the UCMDP framework. Stock randomization enables a more complete understanding of the reward–cost relationship and allows practitioners to select budgets accordingly. It also enables learning policies for multiple budget levels without retraining, removes the need for manually designing modified reward signals (since each signal is learned with its own critic), and allows flexible application of different utility functions to the cost signals.

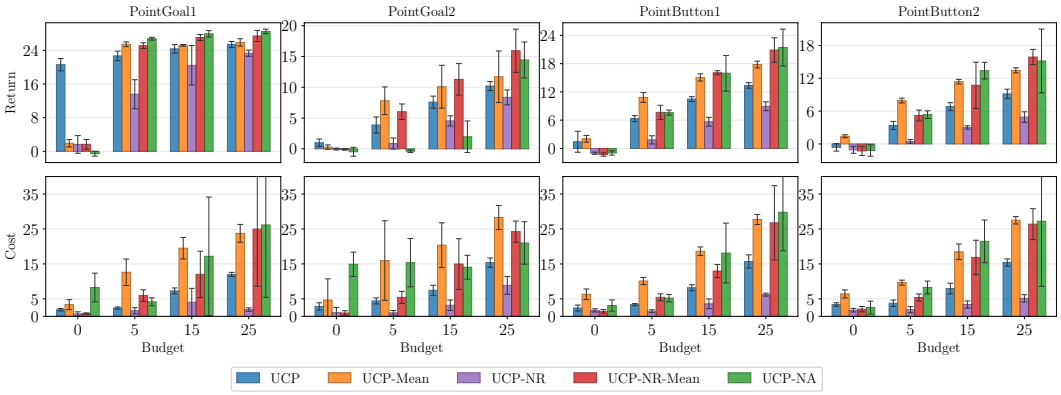


Figure 7: Ablation results on the *Safety Navigation* tasks for UCP, UCP-Mean, UCP-NR, UCP-NR-Mean, and UCP-NA.

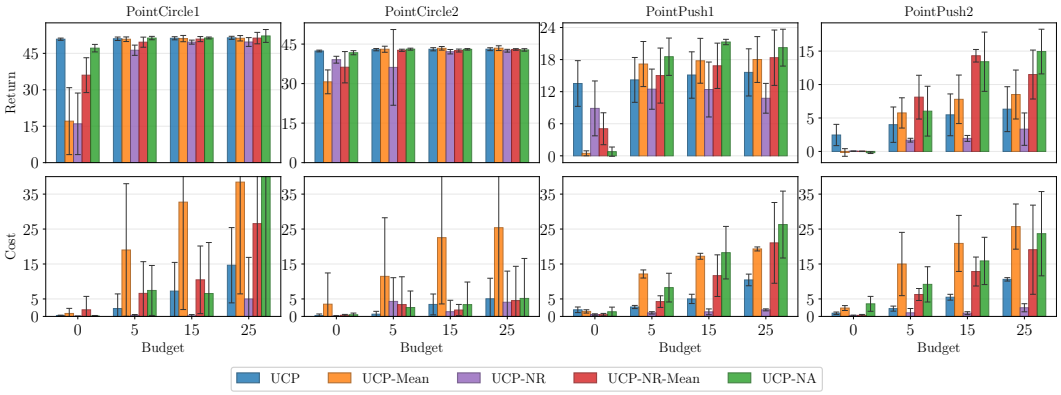


Figure 8: Ablation results on the *Safety Navigation* tasks for UCP, UCP-Mean, UCP-NR, UCP-NR-Mean, and UCP-NA.

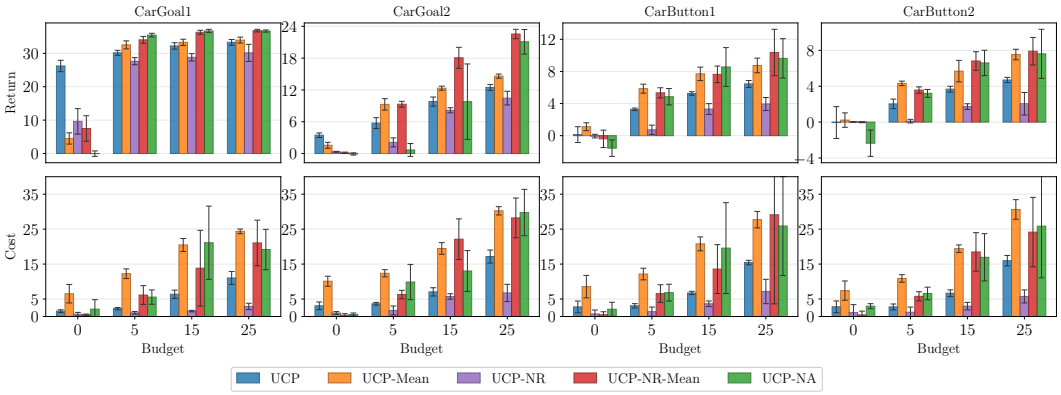


Figure 9: Ablation results on the *Safety Navigation* tasks for UCP, UCP-Mean, UCP-NR, UCP-NR-Mean, and UCP-NA.

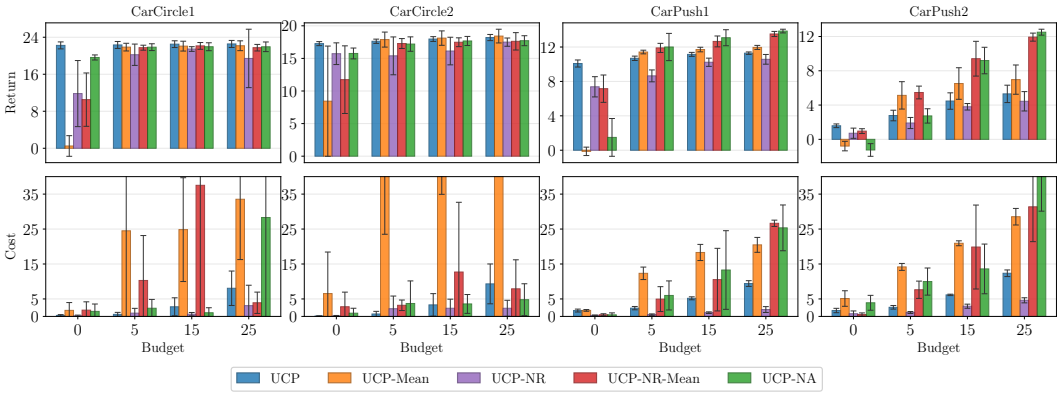


Figure 10: Ablation results on the *Safety Navigation* tasks for UCP, UCP-Mean, UCP-NR, UCP-NR-Mean, and UCP-NA.

E Sensitivity to the Lagrange Multiplier Function Approximation

As discussed in Section 4 and Appendix C, we use function approximation to maintain the estimates of the Lagrange multipliers λ_z, via a K-bin discretization over $[z_{\min}, z_{\max}]$. In this section, we conduct a brief study of the impact of the discretization granularity K on the performance of UCP, and we see that, as expected, return improves with the resolution up to the $K = 31$ that we have used in the main text. Note that a denser grid does not require additional environment interaction or network evaluation, but only additional scalar Lagrange multipliers, so its computational cost is negligible.

We evaluated UCP on the CarGoal1 task with different values of K (4, 13, 31), and the results are given in Table 5. Over the training range $[-30, 0]$, $K = 4$, 13, and 31 give stock spacings of 10, 2.5, and 1. Return improves at every budget, for example from 0.1 to 25.4 to 30.1 at $z_0 = -5$, and from 28.9 to 31.7 to 33.3 at $z_0 = -25$. A denser grid exposes the agent to more initial stocks and assigns distinct multiplier values to nearby budgets, which may improve both interpolation to evaluation stocks and overall learning. For every $z_0 < 0$, all three settings satisfy the benchmark's risk-neutral constraint, so the improvement does not sacrifice mean-budget satisfaction.

Table 5: UCP on CarGoal1 with different discretization levels K, evaluated at different initial stocks z_0. Each cell shows return (top) and cost (bottom, red), with 95% Student's t intervals across five seeds.

Discretization	$z_0 = 0$	$z_0 = -5$	$z_0 = -15$	$z_0 = -25$
$K = 4$	1.5 ± 1.7 0.4 ± 0.2	0.1 ± 0.8 0.5 ± 0.3	23.9 ± 2.4 4.2 ± 1.1	28.9 ± 1.5 7.1 ± 1.3
$K = 13$	17.9 ± 7.8 1.2 ± 0.2	25.4 ± 6.5 1.4 ± 0.4	30.3 ± 2.2 6.1 ± 0.6	31.7 ± 2.4 11.4 ± 0.9
$K = 31$	26.2 ± 1.7 1.5 ± 0.4	30.1 ± 0.8 2.2 ± 0.3	32.2 ± 1.0 6.4 ± 1.2	33.3 ± 0.9 11.0 ± 1.9

F Sensitivity to the Constraint Violation Tolerance ϵ

As discussed in Section 4, the constraint $\mathbb{E}[(z + G_0)_+] \leq 0$ may not be strictly feasible [Boyd and Vandenberghe, 2004, Section 5.2.3, p. 226]. This and the fact that we use imperfect return distribution estimators for the Lagrangian multiplier updates can make the optimization challenging. Therefore, we use a small constraint violation tolerance ϵ which mitigates both issues and has the practical benefit that λ_z decreases when the constraint is approximately satisfied, unlike the case where there is no tolerance.

We can see the practical impact of the tolerance in Table 6, where we report the performance on CarGoal1 of UCP with different values of ϵ. Setting $\epsilon = 0$ prevents the multiplier from decreasing, because $g \geq 0$, so accumulated early increases cannot be undone. This results in an overly conserva-

Table 6: UCP on CarGoal1 with different constraint violation tolerances ϵ, evaluated at different initial stocks z_0. Each cell shows return (top) and cost (bottom, red), with 95% Student's t intervals across five seeds.

Tolerance	$z_0 = 0$	$z_0 = -5$	$z_0 = -15$	$z_0 = -25$
$\epsilon = 0$	13.3 ± 4.4 0.8 ± 0.3	26.8 ± 1.3 1.1 ± 0.5	31.7 ± 0.8 4.7 ± 0.7	32.8 ± 0.5 10.0 ± 1.2
$\epsilon = 10^{-6}$	26.2 ± 1.7 1.5 ± 0.4	30.1 ± 0.8 2.2 ± 0.3	32.2 ± 1.0 6.4 ± 1.2	33.3 ± 0.9 11.0 ± 1.9
$\epsilon = 10^{-4}$	28.4 ± 1.5 1.8 ± 0.4	31.2 ± 1.3 2.5 ± 0.3	33.0 ± 0.8 7.1 ± 0.9	33.9 ± 0.6 11.5 ± 1.4

tive learned policy. At budget 0, $\epsilon = 0$ obtains return 13.3 at cost 0.8, compared with return 28.4 at cost 1.8 for $\epsilon = 10^{-4}$. The return and mean cost increase as ϵ and $-z_0$ increase (though ϵ remains relatively small), and at budgets 5, 15, and 25, all tested settings remain feasible in mean. A positive tolerance lets λ_z decrease whenever the estimated excess is below ϵ, so the multiplier can recover from early over-penalization. Accordingly, $\epsilon = 10^{-6}$ gives an intermediate result, while $\epsilon = 10^{-4}$ is the least conservative and obtains the highest return at every budget.

G Empirical Distributions

In this section, we present the empirical distributions of return and cost for all environments at a budget level of 25. The tables accompanying each figure report the probability that trajectories from each policy exceed the allocated budget, as well as the magnitude of the excess cost.

Across all *Safety Navigation* tasks (except Circle), we observe that the probability of violating the constraint under risk-neutral variants such as UCP-NA, UCP-NR-Mean, and UCP-Mean exceeds 30%. We also find that UCP-NR achieves the lowest probability of budget violations, albeit at a noticeable drop in terms of return. In contrast, UCP achieves a more favorable balance between return and constraint satisfaction.

Another notable observation is that, despite the lower violation probability of UCP-NR, the expected excess cost under UCP is often smaller. This suggests that the increased diversity in the training data in UCP leads to more controlled violations when they occur.

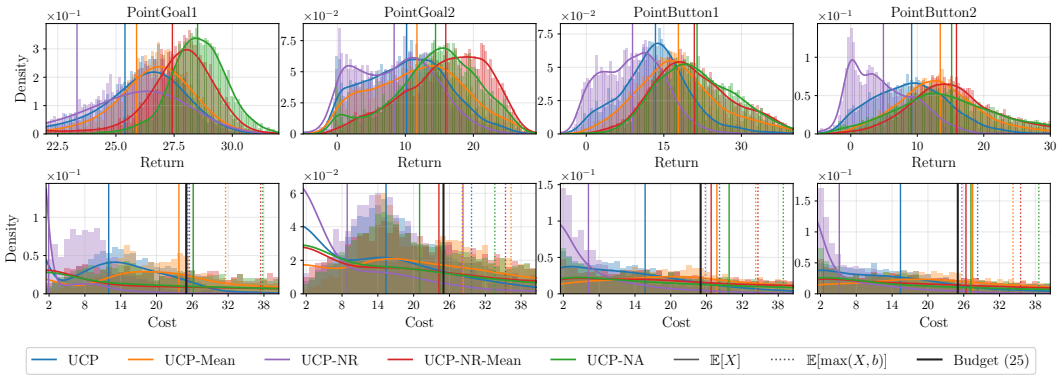


Figure 11: Empirical distributions of episodic return and cost at budget 25 on the *Safety Navigation* task with the *Point* agent.

Table 7: Evaluation statistics on *Safety Navigation* (*Point*) tasks with cost budget of 25. The third column corresponds to the estimated conditional expectation of the undiscounted cost given that it (strictly) exceeds the budget, and fourth to the estimated probability of the undiscounted cost being above the budget.

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	25.4 ± 0.1	12.0 ± 0.3	31.7	0.077
UCP-Mean	25.9 ± 0.1	23.7 ± 0.5	43.7	0.352
UCP-NR	23.3 ± 0.2	1.9 ± 0.2	53.0	0.009
UCP-NR-Mean	27.4 ± 0.1	24.9 ± 0.9	58.4	0.373
UCP-NA	28.5 ± 0.0	26.2 ± 0.9	58.3	0.385

(a) PointGoal1

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	10.2 ± 0.2	15.4 ± 0.6	49.5	0.192
UCP-Mean	11.7 ± 0.2	28.3 ± 0.7	50.7	0.441
UCP-NR	8.4 ± 0.2	8.9 ± 0.7	57.4	0.097
UCP-NR-Mean	15.9 ± 0.2	24.2 ± 0.8	53.3	0.367
UCP-NA	14.4 ± 0.2	21.0 ± 0.9	53.2	0.305

(b) PointGoal2

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	13.4 ± 0.2	15.7 ± 0.5	42.5	0.181
UCP-Mean	17.8 ± 0.2	27.7 ± 0.6	45.8	0.444
UCP-NR	8.9 ± 0.2	6.2 ± 0.4	41.5	0.048
UCP-NR-Mean	20.9 ± 0.2	26.8 ± 0.6	46.8	0.441
UCP-NA	21.4 ± 0.2	29.8 ± 0.9	57.3	0.429

(c) PointButton1

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	9.2 ± 0.2	15.4 ± 0.5	44.1	0.174
UCP-Mean	13.5 ± 0.2	27.5 ± 0.6	46.0	0.441
UCP-NR	4.9 ± 0.1	5.1 ± 0.3	41.8	0.041
UCP-NR-Mean	15.9 ± 0.2	26.4 ± 0.7	50.2	0.417
UCP-NA	15.2 ± 0.2	27.2 ± 1.0	61.6	0.371

(d) PointButton2

Table 8: Evaluation statistics on *Safety Navigation* (*Point*) tasks with cost budget of 25. The third column corresponds to the estimated conditional expectation of the undiscounted cost given that it (strictly) exceeds the budget, and fourth to the estimated probability of the undiscounted cost being above the budget.

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	51.4 ± 0.0	14.6 ± 0.3	28.4	0.164
UCP-Mean	51.2 ± 0.1	38.4 ± 1.4	77.0	0.415
UCP-NR	49.7 ± 0.1	5.0 ± 0.3	33.1	0.078
UCP-NR-Mean	51.3 ± 0.1	26.5 ± 1.4	91.3	0.273
UCP-NA	52.2 ± 0.1	54.9 ± 1.6	99.9	0.537

(a) PointCircle1

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	43.1 ± 0.0	5.0 ± 0.2	26.3	0.002
UCP-Mean	43.4 ± 0.1	25.4 ± 1.8	126.1	0.141
UCP-NR	42.5 ± 0.0	4.1 ± 0.2	30.2	0.067
UCP-NR-Mean	43.0 ± 0.0	4.5 ± 0.5	300.1	0.002
UCP-NA	42.8 ± 0.0	5.1 ± 0.3	35.1	0.079

(b) PointCircle2

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	15.6 ± 0.2	10.4 ± 0.4	35.6	0.108
UCP-Mean	18.0 ± 0.2	19.3 ± 0.6	44.0	0.293
UCP-NR	10.8 ± 0.1	1.9 ± 0.2	38.8	0.014
UCP-NR-Mean	18.4 ± 0.2	21.0 ± 0.7	52.2	0.330
UCP-NA	20.2 ± 0.1	26.3 ± 0.8	53.2	0.413

(c) PointPush1

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	6.3 ± 0.1	10.6 ± 0.5	43.5	0.129
UCP-Mean	8.5 ± 0.2	25.7 ± 0.9	51.8	0.372
UCP-NR	3.3 ± 0.1	2.5 ± 0.2	42.8	0.023
UCP-NR-Mean	11.5 ± 0.1	19.1 ± 0.8	53.2	0.283
UCP-NA	14.9 ± 0.1	23.7 ± 1.3	58.5	0.323

(d) PointPush2

Table 9: Evaluation statistics on *Safety Navigation* (*Car*) tasks with cost budget of 25. The third column corresponds to the estimated conditional expectation of the undiscounted cost given that it (strictly) exceeds the budget, and fourth to the estimated probability of the undiscounted cost being above the budget.

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	33.3 ± 0.2	11.0 ± 0.3	31.4	0.094
UCP-Mean	34.0 ± 0.1	24.4 ± 0.6	46.0	0.364
UCP-NR	30.1 ± 0.2	2.9 ± 0.2	33.3	0.007
UCP-NR-Mean	36.8 ± 0.1	21.1 ± 0.8	53.6	0.326
UCP-NA	36.7 ± 0.1	19.2 ± 0.6	48.1	0.292

(a) CarGoal1

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	12.5 ± 0.2	17.2 ± 0.6	45.8	0.205
UCP-Mean	14.6 ± 0.2	30.2 ± 0.8	50.0	0.465
UCP-NR	10.4 ± 0.2	6.7 ± 0.4	56.2	0.039
UCP-NR-Mean	22.5 ± 0.2	28.2 ± 0.7	50.8	0.459
UCP-NA	21.1 ± 0.2	29.7 ± 0.8	53.2	0.459

(b) CarGoal2

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	6.4 ± 0.1	15.4 ± 0.5	41.7	0.174
UCP-Mean	8.7 ± 0.1	27.7 ± 0.8	48.2	0.456
UCP-NR	3.9 ± 0.1	7.2 ± 0.7	71.9	0.024
UCP-NR-Mean	10.4 ± 0.2	29.1 ± 0.9	58.7	0.417
UCP-NA	9.6 ± 0.2	25.9 ± 1.2	67.1	0.329

(c) CarButton1

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	4.7 ± 0.1	16.0 ± 0.6	43.2	0.178
UCP-Mean	7.5 ± 0.1	30.6 ± 1.0	52.2	0.473
UCP-NR	2.0 ± 0.1	5.7 ± 0.6	59.4	0.036
UCP-NR-Mean	7.9 ± 0.1	24.1 ± 0.9	55.3	0.353
UCP-NA	7.6 ± 0.2	25.8 ± 1.2	67.9	0.327

(d) CarButton2

Table 10: Evaluation statistics on *Safety Navigation* (*Car*) tasks with cost budget of 25. The third column corresponds to the estimated conditional expectation of the undiscounted cost given that it (strictly) exceeds the budget, and fourth to the estimated probability of the undiscounted cost being above the budget.

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	22.6 ± 0.0	8.1 ± 0.3	34.2	0.075
UCP-Mean	22.2 ± 0.0	33.5 ± 1.5	92.9	0.303
UCP-NR	19.4 ± 0.2	3.1 ± 0.2	45.8	0.004
UCP-NR-Mean	21.7 ± 0.0	3.9 ± 0.6	111.6	0.031
UCP-NA	21.9 ± 0.0	28.3 ± 1.1	75.5	0.352

(a) CarCircle1

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	18.2 ± 0.0	9.3 ± 0.3	30.1	0.073
UCP-Mean	18.4 ± 0.0	83.9 ± 3.1	128.5	0.600
UCP-NR	17.5 ± 0.0	2.3 ± 0.1	35.0	0.003
UCP-NR-Mean	17.6 ± 0.0	7.9 ± 1.0	88.9	0.075
UCP-NA	17.7 ± 0.0	4.8 ± 0.3	35.1	0.063

(b) CarCircle2

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	11.3 ± 0.1	9.4 ± 0.4	35.9	0.130
UCP-Mean	12.0 ± 0.1	20.5 ± 0.7	46.9	0.347
UCP-NR	10.6 ± 0.1	2.0 ± 0.2	42.5	0.011
UCP-NR-Mean	13.5 ± 0.1	26.6 ± 0.8	53.0	0.435
UCP-NA	13.9 ± 0.1	25.3 ± 0.8	53.0	0.408

(c) CarPush1

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	5.3 ± 0.1	12.4 ± 0.5	40.7	0.144
UCP-Mean	7.0 ± 0.1	28.5 ± 1.1	54.0	0.424
UCP-NR	4.4 ± 0.1	4.6 ± 0.3	51.8	0.017
UCP-NR-Mean	11.9 ± 0.1	31.4 ± 1.2	60.4	0.437
UCP-NA	12.5 ± 0.1	43.9 ± 2.0	76.5	0.518

(d) CarPush2

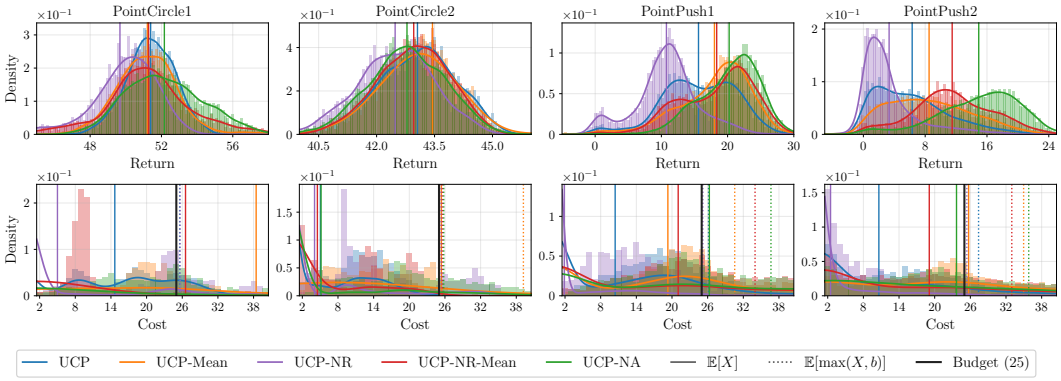


Figure 12: Empirical distributions of episodic return and cost at budget 25 on the *Safety Navigation* task with the *Point* agent.

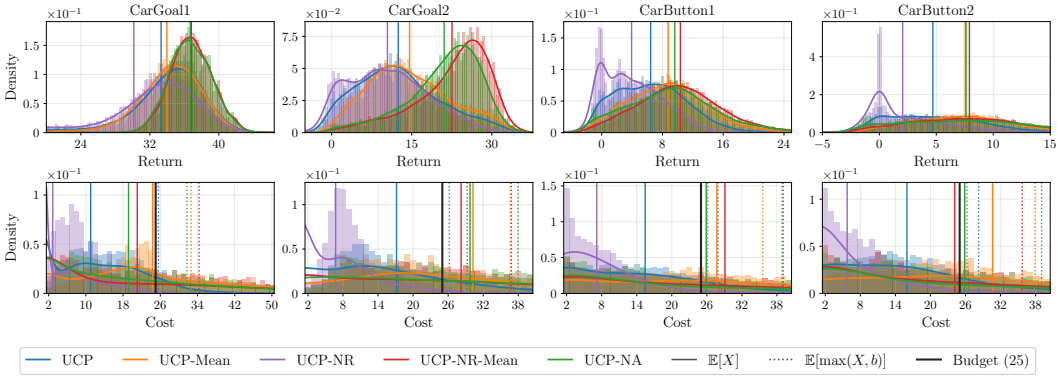


Figure 13: Empirical distributions of episodic return and cost at budget 25 on the *Safety Navigation* task with the *Car* agent.

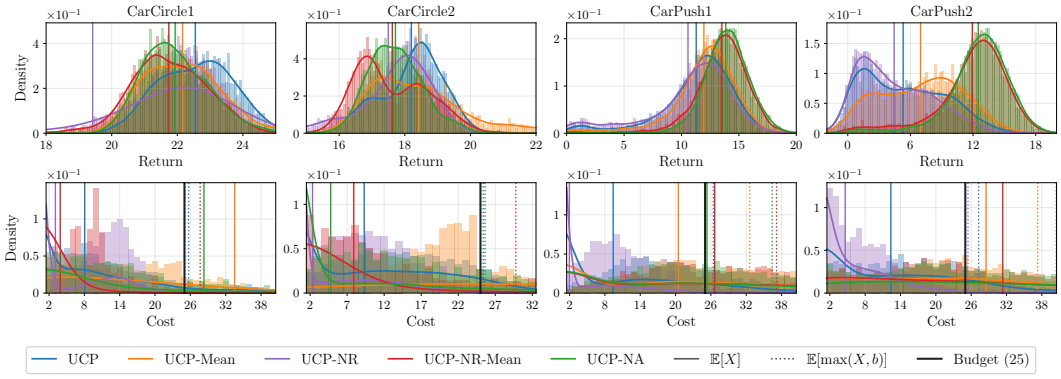


Figure 14: Empirical distributions of episodic return and cost at budget 25 on the *Safety Navigation* task with the *Car* agent.

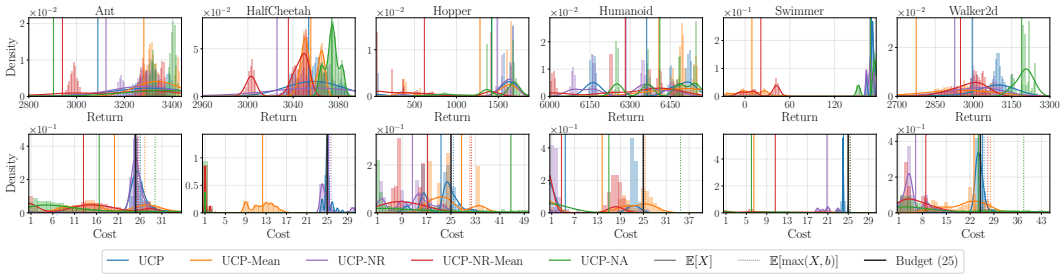


Figure 15: Empirical distributions of episodic return and cost at budget 25 on the *Safety Velocity* tasks.

Table 11: Evaluation statistics on *Safety Velocity* tasks with cost budget of 25. The third column corresponds to the estimated conditional expectation of the undiscounted cost given that it (strictly) exceeds the budget, and fourth to the estimated probability of the undiscounted cost being above the budget.

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	3088.7 ± 24.8	24.8 ± 0.2	27.6	0.395
UCP-Mean	3281.2 ± 13.1	20.2 ± 0.3	30.6	0.371
UCP-NR	3122.8 ± 18.5	24.7 ± 0.1	27.0	0.305
UCP-NR-Mean	2940.7 ± 18.1	13.0 ± 0.3	28.4	0.136
UCP-NA	2903.1 ± 28.5	16.6 ± 0.5	42.3	0.257

(a) Ant

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	3053.8 ± 3.0	25.3 ± 0.0	27.0	0.192
UCP-Mean	3055.7 ± 0.2	12.3 ± 0.1	–	0.000
UCP-NR	3025.8 ± 5.5	25.3 ± 0.1	29.5	0.170
UCP-NR-Mean	3035.8 ± 0.6	0.3 ± 0.0	–	0.000
UCP-NA	3074.2 ± 0.2	0.0 ± 0.0	–	0.000

(b) HalfCheetah

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	1419.1 ± 14.9	21.7 ± 0.2	28.1	0.258
UCP-Mean	1281.7 ± 14.6	28.5 ± 0.4	44.3	0.320
UCP-NR	1486.5 ± 10.2	12.4 ± 0.2	29.2	0.020
UCP-NR-Mean	616.9 ± 16.4	17.3 ± 0.8	89.1	0.104
UCP-NA	1423.9 ± 9.4	44.6 ± 1.6	96.5	0.439

(c) Hopper

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	6361.7 ± 4.6	4.5 ± 0.2	–	0.000
UCP-Mean	6407.5 ± 20.3	14.2 ± 0.3	27.3	0.217
UCP-NR	6284.1 ± 4.8	0.0 ± 0.0	26.0	0.000
UCP-NR-Mean	6281.7 ± 9.6	3.5 ± 0.2	–	0.000
UCP-NA	6410.7 ± 3.5	16.0 ± 0.9	63.9	0.250

(d) Humanoid

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	169.4 ± 0.1	24.1 ± 0.0	–	0.000
UCP-Mean	9.2 ± 0.6	6.5 ± 0.2	28.1	0.002
UCP-NR	167.5 ± 0.1	20.8 ± 0.0	–	0.000
UCP-NR-Mean	21.1 ± 0.6	10.7 ± 0.2	43.5	0.022
UCP-NA	167.9 ± 0.3	6.0 ± 0.3	74.4	0.007

(e) Swimmer

Method	Return	Cost	Conditional Expected Cost	Freq. of Cost above Budget
UCP	2994.9 ± 10.2	25.6 ± 0.1	29.8	0.229
UCP-Mean	2775.0 ± 15.7	23.1 ± 0.4	36.2	0.267
UCP-NR	2926.2 ± 11.1	6.0 ± 0.1	27.5	0.012
UCP-NR-Mean	2948.7 ± 6.8	8.9 ± 0.5	81.3	0.040
UCP-NA	3189.3 ± 3.4	24.4 ± 1.0	62.9	0.338

(f) Walker2d

H Tables

In this section, we present our results in table format for ease of reference and comparison.

In the Button and Push tasks with both Point and Car agents, the OmniSafe baselines perform significantly worse than our results. Comparing with the results reported in Ji et al. [2023, Table 5], we find that our outcomes, even after extensive hyperparameter tuning, are consistent with those reported. However, our UCP-NA baseline, which uses a risk-neutral objective with a distributional critic, shows that the performance of existing methods can be improved using standard techniques such as distributional critics and n-step returns. We report UCP-NA specifically to highlight how much of the performance gain comes from our proposed components versus these standard design choices.

Table 12: Comparison of UCP with OmniSafe baselines on Velocity and Navigation environments.

Environment	**UCP**	**PPOLag**	**PPOSaute**	**PPO-ET**	**SACLag**	**SACPID**	**TD3Lag**	**TD3PID**
Velocity Environments								
Ant	3289.6 ± 104.5 0.9 ± 1.1	3166.8 ± 88.6 27.9 ± 3.7	2711.2 ± 84.6 17.5 ± 0.4	1899.5 ± 111.6 13.3 ± 2.0	2215.5 ± 717.3 11.8 ± 12.7	2756.7 ± 182.1 3.7 ± 2.3	2335.4 ± 915.5 19.8 ± 11.5	1677.5 ± 1686.3 19.8 ± 8.8
HalfCheetah	3044.9 ± 15.6 14.9 ± 0.5	3024.1 ± 31.5 17.3 ± 1.3	2792.7 ± 31.7 17.1 ± 0.2	2091.8 ± 21.5 9.9 ± 0.4	2967.9 ± 886.0 146.0 ± 89.6	2447.0 ± 258.0 3.0 ± 7.5	1977.4 ± 353.3 19.6 ± 13.4	1664.6 ± 294.2 6.3 ± 4.1
Hopper	1593.3 ± 65.9 3.6 ± 4.5	1394.4 ± 607.2 23.5 ± 15.4	1656.4 ± 29.0 17.9 ± 0.4	1442.7 ± 50.2 5.3 ± 1.1	970.1 ± 148.0 63.5 ± 72.7	994.1 ± 121.9 63.5 ± 62.0	1093.8 ± 106.5 53.1 ± 35.7	1107.0 ± 97.5 39.3 ± 43.4
Humanoid	6361.7 ± 205.5 4.5 ± 11.1	6478.3 ± 100.4 25.7 ± 5.2	5986.0 ± 42.5 20.4 ± 0.8	5547.7 ± 86.3 6.0 ± 0.4	6137.2 ± 172.4 8.9 ± 15.1	6068.6 ± 250.1 0.6 ± 1.2	5925.7 ± 202.2 20.4 ± 13.2	6021.6 ± 154.9 24.2 ± 2.6
Swimmer	169.4 ± 4.1 24.1 ± 0.2	82.2 ± 28.3 27.6 ± 2.9	111.6 ± 72.4 12.5 ± 6.7	22.0 ± 1.3 14.8 ± 1.3	4.2 ± 4.4 16.3 ± 14.0	5.1 ± 3.1 4.3 ± 6.6	31.6 ± 16.4 20.7 ± 4.4	43.8 ± 10.3 30.2 ± 13.6
Walker2d	3056.5 ± 45.2 0.3 ± 0.3	3015.3 ± 181.8 23.7 ± 16.8	2064.9 ± 1381.5 18.5 ± 0.8	1349.7 ± 1116.6 14.6 ± 6.0	960.2 ± 271.1 23.3 ± 13.7	2146.1 ± 553.5 23.0 ± 22.6	2538.3 ± 362.4 32.3 ± 21.2	2290.1 ± 752.9 19.9 ± 13.2
Navigation Environments								
PointGoal1	25.4 ± 0.7 12.0 ± 0.6	16.0 ± 1.6 21.1 ± 2.4	6.2 ± 3.3 21.1 ± 3.3	15.7 ± 0.5 18.4 ± 0.9	16.5 ± 5.1 24.2 ± 10.8	20.2 ± 4.1 29.6 ± 8.7	20.9 ± 2.3 29.8 ± 6.5	13.9 ± 7.9 20.8 ± 3.7
PointGoal2	10.2 ± 0.7 15.4 ± 1.3	1.3 ± 0.5 24.4 ± 4.4	0.6 ± 1.0 28.1 ± 3.6	6.9 ± 0.2 25.2 ± 0.2	1.1 ± 2.3 40.8 ± 14.3	0.7 ± 1.7 37.3 ± 23.3	2.5 ± 1.4 38.5 ± 16.4	-0.4 ± 0.3 51.9 ± 28.8
PointButton1	13.4 ± 0.6 15.7 ± 1.9	4.5 ± 1.7 23.2 ± 3.5	0.8 ± 1.3 26.4 ± 10.8	10.6 ± 0.3 24.4 ± 0.1	10.3 ± 2.7 23.9 ± 3.1	9.1 ± 4.2 27.1 ± 6.9	6.2 ± 2.3 26.6 ± 1.9	7.0 ± 3.5 23.5 ± 4.3
PointButton2	9.2 ± 0.9 15.4 ± 1.0	2.0 ± 1.7 24.7 ± 3.4	-0.2 ± 0.3 29.4 ± 12.6	7.8 ± 0.6 25.4 ± 0.1	7.9 ± 3.1 31.8 ± 10.5	4.8 ± 0.7 23.2 ± 4.6	8.0 ± 1.6 29.4 ± 8.9	2.7 ± 3.0 21.9 ± 11.6
PointCircle1	51.4 ± 0.6 14.6 ± 10.8	47.2 ± 1.4 22.0 ± 4.6	42.4 ± 1.8 14.3 ± 6.7	32.3 ± 8.2 13.4 ± 9.1	42.5 ± 1.4 25.2 ± 15.3	40.0 ± 4.1 25.5 ± 2.5	40.7 ± 2.9 27.3 ± 10.9	40.5 ± 2.9 22.4 ± 15.7
PointCircle2	43.1 ± 0.5 5.0 ± 5.9	41.8 ± 0.3 26.7 ± 14.9	38.6 ± 1.2 19.7 ± 10.2	35.5 ± 2.1 6.5 ± 4.4	42.3 ± 0.3 24.2 ± 1.6	42.2 ± 0.4 29.1 ± 19.6	41.8 ± 1.6 30.8 ± 12.6	41.6 ± 1.2 34.1 ± 18.5
PointPush1	15.6 ± 4.4 10.4 ± 1.7	1.3 ± 1.9 24.3 ± 2.1	0.3 ± 0.2 9.9 ± 7.2	1.1 ± 1.2 8.4 ± 1.9	0.2 ± 0.2 12.2 ± 10.0	-0.3 ± 0.5 28.3 ± 14.8	-0.3 ± 0.3 31.9 ± 36.9	-0.3 ± 0.8 38.3 ± 65.4
PointPush2	6.3 ± 3.4 10.6 ± 0.5	0.8 ± 0.1 26.1 ± 6.1	0.1 ± 0.5 13.5 ± 3.7	0.9 ± 0.1 12.6 ± 2.2	-0.0 ± 0.5 25.7 ± 13.8	-0.2 ± 0.3 27.7 ± 26.3	-0.1 ± 0.3 20.8 ± 18.8	-0.5 ± 0.6 20.7 ± 14.5
CarGoal1	33.3 ± 0.9 11.0 ± 1.9	14.8 ± 3.8 22.6 ± 2.8	9.6 ± 2.9 22.6 ± 6.0	18.7 ± 1.5 18.4 ± 0.8	15.4 ± 8.2 31.7 ± 14.0	21.0 ± 16.4 35.6 ± 3.1	16.6 ± 13.1 27.6 ± 8.6	22.3 ± 9.8 29.9 ± 14.1
CarGoal2	12.5 ± 0.6 17.2 ± 1.9	1.5 ± 0.4 28.9 ± 3.2	0.6 ± 0.4 20.4 ± 6.8	6.6 ± 0.2 25.3 ± 0.0	3.3 ± 2.4 29.2 ± 11.4	3.2 ± 2.3 28.8 ± 8.7	2.9 ± 4.6 30.4 ± 20.2	2.8 ± 3.7 31.8 ± 20.6
CarButton1	6.4 ± 0.4 15.4 ± 0.6	0.1 ± 0.3 41.3 ± 12.5	-0.3 ± 0.8 31.6 ± 8.5	4.7 ± 0.6 25.9 ± 0.1	1.8 ± 1.2 28.9 ± 13.2	1.3 ± 1.1 20.0 ± 4.1	1.4 ± 0.6 31.6 ± 8.0	0.7 ± 0.9 28.2 ± 5.9
CarButton2	4.7 ± 0.3 16.0 ± 1.5	0.4 ± 0.5 52.7 ± 15.2	0.0 ± 0.3 40.0 ± 10.6	4.2 ± 0.2 25.9 ± 0.1	1.6 ± 0.6 32.4 ± 8.0	0.6 ± 0.6 20.4 ± 12.2	1.4 ± 0.6 33.6 ± 4.5	0.5 ± 1.1 24.6 ± 13.0
CarCircle1	22.6 ± 0.8 8.1 ± 4.9	18.4 ± 0.1 22.8 ± 5.4	13.5 ± 1.5 13.2 ± 4.8	14.4 ± 1.2 8.6 ± 3.9	18.8 ± 2.0 30.1 ± 14.4	17.7 ± 5.2 86.7 ± 158.6	18.7 ± 1.5 29.1 ± 3.5	18.1 ± 3.7 21.8 ± 12.0
CarCircle2	18.2 ± 0.5 9.3 ± 5.7	16.2 ± 0.3 38.2 ± 14.4	13.0 ± 0.6 16.8 ± 5.0	11.9 ± 0.4 12.1 ± 2.3	15.2 ± 4.1 26.6 ± 7.9	16.1 ± 1.8 25.0 ± 3.0	17.5 ± 0.9 58.0 ± 42.6	16.9 ± 1.0 29.0 ± 10.2
CarPush1	11.3 ± 0.1 9.4 ± 0.8	0.4 ± 0.2 24.5 ± 7.1	0.4 ± 0.2 16.1 ± 6.2	0.5 ± 0.1 6.3 ± 0.7	1.0 ± 0.2 20.0 ± 17.3	1.0 ± 0.4 16.4 ± 2.4	0.8 ± 0.3 18.2 ± 16.4	0.9 ± 0.1 22.3 ± 7.4
CarPush2	5.3 ± 1.0 12.4 ± 0.9	0.5 ± 0.1 29.1 ± 5.3	0.3 ± 0.1 26.8 ± 4.6	0.7 ± 0.1 15.8 ± 0.8	0.6 ± 0.5 33.6 ± 8.0	0.2 ± 0.3 21.1 ± 11.6	0.4 ± 0.4 45.9 ± 15.9	0.3 ± 0.2 24.8 ± 7.4

Table 13: Results of UCP, evaluated at different initial stocks z_0.

Environment	$z_0 = 0$	$z_0 = -5$	$z_0 = -15$	$z_0 = -25$
Velocity Environments				
Ant	3289.6 ± 104.5 0.9 ± 1.1	3285.4 ± 99.3 5.7 ± 1.3	3239.3 ± 127.4 15.7 ± 1.2	3088.7 ± 247.1 24.8 ± 1.9
HalfCheetah	3027.1 ± 19.3 0.0 ± 0.0	3035.7 ± 16.0 4.9 ± 0.5	3044.9 ± 15.6 14.9 ± 0.5	3053.8 ± 10.8 25.3 ± 0.5
Hopper	1506.8 ± 136.3 0.7 ± 2.0	1593.3 ± 65.9 3.6 ± 4.5	1487.4 ± 398.0 13.5 ± 7.7	1419.1 ± 354.4 21.7 ± 7.2
Humanoid	6256.0 ± 198.1 0.0 ± 0.0	6334.0 ± 214.3 0.4 ± 1.1	6349.6 ± 208.1 2.5 ± 6.2	6361.7 ± 205.5 4.5 ± 11.1
Swimmer	157.5 ± 11.1 0.0 ± 0.0	162.6 ± 4.1 3.9 ± 0.5	165.4 ± 3.6 14.1 ± 0.3	169.4 ± 4.1 24.1 ± 0.2
Walker2d	3056.5 ± 45.2 0.3 ± 0.3	3036.6 ± 107.1 5.2 ± 1.3	2974.9 ± 217.3 15.4 ± 1.4	2994.9 ± 120.9 25.6 ± 2.2
Navigation Environments				
PointGoal1	20.6 ± 1.5 1.9 ± 0.4	22.7 ± 1.1 2.4 ± 0.3	24.4 ± 1.0 7.3 ± 0.9	25.4 ± 0.7 12.0 ± 0.6
PointGoal2	1.0 ± 0.6 2.8 ± 1.0	3.9 ± 1.3 4.4 ± 0.9	7.6 ± 1.0 7.5 ± 1.4	10.2 ± 0.7 15.4 ± 1.3
PointButton1	1.4 ± 2.2 2.4 ± 0.8	6.3 ± 0.6 3.3 ± 0.3	10.5 ± 0.5 8.2 ± 0.9	13.4 ± 0.6 15.7 ± 1.9
PointButton2	-0.6 ± 0.7 3.4 ± 0.5	3.4 ± 0.7 3.7 ± 0.9	6.8 ± 0.7 7.9 ± 1.5	9.2 ± 0.9 15.4 ± 1.0
PointCircle1	50.8 ± 0.5 0.2 ± 0.1	51.0 ± 0.7 2.3 ± 4.2	51.2 ± 0.6 7.2 ± 8.2	51.4 ± 0.6 14.6 ± 10.8
PointCircle2	42.4 ± 0.3 0.3 ± 0.4	42.9 ± 0.4 0.6 ± 0.8	43.0 ± 0.6 3.4 ± 3.0	43.1 ± 0.5 5.0 ± 5.9
PointPush1	13.5 ± 4.3 1.9 ± 0.8	14.2 ± 4.2 2.7 ± 0.5	15.1 ± 4.3 5.0 ± 1.3	15.6 ± 4.4 10.4 ± 1.7
PointPush2	2.5 ± 1.6 0.9 ± 0.3	4.0 ± 2.6 2.2 ± 0.7	5.5 ± 3.1 5.5 ± 0.8	6.3 ± 3.4 10.6 ± 0.5
CarGoal1	26.2 ± 1.7 1.5 ± 0.4	30.1 ± 0.8 2.2 ± 0.3	32.2 ± 1.0 6.4 ± 1.2	33.3 ± 0.9 11.0 ± 1.9
CarGoal2	3.5 ± 0.4 3.1 ± 1.1	5.7 ± 1.0 3.6 ± 0.4	9.8 ± 0.9 7.0 ± 1.2	12.5 ± 0.6 17.2 ± 1.9
CarButton1	0.1 ± 1.0 2.7 ± 1.7	3.3 ± 0.2 3.1 ± 0.6	5.3 ± 0.2 6.7 ± 0.5	6.4 ± 0.4 15.4 ± 0.6
CarButton2	-0.1 ± 1.8 2.8 ± 1.7	2.0 ± 0.5 2.7 ± 0.8	3.7 ± 0.3 6.7 ± 1.0	4.7 ± 0.3 16.0 ± 1.5
CarCircle1	22.2 ± 0.8 0.3 ± 0.2	22.3 ± 0.7 0.6 ± 0.6	22.5 ± 0.7 2.8 ± 2.6	22.6 ± 0.8 8.1 ± 4.9
CarCircle2	17.3 ± 0.3 0.1 ± 0.0	17.6 ± 0.3 0.7 ± 0.8	18.0 ± 0.4 3.3 ± 3.2	18.2 ± 0.5 9.3 ± 5.7
CarPush1	10.1 ± 0.4 1.7 ± 0.4	10.7 ± 0.3 2.4 ± 0.5	11.1 ± 0.2 5.2 ± 0.4	11.3 ± 0.1 9.4 ± 0.8
CarPush2	1.6 ± 0.2 1.7 ± 0.6	2.8 ± 0.6 2.6 ± 0.5	4.5 ± 1.0 6.2 ± 0.2	5.3 ± 1.0 12.4 ± 0.9

Table 14: Results of UCP-NR (No Randomization), trained separately for each z_0.

Environment	$z_0 = 0$	$z_0 = -5$	$z_0 = -15$	$z_0 = -25$
Velocity Environments				
Ant	3328.1 ± 27.3 0.8 ± 1.8	3207.5 ± 69.1 6.0 ± 1.1	3051.7 ± 288.1 17.3 ± 6.2	3122.8 ± 105.8 24.7 ± 0.7
HalfCheetah	3027.2 ± 6.4 0.1 ± 0.2	3059.2 ± 15.2 3.6 ± 0.5	2994.6 ± 69.1 14.2 ± 0.2	3025.8 ± 54.4 25.3 ± 1.9
Hopper	1609.2 ± 71.1 0.2 ± 0.4	1478.3 ± 432.7 2.8 ± 2.4	1602.2 ± 111.7 5.3 ± 7.0	1486.5 ± 223.1 12.4 ± 7.5
Humanoid	6410.4 ± 60.7 0.2 ± 0.3	6376.5 ± 49.2 1.4 ± 2.1	6284.2 ± 122.0 0.1 ± 0.1	6284.1 ± 169.0 0.0 ± 0.1
Swimmer	161.8 ± 1.3 0.0 ± 0.0	165.8 ± 1.1 3.1 ± 0.7	157.4 ± 22.7 11.9 ± 3.4	167.5 ± 5.6 20.8 ± 2.3
Walker2d	2852.3 ± 564.8 0.5 ± 0.8	2943.3 ± 223.1 2.0 ± 2.6	2955.2 ± 114.3 6.4 ± 3.7	2926.2 ± 87.0 6.0 ± 3.4
Navigation Environments				
PointGoal1	1.6 ± 2.1 0.7 ± 0.6	13.6 ± 3.5 1.6 ± 0.9	20.5 ± 4.7 4.0 ± 4.0	23.3 ± 0.7 1.9 ± 0.5
PointGoal2	-0.0 ± 0.2 1.0 ± 1.5	0.9 ± 0.9 1.0 ± 0.7	4.5 ± 0.8 3.2 ± 1.5	8.4 ± 1.2 8.9 ± 2.5
PointButton1	-1.0 ± 0.2 1.7 ± 0.5	1.8 ± 0.9 1.5 ± 0.4	5.7 ± 0.9 3.6 ± 1.4	8.9 ± 0.9 6.2 ± 0.4
PointButton2	-1.1 ± 0.6 1.8 ± 0.5	0.5 ± 0.3 1.9 ± 0.9	3.0 ± 0.3 3.4 ± 1.0	4.9 ± 1.0 5.1 ± 1.0
PointCircle1	16.0 ± 12.7 0.1 ± 0.1	46.3 ± 2.1 0.3 ± 0.2	49.6 ± 0.8 0.2 ± 0.4	49.7 ± 1.8 5.0 ± 11.9
PointCircle2	39.1 ± 1.3 0.1 ± 0.1	36.2 ± 14.4 4.3 ± 6.8	42.1 ± 0.8 1.3 ± 3.3	42.5 ± 0.5 4.1 ± 8.9
PointPush1	8.9 ± 5.1 0.5 ± 0.2	12.5 ± 3.8 1.0 ± 0.3	12.4 ± 5.1 1.3 ± 0.8	10.8 ± 2.8 1.9 ± 0.3
PointPush2	0.0 ± 0.0 0.2 ± 0.2	1.7 ± 0.3 1.0 ± 1.2	1.9 ± 0.4 0.9 ± 0.4	3.3 ± 2.4 2.5 ± 1.2
CarGoal1	9.6 ± 3.8 0.7 ± 0.5	27.6 ± 1.1 1.1 ± 0.4	28.8 ± 1.1 1.5 ± 0.2	30.1 ± 2.6 2.9 ± 0.9
CarGoal2	0.3 ± 0.1 1.0 ± 0.4	2.1 ± 0.8 1.7 ± 1.3	8.2 ± 0.5 5.7 ± 0.8	10.4 ± 1.3 6.7 ± 2.5
CarButton1	-0.1 ± 0.2 0.7 ± 1.1	0.7 ± 0.6 1.4 ± 1.3	3.3 ± 0.7 3.6 ± 0.8	3.9 ± 0.8 7.2 ± 3.5
CarButton2	0.0 ± 0.0 1.1 ± 2.3	0.1 ± 0.2 1.2 ± 1.5	1.7 ± 0.3 3.0 ± 1.1	2.0 ± 1.3 5.7 ± 1.9
CarCircle1	11.8 ± 7.1 0.2 ± 0.2	20.2 ± 2.3 0.9 ± 1.4	21.5 ± 0.5 0.6 ± 0.5	19.4 ± 6.3 3.1 ± 5.8
CarCircle2	15.7 ± 1.7 0.1 ± 0.2	15.4 ± 2.9 2.2 ± 3.6	16.1 ± 2.1 2.3 ± 2.6	17.5 ± 0.6 2.3 ± 2.3
CarPush1	7.4 ± 1.2 0.3 ± 0.1	8.6 ± 0.7 0.5 ± 0.2	10.2 ± 0.5 1.1 ± 0.2	10.6 ± 0.6 2.0 ± 0.8
CarPush2	0.7 ± 0.6 0.8 ± 0.8	1.9 ± 0.6 1.1 ± 0.3	3.8 ± 0.4 2.9 ± 0.6	4.4 ± 1.1 4.6 ± 0.7

Table 15: Results of UCP-Mean, evaluated at different initial stocks z_0.

Environment	$z_0 = 0$	$z_0 = -5$	$z_0 = -15$	$z_0 = -25$
Velocity Environments				
Ant	3268.3 ± 131.9 6.3 ± 12.0	3285.3 ± 119.1 7.8 ± 12.0	3290.4 ± 125.8 12.5 ± 12.6	3281.2 ± 108.6 20.2 ± 12.8
HalfCheetah	3051.5 ± 10.9 0.0 ± 0.0	3054.9 ± 12.3 0.5 ± 1.0	3050.9 ± 12.4 2.1 ± 2.3	3055.7 ± 9.2 12.3 ± 2.5
Hopper	841.7 ± 614.2 0.5 ± 0.8	1273.7 ± 430.2 8.9 ± 11.4	1264.8 ± 632.6 13.7 ± 15.7	1281.7 ± 581.0 28.5 ± 12.9
Humanoid	6409.0 ± 57.3 0.9 ± 2.2	6427.6 ± 64.8 3.4 ± 4.9	6433.5 ± 109.0 8.9 ± 9.4	6407.5 ± 193.2 14.2 ± 14.4
Swimmer	-11.2 ± 9.1 0.5 ± 0.9	3.5 ± 15.3 1.3 ± 2.1	1.6 ± 3.0 2.8 ± 6.8	9.2 ± 16.5 6.5 ± 10.2
Walker2d	632.6 ± 622.6 3.4 ± 7.1	2578.3 ± 460.6 10.2 ± 10.6	2728.0 ± 290.5 15.6 ± 9.9	2775.0 ± 333.0 23.1 ± 8.4
Navigation Environments				
PointGoal1	1.9 ± 0.9 3.3 ± 1.4	25.4 ± 0.6 12.6 ± 3.8	25.2 ± 0.2 19.5 ± 3.1	25.9 ± 0.9 23.7 ± 2.6
PointGoal2	0.3 ± 0.4 4.7 ± 6.1	7.8 ± 2.2 16.0 ± 11.4	10.1 ± 3.5 20.4 ± 6.4	11.7 ± 4.2 28.3 ± 3.4
PointButton1	2.1 ± 0.7 6.3 ± 1.5	10.8 ± 1.0 10.1 ± 1.0	15.1 ± 0.8 18.6 ± 1.2	17.8 ± 0.7 27.7 ± 1.4
PointButton2	1.4 ± 0.3 6.4 ± 1.2	7.9 ± 0.4 9.6 ± 0.7	11.4 ± 0.4 18.5 ± 2.2	13.5 ± 0.5 27.5 ± 1.1
PointCircle1	17.1 ± 13.8 0.8 ± 1.5	50.8 ± 1.0 19.0 ± 19.0	51.1 ± 1.3 32.7 ± 30.7	51.2 ± 1.1 38.4 ± 32.0
PointCircle2	30.7 ± 4.5 3.5 ± 8.9	43.0 ± 1.2 11.5 ± 16.8	43.4 ± 0.7 22.5 ± 19.0	43.4 ± 0.9 25.4 ± 32.0
PointPush1	0.5 ± 0.5 1.4 ± 0.5	17.2 ± 4.2 12.1 ± 1.2	17.8 ± 4.2 17.2 ± 0.8	18.0 ± 4.3 19.3 ± 0.6
PointPush2	-0.2 ± 0.6 2.4 ± 0.7	5.8 ± 2.3 15.0 ± 9.1	7.8 ± 3.6 20.9 ± 8.0	8.5 ± 3.7 25.7 ± 6.5
CarGoal1	4.5 ± 1.7 6.5 ± 2.6	32.5 ± 1.2 12.2 ± 1.4	33.3 ± 0.9 20.5 ± 1.9	34.0 ± 0.9 24.4 ± 0.7
CarGoal2	1.6 ± 0.6 10.1 ± 1.4	9.3 ± 1.1 12.4 ± 1.0	12.3 ± 0.4 19.5 ± 1.7	14.6 ± 0.4 30.2 ± 1.2
CarButton1	1.1 ± 0.5 8.6 ± 3.2	5.9 ± 0.6 12.2 ± 1.7	7.7 ± 0.8 20.8 ± 2.0	8.7 ± 0.9 27.7 ± 2.4
CarButton2	0.2 ± 0.8 7.4 ± 2.8	4.3 ± 0.2 10.9 ± 1.1	5.7 ± 1.2 19.4 ± 1.1	7.5 ± 0.6 30.6 ± 2.8
CarCircle1	0.5 ± 2.2 1.7 ± 2.2	21.9 ± 0.8 24.5 ± 24.9	22.1 ± 1.1 24.8 ± 14.9	22.2 ± 1.1 33.5 ± 17.3
CarCircle2	8.4 ± 8.5 6.5 ± 11.9	17.9 ± 1.2 51.0 ± 27.5	18.1 ± 1.1 64.8 ± 29.8	18.4 ± 1.1 83.9 ± 42.2
CarPush1	-0.1 ± 0.5 1.7 ± 0.2	11.4 ± 0.2 12.3 ± 1.8	11.7 ± 0.3 18.3 ± 2.3	12.0 ± 0.2 20.5 ± 2.1
CarPush2	-0.8 ± 0.6 5.1 ± 2.2	5.1 ± 1.6 14.2 ± 1.0	6.5 ± 1.8 20.9 ± 0.7	7.0 ± 1.7 28.5 ± 2.4

Table 16: Results of UCP-NR-Mean (No Randomization), trained separately for each z_0.

Environment	$z_0 = 0$	$z_0 = -5$	$z_0 = -15$	$z_0 = -25$
Velocity Environments				
Ant	3288.1 ± 68.0 1.4 ± 1.8	3230.5 ± 215.1 2.2 ± 3.6	3081.7 ± 192.6 8.9 ± 5.7	2940.7 ± 366.9 13.0 ± 12.7
HalfCheetah	3030.1 ± 10.4 0.0 ± 0.0	3036.6 ± 18.2 0.0 ± 0.0	3037.4 ± 25.9 0.1 ± 0.2	3035.8 ± 31.0 0.3 ± 0.4
Hopper	721.2 ± 877.3 5.6 ± 8.3	508.8 ± 889.1 3.8 ± 1.1	547.5 ± 905.4 12.5 ± 11.7	616.9 ± 795.7 17.3 ± 19.9
Humanoid	6019.4 ± 728.9 1.9 ± 4.7	5463.1 ± 2315.1 3.1 ± 4.9	6296.7 ± 228.7 0.8 ± 1.4	6281.7 ± 184.6 3.5 ± 8.8
Swimmer	-0.5 ± 3.4 0.2 ± 0.5	-1.7 ± 4.9 0.0 ± 0.0	72.0 ± 90.4 4.9 ± 7.0	21.1 ± 21.5 10.7 ± 7.9
Walker2d	2700.7 ± 901.3 0.4 ± 0.7	2333.1 ± 1892.1 2.5 ± 2.5	2795.8 ± 572.8 9.3 ± 11.6	2948.7 ± 84.9 8.9 ± 7.7
Navigation Environments				
PointGoal1	1.7 ± 1.2 0.8 ± 0.2	25.1 ± 0.7 5.9 ± 1.7	27.1 ± 0.7 12.0 ± 6.6	27.4 ± 1.3 24.9 ± 16.4
PointGoal2	-0.1 ± 0.1 0.9 ± 0.6	6.0 ± 1.3 5.4 ± 1.7	11.3 ± 2.6 15.0 ± 7.2	15.9 ± 3.5 24.2 ± 3.0
PointButton1	-1.3 ± 0.3 1.4 ± 0.5	7.7 ± 1.5 5.4 ± 1.0	16.1 ± 0.4 13.0 ± 1.9	20.9 ± 2.6 26.8 ± 10.6
PointButton2	-1.3 ± 0.8 2.1 ± 0.8	5.2 ± 1.0 5.4 ± 1.0	10.7 ± 4.2 16.8 ± 4.9	15.9 ± 1.4 26.4 ± 4.4
PointCircle1	36.0 ± 7.2 1.9 ± 3.8	49.6 ± 2.1 6.6 ± 9.1	50.9 ± 1.0 10.5 ± 9.7	51.3 ± 2.4 26.5 ± 41.7
PointCircle2	36.2 ± 5.9 0.3 ± 0.2	42.6 ± 0.4 3.4 ± 8.0	42.5 ± 0.6 1.8 ± 1.6	43.0 ± 0.3 4.5 ± 9.8
PointPush1	5.1 ± 3.0 0.5 ± 0.4	15.0 ± 5.1 4.2 ± 1.7	16.8 ± 4.2 11.7 ± 6.0	18.4 ± 5.2 21.0 ± 11.6
PointPush2	0.0 ± 0.0 0.3 ± 0.3	8.1 ± 3.3 6.3 ± 1.7	14.3 ± 0.9 12.8 ± 4.1	11.5 ± 3.7 19.1 ± 12.8
CarGoal1	7.5 ± 3.8 0.5 ± 0.2	34.0 ± 1.0 6.1 ± 2.7	36.3 ± 0.6 13.8 ± 10.8	36.8 ± 0.4 21.1 ± 6.5
CarGoal2	0.1 ± 0.1 0.4 ± 0.4	9.3 ± 0.5 6.3 ± 1.2	18.1 ± 2.0 22.1 ± 5.8	22.5 ± 0.9 28.2 ± 5.7
CarButton1	-0.4 ± 1.1 0.5 ± 0.8	5.3 ± 0.6 6.6 ± 2.5	7.6 ± 1.0 13.6 ± 7.0	10.4 ± 2.9 29.1 ± 25.5
CarButton2	-0.0 ± 0.1 0.5 ± 1.1	3.6 ± 0.4 5.8 ± 1.3	6.8 ± 1.0 18.5 ± 5.5	7.9 ± 1.5 24.1 ± 9.9
CarCircle1	10.5 ± 5.8 1.8 ± 2.4	21.7 ± 0.5 10.3 ± 12.8	22.1 ± 0.7 37.5 ± 53.9	21.7 ± 0.7 3.9 ± 3.1
CarCircle2	11.7 ± 5.2 2.8 ± 4.2	17.3 ± 0.8 3.2 ± 1.5	17.5 ± 0.7 12.7 ± 20.0	17.6 ± 1.3 7.9 ± 8.3
CarPush1	7.2 ± 1.6 0.5 ± 0.3	11.9 ± 0.5 5.0 ± 3.6	12.7 ± 0.6 10.5 ± 9.0	13.5 ± 0.3 26.6 ± 0.9
CarPush2	1.0 ± 0.3 0.5 ± 0.5	5.5 ± 0.7 7.6 ± 2.5	9.4 ± 2.0 19.9 ± 12.0	11.9 ± 0.5 31.4 ± 10.0

Table 17: Results of UCP-NA (No Stock-Augmentation), trained separately for each z_0.

Environment	$z_0 = 0$	$z_0 = -5$	$z_0 = -15$	$z_0 = -25$
Velocity Environments				
Ant	3336.2 ± 31.7 0.4 ± 0.3	3322.9 ± 78.4 0.6 ± 0.3	2873.2 ± 770.7 22.4 ± 30.9	2903.1 ± 1053.9 16.6 ± 22.7
HalfCheetah	3037.9 ± 3.2 0.0 ± 0.0	3044.2 ± 8.6 0.0 ± 0.0	3061.6 ± 10.1 0.1 ± 0.0	3074.2 ± 7.2 0.0 ± 0.0
Hopper	998.6 ± 955.9 1.9 ± 5.1	409.4 ± 638.4 7.8 ± 21.3	1183.1 ± 396.8 23.2 ± 23.5	1423.9 ± 305.2 44.6 ± 57.0
Humanoid	5952.3 ± 962.3 2.4 ± 6.0	5745.1 ± 1604.9 1.9 ± 4.4	6307.2 ± 144.4 0.1 ± 0.2	6410.7 ± 178.7 16.0 ± 44.0
Swimmer	-0.2 ± 2.5 0.1 ± 0.2	170.4 ± 1.1 0.0 ± 0.0	165.8 ± 16.4 0.6 ± 1.2	167.9 ± 14.8 6.0 ± 9.9
Walker2d	2780.7 ± 559.1 1.8 ± 2.6	2964.3 ± 518.7 26.9 ± 39.5	3028.4 ± 351.5 32.0 ± 68.6	3189.3 ± 26.6 24.4 ± 37.9
Navigation Environments				
PointGoal1	-0.6 ± 0.6 8.2 ± 4.1	26.8 ± 0.3 4.2 ± 1.1	27.9 ± 0.8 17.2 ± 17.0	28.5 ± 0.5 26.2 ± 20.7
PointGoal2	-0.5 ± 0.7 14.9 ± 3.5	-0.4 ± 0.2 15.4 ± 6.9	2.0 ± 2.6 14.0 ± 3.4	14.4 ± 2.9 21.0 ± 6.1
PointButton1	-0.9 ± 0.5 3.1 ± 1.6	7.6 ± 0.5 5.2 ± 1.1	16.0 ± 3.8 18.1 ± 8.6	21.4 ± 3.9 29.8 ± 11.0
PointButton2	-1.2 ± 1.0 2.5 ± 1.8	5.4 ± 0.7 8.3 ± 1.8	13.4 ± 1.5 21.5 ± 6.1	15.2 ± 5.8 27.2 ± 18.6
PointCircle1	47.1 ± 1.5 0.1 ± 0.2	51.3 ± 0.7 7.4 ± 7.1	51.3 ± 0.4 6.5 ± 14.6	52.2 ± 2.6 54.9 ± 79.9
PointCircle2	41.7 ± 0.8 0.5 ± 0.5	43.1 ± 0.4 2.5 ± 4.7	43.0 ± 0.3 3.4 ± 6.5	42.8 ± 0.5 5.1 ± 11.4
PointPush1	0.8 ± 0.9 1.3 ± 1.3	18.5 ± 3.5 8.2 ± 4.1	21.3 ± 0.5 18.2 ± 7.5	20.2 ± 3.4 26.3 ± 9.6
PointPush2	-0.2 ± 0.1 3.6 ± 2.1	6.0 ± 3.7 9.2 ± 5.0	13.4 ± 4.4 15.9 ± 6.8	14.9 ± 3.3 23.7 ± 12.1
CarGoal1	-0.1 ± 0.8 2.1 ± 2.7	35.5 ± 0.5 5.6 ± 2.1	36.7 ± 0.5 21.1 ± 10.5	36.7 ± 0.3 19.2 ± 5.8
CarGoal2	-0.1 ± 0.2 0.6 ± 0.4	0.7 ± 1.2 9.9 ± 5.0	9.8 ± 7.1 13.0 ± 5.8	21.1 ± 2.3 29.7 ± 6.6
CarButton1	-1.6 ± 1.0 2.1 ± 2.0	4.9 ± 1.0 6.8 ± 2.4	8.6 ± 2.4 19.6 ± 13.0	9.6 ± 2.5 25.9 ± 14.2
CarButton2	-2.4 ± 1.5 3.0 ± 0.7	3.2 ± 0.4 6.6 ± 1.8	6.6 ± 1.4 16.9 ± 6.8	7.6 ± 2.7 25.8 ± 14.8
CarCircle1	19.6 ± 0.5 1.4 ± 2.1	21.9 ± 0.7 2.3 ± 2.5	21.9 ± 0.9 1.0 ± 1.4	21.9 ± 1.0 28.3 ± 53.5
CarCircle2	15.8 ± 0.8 0.9 ± 1.4	17.2 ± 1.1 3.7 ± 6.5	17.7 ± 0.7 3.6 ± 2.7	17.7 ± 0.8 4.8 ± 4.6
CarPush1	1.5 ± 2.2 0.5 ± 0.6	12.0 ± 1.6 6.0 ± 4.2	13.1 ± 0.9 13.3 ± 11.3	13.9 ± 0.2 25.3 ± 6.6
CarPush2	-1.2 ± 0.7 3.9 ± 2.1	2.7 ± 0.8 10.0 ± 3.9	9.2 ± 1.5 13.6 ± 7.1	12.5 ± 0.3 43.9 ± 13.8

Table 18: Results of UCP-DS (Discounted Stock), evaluated at different initial stocks z_0.

Environment	$z_0 = 0$	$z_0 = -5$	$z_0 = -15$	$z_0 = -25$
Velocity Environments				
Ant	2437.5 ± 395.5 0.5 ± 0.4	2853.6 ± 392.4 35.9 ± 29.5	2780.9 ± 411.3 48.6 ± 29.1	2282.9 ± 395.3 57.1 ± 24.5
HalfCheetah	2705.4 ± 400.8 2.7 ± 5.7	3027.6 ± 53.0 92.2 ± 53.3	3043.6 ± 66.7 110.8 ± 62.4	3052.3 ± 72.6 129.7 ± 63.6
Hopper	1051.6 ± 546.3 0.0 ± 0.0	1155.6 ± 708.2 261.5 ± 247.7	1123.6 ± 493.8 295.2 ± 171.2	1016.5 ± 319.5 253.2 ± 92.4
Humanoid	6237.7 ± 45.2 0.0 ± 0.1	6281.4 ± 76.0 0.0 ± 0.1	6290.9 ± 78.5 0.0 ± 0.0	6293.2 ± 82.3 0.0 ± 0.0
Swimmer	12.8 ± 15.5 0.0 ± 0.1	44.8 ± 3.7 71.3 ± 49.0	49.7 ± 3.8 88.7 ± 56.8	45.7 ± 4.6 107.7 ± 68.6
Walker2d	2227.0 ± 460.5 0.4 ± 0.8	2527.0 ± 780.9 249.4 ± 194.7	2252.9 ± 538.6 290.2 ± 210.9	1974.6 ± 707.3 273.5 ± 210.6
Navigation Environments				
PointGoal1	15.2 ± 1.7 2.4 ± 0.7	20.9 ± 1.4 2.9 ± 0.9	24.9 ± 0.8 8.1 ± 1.6	26.0 ± 0.6 16.8 ± 1.9
PointGoal2	1.1 ± 1.4 2.9 ± 1.8	2.9 ± 1.7 3.0 ± 1.8	5.9 ± 3.1 7.4 ± 2.1	10.0 ± 3.3 20.9 ± 3.3
PointButton1	-0.5 ± 0.4 2.1 ± 0.9	5.6 ± 1.4 3.3 ± 0.9	10.3 ± 1.3 8.7 ± 1.0	14.4 ± 0.9 22.4 ± 1.9
PointButton2	-1.4 ± 0.6 3.5 ± 0.9	3.1 ± 0.7 3.3 ± 1.1	6.8 ± 0.7 8.6 ± 1.4	10.5 ± 0.3 21.6 ± 1.5
PointCircle1	45.5 ± 3.6 0.9 ± 1.5	47.5 ± 3.0 13.1 ± 3.9	50.0 ± 1.5 52.1 ± 6.2	51.2 ± 1.2 83.6 ± 4.9
PointCircle2	41.0 ± 0.2 1.9 ± 4.7	40.8 ± 0.9 11.3 ± 3.8	42.7 ± 0.6 49.2 ± 7.7	43.3 ± 1.1 88.5 ± 10.6
PointPush1	10.4 ± 4.4 1.6 ± 0.6	12.5 ± 4.4 2.2 ± 0.5	14.4 ± 4.5 6.4 ± 0.7	15.3 ± 4.7 13.6 ± 1.2
PointPush2	1.2 ± 0.4 0.9 ± 0.2	3.9 ± 1.5 2.7 ± 0.3	6.3 ± 2.4 7.3 ± 1.4	7.2 ± 2.6 14.3 ± 2.6
CarGoal1	1.0 ± 4.2 0.7 ± 0.4	20.0 ± 6.3 1.7 ± 1.2	30.5 ± 1.9 8.7 ± 1.2	32.4 ± 1.3 18.9 ± 1.5
CarGoal2	-1.5 ± 0.8 3.7 ± 1.4	0.0 ± 0.8 2.5 ± 1.8	6.3 ± 3.9 8.3 ± 1.3	10.1 ± 4.8 24.4 ± 1.9
CarButton1	-13.4 ± 14.4 2.9 ± 2.1	-8.4 ± 14.6 3.1 ± 2.3	-6.7 ± 18.9 5.4 ± 3.0	-3.6 ± 14.7 10.1 ± 10.3
CarButton2	-8.3 ± 11.0 3.3 ± 3.9	-1.6 ± 1.6 2.3 ± 2.7	1.6 ± 2.0 4.8 ± 3.9	2.9 ± 2.7 16.0 ± 5.0
CarCircle1	18.2 ± 1.7 4.6 ± 4.9	19.2 ± 1.2 18.6 ± 9.3	20.9 ± 1.0 57.0 ± 2.3	21.8 ± 1.0 90.4 ± 5.0
CarCircle2	16.0 ± 0.9 1.6 ± 3.0	16.1 ± 1.1 8.3 ± 6.1	17.3 ± 0.9 57.2 ± 5.8	17.7 ± 0.9 88.2 ± 7.3
CarPush1	6.9 ± 4.2 1.1 ± 1.0	9.0 ± 2.1 1.9 ± 1.1	10.5 ± 1.0 5.5 ± 1.3	10.9 ± 0.8 10.4 ± 0.7
CarPush2	-0.6 ± 4.8 3.7 ± 5.4	-4.3 ± 19.9 2.9 ± 2.5	-0.9 ± 17.9 10.5 ± 9.2	2.4 ± 11.4 17.5 ± 4.0

I Effect of Lower Budgets on a Baseline

In Figure 3, all 16 Safety Navigation results use the same budget of 25 for UCP and the baselines, yet UCP achieves higher return at lower cost. For Safety Velocity in Figure 2, however, we selected UCP's best feasible budget after training, although its performance remains strong at budget 25. This raises the question of whether training with a lower budget could help other methods too. We therefore trained PPOSaute at budget 5, and we compare the results with the original budget-25 results in Tables 19 and 20. The lower budget reduces mean cost in all 22 environments but generally yields the same or lower return; the only clear return improvements are in Swimmer and Walker2d. Moreover, budget-5 PPOSaute satisfies its nominal budget in only five of the six Velocity tasks and in none of the 16 Navigation tasks. Simply training a baseline with a smaller nominal budget therefore does not reproduce UCP's gains.

Table 19: PPOSaute trained with cost budgets $b = 5$ and $b = 25$, evaluated on Velocity environments.

Environment	PPOSaute ($b = 5$)	PPOSaute ($b = 25$)
Ant	2704.4 ± 45.2 3.9 ± 0.2	2711.2 ± 84.6 17.5 ± 0.4
HalfCheetah	2743.1 ± 56.0 3.7 ± 0.1	2792.7 ± 31.7 17.1 ± 0.2
Hopper	1697.0 ± 22.0 3.3 ± 0.1	1656.4 ± 29.0 17.9 ± 0.4
Humanoid	5896.3 ± 84.0 2.7 ± 0.1	5986.0 ± 42.5 20.4 ± 0.8
Swimmer	141.1 ± 9.9 5.8 ± 3.0	111.6 ± 72.4 12.5 ± 6.7
Walker2d	3023.3 ± 88.7 3.6 ± 0.1	2064.9 ± 1381.5 18.5 ± 0.8

Table 20: PPOSaute trained with cost budgets $b = 5$ and $b = 25$, evaluated on Navigation environments.

Environment	PPOSaute ($b = 5$)	PPOSaute ($b = 25$)
PointGoal1	3.7 ± 1.4 9.6 ± 2.1	6.2 ± 3.3 21.1 ± 3.3
PointGoal2	0.2 ± 0.7 17.4 ± 2.9	0.6 ± 1.0 28.1 ± 3.6
PointButton1	-1.2 ± 0.5 10.9 ± 1.4	0.8 ± 1.3 26.4 ± 10.8
PointButton2	-1.7 ± 0.7 14.4 ± 3.2	-0.2 ± 0.3 29.4 ± 12.6
PointCircle1	41.5 ± 2.1 6.1 ± 3.4	42.4 ± 1.8 14.3 ± 6.7
PointCircle2	38.9 ± 0.5 9.6 ± 3.7	38.6 ± 1.2 19.7 ± 10.2
PointPush1	0.5 ± 0.1 7.4 ± 0.8	0.3 ± 0.2 9.9 ± 7.2
PointPush2	0.3 ± 0.1 10.2 ± 2.7	0.1 ± 0.5 13.5 ± 3.7
CarGoal1	3.7 ± 1.6 11.2 ± 5.4	9.6 ± 2.9 22.6 ± 6.0
CarGoal2	0.1 ± 0.2 10.1 ± 3.6	0.6 ± 0.4 20.4 ± 6.8
CarButton1	0.1 ± 0.2 9.3 ± 3.4	-0.3 ± 0.8 31.6 ± 8.5
CarButton2	-0.1 ± 0.3 18.1 ± 6.5	0.0 ± 0.3 40.0 ± 10.6
CarCircle1	13.4 ± 1.1 8.8 ± 5.8	13.5 ± 1.5 13.2 ± 4.8
CarCircle2	12.7 ± 0.9 10.3 ± 2.9	13.0 ± 0.6 16.8 ± 5.0
CarPush1	0.2 ± 0.4 7.6 ± 2.7	0.4 ± 0.2 16.1 ± 6.2
CarPush2	0.2 ± 0.1 16.4 ± 6.0	0.3 ± 0.1 26.8 ± 4.6

J Multiple Constraints

The main results presented in Section 5 consider the one-cost setting from the Safety Gymnasium, where multiple cost signals are aggregated into one; however, the UCMDP framework supports vector-valued costs, and UCP is algorithmically compatible with such a case. In the CarButton2, CarPush2, and CarGoal2 tasks, the cost (as defined for the Safety Gymnasium benchmark) is a combination of multiple components. Therefore, it is natural for us to consider evaluating UCP with multiple constraints in these tasks by constraining the cost components separately.

In this section, we present a study of UCP under multiple constraints, a variant we call MC-UCP (multi-constraint UCP). We trained MC-UCP on CarButton2, CarPush2, and CarGoal2 for three million steps each, matching the experimental design in Section 5. We used one nonnegative multiplier per constraint and one distributional cost critic per stock dimension for vector-valued stock. Every cost dimension uses the same range and discretization as scalar-cost UCP ($K = 31$ equally spaced values in $[-30, 0]$). The initial stock of each cost dimension is sampled independently, so no component receives a privileged range or additional training. At evaluation, an initial-stock vector specifies the limits independently. For example, $(-25, 0)$ relaxes only the first cost limit, keeping the second at zero. We use five seeds and 1000 evaluation episodes per seed and initial-stock vector.

We highlight three interesting findings from this section:

- MC-UCP is competitive with or better than UCP (which uses a single constraint on the aggregate cost) in the original benchmark evaluation, where its total cost must not exceed 25.
- MC-UCP controls cost components separately (when the task allows).
- MC-UCP does not spend unusable budget (when the task allows).

The results for CarButton2 are presented in Table 21. The three additional cost dimensions of the stock ($(z_{0,2}, z_{0,3}, z_{0,4})$) correspond to the costs from buttons, gremlins and hazards, in this order. The total cost (which is not directly constrained) is the scalar cost as measured in the benchmark version of the task. We see that MC-UCP attains higher return while still satisfying the benchmark limit of

Table 21: MC-UCP on CarButton2, evaluated at different initial cost stocks. Intervals are 95% Student's t intervals across five seeds.

$(z_{0,2}, z_{0,3}, z_{0,4})$	Return	Total cost	Buttons	Gremlins	Hazards
$(0, 0, 0)$	-0.23 ± 1.71	5.07 ± 1.23	1.80 ± 0.68	1.73 ± 0.27	1.63 ± 0.63
$(-15, 0, 0)$	2.17 ± 0.86	8.32 ± 1.22	4.89 ± 1.00	2.10 ± 0.11	1.57 ± 0.39
$(0, -15, 0)$	2.56 ± 0.94	7.38 ± 0.77	1.38 ± 0.52	4.13 ± 0.34	1.96 ± 1.20
$(0, 0, -15)$	1.67 ± 1.00	5.11 ± 1.12	0.87 ± 0.10	1.48 ± 0.16	2.79 ± 1.10
$(-25, 0, 0)$	2.25 ± 0.52	9.38 ± 1.25	5.55 ± 0.77	2.18 ± 0.10	1.94 ± 0.71
$(0, -25, 0)$	2.76 ± 0.96	10.51 ± 0.72	1.43 ± 0.47	7.08 ± 0.80	2.06 ± 1.22
$(0, 0, -25)$	1.57 ± 0.96	6.78 ± 1.28	0.94 ± 0.17	1.65 ± 0.31	4.24 ± 1.17
$(-5, -5, -5)$	3.10 ± 0.78	5.65 ± 0.59	1.59 ± 0.23	2.41 ± 0.27	1.71 ± 0.74
$(-10, -10, -10)$	4.52 ± 0.53	10.06 ± 0.96	3.60 ± 0.83	3.83 ± 0.19	2.86 ± 0.36
$(-15, -15, -15)$	5.64 ± 0.39	16.86 ± 1.21	6.27 ± 1.24	5.81 ± 0.50	5.34 ± 0.43
$(-25, -25, -25)$	7.34 ± 0.47	35.93 ± 3.08	13.56 ± 1.29	12.06 ± 0.98	11.33 ± 1.17

25 on the cost. MC-UCP with $z_{0,(2,3,4)} = (-15, -15, -15)$ attains return 5.64 with a total cost of 16.86, whereas UCP has a return of 4.7 and a total cost of 16 (see Table 12). CarButton2 allows for behaviors that incur different types of costs (contact with more or fewer buttons, gremlins or hazards), and MC-UCP, accordingly, controls the cost components separately. Relative to an all-zero initial stock, relaxing one coordinate to -25 primarily raises the matching cost, with buttons going from 1.80 to 5.55, gremlins from 1.73 to 7.08, and hazards from 1.63 to 4.24, while the other components remain comparatively low.

The results for CarGoal2 are presented in Table 22. The three additional cost dimensions of the stock ($(z_{0,2}, z_{0,3}, z_{0,4})$) correspond to the costs from hazards, contact with vases and the velocity of vases, in this order. The total cost (which is not directly constrained) is the scalar cost as measured in the benchmark version of the task. MC-UCP outperforms UCP at lower total cost with the same number of training steps. MC-UCP with $z_{0,(2,3,4)} = (-15, -15, -15)$ attains return 14.15 with a total cost of 12.45, whereas UCP has a return of 12.5 and a total cost of 17.2 (see Table 12). CarGoal2 also

Table 22: MC-UCP on CarGoal2, evaluated at different initial cost stocks. Intervals are 95% Student's t intervals across five seeds.

$(z_{0,2}, z_{0,3}, z_{0,4})$	Return	Total cost	Hazards	Vases Contact	Vases Velocity
$(0, 0, 0)$	-0.33 ± 0.57	4.30 ± 2.18	2.66 ± 1.74	0.34 ± 0.16	1.67 ± 0.47
$(-15, 0, 0)$	1.78 ± 1.37	3.94 ± 1.22	2.92 ± 0.87	0.19 ± 0.12	1.02 ± 0.43
$(0, -15, 0)$	1.49 ± 1.10	3.67 ± 2.65	1.96 ± 1.23	0.36 ± 0.42	1.73 ± 1.51
$(0, 0, -15)$	4.18 ± 1.85	4.15 ± 0.66	1.44 ± 0.47	0.45 ± 0.11	2.71 ± 0.23
$(-25, 0, 0)$	2.36 ± 1.30	7.40 ± 0.99	5.89 ± 0.82	0.29 ± 0.13	1.52 ± 0.23
$(0, -25, 0)$	2.17 ± 1.65	3.38 ± 1.27	1.38 ± 0.39	0.38 ± 0.22	1.99 ± 0.89
$(0, 0, -25)$	6.58 ± 2.05	7.53 ± 0.88	2.36 ± 0.69	0.87 ± 0.17	5.16 ± 0.28
$(-15, 0, -15)$	9.55 ± 1.44	7.77 ± 1.61	5.23 ± 0.83	0.38 ± 0.11	2.54 ± 0.86
$(-25, 0, -25)$	13.30 ± 1.60	19.25 ± 2.01	12.57 ± 0.60	0.98 ± 0.39	6.76 ± 1.67
$(-5, -5, -5)$	5.24 ± 1.16	2.82 ± 0.32	1.42 ± 0.43	0.22 ± 0.03	1.41 ± 0.13
$(-10, -10, -10)$	10.55 ± 0.32	6.26 ± 1.34	3.41 ± 0.49	0.40 ± 0.16	2.84 ± 0.94
$(-15, -15, -15)$	14.15 ± 0.42	12.45 ± 1.69	6.55 ± 0.58	0.82 ± 0.21	5.92 ± 1.41
$(-25, -25, -25)$	16.36 ± 0.68	27.85 ± 2.39	13.72 ± 0.83	2.19 ± 0.46	14.25 ± 2.45

separates hazard and vase-velocity behavior. Relaxing only hazards raises hazard cost from 2.66 to 5.89, and relaxing only vase velocity raises that cost from 1.67 to 5.16. In CarGoal2, vase contacts are relatively rare at lower constraint budgets, so the policy does not spend the budget when only that constraint is relaxed. With $z_{0,(2,3,4)} = (0, -25, 0)$, vase-contact cost barely changes (from 0.34 to 0.38), while return increases from -0.33 to 2.17. By contrast, jointly relaxing hazards and vase velocity produces a large return improvement (consider $z_{0,(2,3,4)} = (-25, 0, -25)$, with return 13.30, versus 2.36 and 6.58 when relaxing either coordinate alone).

The results for CarPush2 are presented in Table 23. The two additional cost dimensions of the stock ($(z_{0,2}, z_{0,3})$) correspond to the costs from hazards and contact with pillars, in this order. The total cost (which is not directly constrained) is the scalar cost as measured in the benchmark version of the task. In this task, MC-UCP with $z_{0,(2,3)} = (-15, -15)$ attains a similar return at lower total cost,

Table 23: MC-UCP on CarPush2, evaluated at different initial cost stocks. Intervals are 95% Student's t intervals across five seeds.

$(z_{0,2}, z_{0,3})$	Return	Total cost	Hazards	Pillars
$(0, 0)$	0.95 ± 0.43	2.81 ± 0.59	1.59 ± 0.50	1.22 ± 0.14
$(-15, 0)$	3.47 ± 0.91	5.99 ± 0.29	4.56 ± 0.58	1.43 ± 0.31
$(0, -15)$	2.44 ± 0.81	4.69 ± 0.49	1.16 ± 0.17	3.54 ± 0.38
$(-25, 0)$	3.72 ± 0.77	8.89 ± 1.21	7.86 ± 1.00	1.03 ± 0.22
$(0, -25)$	2.64 ± 0.70	6.63 ± 1.09	1.40 ± 0.64	5.23 ± 0.47
$(-5, -5)$	3.24 ± 1.12	3.66 ± 0.30	1.92 ± 0.11	1.74 ± 0.22
$(-10, -10)$	4.61 ± 1.17	6.72 ± 1.23	3.26 ± 0.20	3.47 ± 1.07
$(-15, -15)$	5.16 ± 1.24	9.42 ± 0.77	5.31 ± 0.66	4.11 ± 0.11
$(-25, -25)$	5.55 ± 1.03	17.87 ± 1.48	10.69 ± 0.93	7.19 ± 0.80

with return 5.16 and a total cost of 9.42, whereas UCP has a return of 5.3 and a total cost of 12.4 (see Table 12). Similar to CarButton2, CarPush2 also allows for behaviors that incur different types of costs, and MC-UCP, accordingly, can control them separately. For example, $z_{0,(2,3)} = (-25, 0)$ raises hazard cost to 7.86 while pillar cost remains 1.03, whereas $z_{0,(2,3)} = (0, -25)$ raises pillar cost to 5.23 while hazard cost remains 1.40.

These asymmetric and joint-budget evaluations show more than aggregate cost control as one trained policy can reveal and independently tune distinct modes of unsafe behavior. This also makes the trade-off between cost and reward in each task clearly visible, which allows for better budget selection.

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: Our framework is discussed in Section 4 and our experiments and ablation studies are presented in Section 5.

   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 discuss our limitations in Section 6.

   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: [N/A]

   Justification: The paper does not include any theoretical results.

   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: We have highlighted significant components of the method in Section 4 and provided implementation details in Appendix C. We have used the OmniSafe [Ji et al., 2024] library for our experiments and baselines. We have not changed the tasks or task parameters. We have not changed the baselines, except for some hyperparameters, where they led to improved baseline performance, and we have provided details in Section C.2. We have provided tables with benchmark performance for our main method and all our ablations, so that reproduced results can be adequately compared.

   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 code is available at `https://github.com/MehrdadMoghimi/UCP`.

   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: We have presented the methodology and experimental setup in Sections 4 and 5 to enough detail that the reader will be able to appreciate the experimental results, though we assume some basic knowledge of the Safety Gymnasium benchmark [Ji et al., 2023] tasks. We have provided implementation details in Appendix C. The baselines are established methods, and have been described in detail by Ji et al. [2024].

   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: We have reported results with 95% confidence intervals (Student's t) over five independent runs for each experiment, and we believe the numbers we have presented to be representative of what will be observed in independent runs.

   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: We discuss this in Section C.3.

   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 adheres to the NeurIPS Code of Ethics, as it involves a framework and experimental results in Safe RL, which do not pose ethical risks or misuse concerns.

   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: [N/A]

    Justification: This is foundational algorithmic research. This work introduces an RL methodology evaluated on a standard simulated benchmark without ties to specific applications or deployments, so we do not expect direct societal impact.

    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: The 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 environments used for our evaluation and the code used in our benchmarks are properly credited and cited.

    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: [Yes]

    Justification: We release the code for UCP, including its multi-constraint extension, with documentation and an Apache 2.0 license at `https://github.com/MehrdadMoghimi/UCP`.

    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: The 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: The 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.

    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.