Towards a Practical Understanding of Lagrangian Methods in Safe Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Safe reinforcement learning addresses constrained optimization problems where maximizing performance must be balanced against safety constraints, and Lagrangian methods are a widely used approach for this purpose. However, the effectiveness of Lagrangian methods depends crucially on the choice of the Lagrange multiplier λ, which governs the trade-off between return and cost. A common approach is to update the multiplier automatically during training. Although this approach is standard in practice, there remains limited evidence on the variance in practical performance introduced by the choice of λ, nor on how the over- or undershooting of the cost limit, frequently exhibited by automated multiplier updates, affects the return. Therefore, we study, specifically for on-policy Lagrangian methods, (i) the *practical variance* exhibited by λ for a range of widely studied safety tasks, and show that Lagrange multiplier update methods are sensitive to the choice of cost limit within the same task. We present empirical λ-achievable frontiers that offer a complete visualization of the return-cost trade-off in the underlying optimization problem. Our results reveal the highly sensitive nature of λ and further show that the performance of λ-update mechanisms does not generalize across cost limits within the same task, meaning that evaluation at a single cost limit risks biased conclusions. We therefore urge the safe RL community to adopt testing algorithms across multiple cost limits as standard practice, and provide (ii) *recommendations for benchmarking* in the form of a recommended set of cost limits for each evaluated task, and offer an open-source code base: `https://github.com/anonymous`.

1 Introduction

Reinforcement learning (RL) addresses sequential decision-making problems by enabling agents to learn from feedback in the form of rewards, with the goal of maximizing their long-term cumulative reward (Sutton et al., 2018). Despite their success in achieving high performance on tasks without critical safety concerns, agents deployed in safety-critical domains must often deal with restrictive constraints. For example, a robot learning locomotion must satisfy safety constraints such as torque limits or avoiding collisions (He et al., 2023; Huang et al., 2024) and falling when walking in the real world (Ha et al., 2020), often requiring a detour from the unconstrained optimal policy. Safety also plays a crucial role in operational domains such as power systems, where agents tasked with scheduling or real-time control must ensure stability and reliability while simultaneously optimizing performance (Yan & Xu, 2020; Chen et al., 2025; Su et al., 2025).

Safe reinforcement learning, also referred to as constrained reinforcement learning, provides a framework in which the learning objectives are extended to explicitly incorporate constraints imposed on the agent. In this framework, the optimization problem has multiple conflicting objectives: the agent must learn a policy that maximizes expected return while simultaneously keeping constraint costs below a specified limit. Safety in RL has been extensively studied over the last decade, leading to a variety of approaches that tackle the constrained optimization problem.

Classical Lagrangian methods have emerged as a popular choice in practical applications in which constraints are softly enforced during training (Garcıa & Fernández, 2015) through a constraint cost limit, showing performance close to the optima while respecting constraints in safety-critical tasks (Ray et al., 2019). By

augmenting the objective with a penalty term weighted by a Lagrange multiplier λ, these methods reformulate the constrained problem into an unconstrained one, allowing us to apply any standard RL algorithm while implicitly accounting for safety.

Theoretically, if the optimal Lagrange multiplier λ^* is known, the solution to the relaxed unconstrained problem is equivalent to the constrained problem (Borkar, 2005). In practice, however, identifying λ^* *"can be as hard as solving the RL problem itself"* (Paternain et al., 2019) due to the lack of a direct relation between the multiplier value and the resulting policy performance. A common practical approach is to automatically update λ during training (Tessler et al., 2019; Stooke et al., 2020). While these approaches guarantee zero duality gap given infinite compute, in the practical setting of finite compute we find that they exhibit considerable variance in performance, frequently either over- or undershooting the cost limit.

Yet, it remains unclear what the effect of over- and undershooting of the cost limit has on the practical performance in terms of the return. There is limited empirical evidence on the achievable optimal trade-off between return and cost as a function of λ, and no systematic benchmark study comparing automated update mechanisms against this empirical optimum currently exists. Furthermore, in practice the cost limit is often set to an arbitrary value without accounting for the underlying constraint restrictiveness of the problem, leaving open the question of how cost limit selection influences practically achievable performance, both within individual tasks and across different tasks.

This paper aims to fill the gap in our understanding of the practical performance of Lagrangian methods in safe RL. We present a systematic empirical analysis focusing on the role of the Lagrange multiplier, along with recommendations to the safe RL benchmarking community and a code base that enables empirical analysis of safe RL benchmark tasks. Our contributions are the following:

1. **Practical variance** We systematically analyze how the choice of the Lagrange multiplier influences model performance for Lagrangian-based version of the on-policy algorithm PPO (Schulman et al., 2017; Stooke et al., 2020; Ray et al., 2019), and show that even when λ^* is used during training, practical variance remains high. We further show automated updates to λ during training also exhibit variance and fall short of the zero duality gap claimed by Paternain et al. (2019), frequently under- or overshooting the specified cost limit. To illustrate the consequences, we construct an empirical λ-achievable frontier showing how the return is affected by over- and undershooting. We find that this effect varies considerably across cost limit values, reflecting differences in the underlying constraint geometry within individual tasks.

2. **Recommendations for benchmarking** We show that no single update mechanism on λ consistently outperforms the others across tasks and cost limits. Therefore, the relative performance of each method depends strongly on the chosen cost limit. This is reflected in the varying slopes of the empirical λ-achievable frontiers, which differ both across tasks and across cost limits within the same task. This highlights the importance of careful cost limit selection based on constraint restrictiveness, for which we provide a protocol, and motivates systematic evaluation across multiple constraint settings in safe RL benchmarks.

2 Related work

Lagrangian methods in safe RL Borkar (2005) introduced the dual gradient descent framework for actor–critic methods and showed that updating the Lagrange multiplier via gradient ascent guarantees convergence to the optimal value λ^*. In this framework, they provided that the policy and value function updates must occur on faster, converged timescales, compared to a slower timescale on which the Lagrange multiplier is updated. Subsequent theoretical work by Paternain et al. (2019) showed that constrained RL problems exhibit zero duality gap, providing the theoretical guarantee that the constrained MDP can, in principle, be solved exactly in the dual domain. They further introduced primal–dual approaches for probabilistic constraints (Paternain et al., 2022), demonstrating that safe policies can be obtained under realistic uncertainty models.

Tessler et al. (2019) introduced RCPO, a Lagrangian-based algorithm that updates the multiplier through gradient ascent. Several works have since explored extensions of Lagrangian methods. Ding et al. (2023b) extended the Lagrangian framework to multi-agent RL, and they furthermore proposed a regularized Lagrangian framework to guarantee safety beyond the asymptotic convergence (Ding et al., 2023a). Jayant & Bhatnagar (2022) introduced a model-based Lagrangian method and showed that integrating model dynamics can accelerate convergence while maintaining safety guarantees. Stooke et al. (2020) revisited the Lagrange multiplier update mechanism and introduced an automated update method that relies on proportional-integral-derivative control, and show that this method stabilizes training compared to pure gradient ascent.

Multi-objective RL Constrained RL is closely related to multi-objective RL, as both involve balancing multiple objectives. Noted by Ray et al. (2019), safety requirements typically exhibit a saturation effect: once the safety threshold is satisfied, further improvements no longer make the system any safer. This property corresponds to the constraint threshold in the constrained formulation. To analyze the underlying constraint geometry, we borrow the concept of Pareto frontiers from multi-objective RL (Roijers et al., 2013; Moffaert & Nowe, 2014), applying it to the return-cost trade-off inherent in constrained optimization problems. Sweeping over the Lagrange multiplier λ corresponds to linear scalarization, a standard technique in multi-objective optimization for tracing a Pareto frontier by combining objectives into a single weighted objective. This approach is only guaranteed to recover the convex hull of the true frontier, not necessarily all Pareto-optimal points (Roijers et al., 2013). Although safe RL is not a multi-objective problem in the strict sense, since the cost limit is fixed a priori, in practice cost limits are often chosen heuristically and varied across tasks. This makes it valuable to analyze the frontiers achievable through linear scalarization of λ as return-cost trade-off profiles across common safe RL benchmark tasks, since such profiles reveal how the constraint geometry changes with the choice of cost limit.

Empirical studies of safe RL From an empirical standpoint, Ray et al. (2019) introduced the Safety Gym benchmark suite, providing standardized environments to assess safe RL algorithms. Their study highlighted that simple Lagrangian-based methods perform competitively among safe RL algorithms, but did not explicitly analyze the role of the Lagrange multiplier itself. Focusing on the *update* of the Lagrange multiplier, Stooke et al. (2020) provided the first systematic empirical insights into PID-controlled multiplier updates, examining the individual influence of its three tunable hyperparameters.

Despite theoretical and algorithmic advances in Lagrangian methods, empirical understanding of how λ affects practical performance remains limited. To the best of our knowledge, no prior work has systematically characterized this behavior, nor applied Pareto frontier analysis to study the underlying constraint geometry and evaluate the quality of Lagrange multiplier update methods against it.

3 Constrained Markov Decision Process

The reinforcement learning problem is commonly formalized as a Markov Decision Process (MDP) (Sutton et al., 2018). When the problem additionally contains a set of constraints, we speak of constrained reinforcement learning, for which we use the formal framework of a Constrained Markov Decision Process (CMDP) (Altman, 1999). A CMDP is described by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, p_0, \gamma, \mathcal{C})$, where $\mathcal{S}$ and $\mathcal{A}$ are the set of all possible states and actions, respectively, p is the transition dynamics distribution $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, r is the reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, $p_0 \in \Delta(\mathcal{S})$ is the initial state distribution and $\gamma \in [0, 1)$ is the discount factor that governs the importance of future rewards. A set of cost functions $\mathcal{C} := \{C_1, ..., C_m\}$ with cost limits $d_1, ..., d_m$ maps transition tuples to costs, $\mathcal{C} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^m$. Actions are selected from a policy π_θ, where π defines a mapping $\pi : \mathcal{S} \to \Delta(\mathcal{A}), s \mapsto \pi(\cdot|s)$, and θ is the set of parameters, for a stationary parametrized policy π in the set of all policies Π. We denote $R(\tau) = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}, s_{t+k+1})$ as the return of a trajectory $\tau = (s_t, a_t, s_{t+1}, ...) \sim p_{\pi_\theta}(s'|s, a)$.

The state value function for the return is defined as $V^R_{\pi_\theta}(s_0) \doteq \mathbb{E}_{\tau\sim\pi_\theta}\big[R(\tau)|s_t = s\big]$ and yields the return objective of the CMDP, which is to maximize the expected cumulative discounted reward $J^R(\pi_\theta)$, which is equivalent to $V^R_{\pi_\theta}(s_0)$. The expected cumulative discounted cost of the policy π_θ is defined as $J^{C_i}(\pi_\theta)$,

which is equivalent to the state value function for the cost, $V_{\pi_\theta}^{C_i}(s_0) \doteq \mathbb{E}_{\tau\sim\pi_\theta}\big[C_i(\tau)|s_t = s\big]$, with $C_i(\tau) = \sum_{k=0}^{\infty}\gamma^k C_i(s_{t+k}, a_{t+k}, s_{t+k+1})$. The feasible set of stationary parametrized policies is $\Pi_{\mathcal{C}} \doteq \{\pi_\theta \in \Pi : \forall i, J^{C_i}(\pi_\theta) \leq d_i\}$. The optimization problem of a CMDP can be expressed as:

$$\begin{aligned} &\max_{\pi_\theta\in\Pi_\theta} J^R(\pi_\theta) \\ &\text{s.t.} \quad J^{C_i}(\pi_\theta) \leq d_i, i = 1, ..., m, \end{aligned} \tag{1}$$

where $\Pi_\theta \subseteq \Pi$ denotes the set of parametrized policies with parameters θ. Compared to traditional MDPs (Bellman, 1957), local policy search for CMDPs involves the additional requirement that each policy iteration remains feasible with respect to the CMDP constraints. Therefore, instead of optimizing over Π_θ, the optimization algorithm should optimize over $\Pi_\theta \cap \Pi_{\mathcal{C}}$. The optimal policy π^* of a CMDP is then found by $\pi^* = \arg\max_{\pi_\theta\in\Pi_{\mathcal{C}}} J^R(\pi_\theta)$.

4 Methodology

4.1 Lagrangian methods

The optimization problem in Eq. 1 is in a primal form, implying that the constraints must be strictly satisfied at every step and thus each policy update has to remain feasible. We can soften the constraints during training by moving to the dual problem following the Lagrangian method. This method converts a CMDP into an unconstrained one with Lagrange relaxation (Bertsekas, 1997; Boyd & Vandenberghe, 2023):

$$\min_{\lambda\geq 0}\max_{\theta}\mathcal{L}(\lambda,\theta) = \min_{\lambda\geq 0}\max_{\theta}\Big[J^R(\pi_\theta) - \Big(\sum_{i=1}^{m}\lambda_i(J^{C_i}(\pi_\theta) - d_i)\Big)\Big], \tag{2}$$

where $\mathcal{L}$ is the Lagrangian and λ_i is the Lagrange multiplier for the i-th constraint. For convenience in notation we drop the index i and use λ, $J^{\mathcal{C}}$ and d to encompass the entire set of constraints collectively in the rest of this paper [1]. The inequality constraints of the optimization problem are now relaxed to a penalty loss term $\xi = J^{\mathcal{C}}(\pi_\theta) - d$. This allows us to find the optimal solution of a CMDP using any standard RL algorithm with a modified optimization objective. Intuitively, λ is a penalty parameter in the optimization objective, which can be viewed as a parameter that defines the trade-off between the return and cost.

Fixed Lagrange multiplier The Lagrange multiplier λ can be manually set to a constant value and kept fixed throughout training. The resulting unconstrained problem in Eq. 2 can then be solved by maximizing over the policy parameters θ. When λ is chosen to be the optimal value λ^*, this formulation is equivalent to solving the original constrained problem from Eq. 1 and one would be able to find the optimal solution at the saddle point $\big(\theta(\lambda^*), \lambda^*\big)$ (Borkar, 2005).

4.2 Automated multiplier updates

Finding the optimal value λ^* is often computationally- and time-intensive in practice, which motivates the search for automated alternatives. Eq. 2 is in dual form and convex, which allows us to efficiently solve it using gradient descent. Then, the *dual gradient descent algorithm* alternates between the optimization of the policy parameters θ and the Lagrange multiplier λ (Boyd & Vandenberghe, 2023). First, the unconstrained problem in Eq. 2 is solved to update θ. Subsequently, λ is updated by minimizing the penalty loss following a preferred update rule. If the agent violates fewer constraints, λ will gradually be decreased, and vice versa, until all cost functions satisfy their respective cost limits.

Gradient ascent (GA) update Performing gradient descent by taking $\nabla_\lambda\mathcal{L} = -\xi$ and substituting this in $\lambda_{k+1} = \lambda_k - \eta\cdot\nabla_\lambda\mathcal{L}$, with η a step-size parameter, yields a gradient *ascent* (GA)-update on the Lagrange

[1] Dropping index i is done only to avoid clutter in notation. We still implicitly refer to the entire set of constraints, in which each individual constraint corresponds with its own cost limit. For clarification: this means that it is still possible to assume multiple constraints in the CMDP.

multiplier as in Eq. 3.

$$\lambda_{k+1} = \big(\lambda_k + \eta \cdot \xi\big)_+, \tag{3}$$

where $(\cdot)_+$ denotes the projection onto $\mathbb{R}_+$ and comes from the KKT conditions for inequality-constrained optimization (Boyd & Vandenberghe, 2023; Borkar, 2005; Tessler et al., 2019).

PID-controlled update The gradient ascent update from Eq. 3 integrates the constraint, but its inherent learning dynamics can lead to oscillations when modeled with second-order dynamics (Platt & Barr, 1987). This is because the outputs are adjusted proportional to the *accumulated* constraint violations over time. This results in frequent constraint violations during intermediate iterates. Stooke et al. (2020) proposed an update method that utilizes the derivatives of the penalty term, introducing an additional proportional and derivative control term to the Lagrange multiplier update as shown in Eq. 4.

$$\lambda_{k+1} = \big(K_P \xi_k + K_I I_k + K_D \partial_k\big)_+, \tag{4}$$

where $\xi_k = J^{\mathcal{C}}(\pi_{\theta_k}) - d_k$ is the penalty loss as a function of update iteration k, $I_k = (I_{k-1} + \xi_k)_+$ is the integral of the penalty loss, $\partial_k = \big(J^{\mathcal{C}}(\pi_{\theta_k}) - J^{\mathcal{C}}(\pi_{\theta_{k-1}})\big)_+$ is the derivative of the constraint, and K_P, K_I and K_D are tunable step-size parameters corresponding to the proportional, integral and derivative coefficients, respectively (Åström & Hägglund, 2006).

4.3 λ-achievable frontiers and constraint restrictiveness

In safe RL, the cost limit is fixed a priori as part of the problem formulation. However, in practical applications, cost limits are often chosen heuristically and varied across tasks, making the choice of cost limit a practically relevant design decision. This makes it valuable to analyze Pareto frontiers as return-cost trade-off profiles, as such profiles reveal how the constraint geometry changes with the choice of cost limit. Geometrically, the optimal solution to the optimization problem in Eq. 1 lies on the upper boundary of the achievable return for a given cost limit d, which forms the Pareto frontier of the return as a function of the cost: $R(C)$. Because we construct this frontier by linear scalarization, i.e., sweeping a scalar λ, it is guaranteed to coincide with the true Pareto frontier *only* where the achievable return-cost set is locally convex; in regions of local non-convexity, it instead traces the convex hull (Roijers et al., 2013). We therefore refer to this curve as the *λ-achievable frontier* throughout this paper, to make explicit that it characterizes what Lagrangian-based methods can achieve, rather than the full space of Pareto-optimal policies achievable by any method in the class of safe RL algorithms.

Each point on this frontier corresponds to a policy for which no other policy achieves both higher return and lower cost. The constrained optimization problem therefore amounts to selecting the point on the Pareto frontier corresponding to the desired cost limit d. The local slope of the frontier, $\frac{dR}{dC}$, captures the marginal trade-off between return and cost. Intuitively, it indicates how much additional return can be gained per unit increase in allowed cost. A steep slope indicates that a small reduction in cost would lead to a large decrease in return; in other words, the constraint is highly restrictive in this region. Vice versa, flatter regions of the curve correspond to regimes where tightening the cost constraint has only a limited effect on return.

We therefore propose a general protocol for selecting cost limits when benchmarking Lagrangian methods on any task: (1) sweep λ over a grid to construct the empirical λ-achievable frontier; (2) estimate the local slope $\frac{dR}{dC}$ across the swept range; (3) select at least one cost limit from a high-slope (restrictive) region and at least one from a low-slope (less restrictive) region; (4) evaluate and compare algorithms at both. We apply this protocol to the eight tasks studied in Section 5, and report the resulting cost limits in Appendix Table A.1 as a worked example.

5 Experimental setup

Experiments are conducted on eight benchmark tasks from the Safety Gymnasium task suite (Ji et al., 2023), spanning different levels of task and constraint complexity. These tasks consist of four safe navigation tasks and four safe velocity tasks.

To construct empirical λ-achievable frontiers of the return-cost plane for each task, we train a set of 25 distinct policies with a manually fixed Lagrange multiplier, by sweeping λ over the set $\lambda \in \{10^{\ell_i}\}_{i=1}^{25}$, where ℓ_i are evenly spaced in $(-1, 1)$. For each value of λ, we compute the average return and cost over the final 5% of training. The collection of points induced by sweeping over λ forms an empirical λ-achievable frontier in the return-cost plane. Each fixed λ yields a point on this frontier, representing the approximate optimal return-cost trade-off under that particular λ.

Furthermore, we compare the performance of the fixed multiplier λ^* at the cost limit, the GA-updated λ and PID-updated λ with each other. We train 10 seeds of models for $3.5 \cdot 10^7$ timesteps each. To ensure that multiple levels of cost restrictiveness are taken into account, we analyze models trained at a cost limit of 10.0 and 25.0 for navigation tasks, and at 25.0 and 400.0 for velocity tasks.

We train the Lagrangian version of PPO (Ray et al., 2019; Schulman et al., 2017) using the Omnisafe benchmark suite (Ji et al., 2024). PPO-Lag is employed for the GA-update of the Lagrange multiplier, and CPPO-PID for the PID-controlled update (Stooke et al., 2020). We note that Eq. 3 describes the vanilla GA-update rule. In the Omnisafe benchmark suite, the Lagrange multiplier for the GA update is represented as a trainable parameter and updated using the Adam optimizer. The PID-controlled update is used exactly as described in Eq. 4.

In total, we performed 8 tasks $\times$(25 values of λ + (3 update methods $\times$2 cost limits)) $\times$ 10 seeds = 2480 individual runs for our experiments. The full set of experiments is mostly CPU-heavy and required approximately $57 \cdot 10^3$ CPU core-hours of compute. All additional details regarding the experimental setup can be found in the Supplementary Materials Section SM.1. Code: `https://github.com/anonymous`.

6 Results

In Figure 1, we present the empirical λ-achievable obtained by sweeping over λ, which reveal differences in the constraint geometry across the evaluated tasks. We then study the GA and PID update mechanisms and their sensitivity to different constraint regimes and present the results in Table 1.

Constraint geometry Figure 1 visualizes the empirical λ-achievable frontiers for all evaluated tasks directly by plotting return as a function of cost. The local slope of the curve, $\frac{dR}{dC}$, captures the sensitivity of return to changes in the allowed cost. We highlight three distinct patterns we observe in the frontiers: In Figure 1a, the region with a cost $\lesssim 20$ has a steep slope, meaning that the cost is highly restrictive in this region. In other words, decreasing the allowed cost would result in a huge decrease in return in this region. On the other hand, the region with a cost $\gtrsim 20$ is less restrictive, where a slight increase in the allowed cost yields only marginal improvements in return. In Figure 1c, the slope of the curve is relatively consistent across all values of λ, indicating that the restrictiveness of the cost limit acts relatively linear. Figure 1e shows an example of non-monotonic trade-off behavior: the region with a cost $\lesssim 25$, having a steep slope, is highly restrictive, for $25 \lesssim$ cost $\lesssim 300$ it is comparatively less restrictive with a flatter curve, and for a cost $\gtrsim$ 300 the constraint becomes restrictive again.

These highlighted cases show that not merely the numerical cost limit, but rather the slope of the frontier, determines how restrictive a safety requirement truly is. Comparing the navigation and velocity tasks, we observe that the slope of the frontier of the velocity tasks is generally steeper than that of the navigation tasks. This can be explained by differences in the scale of the return and cost across the two task categories. Consequently, the velocity tasks exhibit a more restrictive constraint regime than the navigation tasks. Alternatively visualized, Figure A.1 in Appendix A.1 shows the return-cost profiles as a function of λ for all evaluated tasks. These profiles differ substantially across tasks, both in scale and shape, indicating that the return-cost dynamics are highly task-specific.

Practical variance Table 1 shows the performance of all three methods in terms of average return and cost, computed over the last 5% of training. The best-performing method in each setting is highlighted in green. Since we consider only those methods that satisfy the specified cost limit, we observe that the *best* method does not always refer to the method with the highest return. We remark that there is no best-performing method for SafetyPointGoal1-v0 at a cost limit of 25.0. We observe that the optimal method differs both across tasks

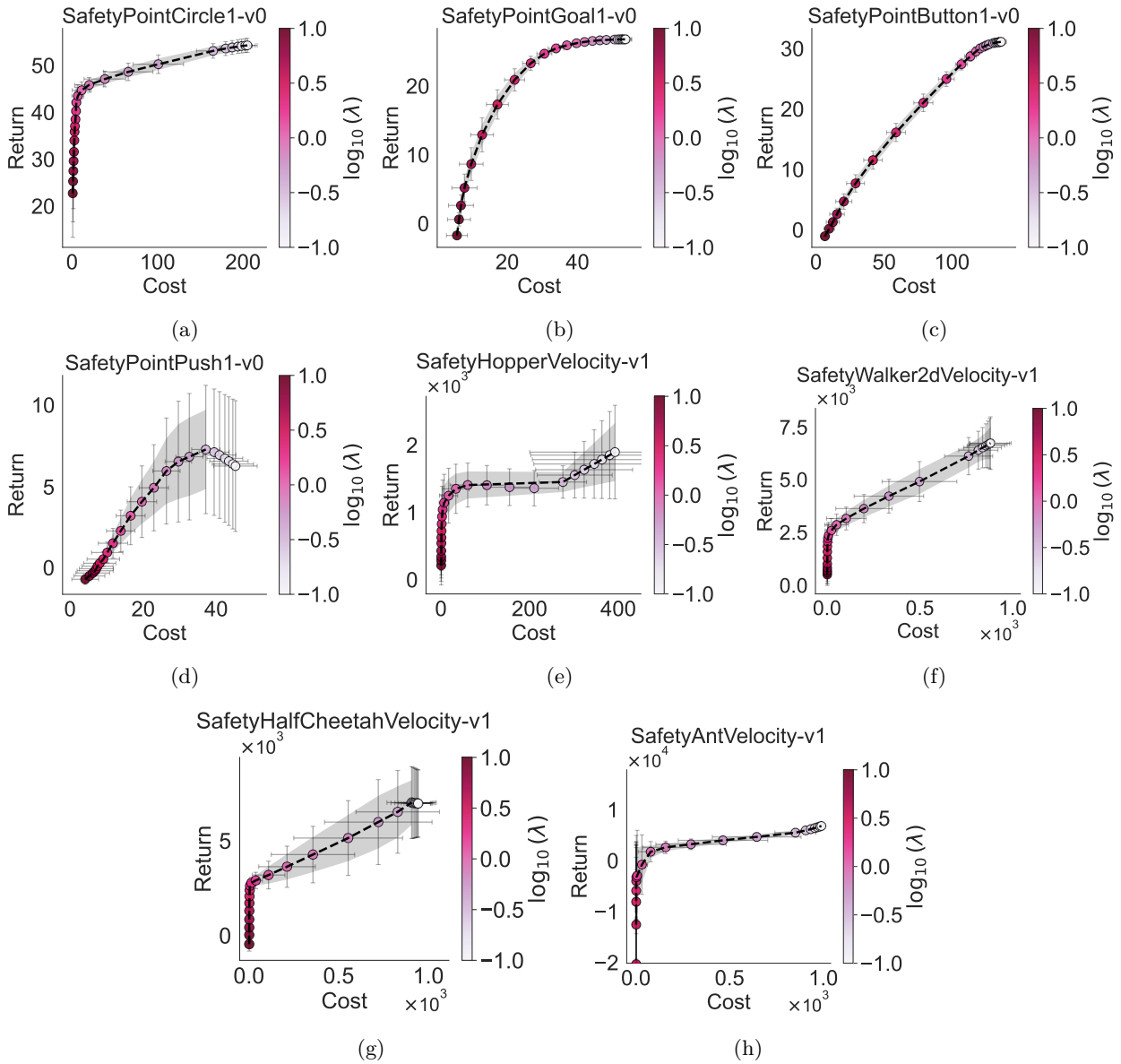

Figure 1: Smoothed empirical λ-achievable frontiers of return versus cost as a function of λ, averaged over 10 seeds. Error bars denote the 1σ across seeds. The dashed line traces only the Pareto-optimal subset of sampled λ-points, and the shaded region shows the 95% confidence interval of the mean empirical Pareto-optimal frontier. The local slope of each curve captures the sensitivity of the return with respect to changes in the allowed cost. E.g., the geometries of SafetyPointCircle1-v0 (1a), SafetyPointButton1-v0 (1c) and SafetyHopperVelocity-v1 (1e) highlight three distinct patterns. Their shapes differ both across and within the tasks themselves, indicating that the restrictiveness of cost limits changes non-linearly.

and across cost limits within the same task, indicating that the constraint restrictiveness influences which update algorithm performs best.

Costs that violate the specified cost limit are highlighted in red. In some cases, a method overshoots or undershoots the target cost. For example, on the SafetyPointCircle1-v0 task, the fixed λ^* method achieved a cost of 4.75 ± 3.93 against a cost limit of 25.0, a substantial undershoot that illustrates how large deviations from the optimum can occur in practice, even when using λ^*. The observed deviations from the specified

Table 1: Performance in average return and constraint cost, computed over the final 5% of training timesteps. A fixed optimal multiplier λ^* (selected per cost limit) is compared against GA- and PID-updated multipliers ($K_P = 10^{-4}, K_I = 10^{-4}, K_D = 0.0$), each trained under the corresponding cost limit. Navigation tasks use cost limits of 10.0 and 25.0, while velocity tasks use 25.0 and 400.0. Results are averaged over 10 seeds with standard deviations. Per task, the best-performing feasible method (satisfying the cost limit) is highlighted in green, while costs violating the limit are shown in red. The results indicate that there is no single method that consistently performs best within a task, but rather depends on the chosen cost limit. For the navigation tasks, PID tends to outperform GA in less restrictive (flatter-slope) regions of the λ-achievable frontier, while GA tends to perform comparatively better in more restrictive (steep-slope) regions; this pattern is less visible and less consistent for the velocity tasks.

Navigation tasks

		Cost limit = 10.0			**Cost limit = 25.0**		
		fixed λ^*	GA	PID	fixed λ^*	GA	PID
SafetyPoint Circle1-v0	Return	**43.42 ±3.35**	42.43 ±5.25	41.30 ±5.06	45.16 ±1.06	41.80 ±10.25	**46.10 ±0.96**
	Cost	3.91 ±2.60	9.61 ±2.43	4.10 ±4.02	4.75 ±3.93	19.94 ±7.73	23.37 ±6.63
SafetyPoint Goal1-v0	Return	5.80 ±2.77	**6.62 ±3.12**	22.34 ±1.64	23.80 ±0.92	22.83 ±1.85	24.73 ±0.70
	Cost	8.49 ±4.84	9.06 ±2.46	27.47 ±2.13	25.48 ±1.82	25.52 ±0.56	30.84 ±1.98
SafetyPoint Button1-v0	Return	0.28 ±0.59	**0.47 ±0.61**	3.83 ±0.95	3.79 ±1.12	5.59 ±0.93	**5.41 ±1.58**
	Cost	11.39 ±2.08	7.78 ±1.51	15.56 ±4.34	14.11 ±4.93	26.73 ±2.30	24.85 ±8.39
SafetyPoint Push1-v0	Return	0.88 ±0.63	**0.59 ±0.41**	1.98 ±1.56	4.96 ±3.52	4.38 ±2.05	**5.49 ±2.84**
	Cost	11.88 ±2.42	8.88 ±3.66	10.68 ±2.26	28.05 ±5.85	25.35 ±2.37	20.36 ±3.27

Velocity tasks

		Cost limit = 25.0			**Cost limit = 400.0**		
		fixed λ^*	GA	PID	fixed λ^*	GA	PID
SafetyHopper Velocity-v1	Return	1292.34 ±603.52	**1682.52 ±37.62**	1443.69 ±510.46	1979.64 ±623.42	1682.93 ±140.68	**1713.47 ±159.91**
	Cost	6.31 ±6.25	24.54 ±6.23	26.56 ±15.69	428.91 ±198.81	357.75 ±35.00	347.49 ±44.96
SafetyWalker2d Velocity-v1	Return	2668.69 ±60.60	**3127.55 ±76.92**	3132.17 ±82.10	4334.02 ±1189.20	**3399.87 ±418.21**	4089.29 ±610.61
	Cost	2.96 ±3.82	22.64 ±5.61	29.82 ±12.71	410.22 ±318.62	381.70 ±58.99	417.78 ±25.41
SafetyHalfCheetah Velocity-v1	Return	2861.85 ±66.76	2745.51 ±540.52	**2965.10 ±86.61**	**4088.51 ±1972.41**	2940.21 ±534.06	3365.50 ±818.29
	Cost	4.82 ±2.95	22.26 ±15.09	16.76 ±28.28	283.79 ±427.59	315.36 ±190.69	386.97 ±62.40
SafetyAnt Velocity-v1	Return	−2602.45 ±6697.29	3387.12 ±22.61	**3332.30 ±72.87**	4023.17 ±981.96	**2727.06 ±1054.88**	2204.24 ±4391.94
	Cost	0.52 ±0.95	27.75 ±4.96	18.60 ±7.60	506.27 ±295.64	323.48 ±162.65	376.74 ±132.07

cost limits are illustrated in Figures SM.2 and SM.3 in Supplementary Materials Section SM.2, which project the results from Table 1 onto the empirical frontiers. The projected GA and PID results further show that no single method consistently performs best across or within tasks. Moreover, the substantial variance reported in Table 1, even across 10 seeds, indicates that methods performing well on average may still produce individual runs that violate the constraint, while methods with lower average performance may occasionally satisfy it. This variability raises the question of whether practitioners should impose stricter cost limits during training to increase the likelihood of satisfying the desired constraint, given that Lagrangian methods are not strict constraint enforcers by design.

7 Discussion & conclusion

Limitations The empirical λ-achievable frontiers in Figure 1 reveal the constraint geometry of the underlying optimization problems and demonstrate that these insights can substantially improve the interpretation and evaluation of safe RL methods by evaluating for multiple cost limits. We point out that these geometric insights do not directly reflect the intrinsic difficulty of learning the tasks; regimes with a steeper slope are not necessarily harder to solve, but rather indicate that the safety requirement is more restrictive.

We highlight that since λ-achievable frontier is constructed via linear scalarization, it is only guaranteed to trace the true Pareto frontier where the underlying achievable return-cost set is locally convex. In non-convex regions, the swept curve instead traces the convex hull, potentially omitting Pareto-optimal policies that no λ can recover. We do not observe large discontinuous jumps in cost between consecutive swept λ values in our results, suggesting the achievable sets are approximately locally convex for the tasks and λ-range studied. However, we cannot rule out non-convex regions outside this range. In particular, Figure 1e shows a slope that does not decrease monotonically with cost, indicating local non-convexity that linear scalarization cannot fully recover.

Moreover, although Lagrangian methods can be combined with any standard RL algorithm, we focused on PPO, which remains state-of-the-art and is the most widely used algorithm for Lagrangian approaches in practice. As such, PPO provides a representative and practically relevant testbed for studying the behavior of the Lagrange multiplier. Though, empirical results may vary across algorithms. We therefore present preliminary results for Lagrangian-based SAC in Supplementary Materials Section SM.2.2 (Haarnoja et al., 2018). These indicate that off-policy algorithms produce constraint geometries different from on-policy methods, motivating further evaluation of our approach across a broader set of Lagrangian-based algorithms.

In addition, the PID-controlled update mechanism was evaluated under a single parameter configuration ($K_P = 10^{-4}$, $K_I = 10^{-4}$, $K_D = 0$), corresponding to PI-control. While this reflects the most commonly used practical configuration, we suggest that broader exploration of PID-parameters in future work could provide deeper insight into the dynamics and robustness of Lagrange multiplier updates. We further note that Adam's momentum in the GA-update could plausibly contribute to the observed variance and over- and undershooting of the cost limit, in addition to the absence of a derivative term relative to PID. Disentangling this effect from the update rule itself would require an ablation study comparing the vanilla SGD-style GA-update to Adam and possibly other choices of optimizers.

Conclusion In this work, we empirically studied the constraint geometry of eight tasks from the Safety Gymnasium suite (Ji et al., 2023), one of the most widely used and standardized benchmarks for safe RL. These tasks were selected to capture diversity in return–cost dynamics, constraint cost restrictiveness, and levels of task complexity, providing a sufficiently broad empirical basis for the results presented in this study. Given the diversity of safe RL domains, we encourage the benchmark community to accompany published benchmarks with analyses of empirical λ-achievable frontiers and to extend such analyses to additional domains, such as visual safety tasks (Tomilin et al., 2025) or operational control benchmarks (Ramanujam et al., 2025). We release an open-source code base that enables researchers to reproduce and extend our experiments. Since the quality of an algorithm can not be properly assessed by evaluating a single point on a frontier, we recommend evaluating Lagrangian safe RL methods across multiple cost limits to capture different levels of restrictiveness. Accordingly, we provide an explicit protocol described in Section 4.3 for this, and we additionally provide PPO-Lag specific recommended cost limits for each task in Appendix Section A.2. We encourage the community to adopt similar practices when proposing new safe RL benchmark tasks, and note that a complete, general evaluation methodology for safe RL algorithms remains an important direction for future work.

References

Eitan Altman. *Constrained Markov Decision Processes*. Routledge, New York, 1999. ISBN 978-1-315-14022-3. doi: 10.1201/9781315140223.

Karl J Åström and Tore Hägglund. Pid control. *IEEE Control Systems Magazine*, 1066:30–31, 2006.

Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. ISSN 0095-9057. Publisher: Indiana University Mathematics Department.

D. P. Bertsekas. Nonlinear Programming. *Journal of the Operational Research Society*, 48(3):334–334, March 1997. doi: 10.1057/palgrave.jors.2600425. Publisher: Taylor & Francis.

V.S. Borkar. An actor-critic algorithm for constrained Markov decision processes. *Systems & Control Letters*, 54(3):207–213, March 2005. ISSN 01676911. doi: 10.1016/j.sysconle.2004.08.007.

Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge New York Melbourne New Delhi Singapore, version 29 edition, 2023. ISBN 978-0-521-83378-3.

Guibin Chen, Lun Yang, and Xiaoyu Cao. A deep reinforcement learning-based charging scheduling approach with augmented Lagrangian for electric vehicles. *Applied Energy*, 378:124706, January 2025. ISSN 03062619. doi: 10.1016/j.apenergy.2024.124706.

Dongsheng Ding, Chen-Yu Wei, Kaiqing Zhang, and Alejandro Ribeiro. Last-iterate convergent policy gradient primal-dual methods for constrained mdps. *Advances in Neural Information Processing Systems*, 36:66138–66200, 2023a.

Dongsheng Ding, Xiaohan Wei, Zhuoran Yang, Zhaoran Wang, and Mihailo Jovanovic. Provably Efficient Generalized Lagrangian Policy Optimization for Safe Multi-Agent Reinforcement Learning. In *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, pp. 315–332. PMLR, June 2023b. ISSN: 2640-3498.

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

Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

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*, pp. 1861–1870. Pmlr, 2018.

Tairan He, Weiye Zhao, and Changliu Liu. AutoCost: Evolving Intrinsic Cost for Zero-Violation Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(12):14847–14855, June 2023. ISSN 2374-3468. doi: 10.1609/aaai.v37i12.26734. Number: 12.

Weidong Huang, Jiaming Ji, Chunhe Xia, Borong Zhang, and Yaodong Yang. Safedreamer: Safe reinforcement learning with world models. In *The Twelfth International Conference on Learning Representations*, 2024.

Ashish K Jayant and Shalabh Bhatnagar. Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm. *Advances in Neural Information Processing Systems*, 35:24432–24445, 2022.

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.

Kristof Van Moffaert and Ann Nowe. Multi-Objective Reinforcement Learning using Sets of Pareto Dominating Policies. *Journal of Machine Learning Research*, 2014.

Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained Reinforcement Learning Has Zero Duality Gap. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Santiago Paternain, Miguel Calvo-Fullana, Luiz FO Chamon, and Alejandro Ribeiro. Safe policies for reinforcement learning via primal-dual methods. *IEEE Transactions on Automatic Control*, 68(3):1321–1336, 2022.

John Platt and Alan Barr. Constrained differential optimization. In *Neural information processing systems*, 1987.

Asha Ramanujam, Adam Elyoumi, Hao Chen, Sai Madhukiran Kompalli, Akshdeep Singh Ahluwalia, Shraman Pal, Dimitri J Papageorgiou, and Can Li. Safeor-gym: A benchmark suite for safe reinforcement learning algorithms on practical operations research problems. *arXiv preprint arXiv:2506.02255*, 2025.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research*, 48:67–113, October 2013. ISSN 1076-9757. doi: 10.1613/jair.3987.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9133–9143. PMLR, November 2020. ISSN: 2640-3498.

Tong Su, Tong Wu, Junbo Zhao, Anna Scaglione, and Le Xie. A review of safe reinforcement learning methods for modern power systems. *Proceedings of the IEEE*, 2025.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 2. 2018.

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2019.

Tristan Tomilin, Meng Fang, and Mykola Pechenizkiy. Hasard: A benchmark for vision-based safe reinforcement learning in embodied agents, 2025.

Ziming Yan and Yan Xu. Real-Time Optimal Power Flow: A Lagrangian Based Deep Reinforcement Learning Approach. *IEEE Transactions on Power Systems*, 35(4):3270–3273, July 2020. ISSN 1558-0679. doi: 10.1109/TPWRS.2020.2987292.

A.1 Additional results

We explicitly visualize the trade-off between return and cost for different fixed values of the Lagrange multiplier λ. These λ-profiles are constructed by training 10 seeds of 25 models for $3.5 \cdot 10^7$ timesteps each, where every model is trained with a distinct, manually fixed multiplier $\lambda \in \{10^{\ell_i}\}_{i=1}^{25}$, with ℓ_i evenly spaced in $(-1, 1)$ (note: $\lambda = 0$ corresponds to standard PPO). For each value of λ, we compute the average return and cost over the final 5% of training, of which the results are plotted in Figure A.1. These profiles differ substantially across tasks, both in scale and shape, indicating that the return-cost dynamics are highly task-specific.

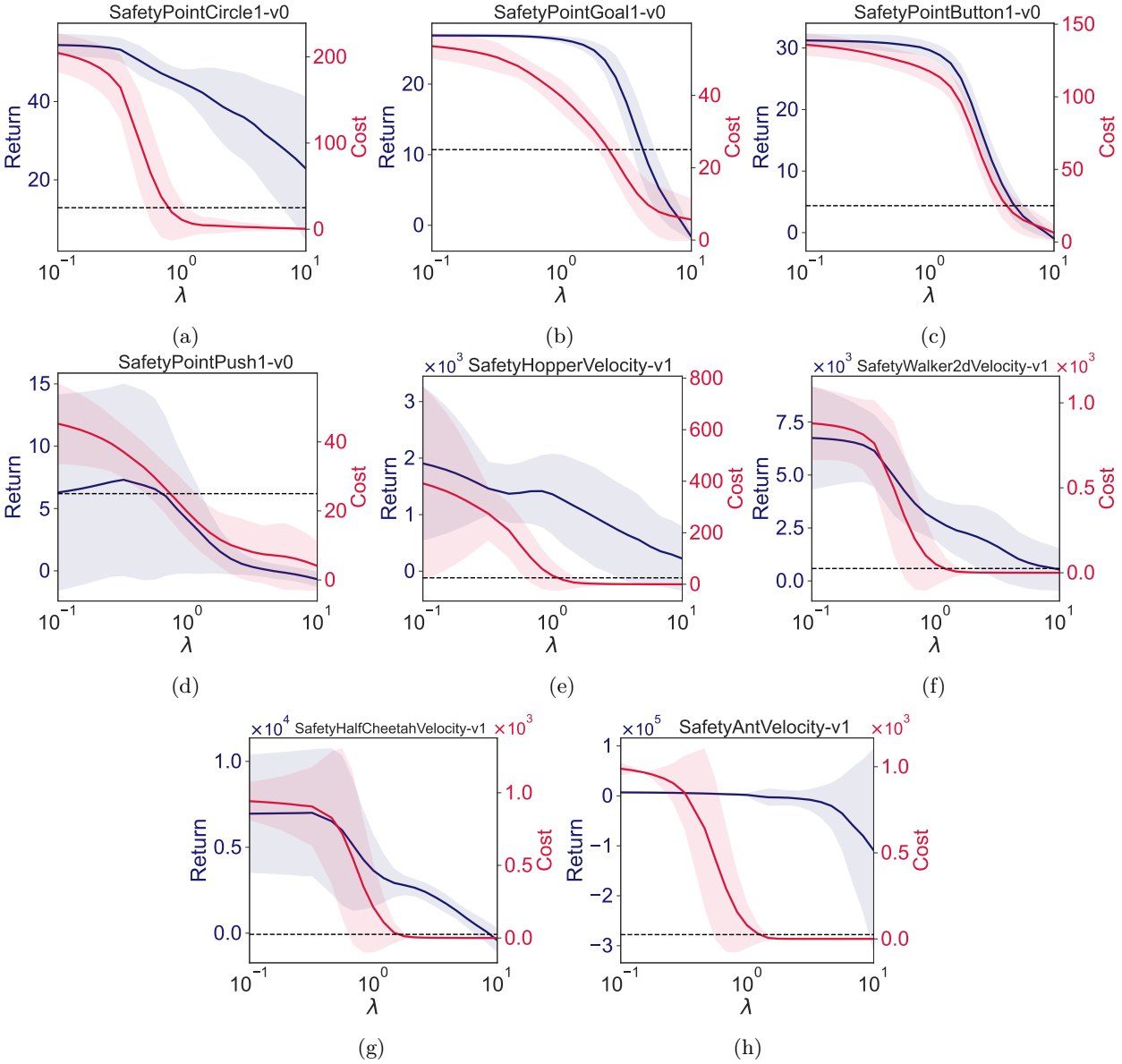

Figure A.1: Smoothened λ-profiles averaged over 10 seeds, with the shaded regions indicating the 95% confidence intervals of the return and cost, for all evaluated tasks. A cost limit of 25.0 is indicated by the dashed lines. These λ-profiles offer practical guidance on tuning λ with respect to a desired cost limit. The return- and cost-curves are monotonically decreasing with increasing λ for all tasks, except for SafetyPointPush1-v0 (A.1d) and SafetyHopperVelocity-v1 (A.1e), where small regions of λ show a slight increase in the return curve. The variance across 10 seeds is high, indicating the highly sensitive nature of λ.

If we draw a horizontal line corresponding to a particular cost limit, e.g., 25.0 in Figure A.1, we observe that relaxing the constraint, i.e. allowing for a higher cost, consistently increases the achievable return, except for SafetyPointPush1-v0 (A.1d) and SafetyHopperVelocity-v1 (A.1e), where small regions of λ show a slight decrease in the return curve for increasing cost/decreasing λ. We attribute these deviations to the following mechanism: our smoothing procedure (LOWESS, frac=0.5) is applied to the return and cost curves independently, each purely as a function of λ. Because return and cost are smoothed separately, this does not guarantee that the resulting joint (return, cost) relationship is monotonic, and thus two individually well-behaved smoothed curves can still produce a non-monotonic return-cost relationship. SafetyPointPush1-v0 and SafetyHopperVelocity-v1 exhibit the highest across-seed variance among their respective task groups as shown in Figures SM.4d and SM.4e. At high variance, a λ that happens to yield slightly higher mean cost can also happen to yield slightly lower mean return purely from sampling noise, even where the true underlying frontier is monotonic. This is compounded by our finite-compute setting.

Due to the difference in shapes and ranges of the cost curves, the optimal value for the Lagrange multiplier, λ^*, i.e., where the cost curve crosses a specified cost limit, varies across tasks. This indicates that the optimal value λ^* is highly task-dependent and dependent on the value of the cost limit. Nevertheless, these λ-profiles can offer practical guidance on tuning λ with respect to a desired cost limit for each of the presented tasks, and offer insights on the return-cost trade-off complementary to the empirical λ-achievable frontiers shown in Figure 1 of the main paper.

A.2 Recommendations on cost limits

Table A.1 presents a list of recommended cost limits for each of the eight evaluated benchmark tasks. These cost limits are selected based on the slopes of the empirical λ-achievable frontiers presented in Figure 1 of the main paper. We emphasize that these recommendations are conditional on the PPO-Lag-specific λ-achievable frontier obtained under our training budget, rather than a provable, algorithm-independent property of the CMDP. As additional compute or analyses of stronger baselines become available, the underlying frontier could shift outward, and the specific numeric thresholds recommended here may require recalibration. We expect, however, that the qualitative transitions between restrictive and less-restrictive regions plausibly reflect real task dynamics, e.g., torque or collision limits inherent to the environment's physics, rather than being purely an effect of algorithm capability. Based on the protocol described in Section 4.3, we selected cost limits in a restrictive region, i.e., with a steep slope, and cost limits in a less restrictive region, i.e., where the curve is flatter. Based on these different cost limits, the best-performing update mechanism for the Lagrange multiplier can vary within the same task, as is shown in Table 1 in the main paper. We therefore encourage the community to use these recommended cost limits when evaluating Lagrangian-based algorithms. Additionally, when proposing new safe RL benchmark tasks, we recommend providing similar lists of cost limits to offer practical guidance to benchmark users.

Table A.1: Recommended cost limits per task, derived from the PPO-Lag-specific λ-achievable frontier under the training budget used in this study (see Section 5). These values are illustrative outputs of the general protocol described in Section 4.3, not fixed, algorithm-independent properties of the underlying CMDPs.

Task	**Cost limits**
SafetyPointCircle1-v0	20; 150
SafetyPointGoal1-v0	15; 40
SafetyPointButton1-v0	50; 150
SafetyPointPush1-v0	10; 50
SafetyHopperVelocity-v1	5; 200; 400
SafetyWalker2dVelocity-v1	5; 400
SafetyHalfCheetahVelocity-v1	5; 400
SafetyAntVelocity-v1	5; 400