An Empirical Study of Lagrangian Methods in Safe Reinforcement Learning

Lindsay Spoor **Álvaro Serra-Gómez** **Aske Plaat** **Thomas Moerland**
Leiden Institute of Advanced Computer Science
Leiden University
{l.j.spoor, a.serra-gomez, a.plaat, t.m.moerland}@liacs.leidenuniv.nl

Abstract

In safety-critical domains such as robotics and navigation, agents must balance performance with safety constraints. Safe reinforcement learning (SRL) addresses this challenge, with Lagrangian methods being a widely used approach. Their effectiveness, however, depends strongly on the choice of the Lagrange multiplier λ, which governs the trade-off between return and constraint cost. We analyze (i) *optimality* and (ii) *stability* of Lagrange multipliers in SRL across multiple benchmark tasks. By constructing λ-profiles, we visualize the sensitivity of performance to λ and show that automated updates of λ can recover or even surpass the performance of optimally tuned fixed multipliers. While PID-controlled updates can reduce oscillations, this method requires careful tuning, emphasizing the need for more robust stabilization strategies for Lagrangian methods in SRL. Code for reproducing our results is available at https://github.com/lindsayspoor/Lagrangian_SafeRL.

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 and Barto, 2018). Despite their success in achieving high performance on tasks without critical safety concerns, agents deployed in safety-critical domains must often deal with conflicting objectives. 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.

Safe reinforcement learning (SRL) 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. SRL has been extensively studied over the last decade, leading to a variety of approaches that tackle the constrained optimization problem, such as CPO (Achiam et al., 2017), IPO (Liu et al., 2020), PCPO (Yang et al., 2020) and a Lyapunov-based approach (Chow et al., 2018).

Classical Lagrangian methods have emerged as a popular choice in practical applications in which strict enforcement of constraints during training is not a hard requirement, showing performance close to the optima while respecting constraints in safety-critical tasks (Garcıa and Fernandez, 2015; Ray et al., 2019). They reformulate the constrained problem into an unconstrained one by augmenting the objective with a penalty term weighted by a Lagrange multiplier λ. This transformation allows us to apply standard RL algorithms while implicitly accounting for safety. However, the practical performance of Lagrangian methods depends heavily on the stability and tuning of λ, which *"can be as hard as solving the RL problem itself"* (Paternain et al., 2019) due to the lack of a direct

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025

relation between the multiplier value and the resulting policy performance. Even though common practical solutions, such as gradient ascent (GA) (Tessler et al., 2019) and PID-control (Stooke et al., 2020), adaptively update λ automatically during training, these methods introduce their own learning dynamics, which are sensitive to parameter settings and can cause practical issues such as overshooting the constraint cost limit during updates.

There is limited empirical evidence on the effectiveness of updating the Lagrange multiplier during training. Moreover, a systematic analysis of the robustness of the currently employed update mechanisms is still lacking, as these methods remain fragile in their ability to ensure truly stable learning dynamics in practice. This paper aims to fill this gap by presenting a systematic empirical analysis focusing on the role of the Lagrange multiplier. Our contributions are the following: (i) **Optimality** (Section 2.1). We analyze how the choice of the Lagrange multiplier influences performance. We do this by visualizing the trade-off between return and cost as a function of λ. We find that the optimal value for λ is highly task-dependent, and our results visually illustrate the lack of general intuition on how to tune the multiplier in practice. We furthermore confirm that automatically updated multipliers are actually able to exceed in performance compared to training with a manually fixed optimal value for λ, and that this is due to a vast difference in learning trajectory of the two methods. (ii) **Stability** (Section 2.2). We study the stability of different automated update mechanisms, including GA- and PID-controlled updates. We find that, while GA-updated multipliers often exhibit unstable behavior, PID-controlled updates actually *shift* the problem of stability rather than *solve* it, as this method is highly sensitive to hyperparameter choices and does not consistently outperform GA-based methods in terms of stability or overall performance.

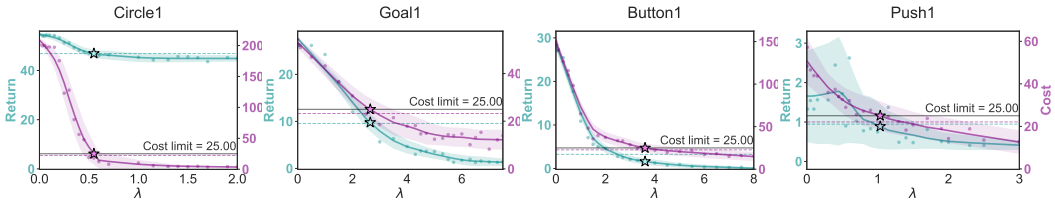
For a formal description of SRL, Lagrangian methods and automated multiplier updates, we refer to Appendix A. All details concerning our experimental setup can be found in Appendix B. Lastly, related work can be found in Appendix E.

2 Results

2.1 Optimality

We analyze the optimality of Lagrangian methods through a visualization of the trade-off between the return and cost under different values of the multiplier λ. For the creation of these profiles we train PPO-Lag agents for 10^7 timesteps with manually fixed values of λ across a wide range (Ji et al., 2024b). Note that an agent trained for $\lambda = 0$ is a standard PPO-agent (Schulman et al., 2017). For each value of λ, we plot the return and cost during the last 5% of training iterations. The results are presented in Fig. 1, and clearly show that performance is highly sensitive to the choice of λ. The optimal trade-off location is found at the evaluated cost limit of 25. As shown in the figure, the optimal multiplier value λ^*, indicated by the ☆, varies across tasks. This highlights an important insight: the optimal value is inherently task-dependent and visualizing the λ-profiles does not provide a general intuition for selecting λ in practice. Looking at values of λ on the right side of λ^* ($\lambda > \lambda^*$), the curves for the return drop further down, which indicates that only values close to the optimal trade-off location yield high-return solutions. For comparison, we also train the GA-update of λ under the same cost limit. In Fig. 1, the dashed horizontal lines indicate the return and cost values at the end of training for the GA-updated multiplier, averaged over the last 5% of the training. These dashed horizontal lines match the ☆ for the Circle1 and Goal1 tasks, indicating that the GA-update successfully recovers the same optimum as the manually fixed λ in these tasks. This demonstrates that automatic updates are able to match the performance of carefully tuned fixed multipliers. For the Button1 and Push1 tasks, the GA-updates surprisingly even exceed the fixed-optimum solutions: the dashed lines for the return in these tasks lie above the ☆.

We refer to Appendix C.1 for the learning trajectories of GA-updated multipliers. The learning trajectories seem to converge to a stable solution in roughly the same amount of timesteps. In contrast, models trained with a fixed optimal multiplier show a more gradual increase in return, exhibiting more conservative behavior with respect to safety during training. This finding indicates that when replacing a fixed multiplier with an automated update mechanism, the learning trajectory fundamentally changes: it initially prioritizes reward maximization before subsequently correcting back into the constraint region, which could explain the increase in performance for the Button1 and Push1 tasks.



(a) (b) (c) (d)

Figure 1: Smoothed profiles of λ, averaged over 10 seeds. The ☆ indicate the return and cost for a manually fixed value of λ at the intersection of the cost curve with the cost limit of 25, which is λ^*. Across all tasks, λ^* varies and the return decreases for $\lambda > \lambda^*$, showing that only values close to ☆ yield high-return solutions. The dashed horizontal lines indicate the return and cost values at the end of training for the GA-updated multiplier, averaged over the last 5% of training (for the training curves, see Appendix C.1). For the Circle1 and Goal1 tasks, the GA-update recovers the same optimum as the manually fixed λ (matching the ☆), while for the Button1 and Push1 tasks, it even exceeds the performance achieved with the fixed multiplier.

2.2 Stability

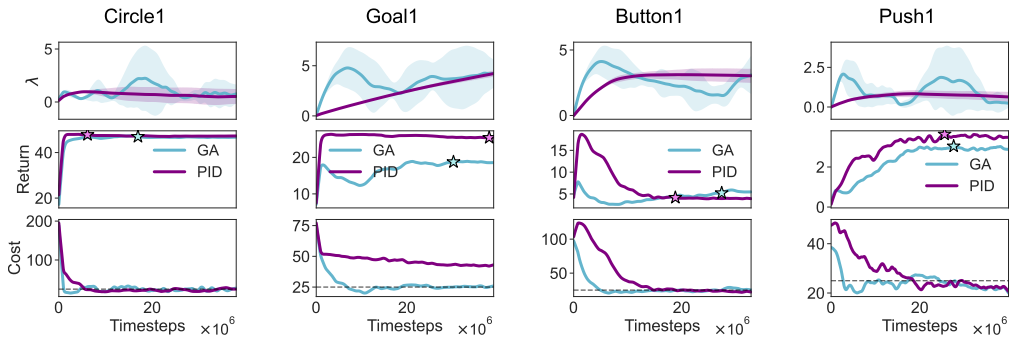
We analyze the stability of automated update rules by comparing GA-updates with PID-controlled updates of the Lagrange multiplier, training models for sufficiently long horizons to capture their oscillatory behavior. We use PI-control ($K_P = 10^{-4}, K_I = 10^{-4}, K_D = 0.0$), which reduces complexity in analyzing the update behavior. As shown in the top plots of Fig. 2, GA-updates exhibit notable oscillations in λ during training. In contrast, PI-controlled updates result in more stable update trajectories. However, this stability does not always translate into fewer constraint violations during training compared to GA-updates based solely on integral control as shown in Table C.1 in Appendix C.4. For the Goal1 and Button1 task, GA-updates result in less constraint violations after the first time reaching the cost limit. Moreover, the best returns reported in this table do not differ substantially between GA and PID. The table reveals that PID-control is not always objectively the better choice for updating the Lagrange multiplier and can thus not be used as a plug-and-play method in practice. PID-control exceeds performance for the Circle1 and Push1 task, but performs worse than the GA-update for the other tasks, where for the Goal1 task the agent does not even manage to reach the cost limit. For a visualization focused on the training curves close to the cost limit, we refer to the Appendix C.4 with zoomed-in plots of the training curves in Fig. 2.

3 Discussion and conclusion

In this section, we discuss our main findings, outline the primary limitations of our study, and provide recommendations for future research. Additional limitations are detailed in Appendix D. Finally, we conclude with a summary of our key contributions.

Across the four benchmark tasks, we observe that performance is highly dependent on the value of the multiplier λ. The λ-profiles introduced in Fig. 1 offer a complete visualization of the trade-off between the return and cost, and further demonstrate that the optimal multiplier value is task-dependent. This non-transferability of the optimal trade-off location of λ across different environments is additionally illustrated in the Appendix C.2. Although most work in SRL is still not entirely scale-invariant, Stooke et al. (2020) addressed the issue of varying reward magnitudes across environments by introducing a reward-scale invariant policy gradient. This also highlights one of the primary limitations of our study: all experiments were conducted without applying reward-scale invariance. Consequently, this emphasizes that the sensitivity of Lagrangian methods to the absolute magnitudes of rewards and costs must be carefully considered when tuning λ in practice. Future work should investigate whether λ-profiles across different tasks become more consistent when reward-scale invariance is applied.

We found that both a manually fixed λ and a GA–updated λ lead to convergence toward the same performance in the less complex tasks we evaluated. This empirical observation supports the theoretical convergence result by Borkar (2005), who proved that gradient ascent on the multiplier converges to the optimal value λ^*. While Borkar's analysis was derived under certain assumptions on



(a) (b) (c) (d)

Figure 2: Smoothed training curves of λ (top), return (middle) and cost (bottom) for a GA-updated λ and for PID-updated λ across training, averaged across 6 seeds. $K_P = 10^{-4}, K_I = 10^{-4}, K_D = 0.0$. The ☆ indicate the point of best return after the cost constraint of 25.0 is first satisfied, considering only those time steps where the cumulative cost remains below the specified cost limit. The GA-update method exhibits more oscillations in the updates in λ, whereas PID shows more stability throughout the training process.

the stochastic approximation conditions, our experiments confirm similar convergence behavior in practice. Surprisingly, automated update methods appear to perform favorably in these more complex tasks, as was shown for the Button1 and Push1 environments in Appendix C.1. Future work should investigate whether this advantage is a fundamental property of automated multiplier updates.

We further emphasize that the original motivation behind PID-controlled updates was to reduce constraint violations during training. However, one might question whether this should indeed be the primary point of improvement for Lagrangian methods, as these algorithms are inherently designed to enforce constraints only asymptotically. The observation that automated update methods exhibit distinctly different and less conservative learning trajectories compared to those with a fixed optimal multiplier can, in fact, be advantageous: such behavior may lead to higher peak performance during training, which can then be leveraged by selecting the best-performing model for deployment. Lagrangian methods should be regarded as approaches that asymptotically converge toward an optimal trade-off between return and cost, rather than as inherently safe methods.

This work provided a systematic empirical analysis of the Lagrange multiplier's role in SRL, focusing on optimality and stability. We introduced λ-profiles that visualize the trade-off between return and cost, showing how the penalty parameter influences performance. These profiles confirm that Lagrangian methods are highly sensitive to λ and that the optimal value λ^* is highly dependent on a given task. We furthermore confirmed the usefulness of automated multiplier updates practice. In particular beyond self-tuning benefits, we showed that the solution they arrive to is on-par or even exceeds the performance of manually fixed value λ^*. We showed that this increase in performance can be attributed to the fact that automatically updating λ during training leads to a less conservative learning trajectory compared to keeping the multiplier at the fixed optimal value during training. Lastly, we analyzed the stability of multiplier updates under GA and PI-control strategies. We observed that GA-based updates are less stable than PI–controlled updates. Although PI-control provides smoother adjustments to the multiplier, it did not consistently perform better than GA-updates across all tasks. Moreover, PID-controlled updates introduce three additional hyperparameters, K_P, K_I and K_D, that still require careful tuning to guarantee the method's stability. In future work we plan to systematically analyze the hyperparameter sensitivity of Lagrangian methods. The pursuit of stable multiplier updates remains an open challenge, and attempts in addressing this instability must be approached with care to avoid merely *shifting* the problem to new forms of instability, rather than *solving* the problem.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. doi:10.48550/arXiv.1705.10528 arXiv:1705.10528.

Eitan Altman. 1999. *Constrained Markov Decision Processes*. Routledge, New York. doi:10.1201/9781315140223

Richard Bellman. 1957. A Markovian Decision Process. *Journal of Mathematics and Mechanics* 6, 5 (1957), 679–684. https://www.jstor.org/stable/24900506 Publisher: Indiana University Mathematics Department.

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

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

Stephen P. Boyd and Lieven Vandenberghe. 2023. *Convex optimization* (version 29 ed.). Cambridge University Press, Cambridge New York Melbourne New Delhi Singapore.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. 2018. A Lyapunov-based Approach to Safe Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/hash/4fe5149039b52765bde64beb9f674940-Abstract.html

Dongsheng Ding, Chen-Yu Wei, Kaiqing Zhang, and Alejandro Ribeiro. 2024. Last-Iterate Convergent Policy Gradient Primal-Dual Methods for Constrained MDPs. doi:10.48550/arXiv.2306.11700 arXiv:2306.11700.

Dongsheng Ding, Xiaohan Wei, Zhuoran Yang, Zhaoran Wang, and Mihailo Jovanovic. 2023. Provably Efficient Generalized Lagrangian Policy Optimization for Safe Multi-Agent Reinforcement Learning. In *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*. PMLR, 315–332. https://proceedings.mlr.press/v211/ding23a.html ISSN: 2640-3498.

Javier Garcıa and Fernando Fernandez. 2015. A Comprehensive Survey on Safe Reinforcement Learning. (2015).

Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. 2020. Learning to Walk in the Real World with Minimal Human Effort. arXiv:2002.08550 [cs.RO] https://arxiv.org/abs/2002.08550

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

Weidong Huang, Jiaming Ji, Chunhe Xia, Borong Zhang, and Yaodong Yang. 2024. SafeDreamer: Safe Reinforcement Learning with World Models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=tsE5HLYtYg

Ashish Kumar Jayant and Shalabh Bhatnagar. 2022. Model-based Safe Deep Reinforcement Learning via a Constrained Proximal Policy Optimization Algorithm. doi:10.48550/arXiv.2210.07573 arXiv:2210.07573.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Juntao Dai, and Yaodong Yang. 2024a. Safety-Gymnasium: A Unified Safe Reinforcement Learning Benchmark. doi:10.48550/arXiv.2310.12567 arXiv:2310.12567.

Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang. 2024b. OmniSafe: An Infrastructure for Accelerating Safe Reinforcement Learning Research. (2024).

Yongshuai Liu, Jiaxin Ding, and Xin Liu. 2020. IPO: Interior-Point Policy Optimization under Constraints. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (April 2020), 4940–4947. doi:10.1609/aaai.v34i04.5932

Santiago Paternain, Miguel Calvo-Fullana, Luiz F. O. Chamon, and Alejandro Ribeiro. 2022. Safe Policies for Reinforcement Learning via Primal-Dual Methods. doi:10.48550/arXiv.1911.09101 arXiv:1911.09101.

Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. 2019. Constrained Reinforcement Learning Has Zero Duality Gap. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/hash/c1aeb6517a1c7f33514f7ff69047e74e-Abstract.html

John Platt and Alan Barr. 1988. Constrained Differential Optimization for Neural Networks. (Jan. 1988).

Asha Ramanujam, Adam Elyoumi, Hao Chen, Sai Madhukiran Kompalli, Akshdeep Singh Ahluwalia, Shraman Pal, Dimitri J. Papageorgiou, and Can Li. 2025. SafeOR-Gym: A Benchmark Suite for Safe Reinforcement Learning Algorithms on Practical Operations Research Problems. doi:10.48550/arXiv.2506.02255 arXiv:2506.02255.

Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning. (2019).

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG] https://arxiv.org/abs/1707.06347

Adam Stooke, Joshua Achiam, and Pieter Abbeel. 2020. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 9133–9143. https://proceedings.mlr.press/v119/stooke20a.html ISSN: 2640-3498.

Richard S Sutton and Andrew G Barto. 2018. Reinforcement Learning: An Introduction. (2018).

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. 2019. Reward Constrained Policy Optimization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SkfrvsA9FX

Tristan Tomilin, Meng Fang, and Mykola Pechenizkiy. 2025. HASARD: A Benchmark for Vision-Based Safe Reinforcement Learning in Embodied Agents. arXiv:2503.08241 [cs.AI] https://arxiv.org/abs/2503.08241

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. 2020. Projection-Based Constrained Policy Optimization. doi:10.48550/arXiv.2010.03152 arXiv:2010.03152.

Acknowledgments and Disclosure of Funding

This work was supported by Shell Information Technology International Limited and the Netherlands Enterprise Agency under the grant PPS23-3-03529461.

A Theoretical background

A.1 Constrained markov decision process

Reinforcement learning is a machine learning approach for sequential decision-making, where an agent learns by interacting with an environment: it takes actions based on the current state, receives feedback in the form of rewards, and iteratively improves its strategy, described by a policy, to achieve optimal performance (Sutton and Barto, 2018). When the set of feasible policies is restricted by 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 $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})$ as the return of a trajectory $\tau = (s_0, a_0, s_1, ...) \sim \pi_\theta$. The state value function for the return is defined as $V^R_{\pi_\theta}(s) \doteq \mathbb{E}_{\tau\sim\pi_\theta}\big[R(\tau)|s_0 = s\big]$ and yields the return objective of the CMDP, which is to maximize the cumulative discounted reward $J^R(\pi_\theta)$, which is equivalent to $V^R_{\pi_\theta}(s)$. The expected 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, $V^{C_i}_{\pi_\theta}(s) \doteq \mathbb{E}_{\tau\sim\pi_\theta}\big[C_i(\tau)|s_0 = s\big]$, with $C_i(\tau) = \sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t, s_{t+1})$. 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 $\pi^* = \arg\max_{\pi_\theta \in \Pi_\mathcal{C}} J^R(\pi_\theta)$.

A.2 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. If strict enforcement of constraints during training is not required, we can move to the dual problem following the Lagrangian method. This method converts a CMDP into an unconstrained one with Lagrange relaxation (Bertsekas, 1997; Boyd and 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.

[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.

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).

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 and Vandenberghe, 2023). On a fast time-scale the unconstrained problem in Eq. 2 is solved to update θ. Then, on a slower time-scale, 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 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 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 and 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 and Barr, 1988). 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.

An important consideration when motivating the use of Lagrangian methods in practical safe RL is whether minimizing constraint violations during training should be the primary objective. While the pursuit of greater stability and robustness is valuable, it is worth recalling that Lagrangian methods are not strict constraint enforcers by design. Rather, they aim to approach the cost limit asymptotically. This raises the question of whether emphasis should be placed on minimizing constraint violations during training, or instead on selecting the best model parameters that yielded the highest performance during training, and use these for deployment *after* training. Therefore, for the purpose of this study, the PID-controlled update method is evaluated on stability throughout training, and the peak performance it managed to achieve during training is compared with that of the GA-update method.

B Experimental setup

We aim to investigate the relationship between the Lagrange multiplier and the resulting model performance in terms of return and cost. Therefore, we train models with a fixed multiplier value across a wide range of λ settings. For each fixed value, we record the corresponding performance and visualize all results together as a λ-profile for each task. These profiles reveal how sensitively both return and cost depend on the choice of λ. We further compare these results with those obtained from models using automatically updated multipliers, allowing us to assess which approach yields better performance for each task. To provide additional insight into the learning dynamics, we also plot the training curves for all evaluated methods.

All experiments are conducted with the Omnisafe benchmark suite (Ji et al., 2024b), in which we evaluate Lagrangian methods across four benchmark tasks from the Safety Gymnasium task suite (Ji et al., 2024a), each involving safe navigation. We train the Lagrangian version of PPO (Ray et al., 2019; Schulman et al., 2017): PPO-Lag is employed for the GA-update of the Lagrange multiplier, and CPPO-PID for the PID-controlled update (Stooke et al., 2020). As mentioned before, Lagrangian methods generalize to any standard RL algorithm, including both on-policy and off-policy settings. However, results could potentially empirically differ per applied RL algorithm. We evaluate on the level-1 Circle, Goal, Button, and Push tasks, denoted throughout the paper as Circle1, Goal1, Button1 and Push1, referred in order of increasing task- and constraint-complexity, with Circle1 being the least complex and Push1 being the most complex. The environments use the Point agent, a robot constrained to a 2D plane with two actuators: a rotational action that controls the angular velocity of the agent around the z-axis, and an action that applies force to the agent for forward/backward movement along the agent's facing direction. Code: `https://github.com/lindsayspoor/Lagrangian_SafeRL`.

B.1 Hyperparameters

The hyperparameters used for all of the experiments in the paper are shown in Table B.1. The manually fixed values for the Lagrange multiplier λ used to create the λ-profiles visualized in Fig. 1 in the paper are shown in Table B.2 for all benchmark tasks.

Table B.1: Hyperparameter settings for GA (PPO-Lag) and PID (CPPO-PID).

Parameter	Value
General PPO Settings	
Steps / epoch	20,000
Update iterations / epoch	40
Batch size	64
Clip ratio	0.2
Target KL	0.02
Entropy coefficient	0.0
Advantage estimation	GAE
γ_r, γ_c	0.99
GAE λ_r, λ_c	0.95
Actor network	[64, 64], tanh
Critic network	[64, 64], tanh
Actor & Critic LR	3×10^{-4}
Lagrangian Update	
λ init	0.001
GA Update (PPO-Lag)	
η	0.035
PID Update (CPPO-PID)	
PID d-delay	10
PID $\Delta p, \Delta d$ EMA α	0.95
Penalty max	100
K_P	10^{-4}
K_I	10^{-4}
K_D	0.0

Table B.2: Values of λ evaluated across all 4 different tasks for constructing the λ-profile plots (Fig. 1 in the paper).

Task	Values
Circle1	0.0, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.7, 1, 1.15, 1.3, 1.4, 1.5, 1.6, 1.7, 1.9, 2
Goal1	0.0, 0.03, 0.1, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.3, 4.5, 4.7, 5, 5.3, 5.5, 5.7, 6, 6.3, 6.5, 6.7, 7, 7.3, 7.5
Button1	0.0, 0.03, 0.1, 0.3, 0.5, 0.7, 1, 1.3, 1.5, 1.7, 2, 2.5, 2.7, 3, 3.3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8
Push1	0.0, 0.01, 0.05, 0.1, 0.15, 0.3, 0.4, 0.45, 0.5, 0.55, 0.6, 0.7, 0.8, 0.9, 1, 1.15, 1.3, 1.4, 1.5, 1.7, 1.9, 2, 2.3, 2.5, 3

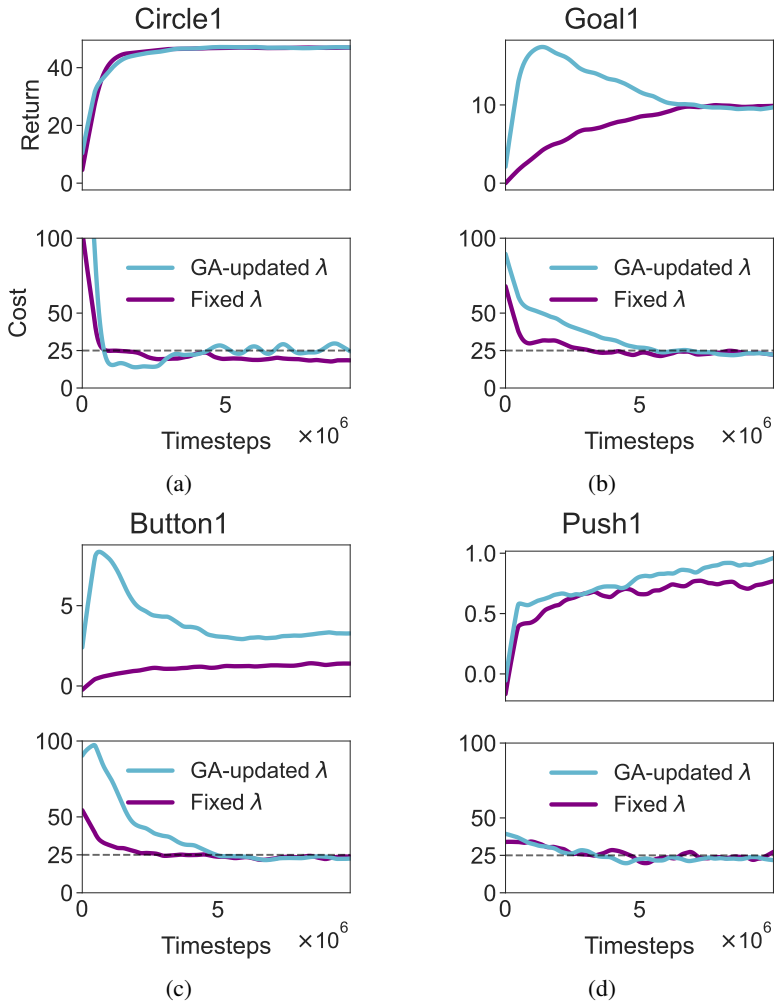


Figure C.1: Smoothed training curves of the return (top) and cost (bottom) for a GA-updated λ and for a manually fixed λ^* across training, averaged across 10 seeds. The training curves for the fixed value of λ correspond to the model trained at the optimal λ^*, indicated by the ☆ in Fig. 1. The learning trajectory for the manually fixed λ evolves more gradually and conservatively compared to the GA-updated λ. Note that the training curves for the Push1 task do not show full convergence at the end of the depicted training iterations.

C Additional results

C.1 Learning trajectories

The learning trajectories for all four tasks are shown in Fig. C.1. The learning trajectory for the manually fixed λ updates more gradually and conservatively as compared to the GA-updated λ. Note that the training curves for the Push1 task do not show full convergence at the end of the depicted training iterates.

C.2 Task dependencies

Fig. C.2 shows the optimal multiplier value λ^* for all 4 benchmark tasks and demonstrates that the optimal multiplier value is not task-agnostic across our evaluated tasks. Note that the multiplier increases with task complexity, however, for the Push1 task (most complex), the value of λ^* drops again.

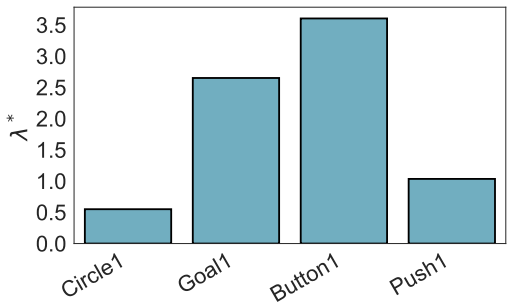


Figure C.2: Optimal multiplier value λ^* for all 4 evaluated benchmark tasks. The optimal value gradually increases with task complexity, except for the Push1 task, which is considered to be the most complex task out of the four. This shows that there is no general intuition on the underlying relation between the optimal multiplier λ^* and the task complexity, and that there is no such thing as a task-agnostic value for λ^*.

Table C.1: Best return and and percentage of timesteps having cost violations after the first time hitting the cost limit of 25.0 for GA- and PID-controlled updates. PID has no results for the Goal1 task since the cost limit remains unreached for this task. The GA- and PID-controlled updates achieve similar returns, but the PID-controlled method shows greater variation in cost violations and does not outperform the GA-updated approach for the Goal1 and Button1 tasks.

	GA		**PID**	
Task	**Best Return**	**Cost Violations**	**Best Return**	**Cost Violations**
Circle1	46.99	44.7%	**47.82**	**17.3%**
Goal1	**18.76**	**47.3%**	–	–
Button1	**5.23**	**24.4%**	4.17	26.6%
Push1	3.04	20.8%	**3.63**	**2.2%**

C.3 Return-cost dependencies

The GA-updated multiplier was evaluated for various cost limits $\in [10, 25, 50, 100, 150, 200]$. Fig. C.3 shows the return-cost curves for all 4 benchmark tasks. Where the line for the GA-updated λ lies above the line for the manually fixed multiplier, the best performance in terms of return is higher than for a manually fixed multiplier. The training curves for the different cost limits that the models are trained on are shown in Fig. C.4.

C.4 Stability near the cost limit

Fig. C.5 shows the training curves for GA-updates and PID-controlled updates with $K_P = 10^{-4}, K_I = 10^{-4}, K_D = 0.0$ (PI-control specifically) for the region of training timesteps in which the cost curves are close to the cost limit of 25.

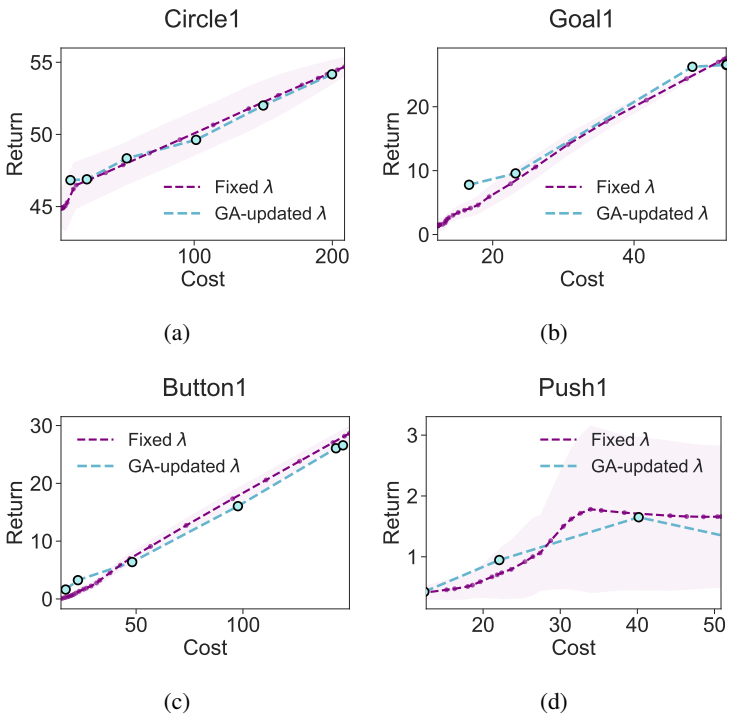


Figure C.3: Return-cost curves for all 4 benchmark tasks with a GA-updated multiplier, averaged over 10 seeds. The return and cost values for the fixed multiplier λ correspond with the last 5% of training iterates, and each data point on the fixed λ line corresponds with one of the values λ was manually kept fixed at. Each data point on the line for the GA-updated multiplier corresponds with a different cost limit set as a target for the penalty loss term during training, which corresponds to the averaged last 5% of the training curves shown in Fig. C.4. Even though these plots illustrate the relationship between two conflicting objectives, none of them exhibit a clear Pareto frontier, which is typically characteristic of multi-objective optimization problems.

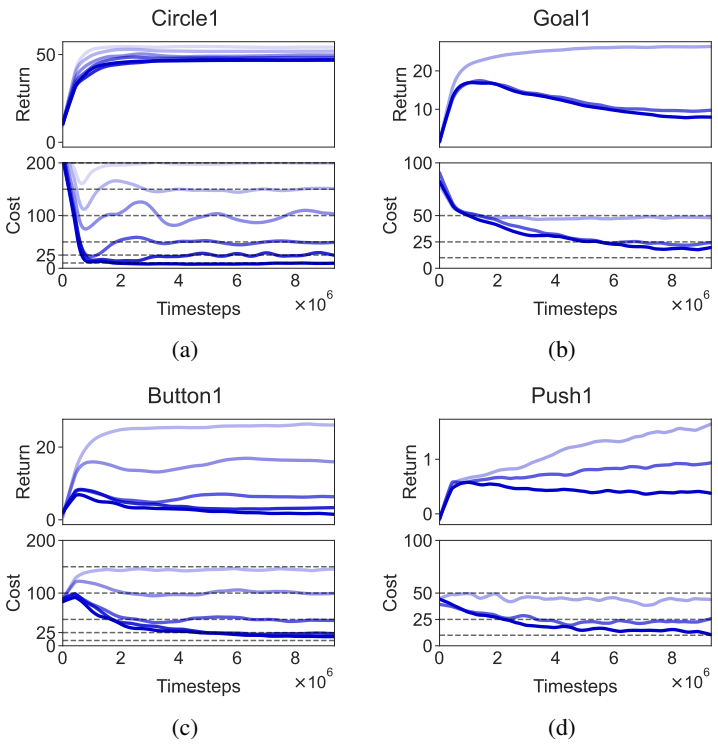


Figure C.4: Smoothed training curves for the return and cost of all four benchmark tasks, averaged across 10 seeds, for different cost limits. The darker the plotted lines, the lower the cost limit. All cost curves approach their respective cost limits, which generally leads the corresponding return curves to converge toward lower overall returns. Note that the return curve corresponding with the cost limit of 50 for the Push1 task does not seem to have converged at the end of training.

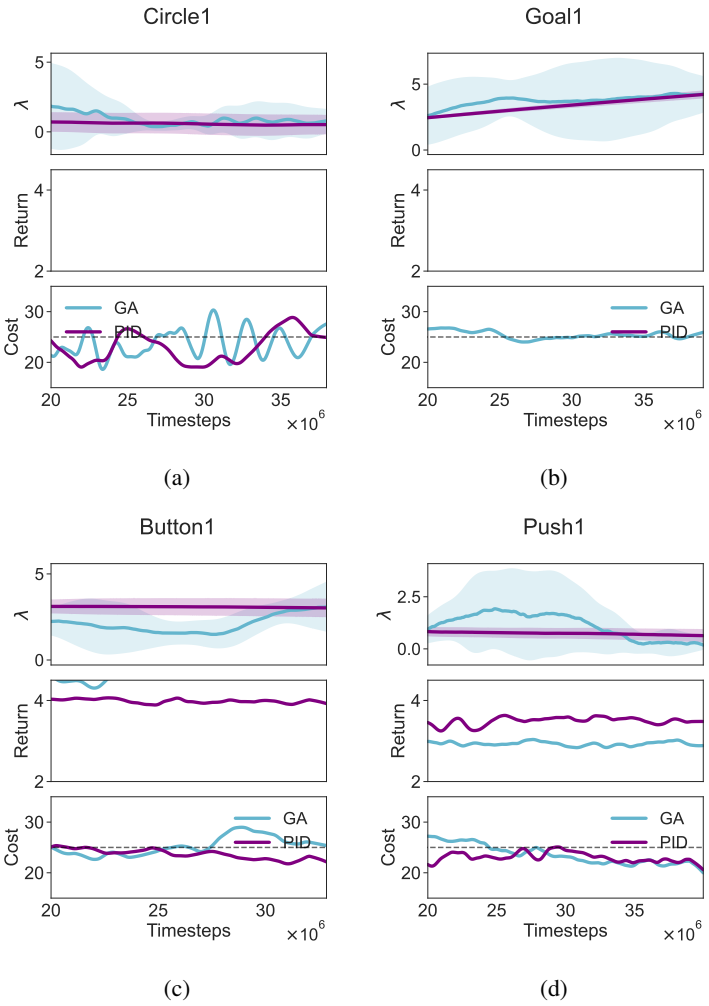


Figure C.5: Smoothed training curves of the Lagrange multiplier (top), return (middle) and cost (bottom) for a GA-updated λ and for PID-updated λ across training, for all 4 tasks, averaged across 6 seeds. These plots show the training curves from timestep $20 \cdot 10^6$ onward. Note that, as $K_P = 10^{-4}, K_I = 10^{-4}, K_D = 0.0$, the PID-curve in these plots actually only exhibit PI-control. Compared to the GA-update method, the PID-controlled updates exhibit much steadier λ trajectories across all tasks; however, both methods display instability in the cost curves near the cost limit.

D Additional limitations

First, we note that our experiments were limited to four level-1 navigation tasks from the Safety Gymnasium suite, all featuring similar collision-avoidance mechanisms. While future work should extend this analysis to other domains such as locomotion tasks (Ji et al., 2024a), visual safety tasks (Tomilin et al., 2025), or operational control benchmarks (Ramanujam et al., 2025), the selected tasks already capture a meaningful range of complexity. Moreover, Safety Gymnasium remains one of the most widely used and standardized benchmarks for evaluating SRL algorithms.

An additional limitation is highlighted by examining how variations in the cost limit influence the relationship between return and cost. As shown in Appendix C.3, the evaluated tasks exhibit an almost linear return–cost relationship, lacking a clear Pareto frontier that would typically be expected in a multi-objective trade-off. It remains unclear whether this behavior arises from the specific design of the reward and cost functions in the selected tasks.

We also note that our experiments on PID-controlled updates were conducted using only a single configuration of K_P, K_I, and K_D, where the derivative component was set to zero. While Stooke et al. (2020) provided a more systematic investigation into the individual influence of each of these hyperparameters, interactions among all three parameters when configured jointly remain unexplored. As a result, there remain open questions regarding how these parameters should be selected in practice to ensure both stability and strong performance, and whether there exists a general relationship or invariant configuration that holds across tasks.

E Related work

Lagrangian methods in safe RL

The underlying idea of Lagrangian methods is to transform the primal constrained optimization problem into its dual form using a Lagrangian relaxation. In 2005, Borkar (2005) formalized this perspective by introducing the dual gradient descent framework for actor–critic methods and showing 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.

In 2018, Tessler et al. introduced RCPO, a Lagrangian-based algorithm that updates the multiplier through gradient ascent (Tessler et al., 2019). Several works have since explored extensions of Lagrangian methods. Ding et al. extended the Lagrangian framework to multi-agent RL (Ding et al., 2023), and they furthermore proposed a regularized Lagrangian framework to guarantee safety beyond the asymptotic convergence (Ding et al., 2024). Jayant and 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.

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.

While prior research has advanced both the theoretical and algorithmic foundations of Lagrangian methods, empirical investigations into the behavior and sensitivity of the Lagrange multiplier remain limited. To our knowledge, no prior work has systematically characterized the empirical behavior of λ in Lagrangian methods. This work addresses this gap by providing a detailed empirical analysis of λ in Lagrangian methods, comparing fixed values of λ with gradient ascent and PID-controlled updates.

Multi-objective RL

Constrained RL is closely related to multi-objective RL, as both involve balancing multiple objectives. However, as 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, which has no direct equivalent in multi-objective optimization.