State-Dependent Lagrange Multipliers for State-Wise Safety in Constrained Reinforcement Learning

Minseok Seo and Kyunghwan Choi

Cho Chun Shik Graduate School of Mobility, KAIST
{seominseok,kh.choi}@kaist.ac.kr
https://kaist-mic-lab.github.io

Abstract. Despite the remarkable success of deep reinforcement learning (RL) across various domains, its deployment in the real world remains limited due to safety concerns. To address this challenge, constrained RL has been extensively studied as an approach for learning safe policies while maintaining performance. However, since constrained RL enforces constraints in the form of cumulative costs, it cannot guarantee state-wise safety. In this paper, we extend the Lagrangian based approach, a representative method in constrained RL, by introducing state-dependent Lagrange multipliers so that the policy is trained to account for state-wise safety. Our results show that the proposed method enables more fine-grained specification of the constraints and allows the policy to satisfy them more effectively by employing state-dependent Lagrange multipliers instead of a single scalar multiplier.

1 INTRODUCTION

Reinforcement learning (RL) learns a policy that maximizes rewards through trial and error. Although this learning paradigm appears simple and straightforward, it has proven highly effective. This effectiveness has been clearly demonstrated in practice. Over the past few years, RL has achieved remarkable success across diverse applications [15], [4], [12], [10]. Nevertheless, deploying RL in physical real-world environments remains a major challenge. To deploy RL-trained agents in real-world environments, two requirements must be satisfied: first, the agent must be able to successfully accomplish the given tasks, and second, it must ensure safety and reliability. In standard RL, such requirements are learned exclusively from reward signals, thereby necessitating careful reward engineering. However, reward engineering is inherently difficult, and even with carefully designed rewards, the learned policy may exploit unintended loopholes (reward hacking) or remain hard to interpret, making it difficult to ensure safety and reliability [3]. These limitations highlight the necessity of an explicit constraint function, separated from the reward, which has motivated research on constrained RL.

Constrained RL is a method that learns a policy to maximize rewards while satisfying constraints, thus enabling agents to behave safely while successfully performing tasks. A common formulation of constrained RL specifies constraints in terms of the expected cumulative cost [5]. For instance, in a driving scenario, a constraint may be to minimize unintended lane intrusions (i.e., crossing into an adjacent lane), in which case the cumulative cost can be defined as the total number of lane intrusions. In this way, the agent can satisfy the constraint in expectation and keep the total number of intrusions within an acceptable limit. However, even a single violation at the level of an individual state—such as an unintended lane intrusion near another vehicle at a critical moment—may nevertheless lead to accidents or catastrophic failures.

This limitation highlights the need to consider state-wise constraints, which explicitly enforce constraints at the level of individual states. Accordingly, we introduce state-wise Lagrange multipliers to encourage the policy to satisfy safety requirements at every timestep. Our contributions are as follows:

- In contrast to most existing methods that utilize a single Lagrange multiplier for cumulative cost constraints, our approach addresses state-wise constraints, which inherently demand state-wise multipliers. To this end, neural networks are utilized to approximate these multipliers, enabling adaptive enforcement of safety at the level of individual states.
- The proposed method allows for finer specification of constraints and achieves better constraint satisfaction compared to existing constrained RL approaches under the same training budget.
- The learned Lagrange multiplier network can be utilized during deployment to assess state-wise safety, providing interpretability and insight into the agent's behavior.

2 Related Work

In the context of constrained RL, Liu et al. [9] survey various formulations in the CMDP setting and provide an overview of corresponding model-free algorithms. Most existing works focus on enforcing constraints based on discounted cumulative cost. Among representative methods is constrained policy optimization (CPO) [1], which uses surrogate functions to approximate both the objective and the constraints. It then employs a projection step to enforce constraint satisfaction. This procedure requires a backtracking line search,thereby making the algorithm computationally expensive. Another line of work includes Proximal Policy Optimization (PPO) Lagrangian and Trust Region Policy Optimization (TRPO) Lagrangian [11], which extend the PPO [14] and TRPO [13] algorithms to the constrained RL setting via Lagrangian relaxation, reformulating the constrained policy optimization problem as an unconstrained max-min optimization problem. By adaptively adjusting the Lagrange multiplier, these methods encourage policy updates toward satisfying the constraints. However, since constraint violations are inevitable for learning a safe policy, the constraints are typically not satisfied during training. In interior-point policy optimization (IPO) [8], logarithmic barrier functions are added to the objective as penalty terms to account for the constraints. While this approach is easy to implement and can handle multiple constraints, it assumes feasible iterates, which can be problematic in situations such as random agent initialization where constraint violations may occur. Stooke et al. [16] show that in Lagrangian-based methods, the Lagrange multiplier is updated only through integral control, resulting in oscillation and overshoot issues, and propose a method that incorporates proportional and derivative terms to stabilize the updates.

3 Methodology

3.1 Preliminaries

Problem Formulations

Markov Decision Processes In RL, problems are typically formulated as a Markov decision process (MDP) [17]. An MDP is defined as a tuple $\langle S, A, P, R, \gamma \rangle$, where S and A denote the state and action spaces, P is the transition probability, R is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The objective of RL is to find an optimal policy π^* that maximizes the cumulative reward defined as:

$$\begin{aligned} J_R(\pi) &= \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \\ \pi^* &= \arg\max J_R(\pi) \end{aligned} \tag{1}$$

Here, $\tau = (s_0, a_0, s_1, a_1, \ldots)$ denotes a trajectory under policy π.

Constrained Markov Decision Processes In contrast, constrained RL is typically formulated as a constrained Markov decision process (CMDP) [2]. A CMDP extends an MDP by introducing cost functions $C_1, \ldots, C_m$ (separate from the reward function) and their corresponding limits $d_1, \ldots, d_m$. Formally, a CMDP is defined as a tuple $\langle S, A, P, R, C, d, \gamma \rangle$. In a CMDP, the set of feasible policies Π_C is defined as:

$$\Pi_C = \{\pi \in \Pi : \forall i \in \{1, \ldots, m\},\ J_{C_i}(\pi) \leq d_i\}. \tag{2}$$

Specifically, the constraint function for constraint i is defined as the expected cumulative discounted cost:

$$J_{C_i}(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t)\right], \quad \forall i \in \{1, \ldots, m\}. \tag{3}$$

In standard RL, the objective is an unconstrained policy optimization problem that aims to find a policy maximizing the expected return, as defined in Eq. (1), whereas constrained RL seeks to maximize the expected return subject to the cost constraints defined in Eq. (3).

$$\pi^* = \arg\max_{\pi} J_R(\pi) \text{ s. t. } \pi \in \Pi_C. \tag{4}$$

State-wise Constrained Markov Decision Process The CMDP framework can be extended to incorporate cost-based constraints of different forms. One such extension is the state-wise constrained Markov decision process (SCMDP) [18], which enforces constraints that bound the expected cost at each state by a specified threshold. In a SCMDP, the set of feasible policies Π_{SC} is defined as:

$$\Pi_{SC} = \{\pi \in \Pi : \forall i \in \{1, \dots, m\},\ J_{SC_i}(\pi) \leq w_i\}. \tag{5}$$

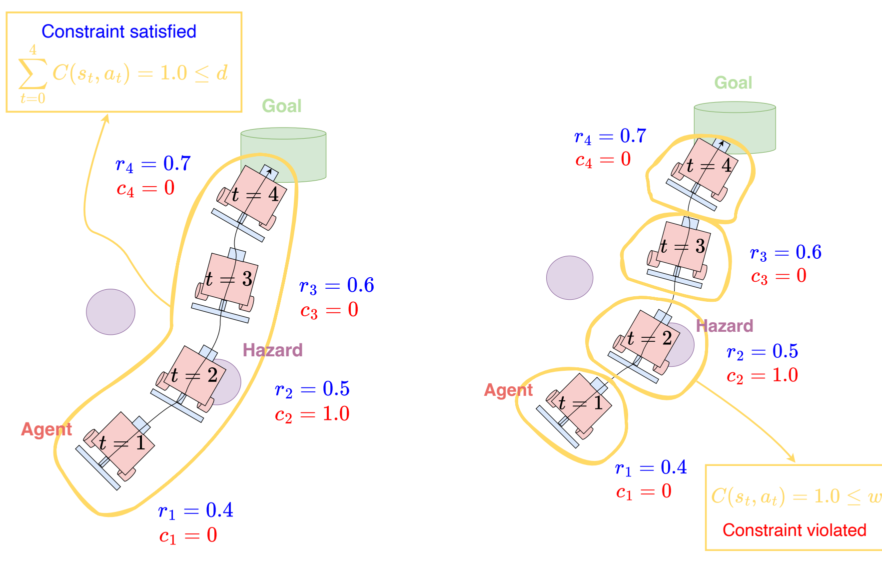
In this definition, the state-wise constraint represents the expected cost incurred at each state:

$$J_{SC_i}(\pi) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \tau,\, \tau \sim \pi}\big[C_i(s_t, a_t, s_{t+1})\big], \quad \forall i \in \{1, \dots, m\}. \tag{6}$$

Consequently, similar to CMDP, the optimization problem in SCMDP can be formulated as:

$$\pi^* = \arg\max_{\pi} J_R(\pi) \text{ s.t. } \pi \in \Pi_{SC}. \tag{7}$$

In a CMDP, constraints are imposed on the cumulative cost, whereas an SCMDP enforces constraints on the expected cost at each state. This enables safety guarantees at the state level, as illustrated in Fig. 1.



(a) Constrained RL

(b) State-wise constrained RL

Fig. 1: Comparison of constrained RL and state-wise constrained RL. In constrained RL, a policy is feasible if the cumulative cost is below the limit. In state-wise constrained RL, a policy is feasible if the cost at every state is below the limit.

Lagrangian Relaxation for Constrained Policy Optimization A common approach to solve constrained optimization problems is to use the Lagrangian relaxation method. In this approach, the constrained optimization problem (4) is reformulated as an unconstrained optimization problem by introducing Lagrange multiplier $\lambda \geq 0$ that penalizes constraint violations. The resulting Lagrangian can be written as:

$$L(\theta, \lambda) = J_R(\pi_\theta) - \lambda^\top (J_C(\pi_\theta) - d), \tag{8}$$

where θ denotes the parameter of the policy. The objective is then to find a saddle point (θ^*, λ^*) that satisfies:

$$L(\theta^*, \lambda) \geq L(\theta^*, \lambda^*) \geq L(\theta, \lambda^*). \tag{9}$$

Since finding a global saddle point is typically intractable, in practice, one seeks a locally optimal solution by iteratively updating the policy parameters and the Lagrange multipliers. A common approach is to apply gradient-based updates of the form

$$\theta_{n+1} = \theta_n + \eta_\theta \nabla_\theta \Big(J_R(\pi_\theta) - \lambda_n^\top J_C(\pi_\theta) \Big), \tag{10}$$

$$\lambda_{n+1} = \Big[\lambda_n + \eta_\lambda \big(J_C(\pi_\theta) - d \big) \Big]_+, \tag{11}$$

where $\eta_\theta, \eta_\lambda > 0$ are step sizes, and $[\cdot]_+$ denotes the projection onto the nonnegative reals to ensure $\lambda \geq 0$.

3.2 Formulations of PPO under MDP and CMDP

PPO Among policy gradient methods, PPO is one of the most widely used algorithms, proposed to solve the optimization problem in Eq. (1). To improve stability, PPO introduces a clipping mechanism that prevents large policy updates. The objective function of PPO is defined as:

$$J^{\mathrm{PPO}}(\theta) = \mathbb{E}_{\pi_{\theta_{\mathrm{old}}}} \Big[\min \Big(r(\theta) A_R^{\pi_{\theta_{\mathrm{old}}}}(s,a), \mathrm{clip}(r(\theta), 1-\epsilon, 1+\epsilon) A_R^{\pi_{\theta_{\mathrm{old}}}}(s,a) \Big) \Big] \tag{12}$$

where $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\mathrm{old}}}(a|s)}$ denotes the probability ratio between the current and the old policies, and $A_R^{\pi_{\theta_{\mathrm{old}}}}(s,a) = Q_R^{\pi_{\theta_{\mathrm{old}}}}(s,a) - V_R^{\pi_{\theta_{\mathrm{old}}}}(s)$ represents the *reward advantage function* under the old policy.

PPO Lagrangian PPO Lagrangian extends PPO to the constrained RL setting, allowing the algorithm to handle explicit constraints. The original PPO algorithm enhances stability by limiting policy updates through clipping, as defined in Eq. (12), but it does not explicitly account for constraints. To address constraints defined in terms of the cumulative cost in Eq. (3), PPO Lagrangian applies the Lagrangian relaxation technique, as discussed in Section 3.1, thereby converting the constrained optimization problem in Eq. (4) into an unconstrained one. For simplicity, we consider a single-constraint case throughout this paper, although the original formulation supports multiple constraints.

$$J^{\text{PPO-Lag}}(\theta) = \mathbb{E}_{\pi_{\theta_{\text{old}}}}\Big[\min\big(r(\theta)A_R^{\pi_{\theta_{\text{old}}}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A_R^{\pi_{\theta_{\text{old}}}}(s,a)\big) - \lambda r(\theta)A_C^{\pi_{\theta_{\text{old}}}}(s,a)\Big] \tag{13}$$

where $\lambda \geq 0$ is the Lagrange multiplier that imposes a penalty on constraint violations, and $A_C^{\pi_{\theta_{\text{old}}}}(s,a) = Q_C^{\pi_{\theta_{\text{old}}}}(s,a) - V_C^{\pi_{\theta_{\text{old}}}}(s)$ denotes the *cost advantage function* under the old policy.

3.3 Proposed Method

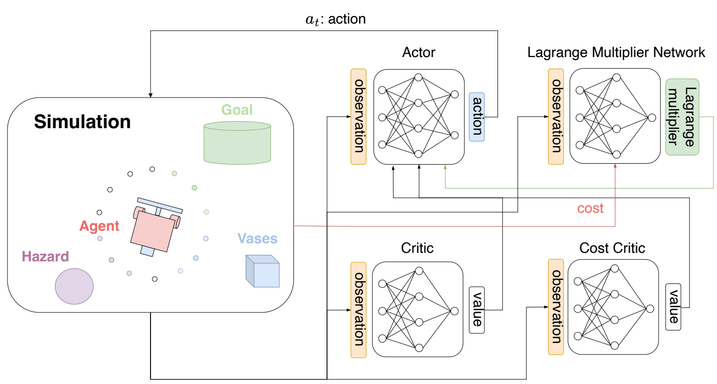


Fig. 2: Overview of the proposed method. Unlike standard PPO-Lagrangian, which employs a scalar Lagrange multiplier since the constraint is defined on cumulative cost, our approach imposes constraints in a state-wise manner, requiring state-varying multipliers. To estimate these multipliers, we introduce an additional neural network, termed the Lagrange multiplier network.

In this paper, we propose a method that extends PPO Lagrangian to the state-wise constrained RL setting by estimating state-wise Lagrange multipliers. Similar to PPO Lagrangian in Eq. (13), which employs a scalar Lagrange multiplier to handle constraints on the cumulative cost, addressing the state-wise constraint defined in Eq. (6) requires a state-varying multiplier. To this end, we introduce an additional neural network, referred to as the Lagrange multiplier network, which maps each state to its corresponding Lagrange multiplier, as illustrated in Fig. 2. Formally, the Lagrange multiplier network is parameterized by ξ and is denoted as $\lambda_\xi(s)$. This enables the application of Lagrangian relaxation to the constrained policy optimization defined in Eq. (7), analogous to the way PPO-Lagrangian handles cumulative cost constraints. The objective function of the proposed method can thus be expressed as:

$$J^{\text{Proposed}}(\theta) = \mathbb{E}_{\pi_{\theta_{\text{old}}}}\Big[\min\big(r(\theta)A^{\pi_{\theta_{\text{old}}}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A^{\pi_{\theta_{\text{old}}}}(s,a)\big) - \lambda_\xi(s)r(\theta)A_c^{\pi_{\theta_{\text{old}}}}\Big] \tag{14}$$

The parameters of the Lagrange multiplier network are updated using the following objective:

$$J_\lambda(\xi) = \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\tau, \tau\sim\pi_{\theta_{\text{old}}}}[\lambda_\xi(s_t)(C(s_t,a_t,s_{t+1}) - w)] \tag{15}$$

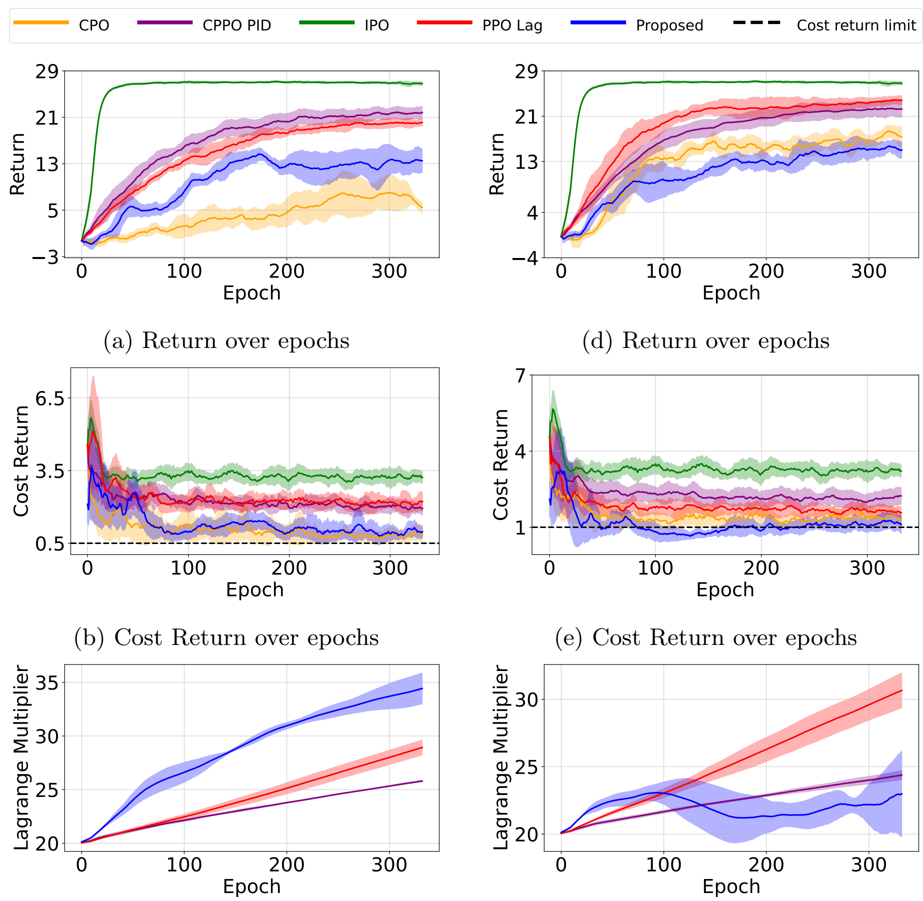
where w denotes the state-wise cost limit. In this formulation, the objective $J_\lambda(\xi)$ encourages the multiplier $\lambda_\xi(s)$ to increase when the observed cost $C(s_t, a_t, s_{t+1})$ exceeds the limit w, and to decrease when it falls below w.

4 Experiments

We evaluate our approach in the Safety Gymnasium [6] environment, focusing on the `PointGoal` task under different cost limits. Safety Gymnasium provides two types of cost definitions: a binary indicator and object-specific values. In all experiments, we adopt the object-specific cost (`constrain_indicator=False`). The proposed method is evaluated in terms of constraint satisfaction performance and reward performance by comparing it with CPO [1], PPO Lagrangian [11], IPO [8], and CPPO [16], using the implementations provided by Omnisafe [7].

4.1 Results

Figures 3 and 4 present the results under different cost limits. Each figure illustrates how return, cost return, and the Lagrange multiplier evolve during training, highlighting how varying cost limits affect the agent's performance and constraint satisfaction. Since all methods except ours define the constraint in terms of cumulative cost, the threshold for our method was set to 1/1000 of theirs (as each episode consists of 1,000 steps). From Fig. 3 (e) and Fig. 4 (b), (e), it can be observed that our proposed method consistently satisfies the constraints across different settings, unlike the other methods. When the constraint is enforced too strictly to be satisfied, as in Fig. 3 (b), none of the methods can fully satisfy it, but our proposed method and CPO come closest to satisfying the constraint, achieving much lower cost returns than the others. In particular, as shown in Fig. 3 (a) and (b), our method achieves a similar level of constraint satisfaction as CPO, while obtaining approximately 1.8 times higher return performance. However, Fig. 3 (d) suggests that the reward performance can be lower than that of some of the other methods. This is because our method applies state-wise cost constraints, unlike the baseline approaches that enforce constraints on the cumulative cost return. As a result, the policy tends to converge in a more local and conservative manner, which leads to well-satisfied constraints but potentially lower return performance.



(a) Return over epochs

(d) Return over epochs

(b) Cost Return over epochs

(e) Cost Return over epochs

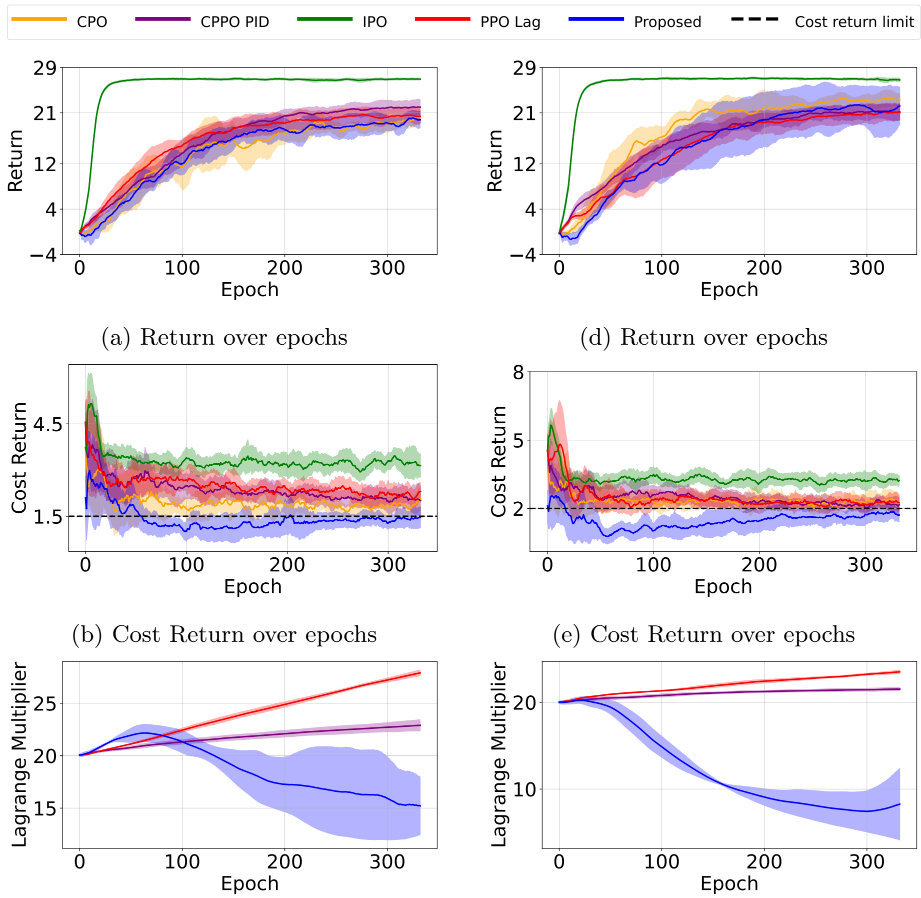
(c) Lagrange multipliers over epochs

(f) Lagrange multipliers over epochs

Cost return limit: 0.5

Cost return limit: 1

Fig. 3: Learning curves. The left column, (a)–(c), illustrates the return, cost return, and Lagrange multiplier when the cost limit is set to 0.5. The right column, (d)–(f), shows the corresponding results under a cost limit of 1.0. Each row presents a different metric, showing how the return (top), cost return (middle), and Lagrange multiplier (bottom) evolve over training epochs.



(a) Return over epochs

(b) Cost Return over epochs

(c) Lagrange multipliers over epochs

Cost return limit: 1.5

(d) Return over epochs

(e) Cost Return over epochs

(f) Lagrange multipliers over epochs

Cost return limit: 2

Fig. 4: Learning curves. The left column, (a)–(c), illustrates the return, cost return, and Lagrange multiplier when the cost limit is set to 1.5. The right column, (d)–(f), shows the corresponding results under a cost limit of 2.0. Each row presents a different metric, showing how the return (top), cost return (middle), and Lagrange multiplier (bottom) evolve over training epochs.

In addition, as shown in Fig. 3 (f) and Fig. 4 (c) and (f), when comparing the values of the Lagrange multiplier, our proposed method satisfies the constraints, resulting in bounded values or decreasing trends. An exception occurs when the cost limit is set to 0.5, which is particularly strict and difficult to satisfy, leading to continuously increasing values.

4.2 Analysis on Lagrange Multiplier

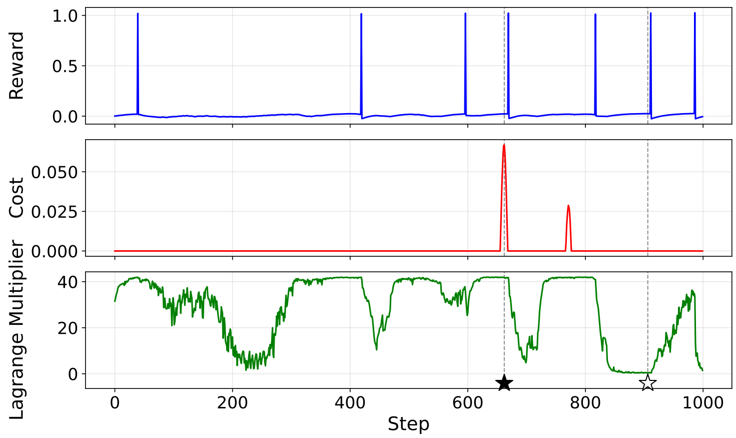
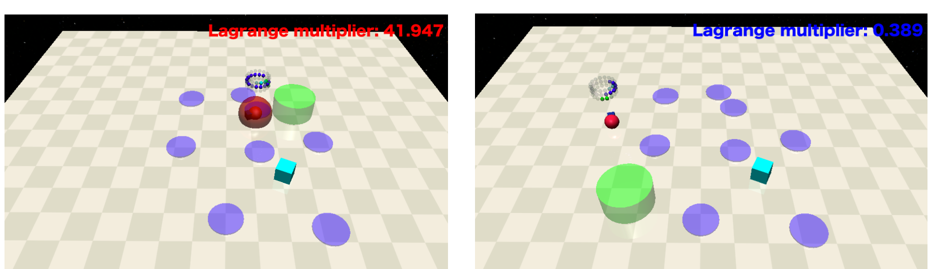


Fig. 5: Evaluation results of a single episode. The plot shows the per-step reward, cost, and the output of the learned Lagrange multiplier network, which takes the state as input and produces a state-wise Lagrange multiplier. This multiplier output increases as the agent approaches obstacles and constraint violations, while it decreases toward zero when the agent remains in safe states. Frames at the timesteps marked with stars are shown in Fig. 6.



(a) (b)

Fig. 6: Subfigure (a) shows the agent violating the constraint, where the network outputs its maximum value, while subfigure (b) shows the agent in a safe situation, where the output remains close to zero.

Figures 5 and 6 present the evaluation results of the learned Lagrange multiplier network. The network takes the state as input and produces a state-wise

Lagrange multiplier at each timestep. Figure 5 shows the per-step reward, cost, and the output of the network throughout a single episode. The output increases as the agent approaches obstacles or potential constraint violations, while decreasing toward zero when the agent remains in safe states, as as shown by the green curve in the bottom subplot of Fig. 5. Figure 6 shows simulation frames corresponding to the two starred timesteps in Fig. 5: the black star corresponds to subfigure (a), where the agent violates the constraint and the multiplier reaches its maximum value, while the white star corresponds to subfigure (b), where the agent remains in a safe state and the output stays close to zero.

5 CONCLUSIONS

In this work, we proposed a method that introduces state-wise Lagrange multipliers to learn policies that satisfy constraints defined in terms of state-wise costs. Unlike most existing approaches that address constraints based on cumulative costs, our method handles state-wise constraints, which inherently require state-dependent multipliers. To this end, neural networks are employed to approximate these multipliers, enabling the policy to more effectively satisfy constraints at the level of individual states. The proposed approach allows for finer specification of constraints and demonstrates consistent satisfaction across experiments. Moreover, the learned Lagrange multiplier network can also be utilized at deployment to assess state-wise safety, thereby providing interpretability and insight into the agent's behavior.

Building on these findings, future research could focus on several directions. One direction is to pursue deterministic safety guarantees, for instance by leveraging the learned Lagrange multiplier network together with mechanisms such as safety filters or control barrier functions. Another important direction is to improve sample efficiency, for example through off-policy or model-based approaches. Additionally, we plan to provide a theoretical discussion on the existence of saddle points, particularly in the context of state-dependent multipliers and function approximation.

Overall, this work demonstrates the potential of state-wise constrained RL and suggests that our approach provides a promising foundation for developing safer and more interpretable RL agents. These contributions are expected to inspire further research on bridging theoretical safety guarantees and practical RL applications.

Acknowledgments. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00554087).

References

1. Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: International conference on machine learning. pp. 22–31. PMLR (2017)
2. Altman, E.: Constrained Markov decision processes. Routledge (2021)
3. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., Mané, D.: Concrete problems in ai safety. arXiv preprint arXiv:1606.06565 (2016)
4. Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al.: Learning dexterous in-hand manipulation. The International Journal of Robotics Research **39**(1), 3–20 (2020)
5. Brunke, L., Greeff, M., Hall, A.W., Yuan, Z., Zhou, S., Panerati, J., Schoellig, A.P.: Safe learning in robotics: From learning-based control to safe reinforcement learning. Annual Review of Control, Robotics, and Autonomous Systems **5**(1), 411–444 (2022)
6. Ji, J., Zhang, B., Zhou, J., Pan, X., Huang, W., Sun, R., Geng, Y., Zhong, Y., Dai, J., Yang, Y.: Safety gymnasium: A unified safe reinforcement learning benchmark. In: Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2023), `https://openreview.net/forum?id=WZmlxIuIGR`
7. Ji, J., Zhou, J., Zhang, B., Dai, J., Pan, X., Sun, R., Huang, W., Geng, Y., Liu, M., Yang, Y.: Omnisafe: An infrastructure for accelerating safe reinforcement learning research. Journal of Machine Learning Research **25**(285), 1–6 (2024)
8. Liu, Y., Ding, J., Liu, X.: Ipo: Interior-point policy optimization under constraints. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 4940–4947 (2020)
9. Liu, Y., Halev, A., Liu, X.: Policy learning with constraints in model-free reinforcement learning: A survey. In: The 30th international joint conference on artificial intelligence (ijcai) (2021)
10. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. Advances in neural information processing systems **35**, 27730–27744 (2022)
11. Ray, A., Achiam, J., Amodei, D.: Benchmarking safe exploration in deep reinforcement learning. arXiv preprint arXiv:1910.01708 **7**(1), 2 (2019)
12. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al.: Mastering atari, go, chess and shogi by planning with a learned model. Nature **588**(7839), 604–609 (2020)
13. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International conference on machine learning. pp. 1889–1897. PMLR (2015)
14. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
15. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. nature **550**(7676), 354–359 (2017)
16. Stooke, A., Achiam, J., Abbeel, P.: Responsive safety in reinforcement learning by pid lagrangian methods. In: International Conference on Machine Learning. pp. 9133–9143. PMLR (2020)
17. Sutton, R.S., Barto, A.G., et al.: Reinforcement learning: An introduction, vol. 1. MIT press Cambridge (1998)
18. Zhao, W., Chen, R., Sun, Y., Wei, T., Liu, C.: State-wise constrained policy optimization. arXiv preprint arXiv:2306.12594 (2023)