Policy Gradient with Worst-Case Constraint for Safe Reinforcement Learning: Application to Safe Legged Robot Navigation

Jason J. Choi, Youngsang Suh
Outside Collaborator: Kai Yun
CS285 Fall 2021 Course Project Report, UC Berkeley
jason.choi, youngsang_suh, kaiyun799@berkeley.edu

Abstract: In this project, we revisited the fundamental formulation of Safe RL problems. In most of the existing Safe RL methods, safety violation is defined as an accumulative unsafety signal exceeding a user-specified threshold, known as the Constrained MDP (CMDP) problem. However, this cannot represent instantaneous constraint violation that is typical in many real-world safety-critical applications. We revisit the optimal control literature that is developed to solve safety problems, and derive a new CMDP formulation that captures the objective of making sure that the worst-case instance of the trajectory still satisfies the safety constraint. Since this new constraint has to evaluate a safety metric over the past history of rollout, we transform this non-Markovian constraint into a Markovian constraint by augmenting th state space with the worst-case state in history. Then, the problem transforms to a terminal-time-constrained RL, whose dual problem can be solved by existing primal-dual methods. We use a modified PPO algorithm to solve this. To evaluate our method, we compare our method to the existing Policy Gradient (PG)-based deep RL methods including the vanilla PPO, PPO with dual update on the accumulative safety constraint (PPO-Lagrangian), and CPO. We evaluate these methods on an OpenAI Safety Gym Benchmark environment (Fig.1, and a realistic simulation of a quadruped navigating an environment with cluttered obstacles (Fig.2). The main results are provided in Table 1 and 2, where we evaluate the average performance of task and safety violation on 100 episodes. The result indicates that PPO-Lagrangian still outperforms our method. We observe that our method is effective in learning safe policies in early phases of the training from fewer iterations, but this induces a more conservative exploration which eventually results in an inferior learning curve. Future work will be taylorizing the PG-based primal-dual update algorithm for the specific use case of our worst-case constrained problem.

Table 1: Evaluation result of PG-based RL algorithms with safety constraint on a modified **Point-Goal1 environment from OpenAI Safety Gym (Fig.1)**[1]. Each quantity is evaluated by averaging over 100 episodes. The rendered simulations are available at https://bit.ly/3yzba4S.

Algorithm	Return	Worst-case Distance	Cum. Binary Cost
CPO	26	0.112	39.2
PPO	**26.5**	0.131	52.2
PPO-Lagrangian	20	**0.0619**	**23.7**
PPO-Worst-Case	21.4	0.131	47.3

Table 2: Evaluation result of PG-based RL algorithms with collision constraint on **the quadruped A1's safe navigation task (Fig.2)** Each quantity is evaluated by averaging over 100 episodes. The rendered simulations are availabe at https://bit.ly/3yzba4S

Algorithm	Return	Worst-case Distance	Collision Rate	Reach-Goal Rate
PPO	392	-0.146	0.47	0.19
PPO-Lagrangian	**1012.1**	-0.185	**0.30**	**0.60**
PPO-Worst-Case	201	**-0.206**	0.42	0.06

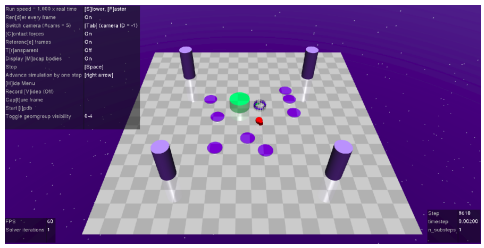

Figure 1: SafetyGym PointGoal Env.

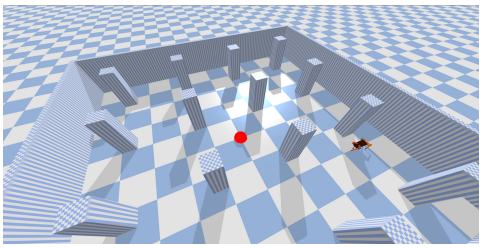

Figure 2: A1 Quadruped Navigation Env.

1 Introduction

1.1 Motivation

In Reinforcement Learning (RL), an agent acts in an environment receiving states and rewards to ultimately learn a policy that would maximize the reward in long term view. Since agents learn policies through numerous exploration and iterations, they might inevitably run into failures. However, in some cases, where physical damage is not permitted, safety of the learning-based policy should be taken into account. Safe RL learns policies by maximizing the objective function while also satisfying such safety constraints. The capability of ensuring safety constraints would enable the RL methods much more practicality for real-world applications.

The main motivation of this project was to develop a new safe RL method that can learn a more realistic safety constraint problem. We restrict our attention to policy gradient (PG)-based safe RL methods although there exist many previous works on safe RL algorithms based on other schemes. Instead of considering an accumulative cost constraint like in most of the previous literature, we are interested in satisfying an instantaneous safety criteria for the entire time span (worst-case analysis [2]). The worst case state is preserved and kept updated during the episode. Then, the worst-case cost is fed into the inequality constraint.

One of the interesting applications of safe RL methods is its usage in legged robots. Safe navigation is one of the most fundamental yet challenging tasks for legged robots due to their complicated dynamics and also their prominent usages in a lot of rugged or cluttered terrains like stairs or mountains. Especially, one of the most exciting application of legged robots is their deployment to risky and hazardous environment, as such, the safety guarantee should be a key challenge for their navigation task.

1.2 Related Work

1.2.1 Safe Reinforcement Learning

A number of previous methods have tackled learning for safe policies by formulating it as a constrained Markov Decision Process (CMDP) [3]. In this setup, the agent aims to learn a typical reward maximization problem, however, it is subjected to a constraint that is defined by a cost function specified separately from the reward objective. Thus, there are two objectives in this problem–the reward maximization and the constraint satisfaction.

Previous Safe RL methods that build on this setup mainly fall into two categories. First, many of these methods learn a separate "critics" that captures the second objective, constraint satisfaction, of the problem [4, 5, 6, 7]. This methods can benefit from value function or actor-critic based approaches in the classic deep RL literature. However, often the trained critic networks do not reliably learn the objective cost function which in this case is a severe issue since extracting wrong information from the critic would directly affect safety violation. In order to mitigate this, the community has devised various ways of reliably using this critic functions, for instance by incentivizing the critics to be more conservative [5] than its groundtruth to account for the error. However, due to these efforts, it is apparent in its nature that the safety critics suffer more from out-of-distribution

which mainly causes the unreliable learning than the critics for the typical reward objective because the constraint violation is always at stake in exploration and exploitation trade-off.

Another category of safe RL based on Constrained MDP formulations are methods that directly learn safe policy through policy gradient [8, 1]. The high-level idea behind these methods is to cast this as a constrained optimization problem, and formulate a dual problem and solve it by doing primal-dual updates. Also, it can benefit from the recent success in trust-region-based methods [9] to more deliberately update the policies based on its KL divergence. These methods often can be used in conjunction with the critics for the safety cost [10]. In [1], OpenAI has released a set of benchmark environments for the evaluation of safe RL algorithms and has shown that PPO-Lagrangian (PPO [10] combined with the dual update on the lagrangian multiplier for the safety cost) achieves the best safety performance. Finally, a recent work has extended the policy gradient method to using an augmented-lagrangian method [11], and has shown that it induces smaller variance during the training.

1.2.2 Worst-case Analysis and Hamilton-Jacobi Reachability

In the optimal control literature, worst-case analysis has a rich history stemming from the early works of [2, 12]. It is shown that Hamilton-Jacobi(HJ) reachability analysis, which formulates the reachability of a target set as an optimal control problem, is equivalently solving the worst-case safety satisfaction problem when the target set is an unsafe set [12, 13]. This problem can be solved with high-precision by solving the resulting HJ Partial Differential Equations (PDE) numerically [14], which can provide safety critics (as in the form of the value function) and the optimal safety controllers together. This method is highly reliable and has been applied to many safety-critical scenarios [15], however, it suffers from the curse of dimensionality because solving the HJ PDE requires gridding up the state space. Recently, Deep learning and Reinforcement Learning are exploited to approximately solve this problem [16, 17]. Granted that these methods show a great promise that Hamilton-Jacobi based methods can be merged nicely with Deep RL, these methods again often suffer from the value function's reliability problem mentioned earlier in Sec.1.2.1.

1.2.3 Legged Robot Navigation

Finally, there are many recent works that leverage on learning rich policies from Deep RL for legged robots to accomplish challenging navigation tasks [18, 19]. Because of the model complexity of legged robots, it is shown recently that Deep RL methods can often outperform classic model-based approaches for solving their locomotion problem [20] Due to their versatile mobility in rugged terrains, the legged robot community often aims at deploying them to a more cluttered or rugged environments [21, 22] where safety is more at stake.

2 Our Methods

2.1 Problem Formulation

The classic safe RL problem based on the Constrained Markov Decision Process model is defined as below:

$$\begin{aligned} \theta^* &= \arg\max_{\theta} \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_t \gamma^t r(s_t, a_t)\right] \\ \text{s.t.} \quad & \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_t \gamma^t c(s_t, a_t)\right] \leq d. \end{aligned} \tag{1}$$

Note that the constraint (1) captures the accumulation of a discounted constraint cost staying under some threshold. However, this formulation in general cannot capture many real-world system failures that instantaneously occur like collisions. An example is highlighted in [11] where the policy that satisfies a cumulative constraint in (1) stills violate safety instantaneously. Instead of (1), we

will consider the notion of safety by examining the worst-case instance–the system is safe if it does not violate the desired constraint even at the worst case. Define $l(s)$ as the safety signal that captures the level of safety. Without loss of generality, we set $l(s) \geq 0$ when s enters an unsafe region. A typical l can be a signed distance to an obstacle. The failure of the system is then defined as the state and action sequence $\{(s_t)\}_{t=0}^{\infty}$ ever entering the unsafe region, i.e. $\exists t$ s.t. $l(s_t) < 0$. The safety objective which takes into account the probabilistic nature of the MDP can be defined as the following way.

$$\mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\max_{t \in [0,T]} l(s_t) \right] \leq 0. \tag{2}$$

Note that the constraint (2) is non-Markovian in that it takes the expectation over the *history* of the value of $l(s_t)$.

Remark 1. Solving for the policy that minimizes the left hand side of the constraint (2) itself amounts to solving a reachability problem where the target set is the unsafe set defined by the zero-superlevel set $\{s|l(s) \geq 0\}$, for more details, please refer to [15, 17].

2.2 Worst-Case Augmentation

In order to deal with non-Markovian constraint (2) with RL algorithm, we need to modify the environment to make use of PG-based Safe RL algorithms. It can be achieved by defining an augmented state $\bar{s}_t$. We define an augmented state $\bar{s}_t = (s_t, z_t)$ as a concatenated vector of the original state s_t and worst-case state z_t. The worst-case state is in this augmented state vector and for each timestep, it is updated by comparing the current safety metric $l(s_t)$ with the previous worst-case record of the safety metric, $l(z_t)$.

Showing that the dynamics of the augmented state is the Markovian process is straightforward: assume that original state s_t satisfies the Markovian process. Since z_{t+1} is determined by comparing z_t and s_t, each current state depends only on the state attained in the previous event. Thus, by definition, the augmented state satisfies the Markovian process. Replacing the $\arg\max_{t \in [0,T]} l(s_t)$ with $l(z_T)$ in (2), the problem is then translated to a typical optimal control problem with a constraint on the terminal state,

$$\mathbb{E}_{\tau \sim p_\theta(\tau)} \left[l(z_T) \right] \leq 0. \tag{3}$$

Remark 2. When the time horizon of the problem is pre-specified and finite, this problem is well-defined. However, the tricky case is when this horizon cannot be specified by the user. For instance, it might be unrealistic that we continue the rollout of the trajectory when the safety constraint is already violated. For such case, the horizon should be cut off when the system encounters failure. This issue will be investigated in our future work.

2.3 Policy Gradient for Worst-Case Safe Policy

We use the augmented state and the worst-case terminal cost for the problem formulation. This Safe RL problem can be solved by lagrangian-based methods. Thus, among various kinds of PG-based RL algorithms, we select PPO-Lagrangian[1] for our base learning algorithm. Lagrangian method uses adaptive variables to enforce constraints. For example, with given objective $f(\theta)$ and constraint $g(\theta)$, by using the lagrangian method, we can transform the original problem into a dual unconstrained max-min optimization problem,

$$\max_{\theta} \min_{\lambda \geq 0} \mathcal{L}(\theta, \lambda) = f(\theta) - \lambda g(\theta). \tag{4}$$

PPO-Lagrangian basically implements the lagrangian method in PPO. $f(\theta)$ and $g(\theta)$ can be represented by our augmented state z_t and the safety constraint (2). Hence, we believe that the PPO-Lagrangian is capable of solving our CMDP problem. In this course project, we implemented PPO-Lagrangian on our customized augmented environments which will be explained in the next section.

3 Results

3.1 Baseline Algorithms

In the result part we will call our method described in Sec.2.3 as **PPO-Worst-Case**. We compare this with the well-know previous RL methods that fall into the Policy Gradient-based category.

The first baseline method is Proximal Policy Optimization (**PPO**) [10], a state-of-the-art PG method but without any explicit safety constraint. To account for the safety constraint in the actual task, we can manually introduce a large negative reward when safety is violated.

The second baseline method is Constrainted Policy Optimization (**CPO**) [8], a trust region-based method that regulates the KL divergence between the updated policy and the previous policy [9]. Finally, we also evaluate a modified PPO algorithm which applies PPO to solve the dual problem of the constrained objective (1), together with the dual update on the lagrangian multiplier for the safety constraint. Dubbed as **PPO-Lagrangian** in [1], this algorithm is known to achieve better safe exploration performance among **PPO** and **CPO**.

3.2 Environments

We tested our methods on two environments. First, we used the **OpenAI SafetyGym environment** to benchmark our method with the baseline methods that is implemented in the library[1]. Due to the time constraint, we were able to evaluate our method only on one environment–PointGoal. This is an environment where a simple pointmass robot has to reach a goal point while navigating a planar terrain under the existence of some cluttered unsafe region (Figure 1). We choose this environment because it resembles the safe navigation task that we consider as the main application of our method. We made a modification to the existing PointGoal1 environment, to bound the robot's state space region. To provide robot an information about its global position, we introduced fixed pillars at each corner of the bounded region, so that they can serve as landmarks. The robot uses a lidar sensor for its observation and returns distance values that are computed from a set of bins around the robot. Thus, when a robot approaches to the pillars or the boundary, robot can realize its location. Plus, we made a trajectory terminate when the robot violates predefined boundary. Subsequently, reward is decreased by 2 in order to indicate boundary violation. This way, the robot indirectly learns to move inside the working space and find a goal within a limited time step. Here we set $4m \times 4m$ space as a state space region.

In this environment, reward is calculated by comparing the last distance to a goal point with a current distance. Hence, the reward equation indicates whether a robot is heading towards a goal or not and it's called once per step. Also, we did not penalize the reward function as a robot violates safety constraints. Cost is computed separately to consider safety in learning process. Whenever the robot enters the random unsafe regions (marked in purple circles in Fig. 1), its safety constraint is violated. In the baseline methods, this is captured by the cumulative costs of the binary signal of entrance to the unsafe regions or collision with the landmark pillars. In our method, this is captured by the signed distance to the unsafe regions or the pillars being positive, as explained in 2.1.

Second, we tested out our method as well as the baseline algorithms on a **simulation environment for navigation of an A1 quadruped robot**. This simulation environment is built on Google Brain's Motion Imitation library[2], which uses PyBullet as the physics engine. To make the task simple enough for this course project, unlike SafetyGym environment, we fixed the obstacles and goal point locations (Figure 2). We introduced 16 pillar obstacles uniformly distributed over the terrain as well as the wall that bounds the terrain, and the goal location is fixed at the center. Finally, another main difference to the SafetyGym setup is that the episode is immediately terminated upon collision to the obstacles or falling down to the ground. In this case, the terminal time in (2) is defined as the minimum of the first hitting time to the unsafe region and the prescribed episode horizon. This setup is chosen to evaluate how much the terminal time issue discussed in Remark 2 actually matters in practical implementation.

The observation of the robot is given as its global position, velocity, and the heading angle and the yaw rate, which consists a 8D vector. Its action is set as a reference velocity and yaw rate signal of its center of mass. To limit the action space, we limit the velocity to be either 0 or a constant velocity in the robot's longitudinal direction, and the yaw rate to be one of $(-\omega_{\max}, 0, \omega_{\max})$. Therefore, there are 6 choices of discrete action. Once this action is decided from the policy, the reference signal is tracked by the built-in low-level controllers of the robot. Finally, the reward is designed similar to the SafetyGym PointGoal environment so that the robot is incentivized to move towards the goal. An additional instanteneuous reward of 1000 is given when it enters the goal region. Finally, for

[1] https://github.com/openai/safety-starter-agents

[2] https://github.com/google-research/motion_imitation

the PPO, no penalty for safety violation is introduced since reaching the goal is only achievable when the episode does not terminate by colliding to an obstacle. Our hope with PPO is that it will implicitly learn that in order to reach the goal (and get a high reward), along the path it has to avoid the obstacles and maintain safety.

3.3 Result - Safety Gym PointGoal Environment

Figure 3 reports the overall learning curves of each algorithm. Table 1 shows the summary of return, worst-case distance, and cumulative binary cost throughout the evaluation averaged over 100 episodes. They were evaluated on the same environment. The only difference in the setup for our algorithm is the usage of the worst-case distance instead of a binary safety violation signal. We provided both worst-case distance and cumulative binary cost for convenient comparison. As we notice, our proposed method is capable of learning a policy that enables a robot to accomplish the Goal task. According to learning curves, every algorithm reaches episode return above 15 at the end of the training (500 epochs). We tested the trained policies and checked that the policies from all algorithms frequently reaches the goal with randomly distributed unsafe regions. Since CPO, PPO, PPO-Lagrangian algorithms consider a cost threshold, trained policies turn out to violate the constraints 39.2, 52.2, 23.7 times average in one episode, respectively. On the other hand, our proposed method aims to maintain the worst-case signed distance to the obstacle below 0. However, as can be seen in the Table 1, the trained policy from PPO-Worst-Case enters unsafe regions 47.3 times average in one episode and achieves 21.4 episode return. From Figure 3 bottom, we notice that the PPO-Worst-Case returns a better reward when safety is more violated and vice versa throughout the training.

3.4 Result - A1 Quadruped Navigation Environment

Figure 4 reports the overall learning curves of each algorithm. Table 2 shows the summary of return, worst-case distance, and cumulative binary cost throughout the evaluation averaged over 100 episodes. Again, the only difference in the setup for PPO-Worst-Case is that we use the signed distance signal for the worst-case constraint. The average return of the PPO-Worst-Case policy is inferior to the other two methods, however, we have observed that the surrogate cost, which is the combination of the reach-goal reward and the dual cost of the worst-case constraint, oscillates severely for PPO-Worst-Case algorithms, which indicates that the primal-dual update is not done properly. However, we observed that the safe policy can be learned surprisingly fast in PPO-Worst-Case algorithm, as can be seen in the cost signal in Fig.4 bottom. The overall behavior of this policy is to stay at the same location for a long time while either standing still or spinning in the same direction. This indicates that in the very beginning, the policy has converged to a very conservative safe policy, which would actually defect the learning curve because it induces less exploration.

4 Conclusion & Future Work

In this project, we revisited the typical Constrained MDP (CMDP) formulation for Safe RL problems. Inspired from the Hamilton Jacobi Reachability literature, we devised a new CMDP formulation that captures the worst-case instance of the trajectory in terms of its safety in the new constraint. In order to transform this non-Markovian constraint into a Markovian constraint, we define a new system where the worst-case state is augmented to the agent's current state. Then, the problem transforms to a terminal-time-constrained RL, which can be solved by existing primal-dual-based Policy Gradient methods. We compare our method resulting from this new formulation to the existing Policy Gradient (PG)-based deep RL methods including the vanilla PPO, PPO with dual update on the accumulative safety constraint (PPO-Lagrangian), and CPO. We have evaluated these methods on an OpenAI Safety Gym Benchmark environment (Fig.1, and a realistic simulation of a quadruped navigating an environment with cluttered obstacles (Fig.2). We have observed that our algorithm can learn safe policies effectively, however, it failed to balance well between the dual cost on safety constraint and the primal cost on the original task reward, which resulted in either a very conservative policy, or a suboptimal policy with mediocre safety performance.

The main issue of our current approach in the theory side is that we need better understanding of how the resulting terminal-time constraint-problem is well defined when we cannot fix the trajectory time horizon a priori; for instance, when the rollout should end whenever it faces unsafe states. This

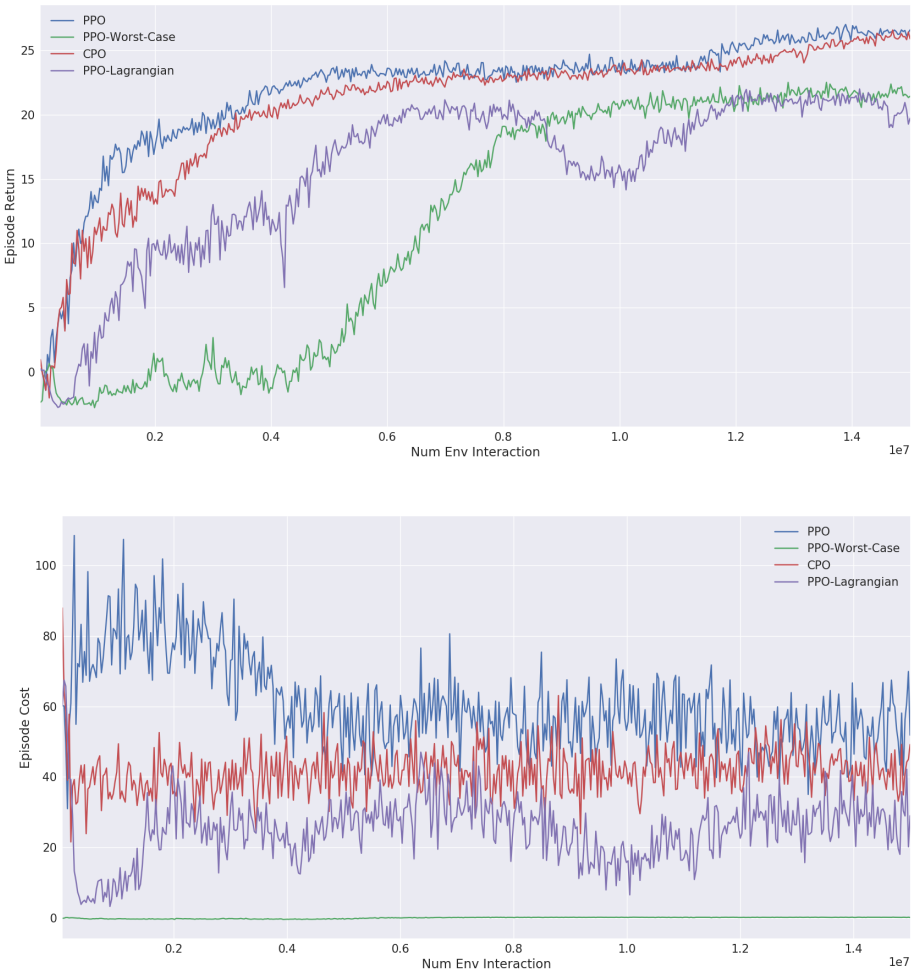

Figure 3: Results from training our method (PPO-Worst-Case) with the baseline algorithms (PPO, CPO, PPO-Lagrangian) on PointGoal Safety Gym environment. Top: Average episode return, bottom: Average episode cost (Note that the PPO-Worst-Case use a different cost signal, which is in a very different scale.)

becomes a bigger problem when one is concerned with infinite-time safety guarantee. Moreover, in many applications, designing a good safety-metric $l(s, a)$ that will be used to construct the worst-case constraint (2) is not straightforward. It must be noted that in the optimal control literature, in order for the value functions to behave nicely, such safety signal should be continuous in the state space. Therefore, it remains an open question whether discontinuous safety signal (for instance, the indicator of failure) can be used under this worst-case formulation.

The main current limitation of our work in the algorithmic is that the primal-dual update during the training is not done properly, we are observing that the value of the dual objective function oscillates severely throughout the training. In fact, in the PointGoal environment, we have observed a persistent oscillation between the average reward return and the average safety cost; the safety is more violated when the agent is learning to achieve higher reward. This indicates that the updates on the lagrangian multiplier is not done properly. We will investigate this problem further, and will consider customizing the PG-based primal-dual update algorithm for the specific use case of our worst-case constrained problem.

Acknowledgments

We would like to express our gratitude to our brilliant collaborator Kai Yun for helping us setting up the training environments and running the training experiments.

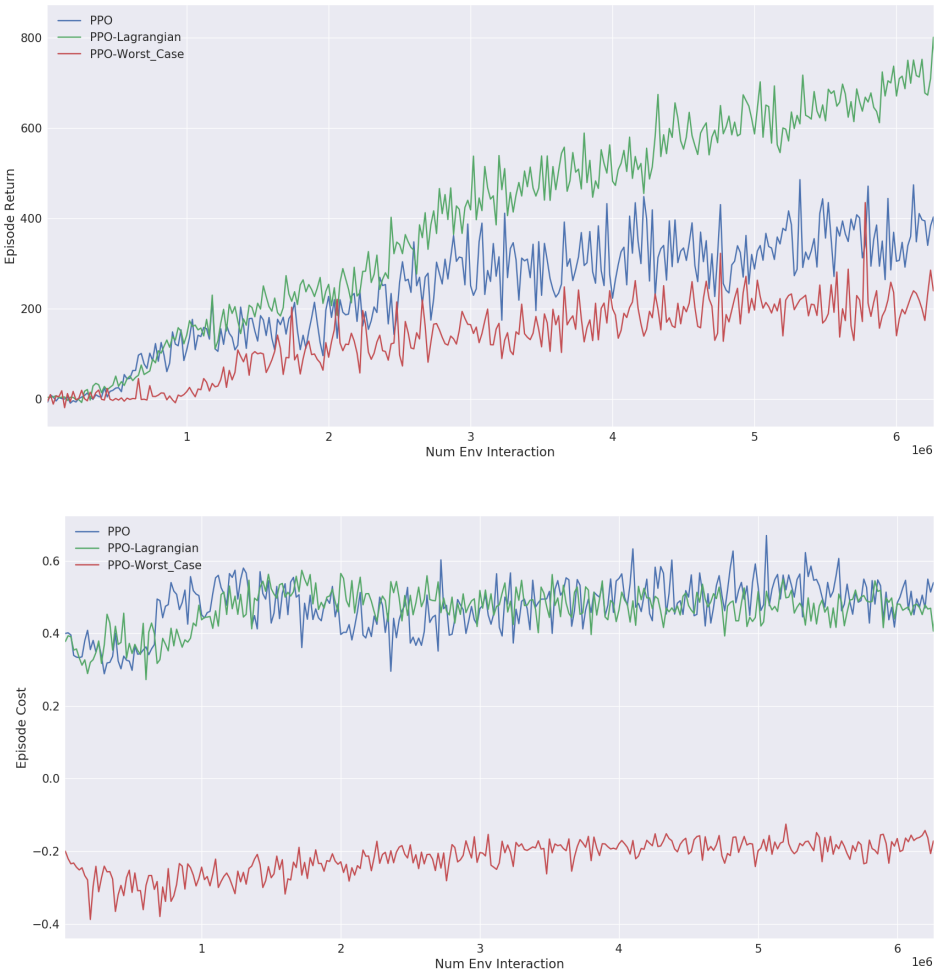

Figure 4: Results from training our method (PPO-Worst-Case) with the baseline algorithms (PPO, PPO-Lagrangian) on A1 Navigation environment. Top: Avg. episode return, bottom: Avg. episode cost (Note that the episode cost for PPO and PPO-Lagrangian represents the robot's collision rate and for PPO-Worst-Case it represents the signed distance to the obstacles)

References

[1] A. Ray, J. Achiam, and D. Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7, 2019.

[2] I. J. Fialho and T. T. Georgiou. Worst case analysis of nonlinear systems. *IEEE TAC*, 1999.

[3] E. Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[4] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *arXiv preprint arXiv:1805.07708*, 2018.

[5] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

[6] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

[7] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021.

[8] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

[9] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[11] J. Li, D. Fridovich-Keil, S. Sojoudi, and C. Tomlin. Augmented lagrangian method for instantaneously constrained reinforcement learning problems. In *IEEE CDC 2021*.

[12] J. Lygeros. On reachability and minimum cost optimal control. *Automatica*, 2004.

[13] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE TAC*, July 2005.

[14] I. M. Mitchell and J. A. Templeton. A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems. In *Hybrid Systems: Computation and Control*, 2005.

[15] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-Jacobi reachability: A brief overview and recent advances. In *IEEE CDC*, 2017.

[16] S. Bansal and C. J. Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. In *IEEE ICRA*, 2021.

[17] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.

[18] J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, D. Batra, and A. Rai. Learning navigation skills for legged robots with learned robot embeddings. *arXiv preprint arXiv:2011.12255*, 2020.

[19] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter. Learning a state representation and navigation in cluttered and dynamic environments. *IEEE Robotics and Automation Letters*, 6 (3):5081–5088, 2021.

[20] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

[21] Z. Xie, H. Y. Ling, N. H. Kim, and M. van de Panne. Allsteps: Curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum*, volume 39, pages 213–224. Wiley Online Library, 2020.

[22] M. Tranzatto, F. Mascarich, L. Bernreiter, C. Godinho, M. Camurri, S. M. K. Khattak, T. Dang, V. Reijgwart, J. Loeje, D. Wisth, S. Zimmermann, H. Nguyen, M. Fehr, L. Solanka, R. Buchanan, M. Bjelonic, N. Khedekar, M. Valceschini, F. Jenelten, M. Dharmadhikari, T. Homberger, P. De Petris, L. Wellhausen, M. Kulkarni, T. Miki, S. Hirsch, M. Montenegro, C. Papachristos, F. Tresoldi, J. Carius, G. Valsecchi, J. Lee, K. Meyer, X. Wu, J. Nieto, A. Smith, M. Hutter, R. Siegwart, M. Mueller, M. Fallon, and K. Alexis. Cerberus: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the darpa subterranean challenge. *Field Robotics*, 2022. doi:10.3929/ethz-b-000489726.