Self-Paced Policy Optimization with Safety Constraints

Fan Yang [1] **Wenxuan Zhou** [1] **Harshit Sikchi** [2] **David Held** [1]

Abstract

A common objective in Safe Reinforcement Learning (RL) is to maximize reward while satisfying safety constraints. However, previous methods in safe RL have difficulties in balancing the reward and the safety objectives. To avoid constraint violations, the policies tend to be over-conservative and converge to local optima with low rewards. In this work, we propose Self-paced Safe Reinforcement Learning which combines a self-paced curriculum on the safety objective with a base safe RL algorithm PPO-Lagrangian. During training, the policy starts with easy safety constraints and gradually increases the difficulty of the constraints until the desired constraints are satisfied. We evaluate our algorithm on the Safety Gym benchmark and demonstrate that the curriculum helps the underlying Safe RL algorithm to avoid local optima and improves the performance for both reward and safety objectives.

1. Introduction

Reinforcement Learning has demonstrated a lot of success in sequential decision making tasks (Mnih et al., 2013; Silver et al., 2016). In most cases, RL algorithms only have a single objective of maximizing the reward during training. However, when safety is a concern in some domains such as human-robot interaction (Pang et al., 2021; Liu & Tomizuka, 2018) or autonomous driving (Muhammad et al., 2020; Hu et al., 2018; Wang et al., 2019), the safety objective is incorporated into the RL objective as a constraint. Thus, the policy will be optimized to maximize the reward while satisfying the constraints.

However, the trade-off between the reward objective and the safety constraints will create difficulties for training(Ray et al., 2019). For example, the policy tends to get stuck at an over-conservative local optima where the safety constraints are satisfied with low rewards. For example, in autonomous driving tasks, the agent is likely not to move at all to avoid any collision with the obstacles.

In this work, we aim to improve the suboptimal solutions in safe RL with curriculum learning. Curriculum learning (Elman, 1993; Bengio et al., 2009) has demonstrated the abilities of stabilizing the training, increasing sample efficiency and avoiding local optima. Previous work (Hacohen & Weinshall, 2019) has demonstrated that curriculum learning can effectively modify the optimization landscape and converge to better final performance. These properties can help the optimization difficulties in safe RL. In addition, to avoid manually designing the curriculum, we consider automatic curriculum (Akkaya et al., 2019) or self-paced learning (Kumar et al., 2010), where the algorithm automatically adjusts the curriculum based on its current performance.

More specifically, we build on top of Lagrangian penalized versions of PPO and propose to improve its performance by applying self-paced learning on the safety constraints. The agent is trained with relaxed constraints first and chooses more challenging constraints when the current constraints are easy to satisfy. For example, the agent can first learn to drive fast and then learn to drive safely to avoid the local optima of driving safely but slowly. This type of training procedure can help exploration and converge to a better final performance in a lot of scenarios. We emphasize that our objective in this work is to obtain a high performance safe policy at deployment but not during trainig. For example, with sim2real transfer, we can learn a policy aggressively in simulation and deploy a safe policy on the real robot.

In summary, we propose a self-paced policy optimization algorithm with safety constraints which could improve exploration and lead to better final performances. We demonstrate the effectiveness of our algorithm on a toy example on constraint optimization and evaluate it on the Safety Gym benchmark (Ray et al., 2019).

[1]Robotics Institute, Carnegie Mellon University [2]Computer Science Department, The University of Texas at Austin. Correspondence to: Fan Yang <fanyang3@andrew.cmu.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

2. Related Work

2.1. Safe RL

Safety is an important consideration in a lot of real-world applications. The goal of safe RL is to train RL policies with safety constraints in addition to the reward objective. Lagrangian method has been widely used in solving Safe RL problems (Altman, 1998; Geibel & Wysotzki, 2005; Chow et al., 2017). The Lagrangian method defines a Lagrangian function which combines reward and cost functions and automatically adjusts the weights between the two objectives. Constrained Policy Optimization (CPO) (Achiam et al., 2017) derives the trust region for the update in safe RL setting and can guarantee monotonic policy improvement while satisfying safety constraints. Other methods apply Lyapunov constraints to ensure a safe policy update (Perkins & Barto, 2002; Chow et al., 2019; Sikchi et al., 2021; Berkenkamp et al., 2017; Chow et al., 2018). However, existing safe RL algorithms still struggle with the trade-off between reward and safety constraints. In addition, methods such as CPO also aim to ensure safety during the exploration process which could be over-conservative. Our method is built on top of PPO-Lagrangian method and improves the exploration with self-paced safety constraints.

2.2. Curriculum Learning

Curriculum learning (Elman, 1993) has been shown to be beneficial in both supervised learning and RL (Bengio et al., 2009; Zaremba & Sutskever, 2014). In addition, generating a curriculum automatically can lead to better asymptotic performance compared to a fixed curriculum. For example, Kumar et al. (2010) propose the idea of self-paced learning. In their method, the easiness of each task is defined by computing the loss of it. The training will focus more on tasks that could be easily achieved for the current network. Florensa et al. (2017) and Graves et al. (2017) also propose the idea of automatically generating the curriculum. Another recent method, Automatic Domain Randomization (Akkaya et al., 2019) allows the agent to increase the difficulties of the tasks automatically based on its current policy.

3. Background

3.1. Markov Decision Process

A Markov decision process (MDP) is described as a tuple $(S; A; r; P; \ ; \ )$, where S is the state set, A is the action set, $r(\mathbf{s}; \mathbf{a}) : S \ A \ ! \ \mathbb{R}$ is the reward function. $P : S \ A \ S \ ! \ [0; 1]$ is the state transition probability function, denotes the initial state distribution, and is the discount factor. Specifically, each episode starts with an initial state $\mathbf{s}_0 \ (\mathbf{s}_0)$, the state is input into a parameterized policy to get the action $\mathbf{a}_t \ (j\mathbf{s}_t)$ The agent takes the action $\mathbf{a}_t$ and the next state is sampled from the environment $\mathbf{s}_{t+1} \ p(j\mathbf{s}_t; \mathbf{a}_t)$. For a given state-action tuple, the reward is given as $r_t = r(\mathbf{s}_t; \mathbf{a}_t)$. The training objective is to maximize the expected discounted sum of reward:

$$\max J_r(\ ) = \mathbb{E} \ [\sum_{t=0}^{1} \ ^t r(\mathbf{s}_t; \mathbf{a}_t)]; \tag{1}$$

where $= (\mathbf{s}_0; \mathbf{a}_0; \mathbf{s}_1; \mathbf{a}_1; :::)$ is the trajectory sampled given the policy , initial state distribution $\mathbf{s}_0 \ (\mathbf{s}_t)$, and the state transition function $\mathbf{s}_{t+1} \ p(j\mathbf{s}_t; \mathbf{a}_t)$.

3.2. Constrained Markov Decision Process

A constrained Markov decision process (CMDP) is an augmented version of MDP (Altman, 1999). Specifically, a constraint function $c(\mathbf{s}_t; \mathbf{a}_t)$ are added to MDP, where $c(\mathbf{s}_t; \mathbf{a}_t) : S \ A \ ! \ \mathbb{R}$. Without loss of generality, we assume that constraint is defined as $c(x) \ 0$(Any $c(x) \ d$ could be rewritten as $c^0(x) = c(x) \ d \ 0$.). Similarly, we could also define the expected discounted cost as:

$$J_c(\ ) = \mathbb{E} \ [\sum_{t=0}^{1} \ ^t c(\mathbf{s}_t; \mathbf{a}_t)]: \tag{2}$$

Then the training objective is defined as:

$$\max_{2 \ f} J_r(\ ): \tag{3}$$

where $_f \doteq f \ 2 \ : J_c(\ ) \ 0g$ is the feasible set.

3.3. PPO-Lagrangian Method

Lagrangian method (Altman, 1998) is one of widely used methods for solving constrained optimization problems. Specifically, in our problem, we define the Lagrangian function $L(\ ; \lambda)$ as:

$$L(\ ; \ ) = \ J_r(\ ) + \ J_c(\ ) \tag{4}$$

The training objective can be written as:

$$\max_{0} \min L(\ ; \ ) \tag{5}$$

PPO-Lagrangian combines the Lagrangian method with Proximal Policy Optimization (PPO) (Schulman et al., 2017). PPO defines the training objective as:

$$\begin{aligned} L_r{}^k(\ ) = \mathbb{E} \ _k[\min(&\frac{(\mathbf{a}j\mathbf{s})}{_k(\mathbf{a}j\mathbf{s})} A_r{}^k(\mathbf{s}; \mathbf{a}); \\ &\text{clip}(\frac{(\mathbf{a}j\mathbf{s})}{_k(\mathbf{a}j\mathbf{s})}; 1 \ ; 1 + \ )A_r{}^k(\mathbf{s}; \mathbf{a}))]; \end{aligned} \tag{6}$$

where $A_r{}^k(\mathbf{s}; \mathbf{a})$ is the advantage function. PPO-Lagrangian uses a similar formulation for reward to define the training objectives $L_c{}^k(\ )$ for cost. Then the training objective of PPO-Lagrangian is defined as:

$$\max_{0} \min \ L_r{}^k(\ ) + \ L_c{}^k(\ ) \tag{7}$$

Algorithm 1 Self-Paced PPO-Lagrangian Method

Input: number of training epochs N_t, number of policy training epochs N.
for $n = 1$ to N_t do
  Estimate the expected discounted sum of costs $J_c(\pi_t)$
  $d_i \quad \arg\max_{d_i \in D} d_i \quad J_c(\quad)$
  $+ \quad (J_c(\quad) \quad d_i)$
  for $t = 1$ to N do
  $+ \quad r \quad (\quad L_r{}^k(\quad) + \quad L_c{}^k(\quad))$
  end for
end for

4. Self-Paced Safe Reinforcement Learning (SPSRL)

4.1. SPSRL Framework

As discussed in Sec 2, directly optimizing Eq. 3 is challenging because it suffers from local optima and training instabilities. We propose self-paced policy optimization with safety constraints, which enables the agent to choose the suitable training objectives automatically and alleviate the problems mentioned above.

Ideally, when an agent is presented tasks with with different difficulties, the agent should gradually learn from the easy ones to the hard ones. In our case, we define a task in safe RL with different difficulties by setting different safety thresholds $d_i : D = \{d_1, d_2, \ldots, d_k\}$. We rank the thresholds with d_1 as the lowest one and d_k as the highest one. During deployment, we may only care about one of the threshold defined in D, which is usually the smallest one $d_1 = 0$. The other thresholds are defined to help the training procedure.

During training, we start from the highest threshold d_k. As long as the safety threshold is satisfied $J_c(\quad) < d_k$, we decrease the threshold $d_k \quad_1$. We iterate such procedure until the minimal threshold d_0 is satisfied. Formally, we solve the following equations iteratively:

$$\min \quad J_r(\quad) \quad \text{s.t.} \quad J_c(\quad) \quad d_i; \tag{8}$$

where

$$d_i = \arg\max_{d_i \in D} \; d_i \quad J_c(\quad): \tag{9}$$

4.2. Self-Paced PPO-Lagrangian method

In this work, we combine the proposed self-paced curriculum with PPO-Lagrangian. However, a similar idea could potentially be applied to other constrained optimization algorithms, and we leave that exploration to future work. To solve the max-min optimization in PPO-Lagrangian, we have two iterations for parameter update: one updates policy parameters with higher frequency and the other updates with much lower frequency. Note that different threshold

(a) $f(x)$ and $c(x)$ (b) Optimization Trajectory

(c) Lagrangian Method (d) SP Lagrangian Method

Figure 1. Fig. 1a visualizes $f(x)$ and $c(x)$. Fig. 1b visualizes the trajectory of x during optimization. Fig. 1c and Fig. 1d visualizes the trajectory of $f(x)$ and $c(x)$. The Lagrangian method converges to a local optima which satisfies the constraint but does not achieve the best possible performance. The self-paced Lagrangian method would relax the constraint first and finally converge to a global optima which is both feasible and optimal.

will only affect the maximization over but will not affect the minimization over . This is because adding a constant value will not affect the gradient update of Thus, at each iteration, is updated according to:

$$_{t+1} = \quad _t + \quad (J_c(\quad) \quad d_i); \tag{10}$$

where d_i is defined as Eq. 9, and is the update rate for. The algorithm is summarized in Alg. 1.

5. Experiments

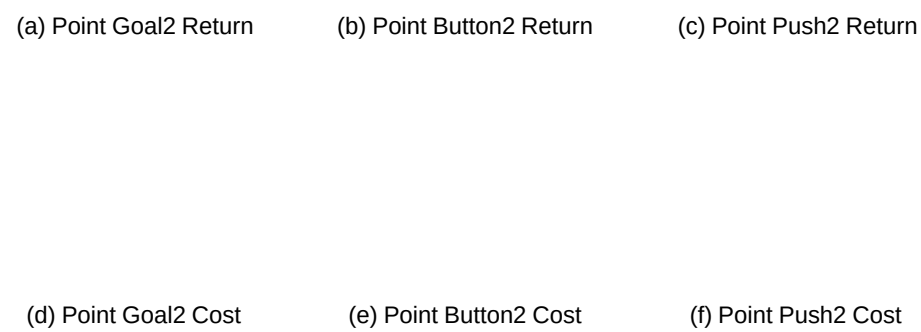
In this section, we first study a toy example to demonstrate the benefits of using self-paced learning on constraints. Then we evaluate the combination of self-paced constraints with PPO-Lagrangian on the Safety Gym Benchmark (Ray et al., 2019). More results can be found in Appendix A.

5.1. Toy Example

We first apply self-paced learning with safety constraints on a toy example to give an intuition why self-paced learning would help constrained optimization. In this experiment, we formulate our problem as the following objective:

$$\min_x f(x); \quad \text{s.t.} \; c(x) \quad 0: \tag{11}$$

Ideally, the proposed self-paced learning over the constraints is supposed to be effective when the feasible set is composed of multiple disconnected subsets. In this case the algorithm may be trapped into a certain constraint set and cannot



(a) Point Goal2 Return (b) Point Button2 Return (c) Point Push2 Return

(d) Point Goal2 Cost (e) Point Button2 Cost (f) Point Push2 Cost

Figure 2.PPO-Lagrangian (orange curve) and SP-PPO-Lagrangian (blue curve) are evaluated in the Safety Gym environment. We choose the point agent and different levels of goal, button, and push tasks. In most of the tasks, our method can converge to a policy with better nal returns while has a similar performance as PPO-Lagrangian in terms of satisfying the safety constraints.

nd the global optima. Thus, we de ne the optimization objective function $f(x) = \frac{1}{50}(x \quad)^2$ and the constraint function $c(x) = \quad 5\cos(x)$. Mathematically, the global optima should be $x = \frac{}{2}$ or $x = \frac{3}{2}$.

We rst apply Lagrangian method to solve Eq. 11. Then we combine SPCPO with Lagrangian method (denoted as SP Lagrangian method). The threshold set $\mathcal{D}$ is de ned as:

$$\mathcal{D} = \{ d_i = 2^{N \quad i} \mid i = 0, 1, 2, \ldots, M \}, \tag{12}$$

where N de nes the initial threshold, while M de nes the number of threshold we use in the set. Adam (Kingma & Ba, 2014) is used in both algorithms for both and x optimization.

The results are shown in Fig. 1. The Lagrangian method is over-conservative, and stuck in a point where it is safe but the doesn't reach the lowest function value of $f(x)$. In constrast, the self-paced method nds the optima without the constraint rst, and then gradually adjusts the parameters to satisfy the constraint.

5.2. Safety Gym Experiment

We evaluate our algorithm on the safety RL benchmark Safety Gym (Ray et al., 2019). The constraints for safety gym environment is de ned as a limited tolerance of interacting with hazards and obstacles. De ning different cost threshold amounts to de ning different budget for unsafe actions. We compare PPO-Lagrangian and the proposed self-paced PPO-Lagrangian method in this experiment. We use 5 seeds for training for each task.

The experiment results are shown in Fig. 2. SP PPO-Lagrangian has better performances in terms of the nal return with similar nal cost as PPO-Lagrangian. At the beginning of training, the return for SP PPO-Lagrangian increases much faster than PPO-Lagrangian and the cost reduces more slowly. This indicates that the policy is doing exploration more aggressively with the self-paced learning. In Point Push2, SP PPO-Lagrangian does not have a signi - cant improvement over PPO-Lagrangian. We hypothesize that it is more challenging than other environments to get high returns even without considering the cost constraints.

6. Conclusion

In this work, we propose Self-Paced Safe Reinforcement Learning (SPSRL). We prede ne a set of safety threshold indicating different dif culties. The agent is allowed to choose a suitable threshold for training based on its current performance. In this way the agent learns to optimize for the reward aggressively rst and gradually netunes to satisfy the safety constraints. We demonstrate the bene ts of our algorithm in a toy example and the Safety Gym environments.

In the future, we aim to evaluate our method in more diverse settings and combine it with other safe RL algorithms besides PPO-Lagrangian. We are also interested in extending this work by experimenting with other ways of incorporating curriculum to study the role of curriculum in safe RL more systematically.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In International conference on machine learning, pp. 22–31. PMLR, 2017.

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik's cube with a robot hand. arXiv preprint arXiv:1910.07113, 2019.

Altman, E. Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program. Mathematical methods of operations research, 48(3):387–417, 1998.

Altman, E. Constrained Markov decision processes: stochastic modeling. Routledge, 1999.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In Proceedings of the 26th annual international conference on machine learning, pp. 41–48, 2009.

Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. Safe model-based reinforcement learning with stability guarantees. Advances in neural information processing systems, 30, 2017.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. Risk-constrained reinforcement learning with percentile risk criteria. The Journal of Machine Learning Research, 18(1):6070–6120, 2017.

Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. A lyapunov-based approach to safe reinforcement learning. Advances in neural information processing systems, 31, 2018.

Chow, Y., Nachum, O., Faust, A., Duenez-Guzman, E., and Ghavamzadeh, M. Lyapunov-based safe policy optimization for continuous control. arXiv preprint arXiv:1901.10031, 2019.

Elman, J. L. Learning and development in neural networks: The importance of starting small. Cognition, 48(1):71–99, 1993.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. Reverse curriculum generation for reinforcement learning. In Conference on robot learning, pp. 482–495. PMLR, 2017.

Geibel, P. and Wysotzki, F. Risk-sensitive reinforcement learning applied to control under constraints. Journal of Artificial Intelligence Research, 24:81–108, 2005.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. Automated curriculum learning for neural networks. In international conference on machine learning, pp. 1311–1320. PMLR, 2017.

Hacohen, G. and Weinshall, D. On the power of curriculum learning in training deep networks. In International Conference on Machine Learning, pp. 2535–2544. PMLR, 2019.

Hu, X., Chen, L., Tang, B., Cao, D., and He, H. Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles. Mechanical systems and signal processing, 100:482–500, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Kumar, M., Packer, B., and Koller, D. Self-paced learning for latent variable models. Advances in neural information processing systems, 23, 2010.

Liu, C. and Tomizuka, M. Robot safe interaction system for intelligent industrial co-robots. arXiv preprint arXiv:1808.03987, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

Muhammad, K., Ullah, A., Lloret, J., Del Ser, J., and de Albuquerque, V. H. C. Deep learning for safe autonomous driving: Current challenges and future directions. IEEE Transactions on Intelligent Transportation Systems, 22 (7):4316–4336, 2020.

Pang, G., Yang, G., and Pang, Z. Review of robot skin: A potential enabler for safe collaboration, immersive teleoperation, and affective interaction of future collaborative robots. IEEE Transactions on Medical Robotics and Bionics, 2021.

Perkins, T. J. and Barto, A. G. Lyapunov design for safe reinforcement learning. Journal of Machine Learning Research, 3(Dec):803–832, 2002.

Rajamani, R. Vehicle dynamics and control. Springer Science & Business Media, 2011.

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

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

Sikchi, H., Zhou, W., and Held, D. Lyapunov barrier policy optimization. arXiv preprint arXiv:2103.09230, 2021.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484–489, 2016.

Snider, J. M. et al. Automatic steering methods for autonomous automobile path tracking. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08, 2009.

Wang, P., Gao, S., Li, L., Sun, B., and Cheng, S. Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm. Energies, 12(12):2342, 2019.

Zaremba, W. and Sutskever, I. Learning to execute. arXiv preprint arXiv:1410.4615, 2014.

(a) Point Goal1 Return (b) Point Button1 Return (c) Point Goal1 Cost

(d) Point Button1 Cost (e) Point Push1 Return (f) Point Push1 Cost

Figure 3.PPO-Lagrangian (orange curve) and SP-PPO-Lagrangian (blue curve) are evaluated in the Safety Gym environment. We choose the point agent and different levels of goal, button, and push tasks. In most of the tasks, our method can converge to a policy with better nal returns while has a similar performance as PPO-Lagrangian in terms of satisfying the safety constraints.

A. Additional Experiment Results on Safety Gym

Additional experiment results for safety gym is shown in Fig. 3.

B. Safe Driving Experiment

Safety is one of the critical issues in the autonomous driving domain. We would like to test our constrained optimization algorithm in autonomous driving. We design the Safe Driving environment, which consists of a circle environment and a trajectory environment.

B.1. Experiment Setup

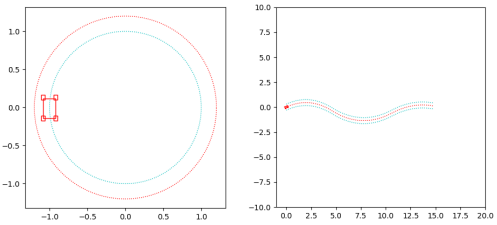
For both environment, we assume a bicycle model as the kinematic model and brush tire model as the dynamics model (Snider et al., 2009; Rajamani, 2011). We take drifting behaviors into account, which would make the task more challenging. The visualization of the two environments is shown in:

For the Circle environment, the vehicle is required to drive along the circle. We assume a xed width of the trajectory and going off the trajectory is considered as dangerous behaviors. The observation space is de ned as the lateral distance to the center of the trajectory, the orientation difference with respect to the tangent of the trajectory, and their corresponding derivatives with respect to time.

For the Trajectory environment, the vehicle is required to drive along the randomly generated trajectory. We assume a xed width of the trajectory and going off the trajectory is considered as dangerous behaviors. In addition to the observation space in the circle environment, the partial trajectory around the vehicle is also added into the observation space.

B.2. Experiment Results

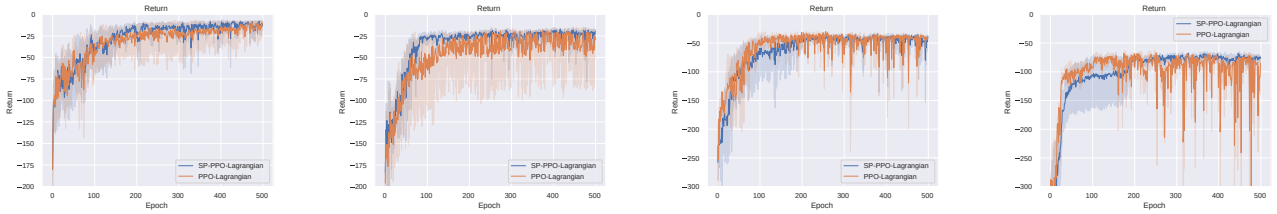
The results of Circle environment is shown in Fig. 5. We could tell that SP PPO-Lagrangian has a slightly better return than PPO-Lagrangian. Additionally, the training of SP PPO-Lagrangian is more stable than PPO-Lagrangian. However, if the



(a) Circle Environment (b) Trajectory Environment

Figure 4. Visualization of the Circle environment and the Trajectory environment. In the Circle environment, the vehicle is required to follow the circle. In the Trajectory environment, the trajectory is randomly generated. The vehicle is required to follow the trajectory. In both environments, we assume a fixed width of the trajectory. Going off the trajectory is considered as dangerous behaviors. We specify different desired velocities in both environments, which defines different levels of difficulties.

desired velocity is too high (e.g. $v = 6m$=s), SP PPO-Lagrangian may not converge to a safe policy within limited training steps. This is probably because it has converged to an aggressive policy, it would takes a long time to converge to a safe one.



(a) Circle Env (v=2) Return (b) Circle Env (v=3) Return (c) Circle Env (v=4) Return (d) Circle Env (v=5) Return

(e) Circle Env (v=2) Cost (f) Circle Env (v=3) Cost (g) Circle Env (v=4) Cost (h) Circle Env (v=5) Cost

Figure 5. PPO-Lagrangian (orange curve) and SP PPO-Lagrangian (blue curve) are evaluated in the Circle environment. We choose different desired velocities denoting different levels of difficulties. The greater the desired velocity is, the larger the gap between two methods in terms of return. But when desired velocity is too high (e.g. 5 m/s), SP PPO-Lagrangian cannot converge to a safe policy.

The results for the Trajectory environment is shown in Fig. 6. The conclusion here is similar to that in the Circle environment. We could see a greater gap between two methods in terms of return. The larger the velocity is, the larger the gap is.

The cost for SP PPO-Lagrangian cannot converge to zero if the desired velocity is too high. This is probably because the policy has already learned to be aggressive, it would take a longer time for it to be safe again because it may need to substantially change the strategy.