2024 IEEE International Conference on Robotics and Automation (ICRA)
May 13-17, 2024. Yokohama, Japan

Projection-Based Fast and Safe Policy Optimization for Reinforcement Learning

Shijun Lin, Hao Wang, Ziyang Chen, and Zhen Kan

2024 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-8457-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICRA57147.2024.10611554

***Abstract*— While reinforcement learning (RL) attracts increasing research attention, maximizing the return while keeping the agent safe at the same time remains an open problem. Motivated to address this challenge, this work proposes a new Fast and Safe Policy Optimization (FSPO) algorithm, which consists of three steps: the first step involves reward improvement update, the second step projects the policy to the neighborhood of the baseline policy to accelerate the optimization process, and the third step addresses the constraint violation by projecting the policy back onto the constraint set. Such a projection-based optimization can improve the convergence and learning performance. Unlike many existing works that require convex approximations for the objectives and constraints, this work exploits a first-order method to avoid expensive computations and high dimensional issues, enabling fast and safe policy optimization, especially for challenging tasks. Numerical simulation and physical experiments demonstrate that FSPO outperforms existing methods in terms of safety guarantees and task completion rate.**

I. Introduction

Deep reinforcement learning (RL) has shown great potentials in many fields, such as e-sports games, self-driving cars. When deploying deep RL in practice, achieving the goal while keeping the agent safe is of great importance. That is, the agent needs to maximize the reward collection while satisfying a set of safety constraints. However, most existing works are either not robust enough [1]–[3] or converge slowly [4] when facing challenging tasks with complex safety requirements. Therefore, this work is motivated to develop a fast and safe policy optimization method.

To keep the agent safe, certificate functions have been widely used. As a common certificate function, control barrier function (CBF) ensures the forward invariance of a safe set [5]–[8]. However, such methods have to rely on prior knowledge of the system models and in most cases, limiting its applications. To overcome this issue, recent works exploit barrier certificates which can be learned from collected data. For instance, the work of [9] learns workspace constraints from human demonstrations and generates robot trajectories satisfying the learned constraints. However, these methods still lack safety guarantees. An optimization-based approach was developed in [10] to learn control barrier functions from expert trajectories with provable safety. The work of [11] replaces the QP controller with a neural network and jointly learns the CBF and controller. These methods still require a lot of effort and high-quality sampling data to train the parameters to learn a qualified CBF.

Constrained Markov decision process (CMDP) is another common approach for safe RL [12]–[14]. Based on the idea of trust region policy optimization (TRPO) [15], the constrained policy optimization (CPO) was developed to guarantee safe exploration [1]. In [2], the projection-based constrained policy optimization (PCPO) replaced the line search of CPO with the projection to improve the convergence. Based on PCPO, the safe policy adaptation with constrained exploration (SPACE) was developed in [16], which uses a baseline policy to further accelerate the learning process. However, these methods all involve convex approximation of non-convex objectives and safe constraints via Taylor approximation, which can lead to poor performance. In addition, these methods require a high-dimensional Fisher information matrix, which increases the computational burden. Instead of convex approximation, the first order constrained optimization in policy space (FOCOPS) [17] exploits the idea of projection, updates the policy in the non-parametric policy space, and then projects it back into the parametric policy space. However, it heavily depends on the current optimal policy. An alternative to address CMDP is the Lagrangian method [18]–[20], which uses the primal-dual method and incorporates the constraint as a penalty signal by multiplying adaptive penalty coefficients. A novel multi-timescale constrained actor-critic approach, namely Reward Constrained Policy Optimization (RCPO), was developed in [4], in which the actor, the critic, and the Lagrangian multiplier share different learning rates. In [3], the cumulative discount cost was reformulated in a maximum form. However, these methods are limited by low convergence rates.

To deal with tasks with complex task objectives and safety constraints, linear temporal logic (LTL) has been used with RL [21]–[24]. For instance, an automata-inspired structure, namely reward machine, was developed in [25] to accelerate the RL training. However, safety was not considered in [25]. In [26], [27], CBF was incorporated with temporal logic-guided RL to satisfy the safety constraints and complex task constraints during exploration. However, the CBF was designed by hand, requiring prior knowledge of the system.

In this work, we propose a novel Fast and Safe Policy Optimization (FSPO) algorithm. Unlike the classical CPO [1], our approach is based on the primal-dual method

This work was supported in part by the National Natural Science Foundation of China under Grant 62173314 and U2013601.

S. Lin, H. Wang, Z. Chen, and Z. Kan (Corresponding Author) are with the Department of Automation at the University of Science and Technology of China, Hefei, Anhui, China, 230026.

with a first-order optimizer, which requires neither convex approximation of the objective and constraints, nor the Fisher information matrix. As a result, our approach requires less computational resources. In particular, to improve the convergence while ensuring policy safety, we consider three steps. We first maximize the expected reward, then project the policy to the region around the baseline policy to accelerate the training process, and finally perform a projection onto the safe sets. In addition, to deal with complex tasks, we also encode the task as a linear temporal logic (LTL) formula and use a reward machine to further accelerate the reward learning process.

The contributions are summarized as follows. The proposed FSPO does not require the convex approximation and Fisher information matrix, and thus is less computational resource demanding. The developed three-step projection optimization method can speed up the training process while keeping the policy safe. We further incorporate reward machines into FSPO to accelerate the learning process and handle complex tasks with temporal and logical constraints. Numerical and experimental studies demonstrate that FSPO outperforms most state-of-the-art baselines, especially for complex tasks.

II. Preliminaries and Problem Statement

A. Safe Reinforcement Learning

Safe reinforcement learning problem are usually formulated as constrained Markov Decision Process (CMDP) [12], which is a tuple $(S, A, \mathcal{T}, R, C)$, where S is the set of states, A is the set of actions, $\mathcal{T}: S \times A \times S \mapsto [0,1]$ is the transition probability, $R: S \times A \mapsto \mathbb{R}$ is the reward function, and $C: S \times A \mapsto \mathbb{R}$ is the cost function. Let $\pi(a|s)$ with $s \in S$ and $a \in A$ denote a policy. That is, given a state s, an action a is selected according to $\pi(a|s)$ and the state transits from s to s' according to the state transition model $\mathcal{T}(s'|s,a)$. A reward $R(s,a)$ and cost $C(s,a)$ can then be received, respectively.

The goal is to find a policy which maximizes the cumulative discounted reward $J^R(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R\left(s_t,a_t\right)\right]$ while keeping the cumulative discounted cost below h_C to satisfy the constraints, i.e., $J^C(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t C\left(s_t,a_t\right)\right] \le h_C$ where $\gamma \in (0,1)$ denotes the discount factor, $\tau = (s_0, a_0, s_1, \cdots)$ denotes the trajectory , and $\tau\sim\pi$ is a trajectory induced by π satisfying $\pi: s_0 \sim \mu, a_t \sim \pi(a_t|s_t), s_{t+1} \sim T(s_{t+1}|s_t,a_t)$, where μ is the initial state distribution.

We define the reward advantage function as $A_\pi^R(s,a) \doteq Q_\pi^R(s,a) - V_\pi^R(s)$, where $Q_\pi^R(s,a) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R\left(s_t,a_t\right) \mid s_0 = s, a_0 = a\right]$ is the expected reward following policy π starting from state s and action a, and $V_\pi^R(s) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R\left(s_t,a_t\right) \mid s_0 = s\right]$ is the expected reward from state s under policy π. Similarly, we can define the cost advantage function $A_\pi^C(s,a) \doteq Q_\pi^C(s,a) - V_\pi^C(s)$, where $Q_\pi^C(s,a) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t C\left(s_t,a_t\right) \mid s_0 = s, a_0 = a\right]$ and $V_\pi^C(s) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t C\left(s_t,a_t\right) \mid s_0 = s\right]$.

Let $P(s = s_t|\pi)$ denote the state distribution at time t under policy π. The discounted state distribution induced by π is $d^\pi(s) \doteq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mu_t(s \mid \pi)$. Then the performance difference between two policies π and π' can be compactly expressed as [28]

$$J^R(\pi') - J^R(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^\pi \\ a\sim\pi'}}\left[A_\pi^R(s,a)\right]. \tag{1}$$

Similarly, we have

$$J^C(\pi') - J^C(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^\pi \\ a\sim\pi'}}\left[A_\pi^C(s,a)\right]. \tag{2}$$

The policy can be safely learned by [1]:

$$\begin{aligned}\pi_{k+1} = &\arg\max_{\pi\in\Pi_\theta} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}}\left[A_{\pi_k}^R(s,a)\right] \\ &\text{s.t. } J^C(\pi_k) + \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}}\left[A_{\pi_k}^C(s,a)\right] \le h^C \quad \forall i \\ &\bar{D}_{KL}(\pi\|\pi_k) \le \delta.\end{aligned} \tag{3}$$

where $\bar{D}_{KL}$ is the KL-divergence between two policies, h^C is cost threshold.

B. Linear Temporal Logic and LDBA

Linear Temporal Logic (LTL) has been widely used in describe complex tasks. Detailed descriptions of the syntax and semantics of LTL can be found in [29].

Definition 3.1 An limit-deterministic Büchi automata (LDBA) [30] is a tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta: Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$ is a (partial) transition function, $q_0 \in Q$ is an initial state, and F is a set of accepting states.

C. Problem Statement

Given a safe reinforcement learning problem formulated in (3), we denote by π_B a baseline policy (i.e., a pre-trained policy that can be obtained by any reinforcement algorithm). The distance between $\pi(s)$ and $\pi_B(s)$ can be measured by KL-divergence as $D(s) \doteq D_{\mathrm{KL}}(\pi(s)\|\pi_B(s))$. Similarly, we can define $J^D(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t D\left(s_t\right)\right]$ and

$$J^D(\pi') - J^D(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\pi'} \\ a\sim\pi'}}\left[A_\pi^D(s,a)\right]. \tag{4}$$

Using the reward machine and KL-divergence between the learning policy and baseline policy, the training process can be improved since more information is utilized. Hence, the goal is to solve the following optimization problem:

$$\begin{aligned}\pi_{k+1} = &\arg\max_{\pi\in\Pi_\theta} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}}\left[A_{\pi_k}^{R'}(s,a)\right] \\ &\text{s.t. } J^C(\pi_k) + \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}}\left[A_{\pi_k}^C(s,a)\right] \le h^C \quad \forall i \\ &\quad\ J^D(\pi_k) + \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}}\left[A_{\pi_k}^D(s,a)\right] \le h^C \quad \forall i \\ &\quad\ \bar{D}_{KL}(\pi\|\pi_k) \le \delta.\end{aligned} \tag{5}$$

where R' is the reward function generated by the reward machine, which will be defined in the following section.

Algorithm 1 FSPO

1: **procedure** INPUT: (An initial policy $\pi_0 = \pi(\cdot \mid \boldsymbol{\theta}_0)$, the initial Lagrangian coefficient μ_0 and v_0, a baseline policy π_B and a trajectory buffer $\mathcal{B}$)
Output: the optimal policy θ^*
2: **while** Reward A^R and cost J^C not converged **do**
3: **for** each $k = 0, 1, 2 \ldots$ **do**
4: Sample a set of trajectories $D = \tau \sim \pi_k = \pi(\theta_k)$
5: Run Reward Improvement Step by (6) or (7) to obtain $\pi_{\theta_{k+\frac{1}{3}}}$
6: Project the $\pi_{\theta_{k+\frac{1}{3}}}$ to baseline policy π_B neighbor set by (10), (11) and (12) to obtain $\pi_{\theta_{k+\frac{2}{3}}}$
7: update v_k by (18) or (20)
8: Project the $\pi_{\theta_{k+\frac{2}{3}}}$ to constrain set by (15), (16) and (17) to obtain $\pi_{\theta_{k+1}}$
9: update μ_k by (19) or (21)
obtain θ_{k+1}, v_k and μ_k, Use $\pi_{\theta_{k+1}}$ in the environment to get new samples

III. Fast and Safe Policy Optimization

A. Approach Overview

Inspired by [16], we use three steps to facilitate the training of the policy, i.e., the reward improvement step, the divergence projection step, and the constraints projection step to solve the optimization problem in (5). CPO [1] can ensure the safety of policy without error occurred. However, in the event of an error, the projection can effectively resolve it, as shown in Fig. 1. To avoid residual errors due to the convex approximation, our approach is based on the dual methods. Unlike existing methods such as RCPO [4] that uses dual methods to mix the constraint and the reward, in this work the dual method is used after policy optimization.

As outlined in Alg. 1, π_0 is an initial policy to be updated and π_B is a pre-trained baseline policy that can be obtained by any trust region based reinforcement algorithm such as PPO [31], CPO [1] and RCPO [4]. For each $k = 0, 1, \ldots$, after sampling trajectories, we first maximize the reward in the trust region by (6) or (7) to get $\pi_{\theta_{k+\frac{1}{3}}}$ in line 5. Then, to accelerate the learning process, we project $\pi_{\theta_{k+\frac{1}{3}}}$ onto the region around π_B using (10), (11) and (12) to obtain $\pi_{\theta_{k+\frac{2}{3}}}$. The Lagrangian coefficients v_k can be update by (18) or (19). Finally, to satisfy the safety constraints, we project $\pi_{\theta_{k+\frac{2}{3}}}$ to the constrain set by (15), (16) and (17), update μ_k by (19) or (21), and output $\pi_{\theta_{k+1}}$ at last.

B. Three Optimization Steps

As illustrated in Fig. 1, our approach consists of three optimization steps.

Reward Improvement Step: inspired by the reward improvement method in TRPO [15] and PPO [31], we first maximize the reward advantage function $A^R_\pi(s, a)$ with the constraint of Kullback_Leibler (KL) divergence to perform a stable policy update. There are two ways in this step:

$$\pi_{\theta_{k+\frac{1}{3}}} = \arg\max_{\pi_\theta \in \Pi_\theta} \{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)}[A^R_{\pi_{\theta_k}}(s,a)]$$

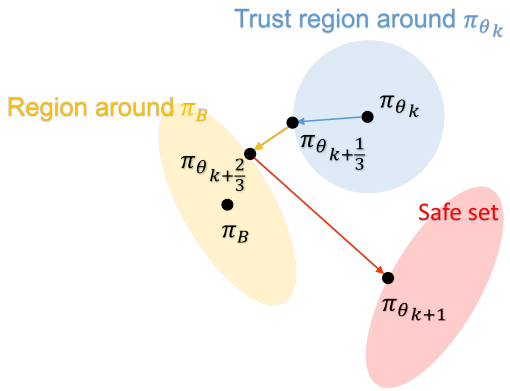


Fig. 1: Three optimization steps of FSPO when π_{θ_k} don't satisfy the constraint

$$-\alpha_k\sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}}[KL(\pi_{\theta_k}, \pi_\theta)[s]]}\}. \tag{6}$$

or update like PPO with clip as

$$\pi_{\theta_{k+\frac{1}{3}}} \approx \arg\max_{\pi_\theta \in \Pi_\theta} \Big\{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)} \min\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_k}(a_t \mid s_t)} A^R_{\pi_{\theta_k}}(s_t, a_t),\right.$$
$$\left. clip\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_k}(a_t \mid s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^R_{\pi_{\theta_k}}(s_t, a_t)\right)\Big\}. \tag{7}$$

Both ways can yield the reward improvement over the neighborhood of the origin policy.

Divergence Projection Step: to accelerate the training process, we leverage the baseline policy π_B and project $\pi_{\theta_{k+\frac{1}{3}}}$ onto the region around π_B. The distance between $\pi_{\theta_{k+\frac{1}{3}}}$ and π_θ is minimized using a distance measure D as

$$\pi_{\theta_{k+\frac{2}{3}}} = \arg\min_{\pi_\theta \in \Pi_\theta} D(\pi_\theta, \pi_{\theta_{k+\frac{1}{3}}}) \tag{8}$$

$$s.t.\ J^D(\pi_{\theta_k}) + \frac{1}{1-\tilde{\gamma}}\mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)}[A^D_{\pi_{\theta_k}}(s,a)]$$
$$+\beta_k\sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}}[D_{KL}(\pi_{\theta_k}, \pi_\theta)[s]]} \le h^D. \tag{9}$$

To avoid the convex approximation like CPO and PCPO, the primal-dual approach is developed as

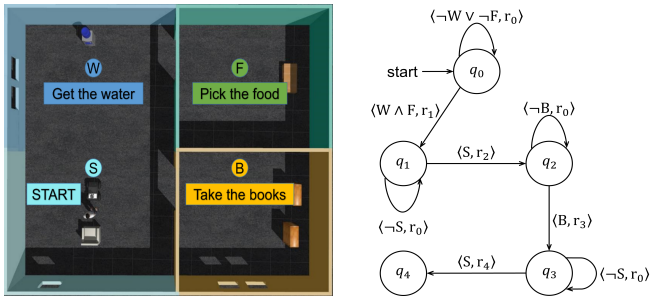
$$(\pi_{\theta_{k+\frac{2}{3}}}, v_{k+1}) = \arg\min_{\pi_\theta \in \Pi_\theta}\max_{v\ge 0} \hat{L}_D(\pi_\theta, \pi_{\theta_k}, \theta_{k+\frac{1}{3}}), \tag{10}$$

$$\hat{L}_D(\pi_\theta, \pi_{\theta_k}, \theta_{k+\frac{1}{3}}) = D_{KL}(\pi_{\theta_{k+\frac{1}{3}}}, \pi_\theta) + v_k\hat{D}(\pi_\theta, \pi_{\theta_k}). \tag{11}$$

The divergence constraint function is

$$\hat{D}(\pi_\theta, \pi_{\theta_k}) = J^D + \frac{1}{1-\tilde{\gamma}}\mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)}A^D_{\pi_{\theta_k}}(s_t, a_t)$$
$$+\beta_k\sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}}[D_{KL}(\pi_{\theta_k}, \pi_\theta)[s]]} - h^D. \tag{12}$$

Constrain projection Step: To ensure policy safety, we project $\pi_{\theta_{k+\frac{2}{3}}}$ onto the safe constrain set in this step. Similar to the last step, the distance between $\pi_{\theta_{k+\frac{2}{3}}}$ and π_θ is



(a) **(b)**

Fig. 2: (a) The simulation environment. (b) The reward machine.

minimized as

$$\pi_{\theta_{k+1}} = \arg\min_{\pi_\theta \in \Pi_\theta} D(\pi_\theta, \pi_{\theta_{k+\frac{2}{3}}}) \tag{13}$$

$$s.t.\ J^C(\pi_{\theta_k}) + \frac{1}{1-\tilde{\gamma}} \mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)}[A^C_{\pi_{\theta_k}}(s,a)] + \eta_k \sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}}[D_{KL}(\pi_{\theta_k}, \pi_\theta)[s]]} \leq h^C. \tag{14}$$

Then we solve the following problem

$$(\pi_{\theta_{k+1}}, \mu_{k+1}) = \arg\min_{\pi_\theta \in \Pi_\theta} \max_{\mu \geq 0} \hat{L}_C(\pi_\theta, \pi_{\theta_k}, \theta_{k+\frac{2}{3}}), \tag{15}$$

$$\hat{L}_C(\pi_\theta, \pi_{\theta_k}, \theta_{k+\frac{2}{3}}) = D_{KL}(\pi_{\theta_{k+\frac{2}{3}}}, \pi_\theta) + \mu_k \hat{C}(\pi_\theta, \pi_{\theta_k}). \tag{16}$$

The safety constraint function is

$$\hat{C}(\pi_\theta, \pi_{\theta_k}) = J^C + \frac{1}{1-\tilde{\gamma}} \cdot \mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)} A^C_{\pi_{\theta_k}}(s_t, a_t) + \eta_k \sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}}[D_{KL}(\pi_{\theta_k}, \pi_\theta)[s]]} - h^C. \tag{17}$$

As for the coefficient v and μ, there are two ways to update:

$$\nu_{k+1} = \nu_k + \eta_\nu (J^D - h_D)_+, \tag{18}$$

$$\mu_{k+1} = \mu_k + \eta_\mu (J^C - h_C)_+, \tag{19}$$

or treat them as parameters of the neural network with the loss functions

$$L_\nu = -\nu * \eta_\nu (J^D - h_D), \tag{20}$$

$$L_\mu = -\mu * \eta_\mu (J^C - h_C). \tag{21}$$

C. Reward Machine

When encountering more challenging tasks (e.g., long-horizon tasks with complex temporal and logic constraints), we encode the task as a linear temporal logic (LTL) formula. Then, we build a reward machine [25] based on the LTL formula to facilitate agent learning.

Definition 3.1 (Reward Machine). Given a tuple $\langle S, A, \mathcal{P}, L\rangle$ where S is a finite set of environment states, A is a set of actions, $\mathcal{P}$ is a finite set of propositional symbols, and L is a labeling function: $L : S \times A \times S \to 2^{\mathcal{P}}$. A reward machine (RM) is a tuple $\langle Q, q_0, \Sigma, \mathcal{R}, \delta, \rho\rangle$ where Q is a finite set of automata states, $q_0 \in Q$ is the initial state, $\Sigma = 2^{\mathcal{P}}$ is the input alphabet, and $\mathcal{R}$ is a finite set where each $R : S \times A \times S \to \mathbb{R}$ in $\mathcal{R}$ is a reward function.

Example 1. Consider a robot tasked to deliver the water and food to the people, and then pick up the books and send them to the people, as shown in Fig. 2. Such a task can be written in an LTL formula as $((\neg start)\mathsf{U}books) \wedge ((\neg books)\mathsf{U}start) \wedge ((\neg start)\mathsf{U}(water \wedge food)) \wedge (\Diamond(water \wedge food))$. A reward machine can be defined over a set of propositional symbols $\mathcal{P} = \{water, food, books, start\}$, where $o \in \mathcal{P}$ occurs when the agent is at location o. The states of reward machine is $Q = \{q_0, q_1, q_2, q_3, q_4\}$ and the reward function set is $\mathcal{R} = \{r_0, r_1, r_2, r_3, r_4\}$ as shown in Fig. 2. The q_0 means the agent haven't got the water and food, and thus the reward is a negative value r_0. If the agent has obtained the water and food, the state of reward machine comes to q_1 and the reward of this transition is r_1. Now the robot needs to send the water and food back to the people. If the agent hasn't go back to S, no matter where it goes, it always stays in state q_1 and gets the negative reward r_0. Once it reaches S, the positive reward r_2 is received, and the RM state comes to q_2. Similarly, only if the agent gets the books, the state can be transited to q_3. The agent then gets the reward r_3 and goes back to the people with state transiting to q_4 to complete the whole task. Otherwise it can only get a negative reward r_0 and stay in q_3 or q_4.

IV. Experiment

In this section, our method is evaluated against recent representative methods. Particularly, we investigate 1) the policy performance: whether our method improves the performance of learned policy in terms of reward collection and convergence; 2) the safety guarantee: whether our method outperforms previous methods in terms of safety guarantee in both training and deployment stages; 3) the task completion: whether our method can complete the tasks.

A. Experiment Setup

1) Environments: To verify the advantages of our method in different environments, we consider two cases. Case 1 is a relatively simple environment with three obstacles as shown in Fig. 3(a)-(b). The LTL task is $((\neg area_y)\mathsf{U}area_b) \wedge (\Diamond area_b)$, which requires the robot to first reach the blue area and then the yellow area. Case 2 is a more challenging environment as shown in Fig. 3(d)-(e), which requires the agent to reach area b, y, and g sequentially, while avoiding the obstacles. Such a task can be written in LTL as $((\neg area_g)\mathsf{U}area_y) \wedge ((\neg area_y)\mathsf{U}area_b) \wedge (\Diamond area_b)$.

The reward machines for the above two cases are shown in Fig. 3(c) and Fig. 3(f), respectively. In the experiment, we set $r_0 = -1$, $r_1 = r_2 = 20$, and $r_3 = 100$. The agent is rewarded based on the accomplishment of subtasks, and punished if no progress is made. The cost is designed as

$$c = \begin{cases} \frac{50}{1+e^{10(d-d_{safe})}}, & \text{if } d > d_{safe}, \\ 50, & \text{if } d \leq d_{safe}, \end{cases}$$

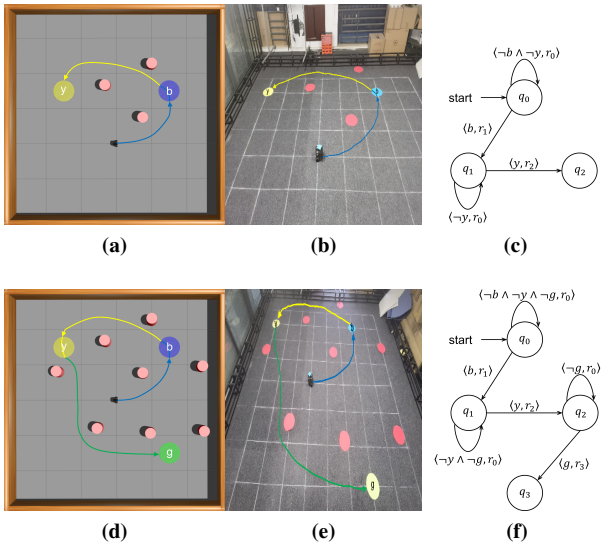


Fig. 3: The top row indicates Case 1 and the bottom row indicates Case 2. (a) The simple environment in simulation. (b) The corresponding physical environment. (c) The reward machine. (d) The challenging environment with more obstacles. (e) The corresponding physical environment. (f) The reward machine.

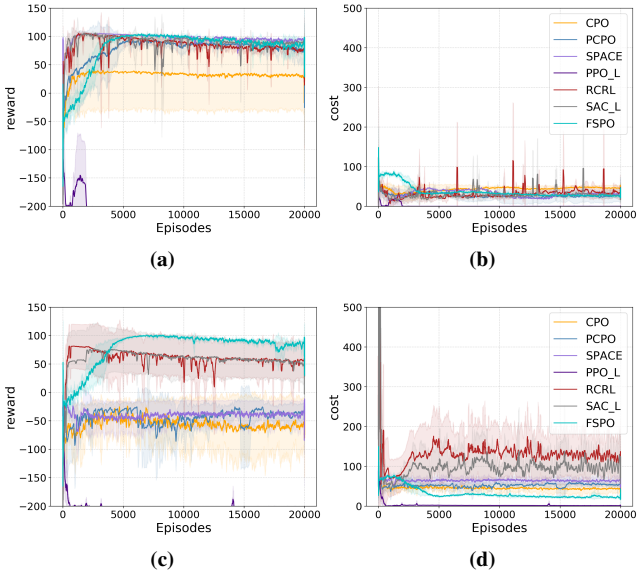


Fig. 4: The reward and cost curves of different algorithm. (a) and (b) show the reward and cost curves for Case 1, while (c) and (d) show the performance for Case 2.

which indicates that, the closer the agent to the obstacle, the greater the cost it will get. If it collides with an obstacle, a large penalty will be received.

2) Baseline: We consider five baselines: constrained policy optimization (CPO) [1], projection-based constrained policy optimization (PCPO) [2], safe policy adaptation with constrained exploration (SPACE) [16], reward constrained policy optimization (RCPO) [4], reachability constrained reinforcement learning (RCRL) [3]. As for RCPO, we use PPO and SAC as the RL base algorithms, denoted as PPO_Lagrangian (PPO_L) and SAC_Lagrangian (SAC_L), respectively. To show the effect of divergence projection step, besides FSPO, we also consider SPO, a simplified version of FSPO in the ablation experiment that only includes the reward improvement step and constrain projection step without the divergence projection step. To compare with the above different update methods, we use FSPO with neural network updated Lagrangian coefficient (FSPO_N) and FSPO with manually updated Lagrangian coefficient (FSPO_M). Similarly, SPO_M and SPO_N are considered.

B. Main Result

This section presents the comparison results of the policy performance, the safety rate, and the task completion rate between our method and the baseline methods. The role of the divergence projection step and the training process of Lagrangian coefficients are also discussed.

1) Safety rate during learning: For Case 1, Fig. 4(a) and (b) show that FSPO converges slower than the baseline methods, but performs similarly to the baseline methods in terms of the reward collection and the cost performance. This is mainly due to the fact that Case 1 is a relatively simple case, for which the baseline methods can generally perform well. However, when encountering more challenging environments and tasks, as in Case 2 with more obstacles and more complex task requirements, the strength of our method emerges. As shown in Fig. 4(c) and (d), the reward of FSPO is higher than baseline algorithms while the cost is significantly lower the others. Note that the SPACE cannot even complete the task. A possible explanation is that, since we do not use the convex approximation method, it results in higher reward and lower cost with high safety guarantees.

2) Safety rate during and after training: We compare the safety rate and task completion rate during and after training of our algorithm with the baseline algorithms for Case 2. We choose the last 2000 episodes in the training to show the results after training. Both SPO and FSPO have high safety rate and task completion rate. Our algorithm also shows high safe guarantees. It is worth mentioning that, although PPO_L shows the highest safety rate, it cannot complete the task.

3) Ablation experiment: As mentioned in (18)-(21), there are two ways to update the Lagrangian coefficients. To verify the effect of the divergence projection step, we compare the performance between the following four algorithms: FSPO_M, FSPO_N, SPO_M and SPO_A. As shown in Fig. 5, FSPO and SPO have similar performance for Case 1. When encountering more challenging Case 2, SPO converges slowly even if it has high reward and low cost at last. However, the convergence of FSPO is greatly improved.

Since Lagrangian coefficient determines the trade off between the safety and performance, we first show the training process of Lagrangian coefficients during the learning process. As shown in Fig. 5(a) and (d), the Lagrangian coefficient updated by neural network slowly decreases as the training progresses. But when updating it

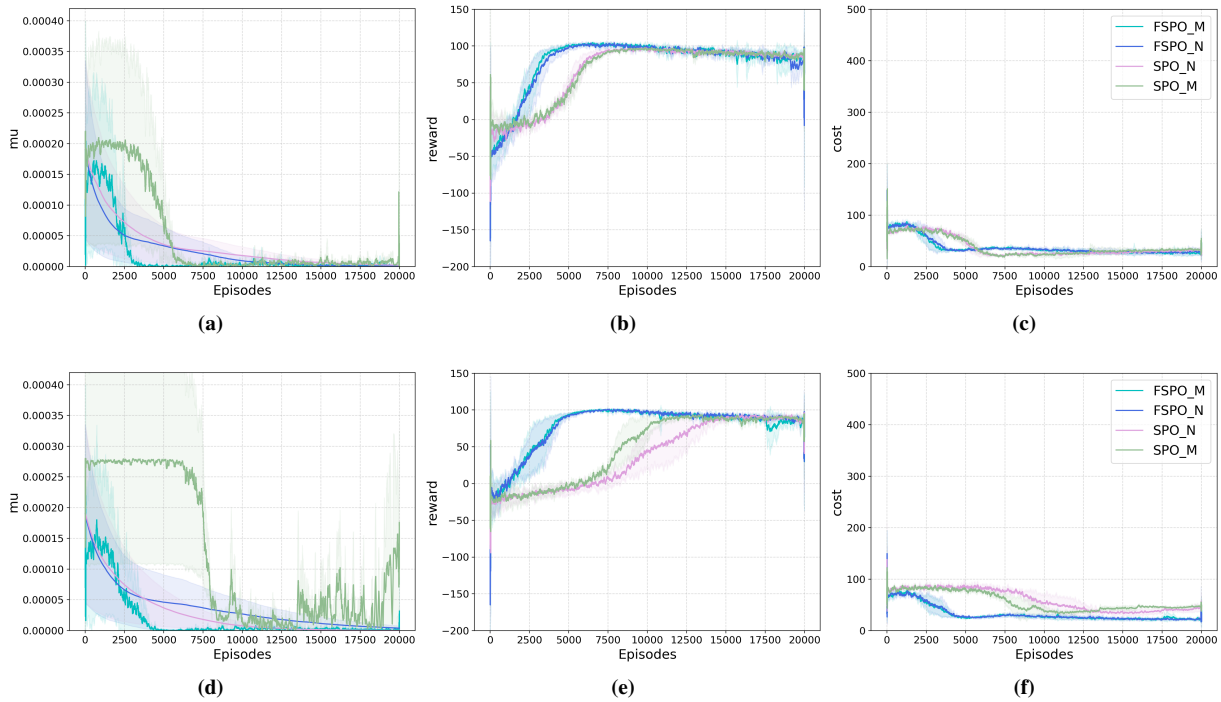


Fig. 5: The evolution of the Lagrangian coefficient, the reward, and the cost using FSPO_M, FSPO_N, SPO_M, and SPO_N. The top row shows the result for Case 1 while the bottom row shows the result for Case 2.

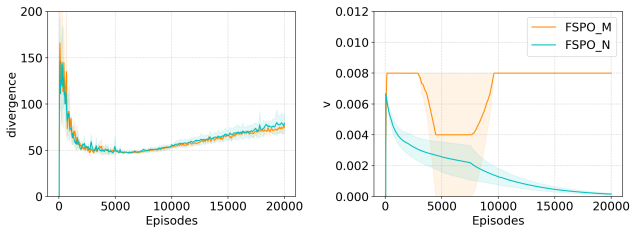


Fig. 6: The evolution of the divergence cost and the Lagrangian coefficient ν for Case 2.

manually, it stays at a high value when the agent is not safe enough. Once the agent has been trained to be sufficiently safe, the value decreases fast. The performance of FSPO_M is slightly better than FSPO_N.

4) Divergence cost of FSPO: The divergence cost means the divergence between the policy and the baseline policy π_B. We show the divergence cost and the Lagrangian coefficients of step 2 in Fig. 6a and Fig. 6b, respectively. It is observed that the value is high at the beginning, and then quickly decreases after some episodes. It is speculated that such a process can help accelerate the training at the beginning and increase the upper limit of the algorithm in the end.

C. Physical Experiment

The effectiveness of our approach is also verified in physical environment for Case 1 and 2 using a turtlebot. As shown in Fig. 3(b) and (e), the turtlebot can safely complete the tasks. The experiment video with more explanations is provided[1].

TABLE I: Safety rate and task completion rate

Method	Final Safety Rate	Task Completion Rate	Safety Rate during Training
CPO [1]	43.15%	0%	39.50%
PCPO [2]	52.75%	0%	49.89%
SPACE [16]	11.1%	0%	25.16%
PPO_L [4]	99.46%	0%	98.69%
SAC_L [4]	84.70%	84.31%	57.47%
RCRL [3]	87.27%	85.38%	63.42%
SPO_M(Ours)	94.97%	94.97%	58.46%
SPO_A(Ours)	93.25%	93.25%	51.81%
FSPO_M(Ours)	**95.10%**	**94.93%**	**86.53%**
FSPO_N(Ours)	**92.70%**	**91.97%**	**86.25%**

V. Conclusion

In this work, a Fast and Safe Policy Optimization (FSPO) algorithm is developed to improve the policy learning efficiency and performance while ensuring policy safety. Numerical simulation and physical experiments demonstrate the effectiveness of FSPO. Future research will consider extending FSPO to further ensure safety in both training and deployment.

[1]https://youtu.be/2RgaH-zcmk

REFERENCES

[1] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Int. Conf. Machin. Learn.* PMLR, 2017, pp. 22–31.

[2] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," in *Int. Conf. Learn. Represent.*, 2019.

[3] D. Yu, H. Ma, S. Li, and J. Chen, "Reachability constrained reinforcement learning," in *Int. Conf. Machin. Learn.* PMLR, 2022, pp. 25 636–25 655.

[4] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," *arXiv preprint arXiv:1805.11074*, 2018.

[5] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *IEEE Control Syst.*, 2014, pp. 6271–6278.

[6] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conf., ECC*, 2019, pp. 3420–3431.

[7] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, "Introducing sostools: A general purpose sum of squares programming solver," in *Proc IEEE Conf Decis Control*, vol. 1, 2002, pp. 741–746.

[8] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC-PapersOnLine*, vol. 40, no. 12, pp. 462–467, 2007.

[9] M. Saveriano and D. Lee, "Learning barrier functions for constrained motion planning with dynamical systems," in *IEEE Int Conf Intell Rob Syst*, 2019, pp. 112–119.

[10] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," in *Proc IEEE Conf Decis Control*, 2020, pp. 3717–3724.

[11] C. Zhang, S. Wang, S. Meng, and Z. Kan, "Safe exploration of reinforcement learning with data-driven control barrier function," in *Proceeding - China Autom. Congr., CAC*, 2022, pp. 1008–1013.

[12] E. Altman, *Constrained Markov decision processes*. CRC press, 1999, vol. 7.

[13] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained markov decision processes," in *Int. Conf. Machin. Learn.* PMLR, 2020, pp. 9797–9806.

[14] V. S. Borkar, "An actor-critic algorithm for constrained markov decision processes," *Syst Control Lett*, vol. 54, no. 3, pp. 207–213, 2005.

[15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Int. Conf. Machin. Learn.* PMLR, 2015, pp. 1889–1897.

[16] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Accelerating safe reinforcement learning with constraint-mismatched baseline policies," in *Int. Conf. Machin. Learn.* PMLR, 2021, pp. 11 795–11 807.

[17] Y. Zhang, Q. Vuong, and K. Ross, "First order constrained optimization in policy space," *Adv. neural inf. proces. syst.*, vol. 33, pp. 15 338–15 349, 2020.

[18] D. Ding, K. Zhang, T. Basar, and M. Jovanovic, "Natural policy gradient primal-dual method for constrained markov decision processes," *Adv. neural inf. proces. syst.*, vol. 33, pp. 8378–8390, 2020.

[19] H. Satija, P. Amortila, and J. Pineau, "Constrained markov decision processes via backward value functions," in *Int. Conf. Machin. Learn.* PMLR, 2020, pp. 8502–8511.

[20] F. Chen, J. Zhang, and Z. Wen, "A near-optimal primal-dual method for off-policy learning in cmdp," *Adv. neural inf. proces. syst.*, vol. 35, pp. 10 521–10 532, 2022.

[21] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible ltl constraints," *IEEE Trans. Autom. Control*, vol. 68, no. 1, pp. 301–316, 2021.

[22] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic ltl motion planning with environment and motion uncertainties," *IEEE Trans Autom Control*, vol. 66, no. 5, pp. 2386–2392, 2020.

[23] P. Vaezipoor, A. C. Li, R. A. T. Icarte, and S. A. Mcilraith, "Ltl2action: Generalizing ltl instructions for multi-task rl," in *Int. Conf. Machin. Learn.* PMLR, 2021, pp. 10 497–10 508.

[24] Z. Zhou, Z. Chen, M. Cai, Z. Li, Z. Kan, and C.-Y. Su, "Vision-based reactive temporal logic motion planning for quadruped robots in unstructured dynamic environments," *IEEE Trans. Ind. Electron.*, vol. 71, no. 6, pp. 5983–5992, 2024.

[25] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *Int. Conf. Machin. Learn.* PMLR, 2018, pp. 2107–2116.

[26] M. Cai and C.-I. Vasile, "Safe-critical modular deep reinforcement learning with temporal logic through gaussian processes and control barrier functions," *arXiv preprint arXiv:2109.02791*, 2021.

[27] M. H. Cohen, Z. Serlin, K. Leahy, and C. Belta, "Temporal logic guided safe model-based reinforcement learning: a hybrid systems approach," *Nonlinear Analysis: Hybrid Systems*, vol. 47, p. 101295, 2023.

[28] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *Int. Conf. Machin. Learn.*, 2002, pp. 267–274.

[29] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[30] M. Y. Vardi, "Automatic verification of probabilistic concurrent finite state programs," in *Proc. Annu. IEEE Symp. Found. Comput. Sci. FOCS*. IEEE, 1985, pp. 327–338.

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