Information Sciences 662 (2024) 120261

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Almost surely safe exploration and exploitation for deep reinforcement learning with state safety estimation

Ke Lin [a], Yanjie Li [a,*], Qi Liu [a], Duantengchuan Li [b], Xiongtao Shi [a], Shiyu Chen [a]

[a] Department of Control Science and Engineering, Harbin Institute of Technology Shenzhen, Shenzhen 518055, China
[b] School of Computer Science, Wuhan University, Wuhan 430072, China

ARTICLE INFO

Keywords:
Reinforcement learning
Risk-sensitive reinforcement learning
State safety estimation
Safe exploration
Gaussian process

ABSTRACT

This study aims to address the challenge of constrained reinforcement learning (CRL), which seeks to maximize cumulative rewards while making agents avoid risks. Existing CRL methods can typically ensure that the final trained policy could meet specified constraints. But during training process, the safety of the agent cannot be guaranteed. In this research, we put forward a safe proximal policy optimization (SPPO) method under the assumption of the continuity of states' safety values. The proposed SPPO can make agents achieve satisfactory cumulative rewards while realizing zero constraint violations in the learning phase. In our method, an uncertainty estimation technique of states' safety value is utilized to get a safe state set $\mathcal{H}_t$. Then, $\mathcal{H}_t$ is applied to intervene in the agent's exploration. As a consequence, the safety of the agent can be achieved. Moreover, we theoretically guarantee that the agent will not violate the safety constraint with almost 100% probability. Considerable empirical experiments demonstrate that the SPPO algorithm can achieve higher cumulative rewards and lower total cost (near zero) than existing state-of-the-art CRL approaches.

1. Introduction

Deep reinforcement learning (DRL) has achieved astounding success in kinds of decision-making problems, e.g., Go [1,2], video games [3,4], robotic control [5,6], and resource scheduling [7–9], where the goal is to maximize the cumulative rewards. Nevertheless, for risk-sensitive scenarios, it is usually unsatisfactory to merely maximize cumulative rewards. Making agents satisfy certain constraints is also necessary. For instance, in self-driving tasks, the agent should reach the goal while ensuring that it can avoid any collisions in movement.

Constrained reinforcement learning (CRL) is commonly utilized to tackle such risk-sensitive sequential decision-making problems. This kind of method can be divided into two categories according to the type of constraint: total cost constraint-based methods and stepwise constraint-based methods. The former usually model the problem as a constrained Markov decision process (CMDP) [10], in which the expected total cost needs to be less than a certain threshold. For the latter, the constrained form usually requires that the cost of each step is less than a threshold. Overall overviews of CRL can be found in García and Fernández [11] and Liu et al. [12].

To make the learned policy have risk-sensitive behaviors in CMDP-based reinforcement learning (RL) methods, researchers commonly integrate the total cost constraints with existing policy gradient-based algorithms, such as policy gradient (PG) [13], proximal policy optimization (PPO) [14], and trust region policy optimization (TRPO) [15]. Afterward, constrained optimization techniques

* Corresponding author.
E-mail address: autolyj@hit.edu.cn (Y. Li).

https://doi.org/10.1016/j.ins.2024.120261
Received 23 May 2023; Received in revised form 25 October 2023; Accepted 28 January 2024

Available online 1 February 2024
0020-0255/© 2024 Elsevier Inc. All rights reserved.

are introduced to deal with such CRL problems, e.g., the Lagrangian relaxation approach [16–19] and Karush-Kuhn-Tucker condition [20]. For stepwise-based CRL algorithms [21–24], the safety constraints are associated with each state. Therefore, to ensure that the agent is safe at each step, it is assumed that the state transition probabilities in environments are known, and a predictive model is used to evaluate whether the policy can meet the constraints in the future time steps. Then, a policy iteration method is applied to find a high-reward and safe policy. In addition, there are also Lyapunov-based [25] and barrier function-based [26] approaches to deal with CRL problems.

CMDP-based approaches are characterized by the fact that the finally learned policy can satisfy the constraints, but the agent will still violate the constraints in the learning phase, which makes the approaches difficult to be deployed in actual risk-sensitive scenarios with the sim-to-real paradigm. On the other hand, similar to the constrained model predictive control methods in control theory [27,28], stepwise-based approaches can guarantee the agents' safety at each time step in the policy learning process. However, it is usually required that the system model is known, i.e., the state transition of the system is known a priori. This paper will make efforts to deal with these issues.

In this study, a safe model-free RL approach is proposed, which can guarantee the agent's safety in the course of the learning process with high probability. Assuming that the safety value of states in the environment does not suddenly change and there is a recovery policy that can restore the state of the environment, our method can evaluate a state's safety at the next moment after an arbitrary action is taken. Thus, the proposed method can ensure the safety of agents. The essential difference from existing stepwise constrained-based methods is that our method does not require a priori known state transition. In addition, different from the CMDP-based methods, our approach can guarantee that the learning process is also safe. The main contributions are summarized as follows:

- The safe proximal policy optimization (SPPO) algorithm is proposed, which is able to learn a policy that can satisfy safety constraints and get high rewards in decision-making environments.
- With the uncertainty estimation for the safety value of each state using Gaussian process, we proposed the safe state set expansion mechanism and theoretically guaranteed that all states encountered by the agent are safe with high probability in the policy learning phase.
- Extensive experiments were conducted, which show that our method achieves superior performance to the existing typical CRL algorithms in terms of safety constraint satisfaction and cumulative rewards.

The remainder of this paper is structured as follows. Section 2 introduces related work. In Section 3, the preliminaries of RL and the Gaussian process (GP) are described. Section 4 gives a detailed description of the proposed SPPO algorithm. Empirical experimental results are presented in Section 5. Finally, Section 6 provides a conclusion and discusses future research directions.

2. Related work

2.1. CMDP-based CRL

Policy optimization approaches are usually used in CMDP-based CRL. Achiam et al. [20] proposed constrained policy optimization (CPO), in which the trust region technique is introduced to ensure the stability of the training process, and the original optimization problem is approximated as a convex optimization problem by Taylor expansion. Then, the Karush-Kuhn-Tucker condition is applied to obtain the next policy, which guarantees that the new policy satisfies constraints. Primal-dual policy optimization (PDO) [16] considers the dual problem of the original problem and employs the Lagrangian multiplier to obtain the hybrid objective function. Then, the gradient with regard to maximizing cumulative rewards and minimizing total costs is used to optimize the policy. Interior-point policy optimization (IPO) was proposed by Liu et al. [29], in which a logarithmic barrier function is applied as a penalty item to make the policy be updated in a direction that would both increase cumulative rewards and reduce total costs. Based on projection gradient methods, Yang et al. [30] put forward the projection-based constrained policy optimization (PCPO). During policy updates, a normal policy improvement step is performed whose goal is to increase the cumulative rewards of the policy. Then, an extra policy projection step is utilized to project the policy onto the feasible domain to make the policy satisfy the constraints. Tessler et al. [17] proposed the reward constrained policy optimization algorithm (RCPO), in which a reward shaping technique was used to penalize policies that fail to satisfy constraints. Experimental results show that RCPO can make the trained policy meet the safety constraints. Yu et al. [31] presented the convergent policy optimization approach, where the original non-convex constrained policy optimization problem is transformed into a series of convex optimization problems, which ensures that the constrained optimization problem can be solved quickly. Zhang et al. [18] proposed a non-parameterized policy search method to tackle the constrained policy optimization problem, named first order constrained optimization, which achieves good performance in constrained robot motion control tasks. Other constrained policy optimization methods include Xu et al. [32] and Ding and Lavaei [33].

Generally, such methods can only ensure that the final trained policy is safe, and the safety constraints cannot be satisfied during the learning process. However, in our research, we address this issue by introducing a safe state set to intervene in the agent's exploration.

2.2. Stepwise-based CRL

Stepwise-based CRL algorithms are usually able to ensure the safety of the agent at each step. Moreover, this feature also holds in the process of policy training. Berkenkamp et al. [21] proposed a safe model-based RL method, where Lyapunov stability presented in control-theoretic results is extended to model-based RL algorithms. Thus, a simulated inverted pendulum can be controlled without falling down, even in the learning phase. A similar framework also appeared in Fisac et al. [22], in which a least-restrictive, safety-preserving control law is safely trained by a policy-gradient method. As a consequence, a quadrotor can be controlled without any crashing. In addition, Turchetta et al. [23] proposed a safe exploration technique for the finite Markov decision process (MDP). Subsequently, Wachi and Sui [24] extended this framework into the stepwise CMDP methods. Polymenakos and colleagues [34] proposed a safe policy search algorithm for safe policy exploration. With a multi-step prediction for the future trajectory of the current policy and the safety checking of the predicted trajectory, the proposed method can judge the safety of a policy. Therefore, an adaptive exploration coefficient adjustment is applied to ensure safe exploration. On the other hand, Bottero et al. [35] proposed an information-theoretic criterion for safe exploitation and exploration, which keeps agents from violating the safety constraint in continuous domains. Moreover, Prajapat et al. [36] have applied similar concepts in the field of multi-agent safe exploration.

In general, in order to guarantee that an agent is safe at every step in the learning process, this kind of method usually requires a known system transition or a known structure of system transition. However, in this paper, we make safe learning possible in model-free settings by building a safe state set $\mathcal{H}_t$ using a state safety estimation model.

3. Preliminaries

3.1. RL with constraint

A CRL problem can be modeled as a safety constrained MDP, which can be defined as a tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, f, r, m, \gamma), \tag{1}$$

where $\mathcal{S}$ denotes a state space, and $\mathcal{A}$ denotes an action space. $f(\boldsymbol{s}, \boldsymbol{a})$: $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$ describes a deterministic transition from state $\boldsymbol{s}$ to next state $\boldsymbol{s}'$ under action $\boldsymbol{a}$. $r(\boldsymbol{s}, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a bounded reward function. $m(\boldsymbol{s}) : \mathcal{S} \to \mathbb{R}$ is a safety value function which measures the safety at state $\boldsymbol{s}$. $\gamma \in [0, 1)$ is a discount factor. Note that $r(\boldsymbol{s}, \boldsymbol{a})$ and $m(\boldsymbol{s})$ are not known a priori. Within the context of an MDP, a policy is a probability distribution denoted as $\pi(\boldsymbol{a}|\boldsymbol{s}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. More specifically, when building a policy with a neural network, the parameter of the neural network can be written as $\boldsymbol{\theta}$. Then, the overall policy can be denoted as $\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$, which describes the probability of taking action $\boldsymbol{a}$ at state $\boldsymbol{s}$. $\rho(\boldsymbol{s}) : \mathcal{S} \to \mathbb{R}$ defines the probability distribution of the initial state $\boldsymbol{s}$ in the environment. $\tau = \{\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{r}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{r}_1, ...\}$ is often used to represent a trajectory, which is obtained during the agent's interaction with the environment. Then, in this deterministic MDP, the probability of a trajectory occurring is

$$P_{\pi_\theta}(\tau) = \rho(\boldsymbol{s}_0) \prod_{t=0}^{\infty} \pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t). \tag{2}$$

In RL, the goal is to find a policy π_θ that is capable of maximizing the discounted cumulative reward

$$J_r(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim P_{\pi_\theta}}[R(\tau)], \tag{3}$$

where $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. To evaluate the performance of a policy π, value function $V^\pi(\boldsymbol{s})$ and $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$ are utilized in RL, which are defined as

$$V^\pi(\boldsymbol{s}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \middle| \boldsymbol{s}_0 = \boldsymbol{s}\right], \quad Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \middle| \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}\right]. \tag{4}$$

In addition, advantage function $A^\pi(\boldsymbol{s}, \boldsymbol{a})$ is given as

$$A^\pi(\boldsymbol{s}, \boldsymbol{a}) = Q^\pi(\boldsymbol{s}, \boldsymbol{a}) - V^\pi(\boldsymbol{s}), \tag{5}$$

which measures how much a certain action $\boldsymbol{a}$ is better than the current policy at state $\boldsymbol{s}$. In the stepwise constrained setting, the agent must be in safe states at each time step t, which means the safety value must be above than a threshold h, i.e., $m(\boldsymbol{s}_t) \geq h, \forall t \geq 0$. Note that the threshold value h is determined by the environment. Thus, we formulate the problem to be solved as follows:

$$\begin{aligned} &\max_{\boldsymbol{\theta}} J_r(\boldsymbol{\theta}) \\ &\text{s.t. } m(\boldsymbol{s}_t) \geq h, \quad \forall t \geq 0. \end{aligned} \tag{6}$$

Furthermore, safe exploration and exploitation should also be achieved in the process of policy learning.

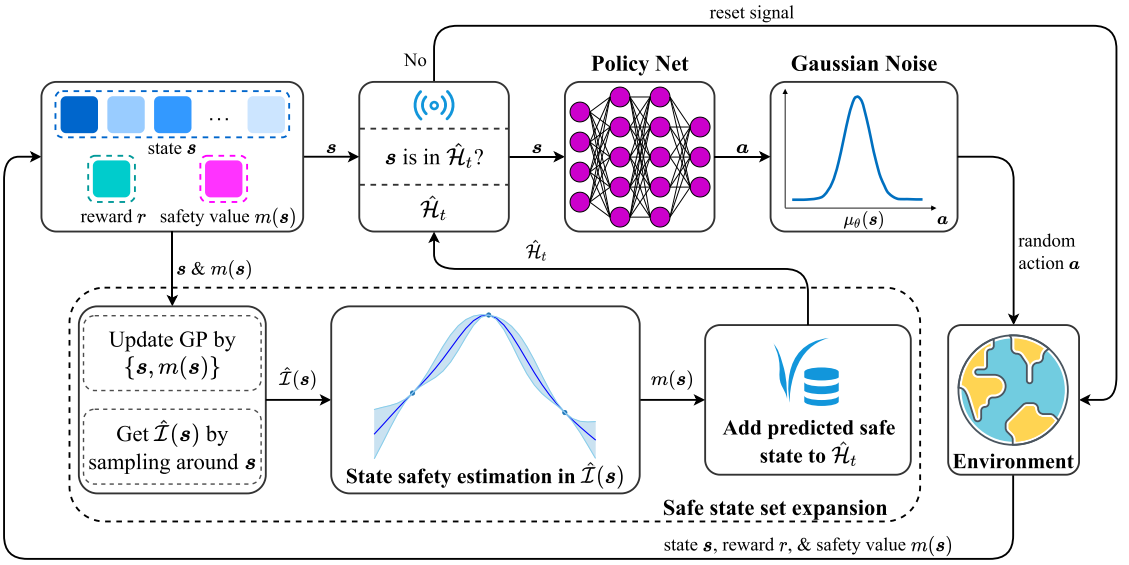


Fig. 1. An illustration of the proposed safe DRL algorithm.

3.2. Gaussian processes

Mean and covariance function $m(s), k(s, s')$ are two important components in a Gaussian process (GP), where $s, s' \in S$ [37]. Thus, a GP model is usually denoted as $\mathcal{GP}\left(m(s), k(s, s')\right)$. Moreover, a covariance function (also named kernel function) $k(s, s')$ is used to represent the distance between two different states. In this paper, the squared exponential function is utilized as the kernel function

$$k(s, s') = \exp\left(-\frac{D(s, s')^2}{2a^2}\right), s, s' \in S, \tag{7}$$

where a is a length-scale parameter, and $D(s, s')$ is the Euclidean distance metric. Given $t+1$ observations of states $\mathcal{D} = \{s_i, y_i\}_{i=0,1,\ldots,t}$, at time step t, the predicted value $\hat{m}_t(s_*)$ of a new state s_* that does not appear in $\mathcal{D}$, and the corresponding covariance $k_t(s_*, s)$ and variance $\sigma_t^2(s_*)$ are

$$\begin{cases} \hat{m}_t(s_*) & = \boldsymbol{k}_t(s_*)^\top \left(\boldsymbol{K}_t + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}_t, \\ k_t(s_*, s) & = k(s_*, s) - \boldsymbol{k}_t(s_*)^\top \left(\boldsymbol{K}_t + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_t(s), \\ \sigma_t^2(s_*) & = k_t(s_*, s_*), \end{cases} \tag{8}$$

where $\boldsymbol{k}_t(s_*) = [k(s_0, s_*), k(s_1, s_*), ..., k(s_t, s_*)]^\top$. In addition, $\boldsymbol{K}_t$ is the kernel matrix generated by $k(s_{i=0:t}, s_{j=0:t})$. σ_n^2 represents the noise level, and $\boldsymbol{y} = [y_0, y_1, ..., y_t]^\top$.

4. Methodology

Overall, we aim to utilize state safety estimation to make the agent's exploration safe and train a final safe policy. The overview of the proposed safe DRL method is shown in Fig. 1. In each step of the interaction, the environment will output a reward r, a state s, and the safety value $m(s)$ corresponding to the state s. Then, a state safety estimation model (a GP model) will be updated by the data $\{s, m(s)\}$. States around s, named $\hat{\mathcal{I}}(s)$, will be checked by the GP model to inspect if they are safe. If it is true, the state s will be added to the safe state set $\mathcal{H}_t$ and the interaction process will continue. Otherwise, the environment will receive a reset signal and be reset to the initial state. Under such an interaction mechanism, the safety of exploration can be achieved.

In this section, we adopt a GP as the state safety estimator. We first introduce the update process of a safe state set $\mathcal{H}_t$. The characteristic of the state in $\mathcal{H}_t$ is that when the agent takes an action arbitrarily at state $s \in \mathcal{H}_t$, the agent's safety is still theoretically guaranteed with high probability. Since $f(s, a)$ is unknown in RL, we then introduce a practical safe state set expansion procedure. Moreover, the safety value $m(s)$ is also not known a priori, so the agent needs to fully explore and exploit the surrounding of the safe state set $\mathcal{H}_t$. Thus, we combine the practical safe state set expansion procedure with the PPO method [14] and give the SPPO algorithm.

Assumption 1. A recovery policy is known that enables environments to recover to its initial state.

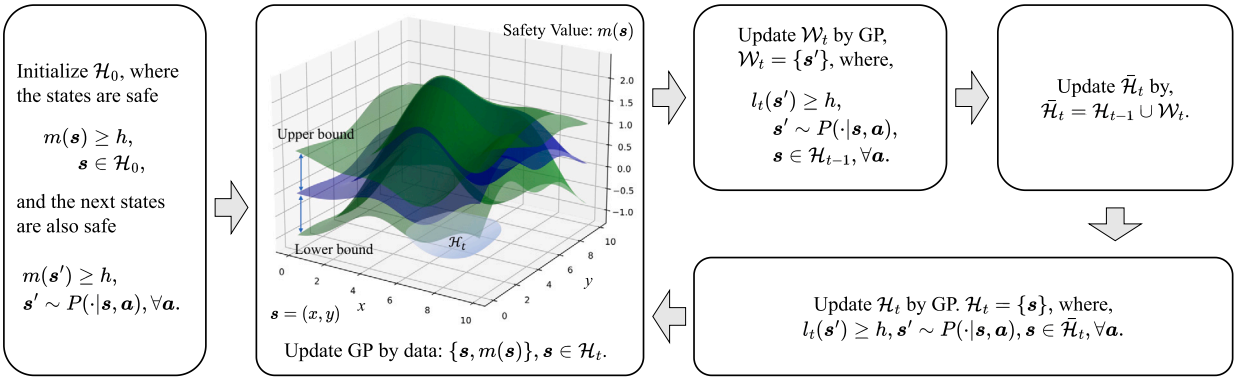


Fig. 2. An overview of the process of the safe state set expansion.

4.1. Safe state set expansion with GP

To achieve safe exploration, an overview of the safe state set expansion process is shown in Fig. 2. To predict the safety values of unknown states from the safety value of known states, we need to assume that the safety value of adjacent states will not suddenly change. Therefore, we give the following assumption:

Assumption 2. The state space $\mathcal{S}$ is endowed with a positive definite kernel function and the safety value function $m(s)$ has bounded norm in the associated reproducing kernel Hilbert space. In addition, $m(s)$ is L-Lipschitz continuous.

Indeed, it is noteworthy that the L-Lipschitz continuous assumption is inherently met by some frequently utilized kernel functions [38]. A GP is used to fit and predict the safety values for each state in the environment. To utilize the GP, we define the predicted lower bound $l_t(s), t \in [1, \infty)$ as

$$l_t(s) = \begin{cases} \hat{m}_{t-1}(s) - \beta_t^{1/2}\sigma_{t-1}(s), & t = 1, \\ \max\left\{ l_{t-1}(s), \hat{m}_{t-1}(s) - \beta_t^{1/2}\sigma_{t-1}(s) \right\}, & t > 1, \end{cases} \tag{9}$$

where $\hat{m}_t(s)$ and $\sigma_t(s), t \in [0, \infty)$ are the predicted safety value and the corresponding predicted variance by the GP model at time step t, respectively. $\beta_t, t \in [1, \infty)$ is a scalar, and we will discuss it later. We expect that the initial state set $\mathcal{H}_0$ is safe, which means states in $\mathcal{H}_0$ are safe, and the next states by taking any action from state in $\mathcal{H}_0$ are also safe. Thus, we give the following assumption:

Assumption 3. For initial safe state set $s \in \mathcal{H}_0$, we assume $m(s) \geq h$, $l_1(s) = \hat{m}_0(s) - \beta_1^{1/2}\sigma_0(s) \geq h$, and $l_1(s') \geq h, s' = f(s,a), \forall a \in \mathcal{A}$.

Under the definition of l_t, we update the safe state set $\mathcal{H}_t, t \geq 1$ by the following way:

$$\mathcal{W}_t = \left\{ s' \mid l_t(s') \geq h, s' = f(s,a), s \in \mathcal{H}_{t-1}, \forall a \in \mathcal{A} \right\}, \tag{10}$$

$$\bar{\mathcal{H}}_t = \mathcal{H}_{t-1} \cup \mathcal{W}_t, \tag{11}$$

$$\mathcal{H}_t = \left\{ s \in \bar{\mathcal{H}}_t \mid l_t(s') \geq h, s' = f(s,a), \forall a \in \mathcal{A} \right\}. \tag{12}$$

After we expand $\mathcal{H}_{t-1}$ and get $\mathcal{H}_t$, we need to update the GP model by all data in $\{(s, m(s)) \mid s \in \mathcal{H}_t\}$. Under such way of updating the safe state set $\mathcal{H}_t$, we have the following conclusions:

Proposition 1. *For all $t > 1$, $l_t(s) \geq l_{t-1}(s), \forall s \in \mathcal{S}$.*

Proof. This can be directly obtained by Equation (9). □

Lemma 1. *For all $t \geq 1$, $\mathcal{H}_{t-1}$ is a subset of $\mathcal{H}_t$, i.e., $\mathcal{H}_{t-1} \subseteq \mathcal{H}_t$.*

Proof. When $t = 1$, let $s \in \mathcal{H}_0$. By the definition of $\mathcal{H}_0$ (Assumption 3), we can get

$$l_1(s') \geq h, s' = f(s,a), \forall a \in \mathcal{A}.$$

By the definition of $\bar{\mathcal{H}}_t$ shown in Equation (11), we know that $\mathcal{H}_0 \subseteq \bar{\mathcal{H}}_1$. Thus, we have $s \in \bar{\mathcal{H}}_1$. According to the definition of $\mathcal{H}_t$ (Equation (12)), we have that $s \in \mathcal{H}_1$, which means $\mathcal{H}_0 \subseteq \mathcal{H}_1$.

When $t = k, k > 1$, let $\boldsymbol{s} \in \mathcal{H}_{k-1}$. By the definition of $\mathcal{H}_{k-1}$, we can obtain

$$l_{k-1}(\boldsymbol{s}') \geq h, \boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a}), \forall \boldsymbol{a} \in \mathcal{A}.$$

By utilizing Proposition 1, we have $l_k(\boldsymbol{s}') \geq l_{k-1}(\boldsymbol{s}') \geq h, \boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a}), \forall \boldsymbol{a} \in \mathcal{A}$. Moreover, we can have $\boldsymbol{s} \in \mathcal{H}_{k-1} \subseteq \bar{\mathcal{H}}_k$ (by the definition of $\bar{\mathcal{H}}_t$). Therefore, $\boldsymbol{s} \in \mathcal{H}_k$, which means $\mathcal{H}_{k-1} \subseteq \mathcal{H}_k$. □

Lemma 2. *Suppose that $\|m\|_p^2 \leq B$ and the noise n_t is conditionally U-sub-Gaussian [39]. When we set $\beta_t = B + U\sqrt{2(\zeta_{t-1} + 1 + \ln 1/\delta)}$ where δ can be an arbitrary real number in $(0, 1]$, and $\zeta_t = \max_{|A|=t} M(m_A; \boldsymbol{y}_A)$, then the inequality $|m(\boldsymbol{x}) - \hat{m}_{t-1}(\boldsymbol{x})| \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \forall t \geq 1$ holds with probability at least $1 - \delta$.*

Proof. This Lemma can be directly reached from Theorem 2 in [39]. □

Proposition 2. *The inequality $m(\boldsymbol{s}) \geq l_t(\boldsymbol{s}), \boldsymbol{s} \in \mathcal{S}, \forall t \geq 1$ holds with probability at least $1 - \delta$.*

Proof. We prove it by induction. When $t = 1$, $l_1(\boldsymbol{s}) = \hat{m}_0(\boldsymbol{s}) - \beta_1^{1/2}\sigma_0(\boldsymbol{s})$, so by Lemma 2 we have

$$\Pr\left\{m(\boldsymbol{s}) \geq l_1(\boldsymbol{s})\right\} = \Pr\left\{m(\boldsymbol{s}) \geq \hat{m}_0(\boldsymbol{s}) - \beta_1^{1/2}\sigma_0(\boldsymbol{s})\right\} \geq 1 - \delta.$$

Assume that when $t = k$, the following inequality holds:

$$\Pr\left\{m(\boldsymbol{s}) \geq l_k(\boldsymbol{s})\right\} \geq 1 - \delta.$$

Thus, when $t = k + 1$, by Lemma 2 we have

$$\Pr\left\{m(\boldsymbol{s}) \geq \hat{m}_k(\boldsymbol{s}) - \beta_{k+1}^{1/2}\sigma_k(\boldsymbol{s})\right\} \geq 1 - \delta.$$

Since

$$l_{k+1}(\boldsymbol{s}) = \max\left\{l_k(\boldsymbol{s}), \hat{m}_k(\boldsymbol{s}) - \beta_{k+1}^{1/2}\sigma_k(\boldsymbol{s})\right\},$$

we can obtain

$$\Pr\left\{m(\boldsymbol{s}) \geq l_{k+1}(\boldsymbol{s})\right\} \geq 1 - \delta.$$

Therefore, we have $\Pr\left\{m(\boldsymbol{s}) \geq l_t(\boldsymbol{s})\right\} \geq 1 - \delta, \forall t \geq 1$. □

Lemma 3. *For all $t \geq 1$, the following inequalities hold with probability at least $1 - \delta$.*

(1) $m(\boldsymbol{s}) \geq h, \forall \boldsymbol{s} \in \mathcal{H}_t$,
(2) $m(\boldsymbol{s}') \geq h, \boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a}), \forall \boldsymbol{s} \in \mathcal{H}_t, \forall \boldsymbol{a} \in \mathcal{A}$.

Proof. We first prove (1) by induction. Let us start from the case when $t = 0$. By the definition of $\mathcal{H}_0$ we have

$$m(\boldsymbol{s}) \geq h, \boldsymbol{s} \in \mathcal{H}_0.$$

Then, when $t = k$, let us assume

$$\Pr\{m(\boldsymbol{s}) \geq h\} \geq 1 - \delta, \boldsymbol{s} \in \mathcal{H}_k.$$

When $t = k + 1$, let $\boldsymbol{s}_\dagger \in \mathcal{H}_{k+1}$. By the definition of $\mathcal{H}_t$, we have

$$\boldsymbol{s}_\dagger \in \bar{\mathcal{H}}_{k+1} = \mathcal{H}_k \cup \mathcal{W}_{k+1}.$$

If $\boldsymbol{s}_\dagger \in \mathcal{H}_k$, by assumption of the case when $t = k$, it yields that

$$\Pr\left\{m(\boldsymbol{s}_\dagger) \geq h\right\} \geq 1 - \delta.$$

If $\boldsymbol{s}_\dagger \in \mathcal{W}_{k+1}$, by the definition of $\mathcal{W}_t$, we have $l_{t+1}(\boldsymbol{s}_\dagger) \geq h$. Thus, it is concluded that

$$\begin{aligned} m(\boldsymbol{s}_\dagger) &\geq l_{t+1}(\boldsymbol{s}_\dagger) \quad \text{(with probability at least } 1 - \delta, \text{ by Proposition 2)} \\ &\geq h. \end{aligned}$$

Therefore, we have $\Pr\{m(\boldsymbol{s}) \geq h\} \geq 1 - \delta, \boldsymbol{s} \in \mathcal{H}_t, \forall t \geq 1$. Then, we prove (2). Let $\boldsymbol{s} \in \mathcal{H}_t, t \geq 1, \boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a}), \forall \boldsymbol{a} \in \mathcal{A}$, we have

Algorithm 1 Safe Proximal Policy Optimization with GP.

Input: Given safe state set $\mathcal{H}_0$ and the corresponding safety value $m(\boldsymbol{s})$. Initial parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ of actor and critic networks. An empty data buffer $\mathcal{D}$.
Procedure:

```
Set t = 1.
Build the GP model by (s, m(s)), s ∈ H_0.
for each training epoch do
    for each time step do
        Observe a state s and take an action a ~ π_θ(· | s).
        Observe the next state s', reward r, safety value m(s'), and done signal d_s.
        Store (s, a, s', r, m(s'), d_s) in D.
        Update the GP model by new data (s', m(s')).
        Update the Ĥ_t by Equation (15) using the updated GP model.
        if s' not in Ĥ_t then
            Set d_s = True.
        end if
        Set t = t + 1.
        if d_s is True then
            Reset the environment state.
        end if
    end for
    Update θ by Equation (16).
    Update φ by Equation (18).
    Clear the buffer D.
end for
```

Output: π_θ.

$$m(\boldsymbol{s}') \geq l_t(\boldsymbol{s}') \quad \text{(with probability at least } 1-\delta, \text{ by Proposition 2)}$$
$$\geq h. \quad \text{(by definition of } \mathcal{H}_t, \text{Equation (12))}$$

Therefore, one can obtain that $\Pr\left\{m(\boldsymbol{s}') \geq h\right\} \geq 1-\delta, \boldsymbol{s}' = f(\boldsymbol{s},\boldsymbol{a}), \boldsymbol{s} \in \mathcal{H}_t, t \geq 1$. □

Theorem 1. *Assume the safety function satisfies $\|m\|_p^2 \leq B$, $m(\boldsymbol{s})$ is L-Lipschitz continuous, and the noise n_t in m is conditionally U-sub-Gaussian. Also, suppose $\mathcal{H}_0 \neq \emptyset$, and $l_1(\boldsymbol{s}) \geq h, l_1(\boldsymbol{s}') \geq h, \boldsymbol{s}' = f(\boldsymbol{s},\boldsymbol{a}), \boldsymbol{s} \in \mathcal{H}_0, \forall \boldsymbol{a} \in \mathcal{A}$, where h is a safety threshold given by environments. If we let $\beta_t = B + U\sqrt{2(\zeta_{t-1} + 1 + \ln 1/\delta)}$ and update $\mathcal{H}_t$ by Equation (12), then the set $\mathcal{H}_t$ expands gradually, and during this expansion process, states are safe with high probability, i.e., $\Pr\{m(\boldsymbol{s}) \geq h\} \geq 1-\delta, \boldsymbol{s} \in \mathcal{H}_t, \forall t \geq 1$ and $\Pr\left\{m(\boldsymbol{s}') \geq h\right\} \geq 1-\delta, \boldsymbol{s}' = f(\boldsymbol{s},\boldsymbol{a}), \boldsymbol{s} \in \mathcal{H}_t, \forall \boldsymbol{a} \in \mathcal{A}, \forall t \geq 1$.*

Proof. By Lemma 1, we have that the set $\mathcal{H}_t$ expands gradually, and by Lemma 3 we have that $\Pr\{m(\boldsymbol{s}) \geq h\} \geq 1-\delta, \boldsymbol{s} \in \mathcal{H}_t, \forall t \geq 1$ and $\Pr\left\{m(\boldsymbol{s}') \geq h\right\} \geq 1-\delta, \boldsymbol{s}' = f(\boldsymbol{s},\boldsymbol{a}), \boldsymbol{s} \in \mathcal{H}_t, \forall \boldsymbol{a} \in \mathcal{A}, \forall t \geq 1$. □

Remark 1. From Theorem 1, if we set δ to a real number which is close to 0, we can know that given an initial safe state set $\mathcal{H}_0$, if we update $\mathcal{H}_t$ by Equation (12), all states $\boldsymbol{s}$ in $\mathcal{H}_t$ are safe with high probability (at least $1-\delta$), i.e., $m(\boldsymbol{s}) \geq h, \boldsymbol{s} \in \mathcal{H}_t$. In addition, at these states, it is still safe to take any action with high probability, i.e., $m(\boldsymbol{s}') \geq h, \boldsymbol{s}' = f(\boldsymbol{s},\boldsymbol{a}), \boldsymbol{s} \in \mathcal{H}_t, \forall \boldsymbol{a} \in \mathcal{A}$.

Note that δ can be set to different values for different environments to make sure that the meaning of "safe with high probability" fits the context of different scenarios. If an environment is extremely risk-sensitive, the value of δ can be set very close to 0 (e.g., 10^{-8}). Conversely, the value of δ can be set slightly larger, such as 0.001.

4.2. Practical safe state set expansion

In the setting of standard RL, $f(\boldsymbol{s},\boldsymbol{a})$ is unknown to the agent, so it is infeasible to directly use Equation (12) to expand $\mathcal{H}_t$ during exploration. Therefore, we first define the neighborhood of the state $\boldsymbol{s}$ as

$$\mathcal{I}(\boldsymbol{s}) = \left\{\boldsymbol{u} \mid \|\boldsymbol{s}-\boldsymbol{u}\|_2 \leq \nu\right\}, \tag{13}$$

where ν is a real number. To obtain $\mathcal{I}(\boldsymbol{s})$ in continuous state spaces, we can obtain an approximation $\hat{\mathcal{I}}(\boldsymbol{s})$ by sampling around $\boldsymbol{s}$ in all directions with different step lengths.

Assumption 4. We assume $D(\boldsymbol{s},\boldsymbol{s}') \leq \nu$ holds, for all $\boldsymbol{s}' = f(\boldsymbol{s},\boldsymbol{a}), \forall \boldsymbol{s} \in \mathcal{S}, \forall \boldsymbol{a} \in \mathcal{A}$, where $D(\cdot,\cdot)$ is the Euclidean distance metric, and ν is a positive real number.

According to Assumption 4, we can know that for all $\boldsymbol{s}' = f(\boldsymbol{s},\boldsymbol{a}), \forall \boldsymbol{a} \in \mathcal{A}$, we have $\boldsymbol{s}' \in \hat{\mathcal{I}}(\boldsymbol{s})$. Therefore, as long as we can make $l_t(\boldsymbol{s}') \geq h, \boldsymbol{s}' \in \hat{\mathcal{I}}(\boldsymbol{s})$ hold, the agent can take any action at state $\boldsymbol{s}$, and we can ensure that the next state $\boldsymbol{s}'$ is safe. Also, noted that the acquisition of $\hat{\mathcal{I}}(\boldsymbol{s})$ does not depend on $f(\boldsymbol{s},\boldsymbol{a})$. Consequently, we can expand the surrogate safety state set $\hat{\mathcal{H}}_t$ by the following form with a model-free style:

Table 1
Algorithm Level Parameters in Experiments.

Parameter	Ours	PPO [14]	CPO [20]	IPO [29]	PCPO [30]
Activation function	Tanh	Tanh	Tanh	Tanh	Tanh
GAE λ^{GAE}	0.97	0.97	0.97	0.97	0.97
Number of hidden layers	2	2	2	2	2
Number of hidden nodes	64	64	64	64	64
Learning rate of the policy net	3e-4	3e-4	N/A	3e-4	N/A
Learning rate of the reward critic net	1e-3	1e-3	1e-3	1e-3	1e-3
Learning rate of the cost critic net	N/A	N/A	1e-3	1e-3	1e-3
Clip ratio	0.2	0.2	N/A	0.2	N/A
Penalty factor	N/A	N/A	N/A	1e-8	N/A
Discount factor γ	0.99	0.99	0.99	0.99	0.99
Line search coefficient	N/A	N/A	0.9	N/A	N/A
Line search accept ratio	N/A	N/A	0.1	N/A	N/A
Max Kullback-Leibler divergence	N/A	N/A	0.01	0.01	0.01

Table 2
Environment Level Parameters in Experiments.

Task	Safe-Reach	Safe-Circle	Safe-Push	Safe-Reach-DO
Epochs	500	1000	5000	2000
Steps per epoch	500	500	1000	500
Safety value threshold h	-0.8	5.0	0.0	-0.8
Max cost threshold	0	0	0	0
Max episode length	50	50	100	50

$$\hat{\mathcal{W}}_t = \left\{ \boldsymbol{s}' \mid l_t(\boldsymbol{s}') \geq h, \boldsymbol{s}' \in \hat{\mathcal{I}}(\boldsymbol{s}), \boldsymbol{s} \in \hat{\mathcal{H}}_{t-1} \right\}, \tag{14}$$

$$\hat{\mathcal{H}}_t = \left\{ \boldsymbol{s} \in \hat{\mathcal{W}}_t \cup \hat{\mathcal{H}}_{t-1} \mid l_t(\boldsymbol{s}') \geq h, \boldsymbol{s}' \in \hat{\mathcal{I}}(\boldsymbol{s}) \right\}, \tag{15}$$

where $t \geq 1, \hat{\mathcal{H}}_0 = \mathcal{H}_0$. After we get $\hat{\mathcal{H}}_t$, we also need to update the GP model by all data in $\{(\boldsymbol{s}, m(\boldsymbol{s})) \mid \boldsymbol{s} \in \hat{\mathcal{H}}_t\}$.

4.3. Safe proximal policy optimization algorithm

Consider the fact that $r(\boldsymbol{s}, \boldsymbol{a})$ and $m(\boldsymbol{s})$ are not known a priori in typical RL settings. Agents need to explore environments to obtain this information. In order to achieve safe exploration and exploitation, we consider incorporating the procedure of the practical safe state set expansion into the existing state-of-the-art PPO algorithm [14]. In the exploration and exploitation phase, the agent gets the reward information and the safety value of visited states, and a GP model is used to evaluate the safety of unobserved states to expand the safe state set $\mathcal{H}_t$. By the reset mechanism in the environment, the agent's safety in the learning phase is achieved. The overall description is shown in Fig. 1.

Then, same as in PPO, at the policy learning phase, the optimization objective is

$$\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\substack{\boldsymbol{s}\sim d^{\pi_\theta} \\ \boldsymbol{a}\sim\pi_{\theta_k}}} \left[\min \left\{ \frac{\pi_\theta(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\theta_k}(\boldsymbol{a}|\boldsymbol{s})} A^{\pi_{\theta_k}}(\boldsymbol{s}, \boldsymbol{a}), \text{clip}\left(\frac{\pi_\theta(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\theta_k}(\boldsymbol{a}|\boldsymbol{s})}, 1-\epsilon, 1+\epsilon \right) A^{\pi_{\theta_k}}(\boldsymbol{s}, \boldsymbol{a}) \right\} \right], \tag{16}$$

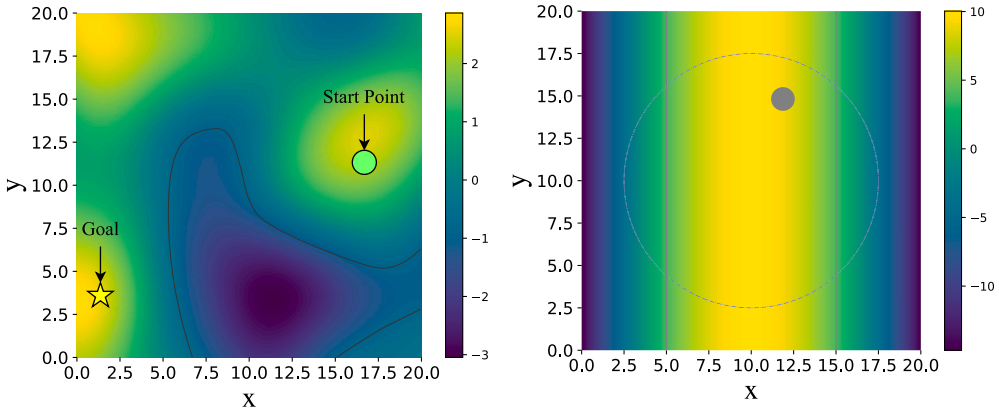
where $d^{\pi_\theta}(\boldsymbol{s}) = \sum_{t=0}^{\infty} \gamma^t P_{\pi_\theta}(\boldsymbol{s}_t = \boldsymbol{s})$. Also, to better estimate $A^\pi(\boldsymbol{s}, \boldsymbol{a})$ and reduce the estimated variance, we adopt generalized advantage estimation trick [40] and estimate $A^\pi(\boldsymbol{s}, \boldsymbol{a})$ by

$$\hat{A}^\pi(\boldsymbol{s}_t, \boldsymbol{a}_t) = \sum_{t'=t}^{\infty} (\gamma\lambda)^{t'-t} \delta_{t'}, \tag{17}$$

in which $\delta_{t'} = r(\boldsymbol{s}_{t'}, \boldsymbol{a}_{t'}) + \gamma \hat{V}_\phi^\pi(s_{t'+1}) - \hat{V}_\phi^\pi(s_{t'})$, and $\hat{V}_\phi^\pi$ is a critic network parameterized by $\boldsymbol{\phi}$ to estimate V^π. Then, $\boldsymbol{\phi}$ can be updated by

$$\boldsymbol{\phi}_{k+1} = \arg\min_{\boldsymbol{\phi}} \sum_{\tau} \sum_{t=0}^{\infty} \left(\hat{V}_\phi(\boldsymbol{s}_t) - \hat{r}_t \right)^2, \tag{18}$$

where $\hat{r}_t$ is the reward-to-go, which can be obtained by $\hat{r}_t = r_t + \gamma \hat{V}_\phi(\boldsymbol{s}_{t+1})$. Thus, we can summarize the proposed SPPO method in Algorithm 1. According to Theorem 1, we can know that the proposed SPPO algorithm is still safe in the process of the learning phase.



(a) Safe-Reach environment. (b) Safe-Circle environment.

Fig. 3. Two risk-sensitive 2-D world environments: Safe-Reach and Safe-Circle.

5. Experiment

5.1. Experimental configuration

In this section, we evaluate the SPPO algorithm in four environments, including: (1) two locomotion control environments, called Safe-Reach and Safe-Circle; (2) a PyBullet-based [41] environment, named Safe-Push; (3) a Safe-Reach-based 2-D world environment with dynamic obstacles, named Safe-Reach-DO. In these environments, the reward information r_t and the safety value m_t at each time step t are given. Also, cost information c_t is also given to indicate whether the agent is in a safe state, i.e., $c_t = \mathbf{1}\{m(\boldsymbol{s}_t) < h\}$, where $\mathbf{1}\{\cdot\}$ is the indicator function. The experimental parameters of the algorithm level are shown in Table 1.

In each environment, the proposed algorithm is compared with the following state-of-the-art methods: a DRL algorithm, PPO [14], and three CRL algorithms, including CPO [20], PCPO [30], and IPO [29]. All experimental results are averaged on 10 different random seeds. The experimental parameters at the environment level are shown in Table 2.

To compare the performances of different algorithms, we adopt average total reward $J_r(\pi)$ and average total cost $J_c(\pi)$ as the evaluation metrics:

$$J_r(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right], J_c(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} c(\boldsymbol{s}_t, \boldsymbol{a}_t) \right], \tag{19}$$

where in addition to maximizing the average total reward like in typical RL, we also aim to minimize the average total costs, which means the total cost should equal to 0.

In addition, different from ordinary DRL algorithms where an action's mean and variance are directly output by the policy network, in our experiment, the policy network of algorithms only outputs the mean of the action, and the action variance will gradually decrease with the exploration process for more stable training. Rewards with curiosity mechanisms [42] are also added to all algorithms for faster convergence.

5.2. 2-D world locomotion control environments

5.2.1. Environment description

The overall description of the two risk-sensitive 2-D world environments is shown in Fig. 3. Among them, the heat map represents all states' safety value. Note that if a state's safety value is less than a certain threshold, it means that the state is not safe.

In the Safe-Reach environment, the goal of the agent is to arrive the target from the start point while avoiding going through the unsafe area. The solid lines in Fig. 3 (a) illustrate the boundary between the safe area and the unsafe area. In this environment, the state space is the current position of the agent (x, y), and the action space is set to the movement distance $(\Delta x, \Delta y)$ along the two axes with a maximum step size of 1. The agent is rewarded for arriving the goal (+1000) and moving to the goal point (distance difference from the goal compared to the last time step). Also, the agent is punished by moving forward ($-0.05 \times$ movement length). In each time step, the agent will also receive a safety value about the current state.

The Safe-Circle environment is mainly based on an environment presented in CPO [20]. In a space of size 20×20, the agent should move along a circle (dotted line in Fig. 3 (b)) while avoiding entering unsafe areas. The state space is also the current position of the agent (x, y), and the action space is identical to that of the Safe-Reach environment. The reward function is designed as

$$r_t = \frac{-\Delta x \cdot (y - 10) + \Delta y \cdot (x - 10)}{1 + \left| \sqrt{(x-10)^2 + (y-10)^2} - 7.5 \right|}. \tag{20}$$

The boundary between the safe area and the unsafe area is shown by the solid line in Fig. 3 (b).

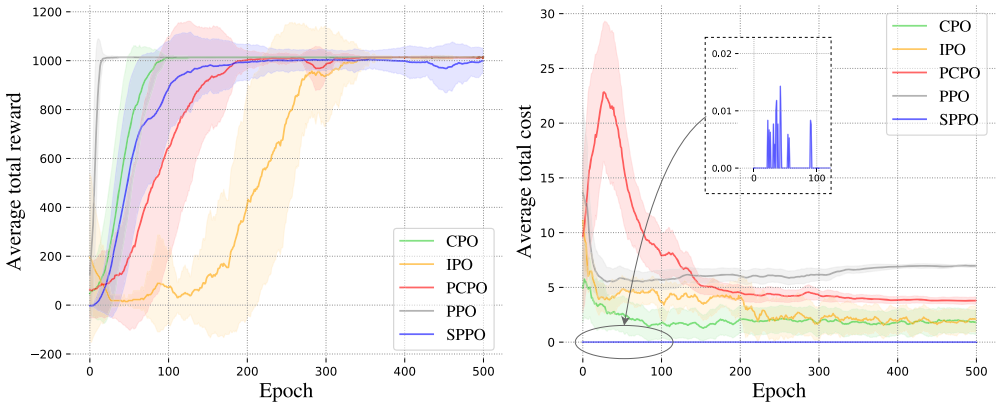


Fig. 4. Comparison of average total reward and average total cost along with policy updates of PPO, CPO, IPO, PCPO, and the proposed SPPO in the Safe-Reach environment.

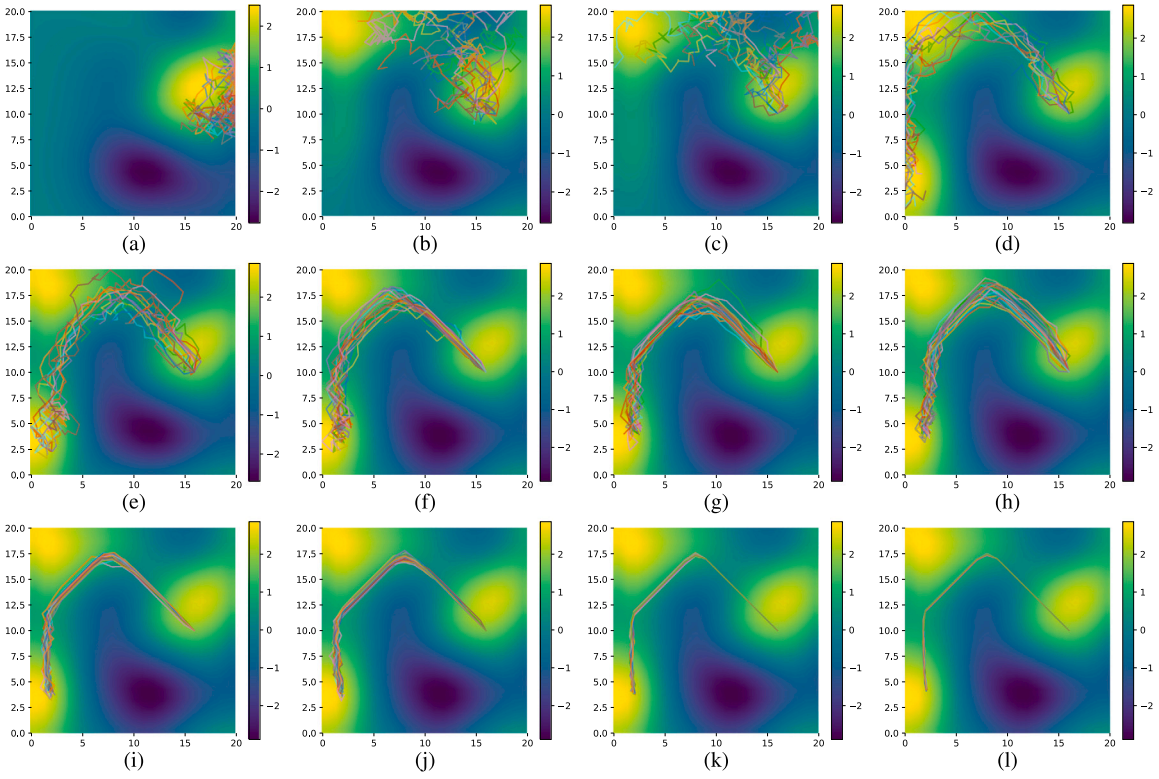


Fig. 5. The trajectory of the agent in the Safe-Reach environment under different epochs, where the heat map represents the current perception of the environment by the agent. The subfigures (a) to (l) represent the agent's trajectories in the 1, 10, 20, 30, 40, 60, 80, 100, 200, 300, 400, and 500 epochs, along with the policy learning process, respectively.

5.2.2. Empirical results in safe-reach environment

Fig. 4 shows the empirical results of the comparison among PPO, CPO, PCPO, IPO, and the proposed SPPO algorithm with metrics of average total reward and average total cost along with policy updates in the Safe-Reach environment.

It can be observed from Fig. 4 that PPO exhibits the most rapid convergence and achieves the highest total reward in the Safe-Reach environment. Nevertheless, since the agent's safety is not considered in PPO, the final total cost of PPO is also the highest. The CPO method also achieves a fast convergence speed. However, for the CRL-based methods such as CPO, IPO, and PCPO, during the policy update process, since the max cost threshold is set to a rather small value (e.g., $h = 0$), it is hard for the learned policy to satisfy the constraint who can only make the total cost as small as possible, which is close to 2 shown in Fig. 4.

The convergence speed of our method is faster than IPO and PCPO algorithms and also achieves a near-optimal total reward. Furthermore, the average total cost of the proposed method is always close to 0 in the whole exploration process, which means the agent only enters the unsafe states in a small number of times.

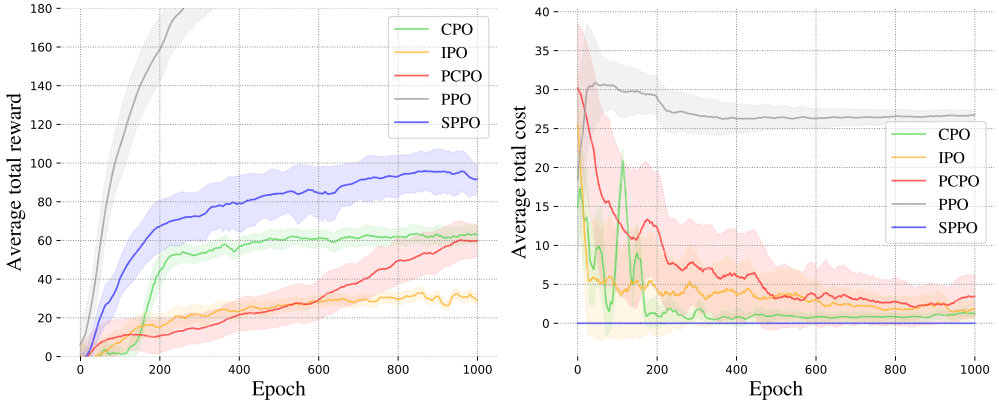


Fig. 6. Comparison of average total reward and average total cost along with policy updates of PPO, CPO, IPO, PCPO, and the proposed SPPO in the Safe-Circle environment.

To visualize whether the agent is safe during exploitation, Fig. 5 shows the exploration trajectories of the agent at different stages along with the policy learning. At the beginning of training, since the agent's perception of the environment is still relatively limited, the agent randomly explores the area near the start point, as shown in subfigures (a) to (c). The agent only explores areas (states) it deems safe. As the exploration progresses, the agent gradually becomes more aware of the environment and gradually explores the goal point, which means the safe state set is gradually expanding. Although the agent can safely reach the target, the variance of the agent's trajectory is still large at this stage, shown in subfigures (d) to (h). Finally, with the further deepening of the exploration, the trajectory of the agent gradually becomes deterministic, as illustrated in subfigures (i) to (l). Note that during the entire exploration process, the agent barely goes to unsafe states.

5.2.3. *Empirical results in safe-circle environment*

We also compared the algorithms mentioned earlier in the Safe-Reach environment. As demonstrated in Fig. 6, PPO attains the highest total reward but also gets the highest total cost. On the other hand, compared with the CRL-based methods, our method can achieve larger returns while maintaining a high probability of safety.

We also visualize the trajectory of the agent in the circle environment during training, as shown in Fig. 7. In the subfigures (a) to (l), the agent's trajectories gradually changed from random exploration to a circle, and the variance of trajectories gradually became smaller. Notably, during the whole exploration process, the agent does not enter unsafe areas.

5.3. *PyBullet-based dynamic control environment*

The Safe-Push environment is based on a physics simulation engine named PyBullet [41]. Fig. 8 shows the description of the environment in subfigure (a), and the definition of the safety values in subfigure (b). The boundaries between the safe areas and the unsafe areas are represented using solid lines. The agent is aimed at pushing the car (gray) into the goal area (green) while ensuring that the car is not pushed into unsafe areas (red areas). To achieve this, the agent (green cylinder) is rewarded for pushing the car to the right ($+1000 \times \Delta x$) and making the car reach the target ($+1000$). The state space is defined by the positional coordinates of both the agent and the car. The action space is defined as the force exerted on the agent.

Fig. 9 shows the comparison results among PPO, CPO, PCPO, IPO, and the proposed SPPO methods in the Safe-Push environment, using the metrics of average total reward and average cumulative cost. It can be observed that our method is close to PPO in terms of convergence speed and final total reward, and the performance is significantly better than CPO, IPO, and PCPO. In addition, the agent trained by our method is in safe areas with high probability. There are only a few cases where the state is unsafe, shown in subfigure (b).

In the training process, the car's trajectories (not the agent) are also visualized in Fig. 10. At the beginning of the training, the agent pushes the car forward in random directions. As the training progresses, the agent gradually pushes the car along the s-shaped safe area to the goal region. Moreover, the car's trajectories are getting smoother and smoother in the training course.

Noted that the agent is able to make good estimates of the safety values of those states that have already been seen and the states near the visited states. Conversely, for those who are far from the states that have already been explored, the estimate is less accurate. This phenomenon can be shown in subfigure (l), in which the safety values of the s-shaped area and its nearby areas are well-fitted, but the safety values of other areas are not well estimated. However, this feature does not affect the safety of the agent's exploration.

5.4. *Environment with dynamic obstacle*

Based on the Safe-Reach environment, we built a more complex environment named Safe-Reach-DO shown in Fig. 11, in which a dynamic obstacle moves periodically to interfere with the agent, and the agent cannot touch the obstacle during

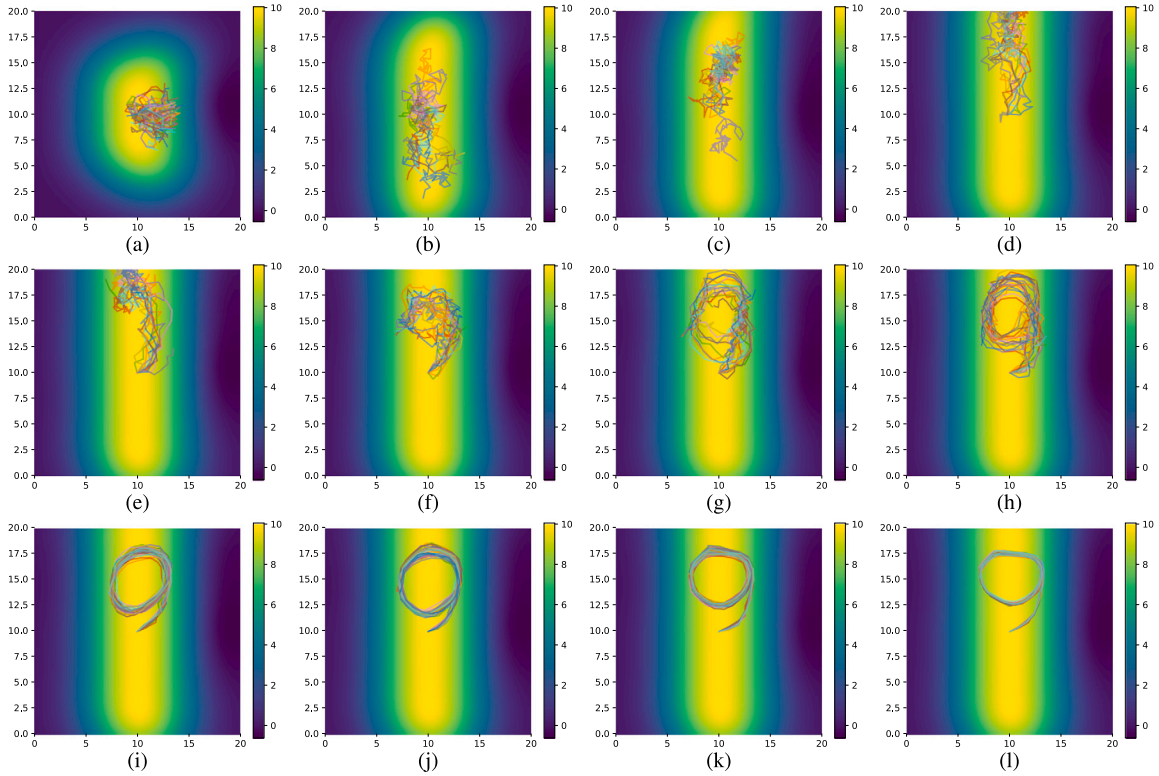


Fig. 7. The trajectory of the agent in the Safe-Circle environment under different epochs. The subfigures (a) to (l) represent the agent's trajectories in the 1, 10, 20, 30, 40, 60, 80, 100, 200, 300, 400, and 500 epochs, along with the policy learning process, respectively.

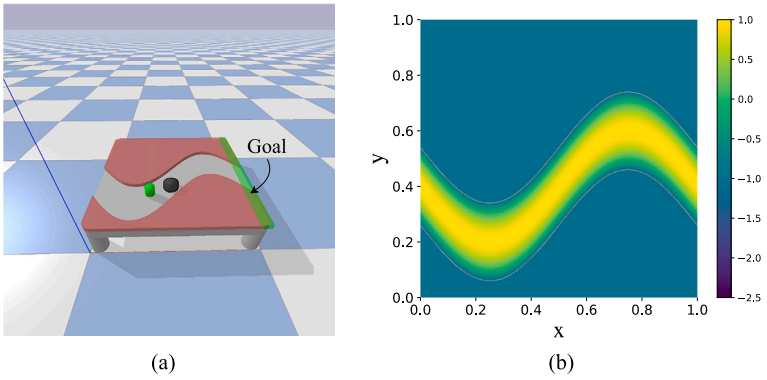


Fig. 8. The Safe-Push environment based on PyBullet, which is a physics simulation engine.

its movement. The state space is defined by the positional coordinates of both dynamic obstacles and the agent. As for the action space and reward design, they are identical to those in the Safe-Reach environment. An additional safety value function $m^{do}(\boldsymbol{s})$ is given in this environment to indicate the distance between the agent and the dynamic obstacle. Specifically, the farther away the agent from the obstacle, the corresponding safety value $m^{do}(\boldsymbol{s})$ is greater. Thus, the cost is defined as $c_t = \mathbf{1}\{\text{the agent is in safe areas and does not collide with the dynamic obstacle}\}$.

Under this setting, we constructed two safe state sets to ensure a high probability of agent safety during exploration. Fig. 12 shows the comparison results among PPO, CPO, PCPO, IPO, and the proposed SPPO methods in the Safe-Reach-DO environment, using reward and cost as metrics. Our methodology still achieves a nearly 0 total cost in environments with dynamic obstacles. As for the cumulative reward, our method can also get a high return. As the environment becomes more complex, more training is required for converging to a near-optimal policy.

Thus, we can conclude that whether in locomotion control environments or dynamic control environments, the proposed SPPO algorithm can achieve zero constraint violations in the course of exploration and exploitation with high probability.

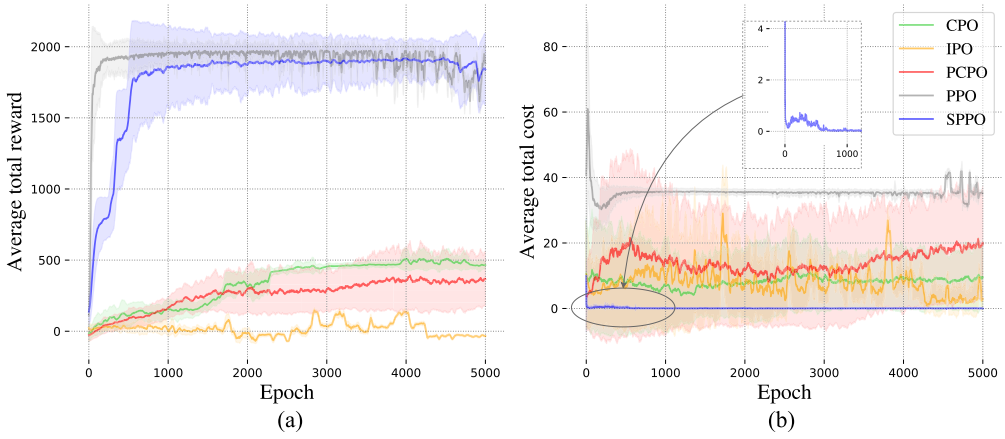


Fig. 9. Comparison of average total reward and average total cost along with policy updates of PPO, CPO, IPO, PCPO, and the proposed SPPO in the Safe-Push environment.

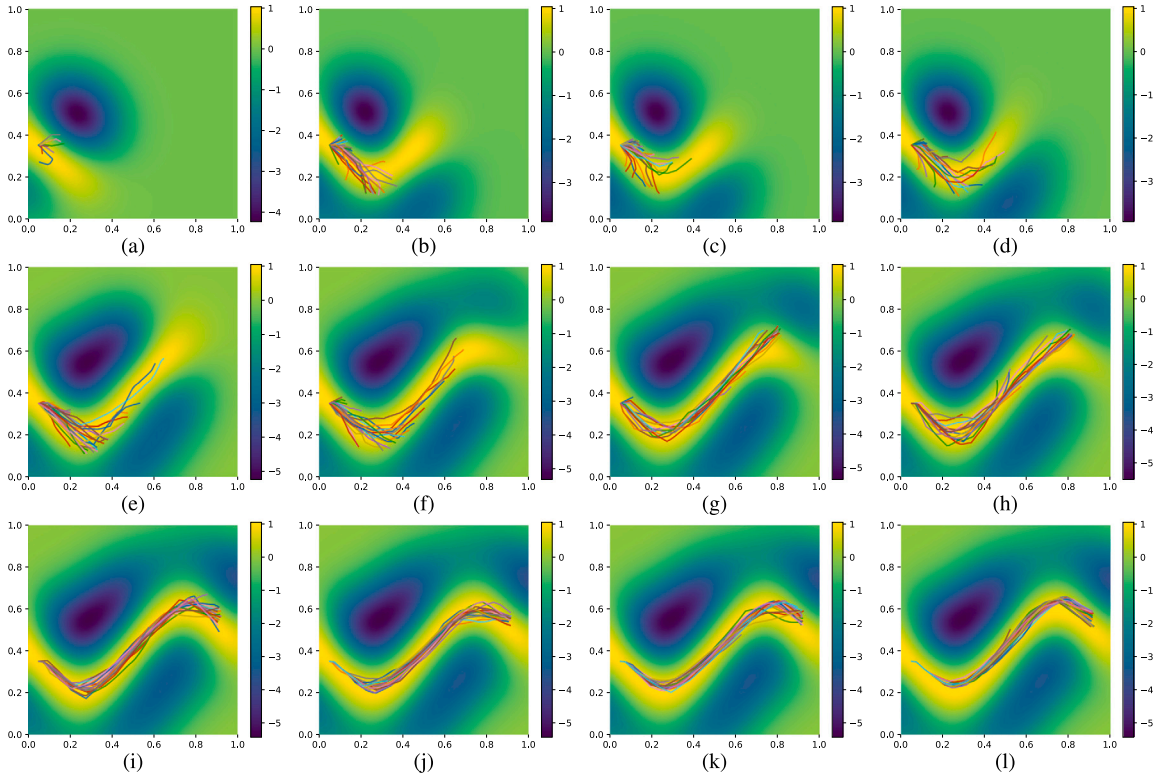


Fig. 10. The trajectory of the car (not the agent) in Safe-Push environment under different epochs. The subfigures (a) to (l) represent the car's trajectories in the 1, 20, 40, 60, 100, 400, 700, 1000, 2000, 3000, 4000, 5000 epochs along with the policy learning process, respectively.

5.5. Blind alley case

To further explore the behavior of the proposed method in the case of a blind alley, the environment shown in Fig. 13 (a) is constructed, in which the start and target points of the agent are in two separate safe areas. As a result, the agent will never reach the target point safely.

During the learning process, the trajectory of the agent is shown in Fig. 13 (b) - (d). As the exploration progresses, the action variance of the agent gradually decreases, but the agent can only move in the safe area all the time. Therefore, for safety, the agent cannot reach the target point.

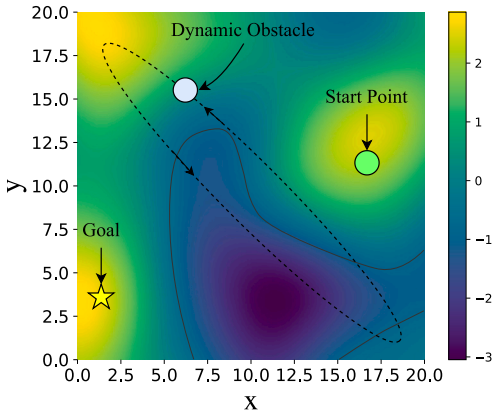


Fig. 11. The schematic diagram of the Safe-Reach-DO environment.

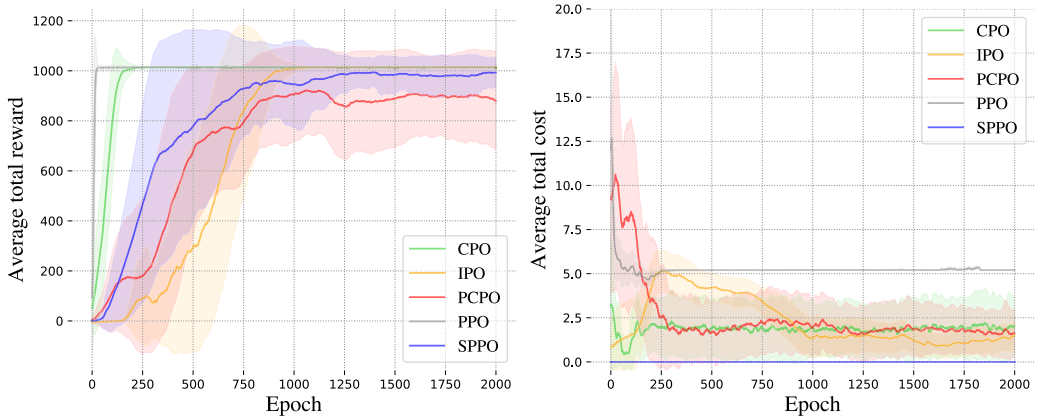


Fig. 12. Comparison of average total reward and average total cost along with policy updates of PPO, CPO, IPO, PCPO, and the proposed SPPO in the Safe-Reach-DO environment.

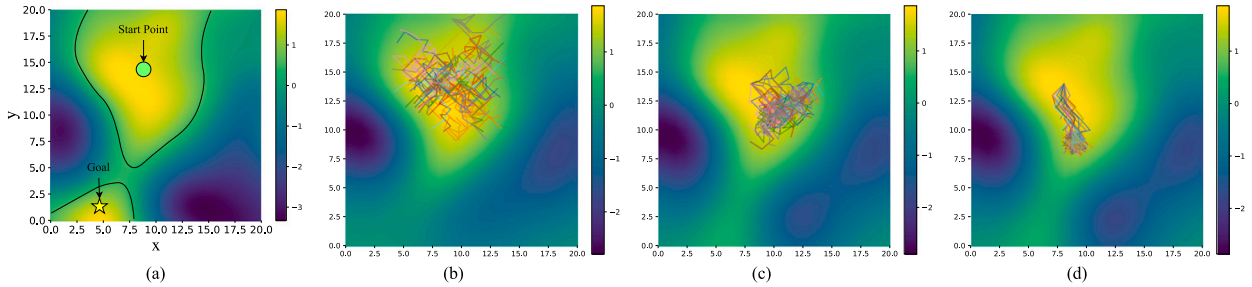


Fig. 13. The schematic diagram of the blind alley case and the trajectory of the agent in this environment under the 1, 100, and 200 epochs along with the policy learning process.

6. Discussion and conclusion

In this article, we proposed the SPPO algorithm, which enables the learned agent to achieve high cumulative rewards with almost zero constraint violations. Furthermore, empowered by the uncertainty estimation of states' safety and the intervention in exploration, we theoretically proved that the agents obtained by our method could satisfy the constraint with high probability even during the training process. Several empirical results demonstrate that the proposed SPPO outperforms other CRL algorithms, e.g., CPO, PCPO, and IPO, in terms of safety constraint satisfaction and cumulative rewards.

It is worth noting that, the reset mechanism in the SPPO algorithm will reduce the exploration efficiency of an agent. When the set of safe states is small with respect to unsafe states, the exploration efficiency will drop even more. This is because, under such circumstances, the reset is triggered more frequently. In fact, there are two sides to every story, the proposed SPPO algorithm ensures the safety of the agent but sacrifices the exploration efficiency to a certain extent.

In addition, Gaussian processes have the disadvantage of limited representation ability in high-dimensional spaces. This is because, as the number of dimensions increases, the computational cost and memory storage cost of Gaussian processes rise dramatically. For high-dimensional state space, in order to use the proposed method, the key lies in how to efficiently predict the safe value of the state. To apply the proposed safe exploration method to higher-dimensional environments, some possible options are listed as follows: (1) Feature extraction and dimensionality reduction operations can be performed on the input data, ensuring that only key information is used for the Gaussian process. Then, the state safety value can be predicted by the proposed method directly. (2) For cases where direct extraction of features cannot be effectively performed, or where the feature dimension is still high, it may be necessary to introduce the local Gaussian process [43] or sparse Gaussian process [44] to perform faster calculations. In such cases, when estimating state safety, only a small number of inputs are used for the calculation, so the calculation speed is greatly improved. Moreover, the proposed method can still ensure the safety of agents' exploration. (3) For more complex environments, deep Gaussian processes [45,46] can be considered. Such methods can handle very large dimensional inputs, even image inputs. In the actual operation process, the effect of such methods may be good, but it lacks theoretical guarantees. Therefore, further research is needed to theoretically guarantee the safety of agents when directly applying the deep Gaussian process to the proposed paradigm.

How to pick the appropriate safety threshold h when building the environment is also a topic worth discussing. h needs to be selected according to specific scenarios. In general, the value of h should be set to a larger safe value than in the event of a disaster (the safety value in an unsafe state). For example, considering the scenario of robot collision, we can design the distance between the robot and the obstacle as the safety value of the state. When the distance is 0, it means that the robot and the obstacle have collided. However, in practice, it is best to design the safety threshold h to be slightly greater than 0 (such as 0.1), so that the robot can explore more safely. This is because, in theory, we can only guarantee that the probability of safe exploration of the agent is close to 1, so it is still possible for the agent to enter an unsafe state, so it is reasonable to set a slightly higher threshold. Also, the value of h should not be set so high that it does not describe the actual safe and unsafe boundary in the environment. A high h will cause the agent's exploration space to be greatly compressed, which makes it impossible for the agent to carry out effective exploration, resulting in a low cumulative return. Consequently, it is a good practice to choose a value that is slightly larger than the value of the boundary between safety and unsafety.

Future research could make the proposed SPPO approach to complex real-world robot control tasks.

CRediT authorship contribution statement

Ke Lin: Methodology, Software, Writing – original draft. **Yanjie Li:** Writing – review & editing. **Qi Liu:** Data curation, Software. **Duantengchuan Li:** Conceptualization, Visualization, Writing – review & editing. **Xiongtao Shi:** Software, Validation. **Shiyu Chen:** Investigation.

Declaration of competing interest

We certify that all co-authors have seen and agreed with the contents of the submission "Almost Surely Safe Exploration and Exploitation for Deep Reinforcement Learning with State Safety Estimation" by Ke Lin et al., which is submitted to Information Sciences as an original research paper, and there is no financial interest to report. We also certify that the submission is our original work and is not under review at any other publications.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work was supported by the National Natural Science Foundation of China [61977019, U1813206] and the Shenzhen Fundamental Research Program [JCYJ20220818102415033, JSGG20201103093802006, KJZD20230923114222045].

References

[1] D. Silver, A. Huang, et al., Mastering the game of Go with deep neural networks and tree search, Nature 529 (2016) 484–489.
[2] D. Silver, A. Huang, et al., A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, Science 362 (2018) 1140–1144.
[3] O. Vinyals, I. Babuschkin, et al., Grandmaster level in StarCraft II using multi-agent reinforcement learning, Nature 575 (2019) 350–354.
[4] K. Łukasz, B. Mohammad, et al., Model based reinforcement learning for Atari, in: Proceedings of the International Conference on Learning Representations, 2020.
[5] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (2016) 1334–1373.
[6] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: Proceedings of the International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.
[7] P.-A. Andersen, M. Goodwin, O.-C. Granmo, Towards safe reinforcement-learning in industrial grid-warehousing, Inf. Sci. 537 (2020) 467–484.
[8] D. Wang, M. Hu, J.D. Weir, Simultaneous task and energy planning using deep reinforcement learning, Inf. Sci. 607 (2022) 931–946.
[9] F. Xue, Q. Hai, T. Dong, Z. Cui, Y. Gong, A deep reinforcement learning based hybrid algorithm for efficient resource scheduling in edge computing environment, Inf. Sci. 608 (2022) 362–374.
[10] E. Altman, Constrained Markov decision processes, Routledge, 2021.

[11] J. García, F. Fernández, A comprehensive survey on safe reinforcement learning, J. Mach. Learn. Res. 16 (2015) 1437–1480.
[12] Y. Liu, A. Halev, X. Liu, Policy learning with constraints in model-free reinforcement learning: a survey, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2021, pp. 4508–4515.
[13] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (1992) 229–256.
[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv:1707.06347, 2017.
[15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: Proceedings of the International Conference on Machine Learning, 2015, pp. 1889–1897.
[16] Y. Chow, M. Ghavamzadeh, L. Janson, M. Pavone, Risk-constrained reinforcement learning with percentile risk criteria, J. Mach. Learn. Res. 18 (2017) 6070–6120.
[17] C. Tessler, D.J. Mankowitz, S. Mannor, Reward constrained policy optimization, in: Proceedings of the International Conference on Learning Representations, 2019.
[18] Y. Zhang, Q. Vuong, K. Ross, First Order Constrained Optimization in Policy Space, Proceedings of the International Conference on Neural Information Processing Systems, vol. 33, Curran Associates, 2020, pp. 15338–15349.
[19] Q. Yang, T.D. Simao, S.H. Tindemans, M.T. Spaan, WCSAC: worst-case soft actor critic for safety-constrained reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 10639–10646.
[20] J. Achiam, D. Held, A. Tamar, P. Abbeel, Constrained policy optimization, in: Proceedings of the International Conference on Machine Learning, PMLR, 2017, pp. 22–31.
[21] F. Berkenkamp, M. Turchetta, A. Schoellig, A. Krause, Safe model-based reinforcement learning with stability guarantees, in: Proceedings of the International Conference on Neural Information Processing Systems, Curran Associates, 2017.
[22] J.F. Fisac, A.K. Akametalu, M.N. Zeilinger, S. Kaynama, J. Gillula, C.J. Tomlin, A general safety framework for learning-based control in uncertain robotic systems, IEEE Trans. Autom. Control 64 (2019) 2737–2752.
[23] M. Turchetta, F. Berkenkamp, A. Krause, Safe exploration in finite Markov decision processes with Gaussian processes, in: Proceedings of the International Conference on Neural Information Processing Systems, Curran Associates, 2016.
[24] A. Wachi, Y. Sui, Safe reinforcement learning in constrained Markov decision processes, in: Proceedings of the International Conference on Machine Learning, PMLR, 2020, pp. 9797–9806.
[25] Y. Chow, O. Nachum, E. Duenez-Guzman, M. Ghavamzadeh, A Lyapunov-based approach to safe reinforcement learning, in: Proceedings of the International Conference on Neural Information Processing Systems, Curran Associates, 2018.
[26] R. Cheng, G. Orosz, R.M. Murray, J.W. Burdick, End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 3387–3395.
[27] A. Aswani, H. Gonzalez, S.S. Sastry, C. Tomlin, Provably safe and robust learning-based model predictive control, Automatica 49 (2013) 1216–1226.
[28] D.Q. Mayne, J.B. Rawlings, C.V. Rao, P.O. Scokaert, Constrained model predictive control: stability and optimality, Automatica 36 (2000) 789–814.
[29] Y. Liu, J. Ding, X. Liu, IPO: interior-point policy optimization under constraints, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 4940–4947.
[30] T.-Y. Yang, J. Rosca, K. Narasimhan, P.J. Ramadge, Projection-based constrained policy optimization, in: Proceedings of the International Conference on Learning Representations, 2019.
[31] M. Yu, Z. Yang, M. Kolar, Z. Wang, Convergent policy optimization for safe reinforcement learning, in: Proceedings of the International Conference on Neural Information Processing Systems, Curran Associates, 2019, pp. 3127–3139.
[32] T. Xu, Y. Liang, G. Lan, CRPO: a new approach for safe reinforcement learning with convergence guarantee, in: Proceedings of the International Conference on Machine Learning, PMLR, 2021, pp. 11480–11491.
[33] Y. Ding, J. Lavaei, Provably efficient primal-dual reinforcement learning for CMDPs with non-stationary objectives and constraints, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 7396–7404.
[34] K. Polymenakos, A. Abate, S. Roberts, Safe policy search using Gaussian process models, in: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems, 2019, pp. 1565–1573.
[35] A. Bottero, C. Luis, J. Vinogradska, F. Berkenkamp, J.R. Peters, Information-theoretic safe exploration with Gaussian processes, in: Proceedings of the International Conference on Neural Information Processing Systems, vol. 35, 2022, pp. 30707–30719.
[36] M. Prajapat, M. Turchetta, M. Zeilinger, A. Krause, Near-optimal multi-agent learning for safe coverage control, in: Proceedings of the International Conference on Neural Information Processing Systems, vol. 35, 2022, pp. 14998–15012.
[37] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, Cambridge, MA, USA, 2006.
[38] N. Srinivas, A. Krause, S.M. Kakade, M.W. Seeger, Gaussian process optimization in the bandit setting: no regret and experimental design, in: Proceedings of the International Conference on Machine Learning, 2010.
[39] S.R. Chowdhury, A. Gopalan, On kernelized multi-armed bandits, in: Proceedings of the International Conference on Machine Learning, PMLR, 2017, pp. 844–853.
[40] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, arXiv:1506.02438, 2015.
[41] E. Coumans, Y. Bai, Pybullet: a Python module for physics simulation for games, robotics and machine learning, http://pybullet.org, 2016–2022.
[42] D. Pathak, P. Agrawal, A.A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in: Proceedings of the International Conference on Machine Learning, PMLR, 2017, pp. 2778–2787.
[43] R.B. Gramacy, D.W. Apley, Local Gaussian process approximation for large computer experiments, J. Comput. Graph. Stat. 24 (2015) 561–578.
[44] M.M. Dunlop, M.A. Girolami, A.M. Stuart, A.L. Teckentrup, How deep are deep Gaussian processes? J. Mach. Learn. Res. 19 (2018) 1–46.
[45] A. Damianou, N.D. Lawrence, Deep Gaussian processes, in: Artificial Intelligence and Statistics, PMLR, 2013, pp. 207–215.
[46] M. Bauer, M. van der Wilk, C.E. Rasmussen, Understanding probabilistic sparse Gaussian process approximations, in: Proceedings of the International Conference on Neural Information Processing Systems, vol. 29, 2016.