2018 Annual American Control Conference (ACC)
June 27–29, 2018. Wisconsin Center, Milwaukee, USA

Safe Reinforcement Learning: Learning with Supervision Using a Constraint-Admissible Set

Zhaojian Li, Uroš Kalabić, Tianshu Chu

Abstract— **Despite recent advances in Reinforcement Learning (RL), its applications in real-world engineering systems are still rare. The primary reason is that RL algorithms involve exploratory actions that can lead to system constraint violations. These violations can damage physical systems and even cause safety issues, *e.g.*, battery overheat, robot breakdown, and car crashes, hindering RL deployment in many engineering applications. In this paper, we develop a novel safe RL framework that guarantees safety during learning by exploiting a constraint-admissible set for supervision. System knowledge and recursive feasibility techniques are exploited to construct a state-dependent constraint-admissible set. We develop a new learning scheme where the constraint-admissible set regulates the exploratory actions from the RL agent and simultaneously guides the agent to learn the system constraints with a penalty for control regulation. The proposed safe RL algorithm is demonstrated in an adaptive cruise control example where a nonlinear fuel economy cost function is optimized without violating system constraints. We demonstrate that the safe RL agent is able to learn the system constraints to gradually fade out the control supervisor.**

I. Introduction

Reinforcement Learning (RL) has attracted significant research efforts in the past decades. It has demonstrated its power in enabling computers to beat world champions at the game Go [1], automatically play ATARI games at human's level [2], and train simulated quadrupeds to run and leap [3]. However, RL applications in practical engineering systems are rare. The main reason is that RL is a trial-and-error learning process that can lead to disastrous consequences due to system constraint violations during the training process.

As such, several studies have been conducted to treat system constraints during RL training and deployment. Worst-case scenario based approaches have been developed to maximize worst-case return, mitigating the effects of variability and uncertainty [4], [5]. Risk management approaches have also been developed to control risks such as variance of the return [6] and the probability of entering into an error state [7]. A comprehensive literature review on the topic of Safe RL can be found in [8]. While the aforementioned methods can manage risks in RL to some extent, they provide no guarantees in avoiding constraint violations. Therefore, disastrous consequences can still happen during the learning process. Recently, reachable set-based analysis is also considered [9]. However, extensive computations are needed, making it undesirable for online learning.

In this work, we propose a novel safe RL framework that exploits system knowledge in the design of an admissible control set that regulates RL explorations to ensure system safety and simultaneously guides RL training to learn the system constraints. Note that system knowledge are typically available or partially available for most engineering systems. The constraint-admissible control set is derived using recursive feasibility techniques that are frequently used in the field of model predictive control [10], [11] and reference governor [12].

In this paper, we use adaptive cruise control, a safety-critical automotive application, to demonstrate the proposed safe RL algorithm where a nonlinear fuel consumption function is optimized without violating system constraints. We show that our developed safe RL agent is able to learn the constraints and eventually avoid actions that can lead to constraint violations. We also show that the safe RL algorithm learns more efficiently than the standard RL algorithm since the safe RL does not have early episode termination due to constraint violations.

The contributions of this paper include the following. First and foremost, we develop a novel safe RL framework that exploits system dynamics and recursive feasibility techniques to construct a constraint-admissible set to guarantee safety during training. Secondly, a new training scheme is developed by integrating the constraint-admissible set with conventional RL to ensure safety while efficiently guiding the RL agent to learn the system constraints. Last but not least, the proposed safe RL framework is demonstrated in the adaptive cruise control example to show the effectiveness.

The rest of the paper is organized as follows. Section II presents the background of conventional RL and an overview of our proposed safe RL. In Section III, the construction of the constraint-admissible control set is derived. The safe RL algorithm is illustrated in Section IV and the ACC example is demonstrated in Section V. Conclusions are drawn in Section VI.

II. From RL to safe RL

A. Conventional RL

RL is a trial-and-error learning algorithm that optimizes the agent's actions to minimize accumulated costs (or maximize accumulated rewards) during its interaction with the environment. As illustrated in Figure 1, at each discrete time step t, the agent receives an observation $x(t) \in \mathbb{R}^{n_x}$, takes

Zhaojian Li is with the Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824, USA. Email: lizhaoj1@egr.msu.edu

Uroš Kalabić is with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA. Email: kalabic@umich.edu

Tianshu Chu is with the Department of Civil and Environmental Engineering, Stanford University, Stanford, CA, USA. Email: cts1988@stanford.edu

a real-valued action $a(t) \in \mathbb{R}^{n_a}$ and receives a scalar cost (or reward) $r(t) \in \mathbb{R}$. The system is typically impacted by some disturbance $w(t)$. The entire history of observation and action is an information state, $s(t) = \big(x(0), a(0), \ldots, a(t-1), x(t)\big)$. We assume a Markovian environment and redefine information state as $s(t) = x(t)$.

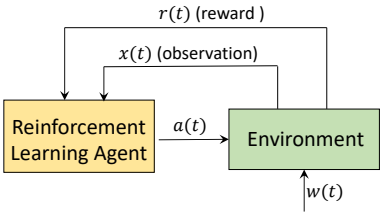

Fig. 1. A schematic diagram of conventional RL.

A policy, $\pi : \mathcal{S} \to \Pr(\mathcal{A})$, is a mapping from states to a probability distribution over the actions and it describes the agent's behavior. Note that the probability of a deterministic policy is one for the chosen action and zero for all other actions. A RL agent learns a control policy from the experience $\{s(0), a(0), r(0), s(1), a(1), \cdots\}$ to maximize the overall future rewards.

$$G = R_0 + \gamma R_1 + \ldots = \sum_{k=0}^{\infty} \gamma^k R_k. \tag{1}$$

Many times the RL policy is parameterized by some parameters θ, i.e., $\pi(\mathcal{S}; \theta)$. The goal is to learn a set of parameters θ that maximizes the reward function (1). Note that during the RL learning process, exploratory actions need to be performed to explore the action space. Therefore, system constraints in engineering systems can be violated due to this exploration. We next introduce our safe RL framework that can train the system safely.

B. Safe RL with a supervisor

The safe RL that we propose is a modification of the conventional RL algorithm. Specifically, we introduce a supervisory element between the RL agent and the control system to ensure safe and reliable system operation and to provide feedback to the RL agent on the safety of the computed RL action. A schematic of the proposed setup is provided in Figure 2.

The control system that we consider is modeled as a linear system. Even though the true system may be nonlinear, we assume that we have at least a linear model to represent the control system. Safe RL with nonlinear systems will be considered in future research work. The model is given as,

$$x(t+1) = Ax(t) + Bu(t) + B_w w(t), \tag{2}$$

$$y(t) = Cx(t) + Du(t) + D_w w(t) \in Y, \ t \in \mathbb{Z}_+, \tag{3}$$

where $x(t) \in \mathbb{R}^{n_x}$ is the state, $u(t) \in \mathbb{R}^{n_u}$ is the control input, $w(t) \sim \mathcal{N}(0, W) \in \mathbb{R}^{n_w}$ is a vector of Gaussian-distributed disturbance inputs with covariance matrix W, and $y(t)$ is the system output. The output is required to remain in the constraint set $Y \subset \mathbb{R}^{n_y}$, which is a polyhedral set containing 0 that represents known safety and reliability constraints, *i.e.*, it is a set for which it is known that enforcing $y(t) \in Y$ ensures safe and reliable system operation.

The control system we consider is a closed-loop system, *i.e.*, it is asymptotically stable, either naturally or by design, such as with a Lyapunov method [13]. The control input $u(t)$ is determined by the RL agent, with the goal of modifying the stabilizing closed-loop control action to change the behavior of the system. Unlike conventional RL where the RL action is directly applied to the control system, the supervisory element modifies the RL action to ensure enforcement of the output constraints (3). This is done by forming a prediction based on the current desired action $a(t)$ obtained from the RL agent and the current measurement or estimate of the system state obtained from the control system $\hat{x}(t)$, and modifying the desired action $a(t)$ to a constraint-admissible control input $u(t)$. The prediction is performed by precomputing offline a set of constraint-admissible system states and corresponding safe actions. The construction of this constraint-admissible control set will be described in Section III.

Simultaneous to its operation as a safety-enforcement mechanism, the supervisor sends a feedback signal to the RL agent, informing the agent on the discrepancy between the desired and constraint-admissible signals. This penalty is given as some norm of the difference between the two inputs. After the RL agent has learned the system and its constraints, the supervisor is removed and the agent acts as a conventional RL-based controller. A detailed description of the safe RL algorithm will be presented in Section IV.

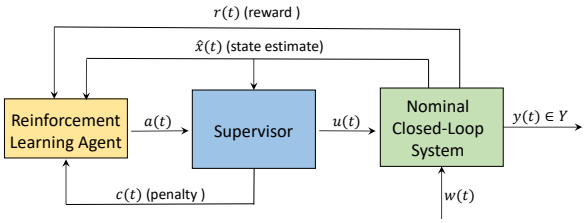

Fig. 2. A safe RL framework with a supervisor. The supervisor modifies the RL action by using a constraint-admissible control set that has been computed offline based on a linear model of the system. Constraint violation penalties are passed to the agent to guide the agent gradually learn a safe policy.

III. Constraint-admissible set

In this section, we describe the construction of the constraint-admissible set used in determining a safe policy. Consider the system (2)-(3). As mentioned, the constraint set Y is polyhedral and contains 0 in its interior, and as such can be represented by a set of linear inequalities,

$$Y = \{y : Sy \leq s\}, \tag{4}$$

where S is a $n_s \times n_y$ matrix and $s > 0$ is a vector of n_s entries, where n_s is the number of constraints.

The safe set we construct is based on the maximal output admissible set of [14], [15], with a modification made in

order to consider stochastic disturbances. It consists of all initial conditions $x(0)$ such that, when the control input $u(t)$ is held constant at 0, the constraints are enforced with some probability p for all present and future time. Mathematically, the set is given by,

$$O^p_\infty = \{x : x(0) = x,\ u(t) = 0, \\ \Pr(y(t) \in Y) \geq p,\ \forall t \in \mathbb{Z}_+\}, \quad (5)$$

where p is treated as a parameter. The main property of the set O^p_∞ is that it is recursively feasible in probability, *i.e.*,

$$x(t) \in O^p_\infty \implies \Pr(Ax(t) + B_w w(t) \in O^p_\infty) \geq p. \quad (6)$$

We are required to consider chance constraints and make guarantees in probability because robust constraint enforcement is not possible with Gaussian disturbances, as the set of all possible disturbances is not bounded. For more details on the properties of O^p_∞, see [16].

The set O^p_∞ is used in modifying the RL control action. The goal of safe RL is to facilitate the learning of a control policy while ensuring safe operation. This can be done by enforcing the constraint,

$$Ax(t) + Bu(t) \in O^p_\infty. \quad (7)$$

By enforcing this constraint, we make a probabilistic guarantee (6) that there exists a sequence of control actions, *i.e.*, $u(t+1) = u(t+2) = \cdots = 0$, for which constraints will be enforced. The set of admissible controls is given by,

$$\mathcal{A}^p(x) = \{u : Ax + Bu \in O^p_\infty\}, \quad (8)$$

and, as we will show in the rest of this section, can be represented as a set of linear inequalities,

$$\mathcal{A}^p(x) = \{u : H(Ax + Bu) \leq h\}. \quad (9)$$

A. Computation of O^p_∞

Here we provide a quick summary of the computation of the admissible set O^p_∞; for details, see [14]–[16]. As mentioned, the set can be represented as a finite set of linear inequalities,

$$O^p_\infty = \{x : Hx \leq h\}. \quad (10)$$

To show this, consider the system (2)-(3) with $u(t) = 0$ for all $t \in \mathbb{Z}_+$. The solution to $y(t)$ is given by,

$$y(t) = CA^t x_0 + y_w(t), \quad (11)$$

where $y_w(t) \sim \mathcal{N}(0, \Upsilon_t)$ and the covariance matrix Υ_t is given by the solution to the dynamic update equations,

$$\Xi_{k+1} = \bar{A}\Xi_k\bar{A}^T + \bar{B}_w W \bar{B}_w^T, \\ \Upsilon_k = \bar{C}\Xi_k\bar{C}^T + D_w W D_w^T.$$

According to (4) and (11), we wish to enforce the following constraints,

$$\Pr(S(CA^t x_0 + y_w(t)) \leq s) \geq p,\ \forall t \in \mathbb{Z}_+. \quad (12)$$

According to [16], we can treat $Sy_w(t)$ as an uncertainty contained in a confidence interval, whose size depends on the probability p. The confidence interval depends on whichever linear inequality is under consideration, *i.e.*, it can be different for each row of the matrix S, and at time t is given by,

$$G\sqrt{S_i \Upsilon_t S_i^T}, \quad (13)$$

where S_i is the i-th row of the matrix S and $G = \sqrt{\chi^{-2}(2p-1, 1)}$, where $\chi^{-2}(\cdot, 1)$ is the inverse of the χ^2 distribution with one degree of freedom.

Knowing this, we are ready to construct O^p_∞. The set is computed recursively, beginning with,

$$H^{(0)} = SC,\ h^{(0)} = s^{(0)},$$

where $s_i^{(t)} := s_i - G\sqrt{S_i \Upsilon_t S_i^T}$ for every i-th entry s_i of s. The computation then continues adding additional entries to H on every iteration,

$$H^{(t)} = \begin{bmatrix} H^{(t-1)} \\ SCA^t \end{bmatrix},\ h = \begin{bmatrix} h^{(t-1)} \\ s^{(t)} \end{bmatrix}.$$

According to [16], there exists a time t^* such that the additional of any constraints will become redundant. Provided that we have been checking for the addition of redundant constraints, we complete the computation by setting,

$$H = H^{(t^*)},\ h = h^{(t^*)}.$$

IV. Safe RL Algorithm

In this section, we present a Safe RL algorithm that can learn a safe policy without violating system constraints. While the proposed Safe RL framework can work with many RL algorithms, in this paper, we apply the deep deterministic policy gradient (DDPG) [17] as the base RL module due to its ability to deal with continuous control space that exists in many engineering systems.

The algorithm is illustrated in Algorithm 1. The parameters include matrices H and h from (10) that define the polyhedral of recursive safe states; M is the number of training episodes; T is the number of time steps in one episode; $C >> 0$ is the penalty of safety violations, i.e., $y \notin Y$ where Y is the safe set defined in (4); $C' > 0$ is the penalty if RL action is not in the admissible set, i.e., $a \notin \mathcal{A}^p(x)$ with $\mathcal{A}^p(x)$ defined in (9); K_b and γ represent the minibatch size and the discount factor, respectively; L is the vector of hidden neuron sizes; and $\alpha = [\alpha_a\ \alpha_c]$ is the vector of learning rates for the actor network and critic network.

We use an experience replay $\mathcal{D}$ that stores the most recent 200 episodes, with first 50 episodes as the warm-up stage for critic network updating only. The term ϵ_{OU} is the Ornstein-Uhlenbeck noise for random exploration, where both the mean value and the degree of volatility are 0.1. For the network structure, both actor and critic networks are two-layer $ReLu$ activated neural networks with the same hidden layer sizes as specified by L. All other training related parameters, such as smoothing factor, are set as default[1].

[1]Package site: `https://github.com/songrotek/DDPG`.

Lines 5-6 are the action selection steps using the Ornstein-Uhlenbeck approach. Lines 7-11 are the control regulation steps. Specifically, Line 7 exploits the recursive feasibility set derived in Section III to check whether the RL action is in the admissible set. If the RL action is admissible, we retain the control action. Otherwise, we assign a violation penalty C' and project the RL action $a(t)$ to an admissible control $u(t)$. The projection typically takes an action close to the center of the polyhedral since the actions on the boundary of the polyhedral can take the state to a value close to the constraint boundaries.

Lines 12-17 collect experiences to form a minibatch which is later sampled to update the actor network as well as the critic network. Note two experience collections are involved if RL action is not admissible. We first collect the experience $(s(t), a(t), -C', s(t+1))$ to inform the RL agent that $a(t)$ is not admissible for the state $s(t)$, steering the RL agent to avoid bad explorations. We use the actual state $s(t+1)$ as a result of control regulation for this experience. In addition, we store the actual transition $(s(t), u(t), \tilde{r}(t), s(t+1))$, where $u(t)$ is the regulated control and $r(t)$ is the reward function if there is no actual safety violation and $-C$ otherwise. Further, we clip $\tilde{r}(t) = \max(-C', r(t))$ for stable gradient updating in RL training. The parameter updates are algorithm dependent that typically Adam optimizer [18] is used for DDPG.

Algorithm 1: Safe Reinforcement Learning

1 **Parameters:** H, h, M, T, C, C', K_b, γ, L, α
2 **initialize** actor network θ, critic network β, corresponding target networks $\theta' = \theta, \beta' = \beta$, and replay buffer $\mathcal{D} = \emptyset$;
3 **for** $j = 0 \rightarrow M - 1$ **do**
4 **initialize** $t = 0$ and a safe state $s(0)$ **while** $t < T$ **do**
5 **measure** state $s(t)$;
  /* OU exploring step */
6 **perform** action $a(t) = \mu_\theta(s(t)) + \epsilon_{OU}$;
  /* control regulation step */
7 **if** $H(Ax(t) + Ba(t)) \leq h$ **then**
8 **retain** the control action $u(t) = a(t)$;
9 **else**
10 **project** the action $a(t)$ to the admissible polyhedral in (9) to obtain an admissible control action $u(t) = Proj_{\mathcal{A}^p(x)} a(t)$;
11 **end**
  /* interacting step */
12 **perform** $u(t)$, **observe** $s(t+1)$, $r(t)$, and **calculate** $\tilde{r}(t)$;
13 **store** $(s(t), a(t), -C', s(t+1))$ to $\mathcal{D}$ **if** $u(t) \neq a(t)$;
14 **store** $(s(t), u(t), \tilde{r}(t), s(t+1))$ to $\mathcal{D}$;
  /* learning step */
15 **sample** minibatch of size K from $\mathcal{D}$;
16 **update** actor and critic networks θ and β using Adam optimizer, with given learning rates;
17 **update** target actor and critic networks with given smoothing factor;
  /* violation step */
18 **if** $r(t) = -C$ *and testing* **then**
19 **stop** the episode;
20 **else**
21 **update** $t \leftarrow t + 1$ and initialize a safe state $s(t+1)$;
22 **end**
23 **end**
24 **end**
25 **return** θ

V. Safe RL for Adaptive Cruise Control

The adaptive cruise control (ACC) system can automatically adjust the ego vehicle speed to maintain a safe distance from the lead vehicle; it has become an increasingly popular feature in modern vehicles. The ACC involves a vehicle following problem illustrated in Figure 3. The relative motion dynamics can be described as:

$$\begin{aligned} \dot{d} &= v_l - v_f, \\ \dot{v}_f &= u, \end{aligned} \tag{14}$$

where d represents the longitudinal distance (often referred to as the range) between the lead vehicle (right) and the ego/host vehicle (left). The variables v_l and v_f are the longitudinal speed of the lead vehicle and the ego vehicle, respectively. The control u is the longitudinal acceleration of the ego vehicle. Note that the velocity of the lead vehicle is typically not available and treated as disturbance.

The following constraints are considered to ensure comfort, safety, and acceptable fuel economy:

$$u_{\min} \leq u \leq u_{\max}, \tag{15}$$

$$0 \leq v_f \leq v_{f,\max}, \tag{16}$$

$$T_{h,\min} \leq T_h \leq T_{h,\max}, \tag{17}$$

$$d \geq d_{\min}, \tag{18}$$

where $u_{\max}$ and $u_{\min}$ are, respectively, the maximum and minimum acceleration limits for good ride comfort and fuel economy; $v_{f,\max}$ represents the maximum speed due to speed limit; $T_h \triangleq \frac{d}{v_f}$ is referred to as time headway, which is constrained between $T_{h,\min}$ and $T_{h,\max}$ for safety and comfort considerations; $d_{\min}$ designates the minimum distance the vehicle should maintain from the lead vehicle.

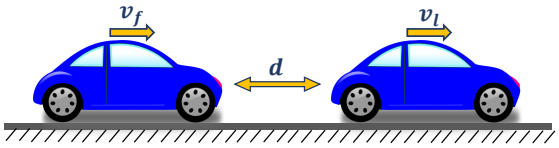

Fig. 3. Schematic diagram of the vehicle-follow dynamics.

With a sampling time Δ, (14) can be discretized as

$$\begin{aligned} d(k+1) &= d(k) + (v_l(k) - v_f(k)) \cdot \Delta, \\ v_f(k+1) &= v_f(k) + u(k) \cdot \Delta, \end{aligned} \tag{19}$$

By defining $x = [d,\, v_f]^{\mathrm{T}}$ and $w = v_l$, system (19) can be written as

$$x(k+1) = Ax(k) + Bu(k) + B_w w(k), \tag{20}$$

where $A = \begin{bmatrix} 1 & -\Delta \\ 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, and $B_w = \begin{bmatrix} \Delta \\ 0 \end{bmatrix}$.

Let us define $y = [d,\, v_f,\, u]^{\mathrm{T}} = Cx + Du + D_w w$, where

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad D_w = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Then the constraints in (15)-(18) can be written as $Sy \leq s$, where

$$S = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \\ 1 & -T_{h,\max} & 0 \\ -1 & T_{h,\min} & 0 \end{bmatrix}, \quad s = \begin{bmatrix} u_{\max} \\ -u_{\min} \\ v_{f,\max} \\ 0 \\ -d_{\min} \\ 0 \\ 0 \end{bmatrix}. \tag{21}$$

We consider the fuel consumption as the reward/cost function. Specifically, we use a fuel consumption model adopted from [19]:

$$f(v_f, a) = \max(p_3 v_f^2 + p_2 v_f a + p_1 v_f + p_0, 0), \tag{22}$$

where $p_3 = 0.0112$, $p_2 = 0.5284$, $p_1 = 0.0409$, $p_0 = -0.1868$.

Note that the admissible set construction in Section III requires the system is stable, i.e., the matrix A is Schur. Towards that end, we pre-stablize the system with a Linear Quadratic Regulator gain K. The gain K is obtained with a state weighting matrix $Q = \mathbb{I}_2$ and control weighting matrix $R = 1e7$. Therefore, we have $A \leftarrow A - B * K$ and $C \leftarrow C - D * K$. The speed of the lead vehicle is modeled as a Gaussian process with mean $\bar{v} = 25\, m/s$ and standard deviation $\sigma_v = 1\, m/s$.

In this example, we use Deep Deterministic Policy Gradient (DDPG) algorithm as the RL agent where a deep neural network is exploited as the policy function. Our DDPG implementation is based on an open-source package DDPG[2]. The simulation parameters are listed in Table I.

TABLE I

SIMULATION PARAMETERS.

$v_{f,max}\ (m/s)$	$u_{\min}\ (N)$	$u_{\max}\ (N)$	$\Delta\ (s)$	C
33	-5	5	1	2000
$T_{h,\min}\ (s)$	$T_{h,\max}\ (s)$	$d_{\min}\ (m)$	M	T
2	6	5	1000	200
L	K_b	γ	α	C'
[160 120]	32	0.99	[5e-5 5e-4]	200

With $p = 0.9$ and based on Section III, the O_∞^p set in (10) is computed and illustrated in Figure 4 as the shaded polyhedral. This set defines the states (distance headway and vehicle speed) that can recursively satisfy the system constraints in an infinite horizon with a probability greater than p when there is no acceleration control. Based on this set, we can compute the admissible set based on (9). As we can see from the sample trajectory in Figure 4, it is possible that states can go beyond the polyhedral. This is because based on (5) we only guarantee the recursive feasibility in terms of Y set not the state set. However, (5) indicates that as long as we take zero controls, the system is still safe. So the states can evolve and eventually admissible control will be available. No system constraint violations happen in this episode.

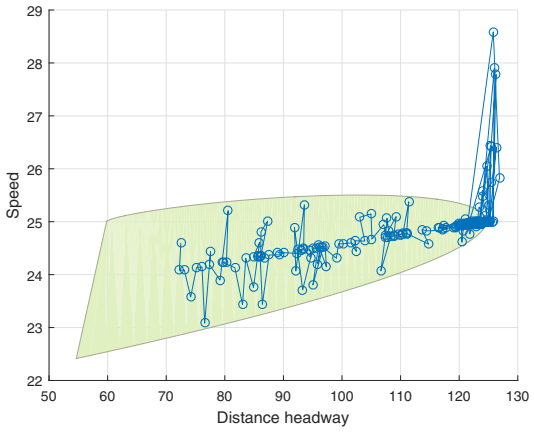

Fig. 4. O_∞^p set and a sample trajectory of the ACC problem.

We compare two frameworks, conventional DDPG without the control regulator and our proposed Safe RL algorithm. The training history of our Safe RL algorithm is illustrated in Figure 5. No system constraints are reported during training. As we can see from Figure 5, the RL agent is able to learn a safe policy as no action adjustments are made after about 850 episodes. The fuel consumptions are also significantly improved through learning. The training history of average weights of the DDPG model is illustrated in Figure 6. It can be seen that the model converges after about 700 episodes.

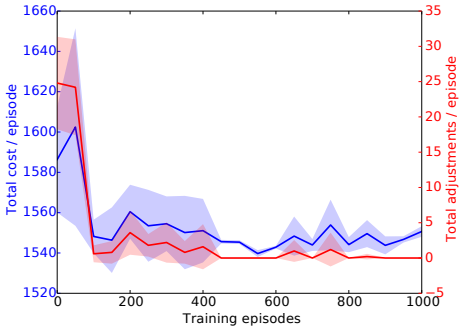

Fig. 5. Training history of our proposed Safe RL algorithm.

The training history of the conventional DDPG is illustrated in Figure 7. As we can see that system violations

[2]Package site: https://github.com/songrotek/DDPG.

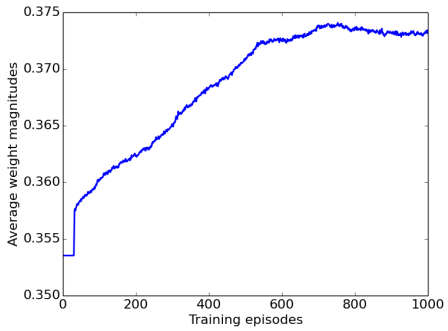

Fig. 6. Training history of average model weights. The model converges after about 700 episodes.

happens in all episodes and the model does not converge after 1000 episodes as illustrated in Figure 8. The reason is that system violations lead to early termination of the episodes which slows the RL training.

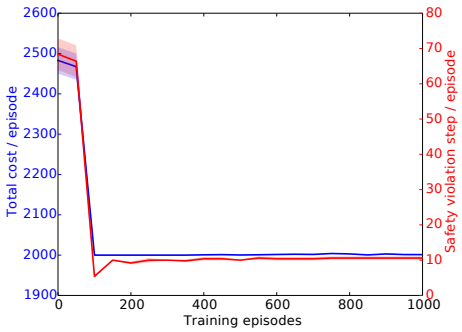

Fig. 7. Training history of conventional DDPG algorithm.

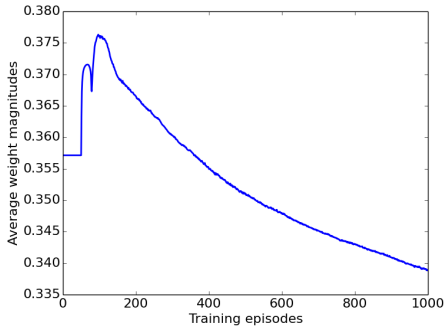

Fig. 8. Training history of average model weights of conventional DDPG. The model does not converge in 1000 episodes.

VI. Conclusions

In this paper, we developed a novel safe reinforcement learning framework that can guarantee system safety during the training process. System dynamics is incorporated in the construction of the admissible control set using recursive feasibility techniques. A scheme of control regulation and penalty feedback is developed to guide the RL agent to asymptotically learn the system constraints. The proposed framework is demonstrated in the adaptive cruise control example where a control policy is learnt to optimize fuel consumption without violating system constraints.

In our future work, we will consider nonlinear and systems with uncertainties. Furthermore, we will demonstrate the proposed safe RL algorithm in lab ground robots.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] X. B. Peng, G. Berseth, and M. van de Panne, "Terrain-adaptive locomotion skills using deep reinforcement learning," *ACM Transactions on Graphics (Proc. SIGGRAPH 2016)*, vol. 35, no. 4, 2016.

[4] M. Heger, "Consideration of risk in reinforcement learning," 1994.

[5] A. Nilim and L. El Ghaoui, "Robust control of markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[6] M. Sato, H. Kimura, and S. Kobayashi, "Td algorithm for the variance of return and mean-variance reinforcement learning," *Transactions of the Japanese Society for Artificial Intelligence*, vol. 16, no. 3, pp. 353–362, 2001.

[7] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Int. Res.*, vol. 24, pp. 81–108, July 2005.

[8] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[9] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. H. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *CoRR*, vol. abs/1705.01292, 2017.

[10] J. Löfberg, "Oops! i cannot do it again: Testing for recursive feasibility in mpc," *Automatica*, vol. 48, no. 3, pp. 550 – 555, 2012.

[11] M. Maiworm, T. Bäthge, and R. Findeisen, "Scenario-based model predictive control: Recursive feasibility and stability," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 50 – 56, 2015. 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015.

[12] E. Garone, S. Di Cairano, and I. Kolmanovsky, "Reference and command governors for systems with constraints: A survey on theory and applications," *Automatica*, vol. 75, pp. 306–328, 2017.

[13] M. I. Weinstein, "Lyapunov stability of ground states of nonlinear dispersive evolution equations," *Communications on Pure and Applied Mathematics*, vol. 39, no. 1, pp. 51–67, 1986.

[14] E. G. Gilbert and K. T. Tan, "Linear systems with state and control constraints: The theory and application of maximal output admissible sets," *IEEE T. Automat. Contr.*, vol. 36, pp. 1008–1020, 1991.

[15] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Math. Problems Eng.*, vol. 4, no. 4, pp. 317–367, 1998.

[16] U. Kalabić, C. Vermillion, and I. Kolmanovsky, "Constraint enforcement for a lighter-than-air wind-energy system: An application of reference governors with chance constraints," in *Proc. IFAC World Congress*, (Toulouse, France), pp. 13800–13805, Jul. 2017.

[17] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] C. R. He, H. Maurer, and G. Orosz, "Fuel consumption optimization of heavy-duty vehicles with grade, wind, and traffic information," *Journal of Computational and Nonlinear Dynamics*, vol. 11, no. 6, p. 061011, 2016.