Proceedings of Machine Learning Research vol xxx:i–xiii, 2021

Safe Reinforcement Learning Using Robust Action Governor

Yutong Li YUTLI@UMICH.EDU
University of Michigan, Ann Arbor, MI, USA

Nan Li NANLI@UMICH.EDU
University of Michigan, Ann Arbor, MI, USA

H. Eric Tseng HTSENG@FORD.COM
Ford Motor Company, Dearborn, MI, USA

Anouck Girard ANOUCK@UMICH.EDU
University of Michigan, Ann Arbor, MI, USA

Dimitar Filev DFILEV@@FORD.COM
Ford Motor Company, Dearborn, MI, USA

Ilya Kolmanovsky ILYA@UMICH.EDU
University of Michigan, Ann Arbor, MI, USA

Abstract

Reinforcement Learning (RL) is essentially a trial-and-error learning procedure which may cause unsafe behavior during the exploration-and-exploitation process. This hinders the application of RL to real-world control problems, especially to those for safety-critical systems. In this paper, we introduce a framework for safe RL that is based on integration of an RL algorithm with an add-on safety supervision module, called the Robust Action Governor (RAG), which exploits set-theoretic techniques and online optimization to manage safety-related requirements during learning. We illustrate this proposed safe RL framework through an application to automotive adaptive cruise control.

Keywords: safety-critical systems, reinforcement learning, action governor, automotive applications

1. Introduction

In Reinforcement Learning (RL), the agent interacts with the environment by perceiving environment's states and selecting the action that maximizes the long-term return based on a real-valued reward signal (Sutton and Barto, 2018). The success of RL has been apparent in a broad spectrum of applications (Mnih et al., 2013; Kober et al., 2013; Abbeel et al., 2007). However, RL is essentially a trial-and-error learning process which may cause unsafe behavior during the learning process. This hinders the RL real-world applications, especially to safety-critical systems.

One approach to addressing such safety issues is the safe RL approach (Garcıa and Fernández, 2015). In particular, risk-sensitive safe RL aims to promote the constraint satisfaction via balancing the long-term return and the risk of reaching the unsafe region (Geibel and Wysotzki, 2005). Methods based on policy optimization with constraints were also proposed, where a constraint on the probability of the system being able to return to the safe region is enforced (Wachi et al., 2018; Moldovan and Abbeel, 2012). However, safety cannot be guaranteed via these model-free

approaches, as the agent needs to learn to operate safely via its interactions with the environment, which may lead to constraint violations.

To guarantee the safety constraint satisfaction during the learning process, an effective approach is to incorporate the system model information. In general, the safe region can be determined by the system model, and a control policy that keeps system state staying inside this safe region can then be computed (Aswani et al., 2013; Fisac et al., 2018; Larsen et al., 2017; Sloth et al., 2012). The main advantage of these model-based methods is that within the interior of the safe region, the RL agent can explore safely to improve the performance. However, the optimality of the trained policy is highly dependent on the size of the safe region computed based on the system model.

Following the idea of this model-based direction, in this paper, we propose a novel safe RL framework that also exploits the system model to design a safe set that regulates the RL explorations to guarantee system safety. In particular, the proposed framework exploits an add-on module, called the Robust Action Governor (RAG), to manage safety. With the RAG, an arbitrary RL algorithm can be integrated into the framework and lead to safe RL. The RAG enforces safety constraints by monitoring, and minimally modifying when necessary, the control signal produced by the nominal RL policy to a constraint-admissible one.

The proposed safe RL framework based on RAG operates on the basis of set-theoretic techniques and online optimization. Similar approaches were proposed by integrating the Reference Governor (RG) with RL to enforce constraint satisfaction (Li et al., 2018, 2019). A distinguishing feature of our safe RL framework based on RAG is that, unlike the RG which modifies the reference input to the controller, the RAG modifies controller output signal, i.e. the RAG can be placed closer to the environment/plant. A direct consequence is that our safe RL framework based on RAG can be used to train lower-level controls, while the approach based on RG in Li et al. (2018, 2019) can only be used to train higher-level planners where the plant must have already been stabilized by some controller that is fixed. Another advantage of RAG compared to RG is that the use of RAG yields a larger safe set, and thereby, can potentially achieve better control performance (Li et al., 2021). On the other hand, unlike safe RL approaches based on the use of control barrier functions (Cheng et al., 2019; Sloth et al., 2012), our RAG is formulated based on discrete-time models from the start and thereby can be more directly applied in a digital setting (e.g., one does not need to modify the algorithm to account for sampling time and discrete updates).

In summary, the contributions of this paper include: 1) establishing a safe RL framework based on RAG, 2) extending the theoretical results developed for the case of the Action Governor for discrete-time linear systems in Li et al. (2021) to its robust version for linear systems with additive bounded disturbances, and 3) illustrating effectiveness of the proposed safe RL framework using an example relevant to automated driving and adaptive cruise control.

2. Conventional RL and Safe RL

Conventional RL optimizes the agent's action via trial-and-error to maximize its accumulated reward (or minimize accumulated cost) through continuously interacting with the environment, as illustrated in Figure 1a. Specifically, at each time instant $k \in \mathbb{Z}_{\geq 0}$, the agent takes a measurement of the state $x(k) \in \mathbb{R}^n$, executes a control $u_\phi(k) \in \mathbb{R}^m$ and collects a reward $r(k) \in \mathbb{R}$. A control policy, $\pi : \mathbb{R}^n \to \mathbb{R}^m$, which is a mapping from the state space $\mathbb{R}^n$ to the action space $\mathbb{R}^m$, describes the agent's behavior. The control policy is learned from the experience $\{x(k), u_\phi(k), r(k)\}$ to maximize the long-term reward $R = \sum_{k=0}^{\infty} \gamma^k r(k)$, where $\gamma \in (0, 1)$ is a discount factor. However,

to maximize the long-term reward, RL agent needs to explore within the action space, which may cause unsafe behaviors (i.e., violation of certain safety constraints) during the learning process. This feature hinders the application of RL to actual engineering systems, and motivates us to propose a safe RL framework enabling the agent to learn a policy safely.

The proposed safe RL framework is illustrated in Figure 1b. An add-on module, termed Robust Action Governor (RAG), is introduced between the RL agent and the environment (which represents the system that the control policy is acting on, e.g., the plant in a conventional control setting). The RAG monitors the control signal $u_\phi(k)$ generated by the RL agent and corrects the ones that may cause unsafe behavior. We will describe the properties and design procedure of the RAG in the subsequent section.

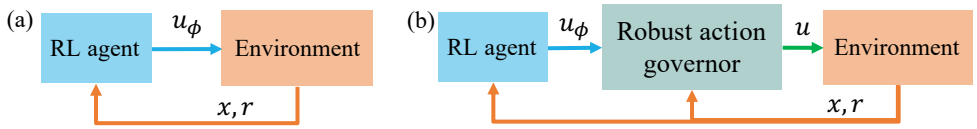

Figure 1: Schematic diagram of conventional RL (a) and safe RL (b).

3. Robust Action Governor

3.1. Problem Formulation

In this section, we review the main results in Li et al. (2021) on the Action Governor and extend them from disturbance-free systems to systems subject to bounded disturbances, enabling the Action Governor to be applicable to more general cases. Consider a discrete-time linear model with additive disturbances as follows:

$$x(k+1) = Ax(k) + Bu(k) + Ew(k), \tag{1}$$

where $x(k) \in \mathbb{R}^n$ is the state at discrete time instant $k \in \mathbb{Z}_{\geq 0}$ and $u(k) \in \mathbb{R}^m$ is the control input. The vector $w(k)$ represents an unmeasured disturbance input and we assume that it is bounded in a known set, i.e., $w(k) \in \mathcal{W} = \{w : Mw \leq m\}$, where $\mathcal{W}$ is a known polytope in $\mathbb{R}^{n_w}$. In applications to RL, $w(k)$ can account for model mismatch, e.g., differences between the actual lead vehicle acceleration and assumed zero lead vehicle acceleration in an adaptive cruise control (ACC) setting. We further assume that state, input and disturbance constraints are not coupled, i.e., $\mathcal{T} = \{(x, u, w) | x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}\} = \mathcal{X} \times \mathcal{U} \times \mathcal{W}$, where $\mathcal{X}$ represents the feasible set of states and $\mathcal{U}$ is the control input set. We assume that a nominal control policy ϕ has been designed for the system (1),

$$u_\phi = \phi(x(k), x_r(k), w(k), k), \tag{2}$$

where $x_r(k) \in \mathbb{R}^r$ denotes a reference signal for the system (1), which defines the control objective. Note that there is no further assumption on the nominal control ϕ, and it can take any form, e.g. nonlinear and time-varying. In particular, in this paper, ϕ represents the RL policy.

We assume that the system is subject to a safety requirement of the form

$$x(k) \in \mathcal{X} = \mathbb{R}^n \setminus X_0, \quad \forall k \in \mathbb{Z}_{\geq 0}, \tag{3}$$

where the unsafe set X_0 can be expressed as a finite union of polytopes, i.e.,

$$X_0 = \bigcup_{i=1}^{n_g} \{x \in \mathbb{R}^n : G_i x < g_i\}, \tag{4}$$

where $G_i \in \mathbb{R}^{n_g \times n}$ and $g_i \in \mathbb{R}^{n_g}$. Note that the expressions (3) and (4) can represent a broad range of safety requirements. For instance, the set $\mathcal{X}$ can be non-convex.

As the safety requirement (3) may not be strictly handled by the nominal control policy (2) (especially when (2) represents an RL policy), this motivates us to design an add-on scheme to enforce (3). In particular, we exploit a supervisory solution, the RAG, to satisfy this requirement, as illustrated in Figure 1b. The RAG monitors the nominal control input u_ϕ and, if necessary, minimally modifies u_ϕ to guarantee that the system state can stay outside X_0 for the present and all future time instants even in the presence of disturbances.

In particular, at each time instant, the RAG solves the following constrained optimization problem to enforce the safety requirement (3),

$$u(k) = \arg\min_{u \in \mathcal{U}} \ \|u - u_\phi(k)\|_S^2 \tag{5a}$$

$$\text{subject to } \ Ax(k) + Bu + Ew \in X_{safe}, \forall w \in \mathcal{W} \tag{5b}$$

where $X_{safe} \subset \mathcal{X}$ is a "safe set" which will be introduced in the next section. The function $\|\cdot\|_S = \sqrt{(\cdot)^T S(\cdot)}$ is used to penalize the difference between the nominal control u_ϕ and the modified control u, where the matrix $S \in \mathbb{R}^{m \times m}$ is positive-definite.

3.2. Safe Set and Unrecoverable Sets

To enforce both present and future safety, the safe set X_{safe} is characterized by the following requirements: There exists a state-feedback control $u(x)$ such that

- $u(x) \in \mathcal{U}$ for all $x \in X_{safe}$;
- Given $x(0) \in X_{safe}$, this control can keep all future states $\{x(1), x(2), ...\}$ within $\mathcal{X}$ even in the presence of disturbances $\{w(0), w(1), ...\} \subset \mathcal{W}$.

The derivation of X_{safe} relies on its complementary sets, termed unrecoverable sets, which are recursively defined as follows,

$$\begin{aligned} X_k &= X_0 \cup \{x \in \mathbb{R}^n : \forall u \in \mathcal{U}, \exists w \in \mathcal{W}, \text{s.t. } Ax + Bu + Ew \in X_j \\ &\qquad \text{for some } j = 0, ..., k-1\} \\ &= X_0 \cup \{x \in \mathbb{R}^n : Ax \in \bigcup_{j=0}^{k-1} X_j \oplus ((-E) \circ \mathcal{W}) \sim (B \circ \mathcal{U})\}, \end{aligned} \tag{6}$$

where $\oplus$, $\sim$, and $\circ$ represent the Minkowski sum, Pontryagin difference, and affine mapping operations of sets. Note that in the derivation of (6) we have used the fact that $S = \{v : \forall u \in U, \exists z \in Z, \text{s.t. } v = z - u\} = \{v : \forall u \in U, \exists z \in Z, \text{s.t. } z = v + u\} = \{v : v + u \in Z, \forall u \in U\} = Z \sim U$.

The unrecoverable set X_k in (6) has the following properties elaborated in Propositions 1 to 3.

Proposition 1: 1) If $x_0 \in X_k$, then for any state-feedback control sequence $\{u_0(x_0)..., u_{k-1}(x_{k-1})\} \in \mathcal{U} \times ... \times \mathcal{U}$, there exists a disturbance sequence $\{w_0, ..., w_{k-1}\} \in \mathcal{W} \times ... \times \mathcal{W}$ such that $x_j \in X_0$ for some $0 \leq j \leq k$.
2) Let x_0 be given. If for any state-feedback control sequence $\{u_0(x_0), ..., u_{k-1}(x_{k-1})\} \in \mathcal{U} \times ... \times \mathcal{U}$, there exists a sequence $\{w_0, ..., w_{k-1}\} \in \mathcal{W} \times ... \times \mathcal{W}$ such that $x_j \in X_0$ for some $0 \leq j \leq k$, then $x_0 \in X_k$.

The proof of Proposition 1 can be constructed following similar steps as in the proofs of Propositions 1 and 2 in Li et al. (2021) and thus is omitted. Proposition 1 implies that for any state-feedback control sequence $\{u_0(x), ..., u_{k-1}(x)\} \in \mathcal{U} \times ... \times \mathcal{U}$, there exists a disturbance sequence $\{w_0, ..., w_{k-1}\} \in \mathcal{W} \times ... \times \mathcal{W}$ that causes the state trajectory to enter X_0 during the steps $0, ..., k$ if and only if $x_0 \in X_k$. Or equivalently, for any admissible disturbances $\{w_0, ..., w_{k-1}\} \in \mathcal{W} \times ... \times \mathcal{W}$, there exists a state-feedback control sequence $\{u_0(x), ..., u_{k-1}(x)\} \in \mathcal{U} \times ... \times \mathcal{U}$ that can prevent the state trajectory from entering X_0 over steps $0, ...k$ if and only if $x_0 \in \mathcal{X} \setminus X_k$.
Proposition 2: For each $k = 0, 1, ...,$ we have $X_k \subset X_{k+1}$, i.e. X_k is an increasing sequence of sets. In turn, $X_\infty = \lim_{k\to\infty} X_k$ exists and satisfies $X_k \subset X_\infty$ for all k.

On the basis of the unrecoverable sets X_k, we define the safe set as $X_{safe} = \mathcal{X} \setminus X_\infty = \lim_{k\to\infty}(\mathcal{X} \setminus X_k)$.
Proposition 3: For any $x \in X_{safe}$, it holds that (i) $x \in \mathcal{X} = \mathbb{R}^n \setminus X_0$ and (ii) there exists $u \in \mathcal{U}$ such that $Ax + Bu + Ew \in X_{safe}$ for all $w \in \mathcal{W}$.

The proofs of the above Propositions 2 and 3 can also be constructed following the proofs of the Propositions 3 and 4 in Li et al. (2021). Specifically to the case of this paper, one needs to use the formula $\bigcup_{k=0}^{k} X_k \oplus W = \bigcup_{k=0}^{k} (X_k \oplus W)$ to complete these proofs, which can be easily shown to hold using the definitions of Minkowski sum and union of sets.

Proposition 3 ensures that if the RAG operates based on (5), then a feasible solution exists to (5) for all k, and the safety requirement (3) is satisfied for all k in spite of disturbance $w \in \mathcal{W}$. Note that the exact determination of X_{safe} relies on the set X_k iteratively computed according to (6) with $k \to \infty$. In practice, we approximate X_{safe} by $X_{safe,k} = \mathcal{X} \setminus X_k$ with k being sufficiently large.

3.3. Offline and Online Computations

Proposition 4: Suppose A is invertible. Then, for each $k = 1, 2, ...,$ we have (i) X_k can be represented as the union of a finite number of polytopic sets, i.e., $X_k = \bigcup_{j=1}^{r_k} X_{k,j}$ where $X_{k,j}$ is a polytopic set for each $j = 1, ..., r_k$; and (ii) X_k can be numerically computed using Algorithm 1.

The proof of Proposition 4 is similar to that of Proposition 6 in Li et al. (2021). RAG operates by solving the optimization problem in (5) at each time step. Using the fact that $\mathcal{X}\setminus(X_k \oplus (-E \circ \mathcal{W})) = \mathcal{X}\setminus X_k \sim E \circ \mathcal{W}$, and $X_k \oplus (-E \circ \mathcal{W}) = \bigcup_{j=1}^{r_k} \bigcap_{i=1}^{s_j} \{x \in \mathcal{X} : G_{i,j}x < g_{i,j}\}$, (5) can be transformed into a Mixed-Integer Quadratic Programming (MIQP) problem with the constraints as following

$$G_{i,j}(Ax(k) + Bu) \geq g_{i,j} - M(1 - \delta_{i,j}), \tag{7a}$$

$$\delta_{i,j} \in \{0, 1\}, \forall i = 1, ..., s_j, \forall j = 1, ..., r_{k'}, \tag{7b}$$

$$\sum_{i=1}^{s_j} \delta_{i,j} = 1, \forall j = 1, ..., r_{k'}, \tag{7c}$$

where $M > 0$ is a sufficiently large number.

Algorithm 1 Offline computation of X_k for systems in (1)

Input: $A, B, E, X_0, X_{k-1}, \mathcal{U}, \mathcal{W}$
Output: X_k

1: $\mathcal{H} \leftarrow \text{convhull}(X_{k-1} \oplus ((-E) \circ \mathcal{W}))$
2: $\mathcal{D} \leftarrow \mathcal{H} \sim ((B) \circ \mathcal{U}))$
3: $\mathcal{E} \leftarrow \mathcal{H} \setminus (X_{k-1} \oplus ((-E) \circ \mathcal{W})))$
4: $\mathcal{F} \leftarrow \mathcal{E} \oplus ((-B) \circ \mathcal{U})$
5: $\mathcal{G} \leftarrow \mathcal{D} \setminus \mathcal{F}$
6: $X_k \leftarrow X_0 \bigcup A^{-1}\mathcal{G}$

4. Safe RL With RAG

Given the constraint enforcement property of RAG, in this section, we integrate RL with RAG to achieve safe learning without violating system constraints. Indeed, the proposed safe RL framework can be combined with any RL algorithms. In this paper, we consider the Neural-Fitted Q-learning (NFQ) as the base RL module due to its ability to deal with continuous state and control spaces (Riedmiller, 2005).

In NFQ, we use a neural network to approximate the Q-function, and we update this Q-function approximation using the following formula,

$$Q(x(t), u(t)) \leftarrow \lambda \tilde{Q}(x(t), u(t)) + (1 - \lambda)\big(R(x(t), u(t)) + \gamma \tilde{V}(x(t+1))\big), \tag{8}$$

where $\lambda \in (0, 1)$ is the learning rate, $\gamma \in (0, 1)$ is a factor to discount future rewards and avoid the Q-values increasing to infinity, and

$$\tilde{V}(x) = \max_{u \in \mathcal{U}} \tilde{Q}(x, u), \tag{9}$$

where $\tilde{Q}(x, u)$ represents the approximated Q-values extracted from the neural network. Note that the data points used to update Q-function in (8) can be obtained through black-box simulations or hardware experiments, which makes the NFQ a model-free method.

The proposed safe RL framework based on RAG is formally presented as Algorithm 2. We first initialize a neural network to approximate Q-function for the continuous state and action spaces. The action space $\mathcal{U}$ is discretized with step d_s to reduce the computational cost of selecting the optimal control action $u(t)$ based on currently approximated Q-function in (9). The n_t is the collected trajectory number in one episode, and T is the length of one trajectory. We use a replay buffer B to store the experience collected in the most recent episode.

The RL agent balances exploration and exploitation using an ϵ-greedy action selection rule in Lines 5-9. Line 10 is the control modification step. By solving (5), any action that may lead to constraint violation will be modified to u^{safe}. Note that we only discretize the action space $\mathcal{U}$ in Line 8, (5) is solved on the continuous action space.

Lines 11-16 collect experience and store it into replay buffer B for later use in training the Q-function NN $\tilde{Q}(x, u)$. In particular, we update the Q-value of the current state $x(t)$ and nominal control $u(t)$, $Q(x(t), u(t))$, with the reward $R(x(t), u^{safe}(t))$ brought by safe action $u^{safe}(t)$. Furthermore, we store the tuple of current state, nominal control and updated Q-value, $(x(t), u(t), Q(x(t),$

$u(t)))$, to the replay buffer. This way, the action modification by RAG is not perceived by the agent, and the agent will explore all actions. This may be beneficial as this will potentially improve convergence to the optimal control policy.

Algorithm 2 Safe RL algorithm

Input Initialized Q-value NN $\tilde{Q}(x,u)$, empty replay buffer B, discretized action space $\mathcal{U}_d$, and the maximum trajectory number within a training episode N.

for each training episode **do**
  while $n_t < N$ **do**
    Pick a safe initial state $x(0) \in X_{safe}$
    while $t < T$ **do**
      if $rand() < \epsilon$ **then**
        $u(t)$ takes a random value within $\mathcal{U}_d$ ▷ Exploration
      else
        $u(t) = \arg\max_{u \in \mathcal{U}_d} \tilde{Q}(x(t),u)$ ▷ Exploitation
      end if
      Solve (5) to modify $u(t)$ to $u^{safe}(t)$ ▷ RAG
      Apply $u^{safe}(t)$ to system
      Observe next state $x(t+1)$ and reward $R(x(t),u^{safe}(t))$
      Update Q value:
      $\tilde{V}(x(t+1)) = \max_{u \in \mathcal{U}_d} \tilde{Q}(x(t+1),u)$
      $Q(x(t),u(t)) \leftarrow \lambda \tilde{Q}(x(t),u(t)) + (1-\lambda)\big(R(x(t),u^{safe}(t)) + \gamma \tilde{V}(x(t+1))\big)$
      Store $(x(t),u(t),Q(x(t),u(t)))$ to replay buffer B
    end while
  end while
  Train Q-value NN $\tilde{Q}(x,u)$ using sampled data from replay buffer B
  $n_t \leftarrow 0$
end for

5. Safe RL for Adaptive Cruise Control

In this section, we apply the proposed safe RL framework to the Adaptive Cruise Control (ACC) problem for an automated vehicle. The dynamics of relative motion between the lead vehicle and the following ego vehicle are represented as

$$\begin{bmatrix} \Delta s(k+1) \\ \Delta v(k+1) \\ v^{ego}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T_s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta s(k) \\ \Delta v(k) \\ v^{ego}(k) \end{bmatrix} + \begin{bmatrix} -\frac{T_s^2}{2} \\ -T_s \\ T_s \end{bmatrix} u(k) + \begin{bmatrix} \frac{T_s^2}{2} \\ T_s \\ 0 \end{bmatrix} w(k), \tag{10}$$

where Δs[m] and Δv[m/s] are the longitudinal distance and relative speed between the lead and ego vehicles, respectively. The state v^{ego}[m/s] represents the ego vehicle's velocity. The control input u[m/s^2] represents the ego vehicle's acceleration, and to improve the driving comfort, we impose the following input constraints $u \in \mathcal{U} = \{u : -3 \leq u \leq 3\}$. $T_s = 0.5$s is the sampling period. We treat the lead vehicle's acceleration $w \in \mathcal{W} = \{w : -1.5 \leq w \leq 1.5\}$ as a disturbance. The upper

and lower bounds of the lead vehicle's acceleration are derived based on the FTP75 driving cycle, which represents the typical city driving behavior.

Our goal is to achieve a safe and efficient car-following performance. We use headway time as the metric and set its target value to 1.5s. The reward function of the RL agent is design as follwing:

$$R = \begin{cases} -(\frac{\Delta s}{v^{ego}} - 1.5)^2 & \text{if } v^{ego} \geq 5, \\ -(\Delta s - 7.5)^2 & \text{if } 0 \leq v^{ego} \leq 5. \end{cases} \tag{11}$$

Also, we consider the following constraints on the headway time to enforce safety:

$$1 \leq \frac{\Delta s}{\max(v^{ego}, 5)} \leq 2. \tag{12}$$

As the set defined by (12) is the region where the system state is supposed to be in, X_0 is its complement and can be expressed in the form of (4).

The nominal control u_ϕ (before training) is a state-feedback control policy tracking the desired headway time of 2.5s, which is different from the target, i.e. 1.5s. Two RL schemes, namely, conventional RL and safe RL, as shown in Figure 1, are employed to train the nominal control policy u_ϕ. During training, we randomly sample segments (with length of $T = 30$s) within the FTP75 driving cycle as the lead vehicle's speed trajectory. The unrecoverable set X_k and safe set $X_{safe,k}$ are computed with the MPT3 toolbox offline (Herceg et al., 2013). The online MIQP optimization problem in (5) is solved by OPTI with SCIP (Currie and Wilson, 2012; Achterberg, 2009). We compute X_k according to Algorithm 1 with $k = 60$, the results are shown in Figure 2.

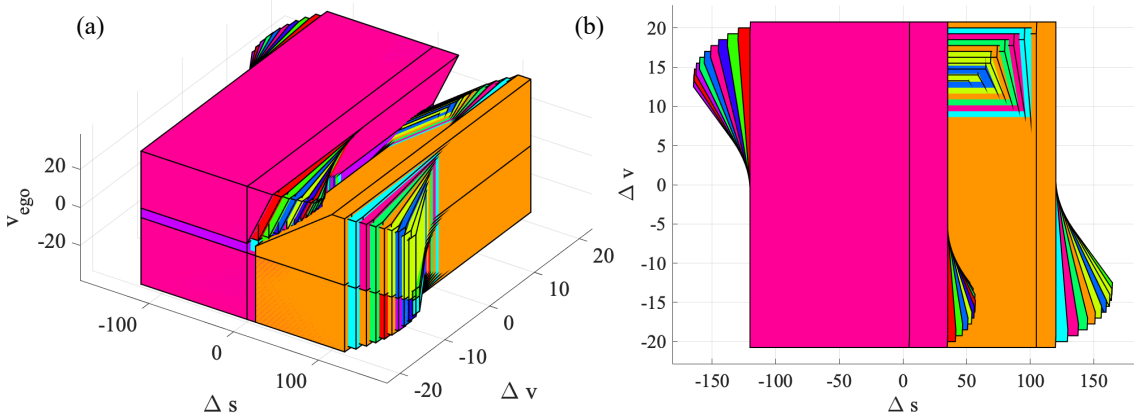

Figure 2: Illustration of the computed unrecoverable set X_k, $k = 60$.

The training histories of conventional and safe RL algorithms are illustrated in Figure 3. As the nominal feedback control policy tracks the headway time of 2.5s before training, which is not within the range of constraints in (12), the constraint violation rate of conventional RL is high at the beginning of training. In contrast, no constraint violation is exhibited for safe RL during the entire training process. With RAG, the RL agent can learn the control policy safely without any constraint violation. Moreover, with RAG the headway time of ego vehicle is always within the range of $[1, 2]$s, leading to a smaller reward variation and faster learning compared with conventional RL, as shown in Figure 3b.

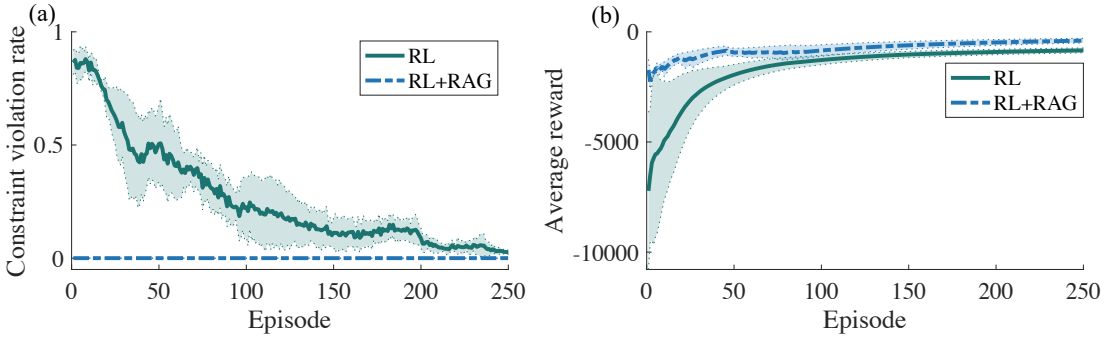

Figure 3: Training histories of conventional RL and safe RL. (a) Constraint violation rates of each episode. (b) Average reward values of each episode. Solid lines represent average values and shaded areas represent the standard deviation values over 20 experiments.

The validation results of trained policies are shown in Figure 4. The policy trained with safe RL is implemented with RAG. As shown in Figure 4a, 4b and 4c, the velocity and headway time tracking performances of conventional and safe RL policies are both satisfactory. However, under the control of conventional RL policy, there are still occasional constraint violations in Δs, as shown in Figure 4d. In comparison, there is no constraint violation with the control of safe RL policy, which is attributed to the fact that the RAG monitors and modifies the control input to guarantee the constraint satisfaction as illustrated in Figure 4f.

We remark that the fact that our safe RL algorithm based on the use of RAG guarantees no safety constraint violation during both the training and the operating phases yields that it can be used for onboard applications. For instance, it can be used to continuously improve the performance of a controller during its onboard operation.

The computational time performances are shown in Figure 5. The simulations are performed on the Matlab R2019b platform using an Intel Xeon E5-1650 3.50 GHz PC with Windows 10 and 16.0 GB of RAM. As the optimization problem (5) needs to be solved at each time instant, safe RL takes around 130 s on average to complete an episode, which is longer compared with the one with conventional RL. Furthermore, we can observe that the average online solving time for (5) is around 30 ms, which is feasible for real-time control scenarios.

6. Conclusions and Future Work

In this paper, we developed a safe RL framework based on RAG that integrates model-based safety supervision and model-free learning. We exploit the underlying dynamics and exclusion-zone requirement to construct a safety set for constraining learning exploration using set-theoretic techniques and online optimization. We applied the proposed safe RL framework to an Adaptive Cruise Control system and showed that we can conduct the online learning with no safety constraint violations. Future work will include the investigation of how to improve the exploration efficiency of the proposed safe RL framework, and how to extend the current framework to general nonlinear systems.

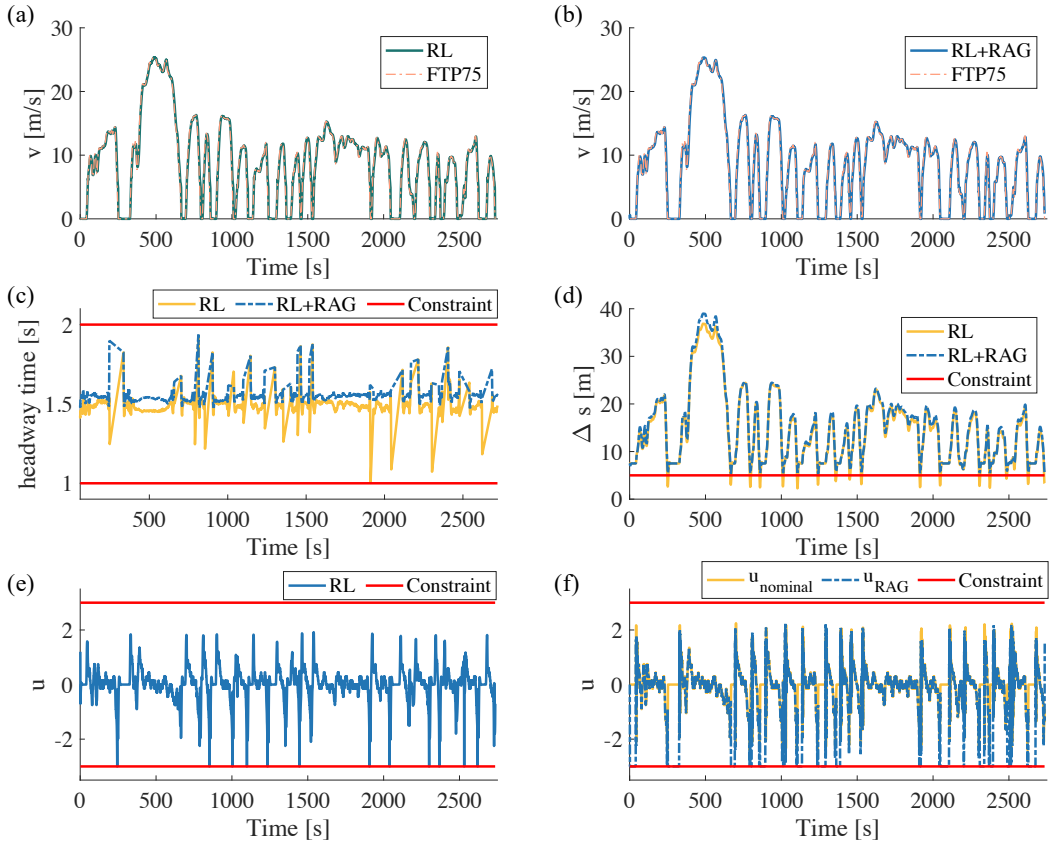

Figure 4: Validation results of conventional RL and safe RL. (a, b) Speed tracking performances. (c) headway time tracking performances (target value: 1.5s). (d) Relative distance between lead vehicle and ego vehicle. (e, f) Control inputs.

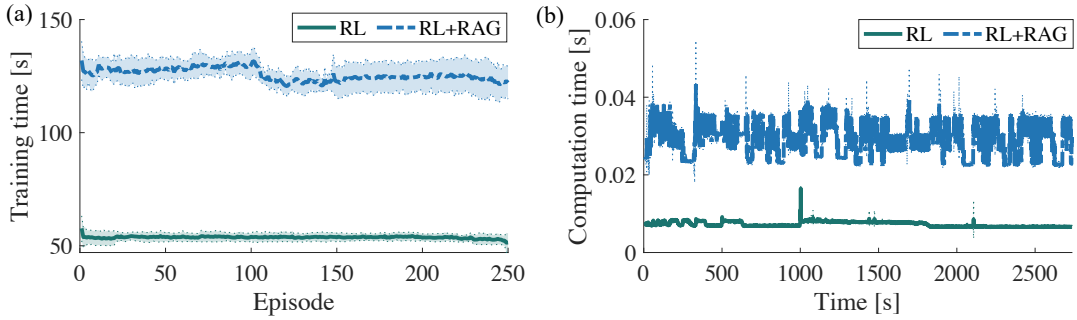

Figure 5: Comparison of computational times. (a) Training time of each episode. (b) Computational time of each step during validation. Solid lines represent average values and shaded areas represent the standard deviation values over 20 experiments.

Acknowledgments

This work is supported by Ford company. We also thank Mr. Xintao Yan for helpful discussions about implementing reinforcement learning algorithm in the ACC example.

References

Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in neural information processing systems*, pages 1–8, 2007.

Tobias Achterberg. Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, Jul 2009. ISSN 1867-2957. doi: 10.1007/s12532-008-0001-1. URL https://doi.org/10.1007/s12532-008-0001-1.

Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel 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*, volume 33, pages 3387–3395, 2019.

Jonathan Currie and David I. Wilson. OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User. In Nick Sahinidis and Jose Pinto, editors, *Foundations of Computer-Aided Process Operations*, Savannah, Georgia, USA, 8–11 January 2012.

Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

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

M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. http://control.ee.ethz.ch/~mpt.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Rie B Larsen, Andrea Carron, and Melanie N Zeilinger. Safe learning for distributed systems with bounded uncertainties. *IFAC-PapersOnLine*, 50(1):2536–2542, 2017.

N. Li, K. Han, A. Girard, H. E. Tseng, D. Filev, and I. Kolmanovsky. Action governor for discrete-time linear systems with non-convex constraints. *IEEE Control Systems Letters*, 5(1):121–126, 2021. doi: 10.1109/LCSYS.2020.3000198.

Zhaojian Li, Uroš Kalabić, and Tianshu Chu. Safe reinforcement learning: Learning with supervision using a constraint-admissible set. In *2018 Annual American Control Conference (ACC)*, pages 6390–6395. IEEE, 2018.

Zhaojian Li, Tianshu Chu, and Uroš Kalabić. Dynamics-enabled safe deep reinforcement learning: Case study on active suspension control. In *2019 IEEE Conference on Control Technology and Applications (CCTA)*, pages 585–591. IEEE, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Teodor Mihai Moldovan and Pieter Abbeel. Safe exploration in markov decision processes. *arXiv preprint arXiv:1205.4810*, 2012.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

Christoffer Sloth, George J Pappas, and Rafael Wisniewski. Compositional safety analysis using barrier certificates. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 15–24, 2012.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. Safe exploration and optimization of constrained mdps using gaussian processes. In *AAAI*, pages 6548–6556, 2018.