

The Feasibility of Constrained Reinforcement Learning Algorithms: A Tutorial Study^{*}

Yujie Yang^{a,1}, Zhilong Zheng^{a,1}, Shengbo Eben Li^{a,*}, Masayoshi Tomizuka^b,
Changliu Liu^c

^a*School of Vehicle and Mobility and State Key Lab of Intelligent Green Vehicle and Mobility, Tsinghua University, Beijing, 100084, China*

^b*Department of Mechanical Engineering, University of California, Berkeley, 94720, CA, USA*

^c*Robotics Institute, Carnegie Mellon University, Pittsburgh, 15289, PA, USA*

Abstract

Satisfying safety constraints is a priority concern when solving optimal control problems (OCs). Due to the existence of infeasibility phenomenon, where a constraint-satisfying solution cannot be found, it is necessary to identify a feasible region before implementing a policy. Existing feasibility theories built for model predictive control (MPC) only consider the feasibility of optimal policy, since MPC itself can be viewed as an optimal controller that solves an optimal action at every step. However, reinforcement learning (RL), as another important control method, solves the optimal policy in an iterative manner, which comes with a series of non-optimal intermediate policies. Feasibility analysis of these non-optimal policies is also necessary for iteratively improving constraint satisfaction; but that is not available under existing MPC feasibility theories. This paper proposes a feasibility theory that applies to both MPC and RL by filling in the missing part of feasibility analysis for an arbitrary policy. The basis of our theory is to decouple

^{*}This study is supported by Tsinghua University Initiative Scientific Research Program. It is also partially supported by NSF China with 52221005.

^{*}Corresponding Author

Email addresses: yangyj21@mails.tsinghua.edu.cn (Yujie Yang),
zheng-zl22@mails.tsinghua.edu.cn (Zhilong Zheng), lishbo@tsinghua.edu.cn
(Shengbo Eben Li), tomizuka@berkeley.edu (Masayoshi Tomizuka),
cliu6@andrew.cmu.edu (Changliu Liu)

¹Y. Yang and Z. Zheng contributed equally to this work.

policy solving and implementation into two temporal domains: virtual-time domain and real-time domain. This allows us to separately define initial and endless, state and policy feasibility, and their corresponding feasible regions. Based on these definitions, we analyze the containment relationships between different feasible regions, which enables us to describe the feasible region of an arbitrary policy. We further provide virtual-time constraint design rules along with a practical design tool called feasibility function that helps to achieve the maximum feasible region. The feasibility function either represents a control invariant set or aggregates infinite steps of constraints into a single one. We review most of existing constraint formulations and point out that they are essentially applications of feasibility functions in different forms. We demonstrate our feasibility theory by visualizing different feasible regions under both MPC and RL policies in an emergency braking control task.

Keywords: feasibility, constrained optimal control, model predictive control, reinforcement learning

1. Introduction

Optimal control is an important theoretical framework for sequential decision-making and control. The goal of solving an optimal control problem (OCP) is to find a policy that maximizes some performance index, usually measured through cumulative rewards. In many real-world control tasks, such as robotics (Brunke et al., 2022), aerospace engineering (Ravaioli et al., 2022), and autonomous driving (Guan et al., 2022), the policy not only needs to optimize performance but also must take safety into consideration. Such problems can be formulated as constrained OCPs, where some state constraints must be strictly satisfied at every time step.

In a constrained OCP, satisfying constraints at a single time step is not enough to ensure long-term safety. The policy may still run into a state where no constraint-satisfying solution can be found in future steps. This issue is called the infeasibility phenomenon (Li, 2023), which is one of the most important problems in constrained OCPs. Due to the existence of this phenomenon, we must ensure that any future state encountered by the policy is always feasible. In other words, the policy must work in a feasible region where the infeasibility phenomenon never happens. Therefore, it is necessary to identify the feasible region before implementing a policy in real-

world environments. To rigorously define and analyze feasible regions, a set of feasibility theories for constrained OCPs is needed.

Existing feasibility theories are built for model predictive control (MPC), which solves the optimal action online through receding horizon optimization. A central concept in these theories is *persistent feasibility* (Zhang et al., 2016; Borrelli et al., 2017), also referred to as *recursive feasibility* in some literature (Löfberg, 2012; Boccia et al., 2014), which describes long-term constraint satisfaction in a receding horizon control (RHC) problem. Specifically, an RHC problem is persistently feasible if its initially feasible set equals its maximal positive invariant set. Here, the initially feasible set contains all states where a constraint-satisfying solution can be found. The maximal positive invariant set contains all states whose successive trajectories always stay in this set. Persistent feasibility ensures that as long as there is a constraint-satisfying solution at the initial step, the receding horizon optimization can proceed without encountering the infeasibility phenomenon. A limitation of persistent feasibility is that it only applies to the optimal policy of a constrained OCP. This is because the maximal positive invariant set is defined in the closed-loop system under the MPC controller, which is the optimal controller of RHC problem. However, reinforcement learning (RL), as another important control method, solves the optimal policy in an iterative manner, which comes with a series of non-optimal intermediate policies. For such non-optimal policies, the infeasibility phenomenon may still be encountered even if the RHC problem is persistently feasible. It is necessary to analyze feasible regions of these non-optimal policies for improving safety through iteration, which is not available with the concept of persistent feasibility. Another concept called *strong feasibility* (Gondhalekar et al., 2009; Gondhalekar and Jones, 2011) expands the scope of persistent feasibility from the optimal policy to any initially feasible policy. An RHC problem is strongly feasible if, from every state in the initially feasible set, the trajectory under any feasible solution remains in this set. In other words, the feasible region of any initially feasible policy exactly equals the initially feasible set in a strongly feasible RHC problem. However, fulfilling strong feasibility requires a specific type of constraint design where the endlessly feasible region exactly equals the initially feasible region, a condition that may not be met for a general constrained OCP. Therefore, a more general set of theoretical tools is needed for analyzing the feasibility of an arbitrary policy.

In order to fill this gap, we propose a set of feasibility theories for con-

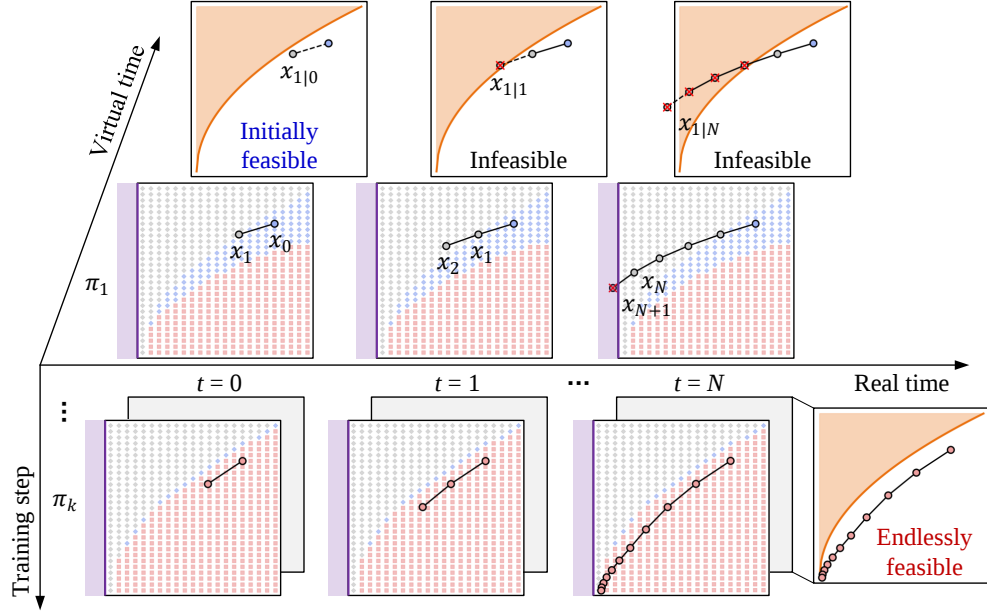


Figure 1: Illustration of core concepts in feasibility theory. The first two rows show state trajectories and feasible regions of an RL policy trained for 50 steps under a Hamilton-Jacobian (HJ) reachability constraint. The third row shows the results of 10000 training steps. The red marks represent endlessly feasible states, the blue marks for initially feasible states, and the gray marks for infeasible states. The purple-shaded areas represent feasible real-time constraints, and the orange-shaded areas represent virtual-time constraints. A state with a red cross indicates violation of real-time or virtual-time constraints. The data in the figure comes from a numerical example of emergency braking control, and details can be found in Section 8.

strained OCPs that apply to both MPC and RL. Our theory is based on an extension of the temporal domain decoupling view in MPC to RL. We point out that any optimal control method follows the practice of solving OCP in virtual-time domain and implementing optimal policy in real-time domain. This decoupling view allows us to define the feasibility of an arbitrary policy, which is necessary for analyzing the feasible regions of intermediate non-optimal policies during RL training. Figure 1 illustrates the core concepts of our feasibility theory. Each real-time step corresponds to a virtual-time constrained OCP, whose optimal solution is either computed at each step online (like MPC) or solved offline before policy implementation (like RL). The constraint in virtual-time domain can be different from that in real-time domain. Whether the virtual-time constraint can be satisfied and how long it can be satisfied leads to different types of feasibility and feasible regions. For a typical safe RL algorithm, the feasible region of its policy starts as a small set and gradually expands during training, finally approaching the feasible region of the optimal policy. MPC, on the other hand, directly computes the optimal control sequences, and therefore, its feasible region is always the same. The main contributions of this paper are summarized as follows.

- We propose a set of feasibility theories for constrained OCPs that apply to both MPC and RL. Based on a decoupling view of virtual-time and real-time domains, we separately define initial and endless feasibility, which describe short-term and long-term constraint satisfaction respectively. We further distinguish state and policy feasibility, which are defined based on the existence of a policy and on a given policy. By combining different types of feasibility, we derive four kinds of feasible regions. We further define the maximum feasible region and reveal its relationship to real-time constraints.
- We give and prove containment relationships of all kinds of feasible regions, which provides a tool for analyzing the feasibility of an arbitrary policy. In particular, we analyze the containment relationship between a policy-specific feasible region and the maximum feasible region, which is a priority concern when solving a constrained OCP. For an arbitrary policy, its feasible region is also bounded from above and below, appearing as an intermediate between some known feasible regions.
- We provide a collection of virtual-time constraint design rules along with a practical design tool called feasibility function that helps to

achieve the maximum feasible region. Specifically, it is revealed by the above-mentioned containment relationships that the maximum feasible region of the optimal policy can be achieved by simply designing a maximum initially feasible region when initially and endlessly feasible regions are equal. The feasibility function ensures this equality by either constructing a control invariant set or aggregating infinite steps of constraints into a single-step one. We review most of existing constraint formulations and point out that they are essentially applications of feasibility functions in different forms.

The rest of this paper is organized as follows. Section 2 introduces a decoupling view of two temporal domains. Section 3 illustrates the infeasibility phenomenon. We define feasibility and feasible regions in Section 4 and further analyze containment relationships of feasible regions in Section 5. After that, we provide feasibility function for constraint design in Section 6 and review existing constraint formulations in Section 7. Finally, Section 8 demonstrate feasible regions with experiments and Section 9 concludes the paper.

2. Constrained Optimal Control Problems in Two Temporal Domains

2.1. Constrained optimal control problems

A constrained OCP can be described by a deterministic Markov decision process $(\mathcal{X}, \mathcal{U}, f, r, \gamma, d_{\text{init}})$, where $\mathcal{X} \subseteq \mathbb{R}^n$ is the state space, $\mathcal{U} \subseteq \mathbb{R}^m$ is the action space, $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$ is the dynamic model, $r : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ is the reward function, $\gamma \in (0, 1]$ is a discounting factor and d_{init} is the initial state distribution.

The constraint is specified through inequalities

$$h(x_{t+i}) \leq 0, i = 0, 1, 2, \dots, \infty, \quad (1)$$

where $h : \mathcal{X} \rightarrow \mathbb{R}$ is the constraint function. The constrained set is defined as $X_{\text{cstr}} = \{x \in \mathcal{X} | h(x) \leq 0\}$. Its complement is the unconstrained set, $\bar{X}_{\text{cstr}} = \mathcal{X} \setminus X_{\text{cstr}}$, which contains all states violating the constraints. Our aim is to find a policy $\pi : \mathcal{X} \rightarrow \mathcal{U}$ that maximizes the expected cumulative

rewards under the state constraints,

$$\begin{aligned}
\max_{\pi} \quad & \mathbb{E}_{x_t \sim d_{\text{init}}(x)} \left\{ \sum_{i=0}^{\infty} \gamma^i r(x_{t+i}, u_{t+i}) \right\}, \\
\text{s.t.} \quad & x_{t+i+1} = f(x_{t+i}, u_{t+i}), \\
& h(x_{t+i}) \leq 0, i = 0, 1, 2, \dots, \infty.
\end{aligned} \tag{2}$$

2.2. Real-time domain and virtual-time domain

To explain what feasibility is, it is necessary to recall the working mechanism of optimal control. In essence, any optimal controller works in two separated temporal domains, i.e., virtual-time domain and real-time domain. Under this perspective, the optimal policy is implemented in the real-time domain, while the solving algorithm is designed in the virtual-time domain. That is to say, constrained MPC/RL algorithm design should be discussed in the virtual-time domain, rather than in the real-time domain. This new viewpoint is our basis for understanding what feasibility is. The separation of the virtual-time domain from the real-time domain inspire us that a different constraint can be used in virtual-time domain from that in real-time domain since constrained OCP is not defined in the real-time domain. Therefore, one can choose a new constraint function in the virtual-time domain:

$$g(x_{i|t}) \leq 0, i = 0, 1, 2, \dots, n, \tag{3}$$

where $g(\cdot)$ is the virtual-time constraint function, and n is the horizon length of virtual-time constrain. This length can be either finite or infinite. With virtual-time constraint (3), we can define the virtual-time OCP as follows.

$$\begin{aligned}
\max_{\pi} \quad & \mathbb{E}_{x_{0|t} \sim d_{\text{init}}(x)} \left\{ \sum_{i=0}^N \gamma^i r(x_{i|t}, u_{i|t}) \right\}, \\
\text{s.t.} \quad & x_{i+1|t} = f(x_{i|t}, u_{i|t}), \\
& g(x_{i|t}) \leq 0, i = 0, 1, 2, \dots, n,
\end{aligned} \tag{4}$$

where N is the horizon length of virtual-time objective, which can be either finite (as in MPC) or infinite (as in RL).

It must be noted that (2) and (4) are not defined in the same temporal domain. The former is defined in the real-time domain, and the latter is defined in the virtual-time domain. The two domains are distinguished by

their subscripts, of which $t + i$ represents the $(t + i)$ -th step in the real-time domain and $i|t$ represents the i -th point in the virtual-time domain starting from time t . The virtual-time constraint (3) can be different from the real-time constraint (1) in two aspects: (1) constraint function and (2) time horizon. The real-time constraint is determined by the control task itself and cannot be modified by algorithm designers. Moreover, this constraint should be satisfied in infinite horizon because any real-world control task is continuing without termination. In contrast, the virtual-time constraint does not need to be posed at every step. It even does not need to have the same function form at all steps. That is to say, except at some special time instances, the constraints in virtual-time domain can be changed or even removed. In fact, a virtual-time constraint can be designed freely as long as it satisfies one requirement: it must be not weaker than the real-time constraint in the current step, i.e., if $g(x_{0|t}) \leq 0$, then $h(x_t) \leq 0$. Besides building the basic structure of defining feasibility, this domain-separating perspective provides us with great flexibility to build various virtual-time constraints, as well as the formulation of their constrained OCPs. As an example, a commonly used constraint design in MPC is that the first $n - 1$ virtual-time constraints are the same as the real-time constraint h , and the last constraint $g(x_{n|t})$, called the terminal constraint, is designed differently, typically stronger than h to ensure that h is satisfied at every real-time step.

3. Illustration of Infeasibility Phenomenon

Infeasibility is the most important concept in constrained OCPs. The domain separation perspective can help us understand the infeasibility phenomenon more straightforwardly. Formally, “infeasibility” describes the phenomenon that the constraint of a virtual-time OCP cannot be satisfied by any policy or by a given policy at some state. Note that infeasibility concerns virtual-time OCP instead of real-time OCP because the policy is obtained from solving the former instead of the latter. We say a state is infeasible when no solution to the virtual-time OCP starting from it can be found. We say a policy is infeasible when it is not a valid solution to a virtual-time OCP. There are two reasons that lead to the infeasibility phenomenon: 1) insufficient training of policy in RL, and 2) improper design of virtual-time constraints.

To better explain how the infeasibility phenomenon occurs, let us consider the following example. Figure 1 shows an infeasible emergency braking

control of an RL policy under a 1-step Hamilton-Jacobian (HJ) reachability constraint. This constraint requires the state not to enter the orange-shaded region in virtual-time domain. Here, the environment model is assumed to be perfect, i.e., the action and state trajectories are the same in the virtual-time and real-time domains. In the first two rows, at time $t = 0$, the action given by the policy π_1 ensures that the virtual-time constraint is satisfied, i.e., $x_{1|0} \in X_{\text{cstr}}$. Since the model is perfect, this action transfers the current state to a new state x_1 in the real-time domain, which equals $x_{1|0}$. At time $t = 1$, the action given by π_1 cannot satisfy the virtual-time constraint, $x_{1|1} \notin X_{\text{cstr}}$. This is how the infeasibility phenomenon occurs, and in this situation, we say π_1 is infeasible in state x_1 . Note that constraint violation in virtual-time domain does not mean that there is no admissible action in real-time domain, where “admissible” means leading to a constraint-satisfying state. Actually, at time $t = 1$, the action is still admissible in real-time domain because the resulting next state x_2 is still inside the real-time constrained set. However, since the virtual-time constraint is already violated, the violation of the real-time constraint is inevitable sometime in the future as long as the virtual-time constraint is properly designed. As shown in the figure, at time $t = N$, the next state x_{N+1} finally goes out of the real-time constrained set.

The above infeasibility phenomenon is mainly caused by insufficient training. Actually, extending the training step from 50 to 10000 basically solves the infeasibility problem. As shown in the third row in Figure 1, the state trajectory is fully contained in the virtual-time constrained set for all times, achieving a desired property called endless feasibility. The other reason that leads to the infeasibility phenomenon, i.e., improper design of virtual-time constraints, is illustrated in Figure 2. Here, the virtual-time constraint function is the same as the real-time one, i.e., $g = h$, but with a finite horizon n . This constraint is called a pointwise constraint. The policy is an MPC controller, which is the optimal policy of the virtual-time OCP. It shows that when n is too small, infeasibility phenomenon occurs easily, and when n is large enough, the state is always feasible. Usually, weaker and fewer steps of constraints have a worse ability to guarantee feasibility in the long run because they cannot provide sufficient confinement to future state trajectories. This problem cannot be mitigated by increasing training steps because even the optimal policy is infeasible. Regarding virtual-time constraint design, we provide a collection of constraint design rules along with a practical design tool called feasibility function that can avoid the infeasibility phenomenon and help to achieve the maximum feasible region.

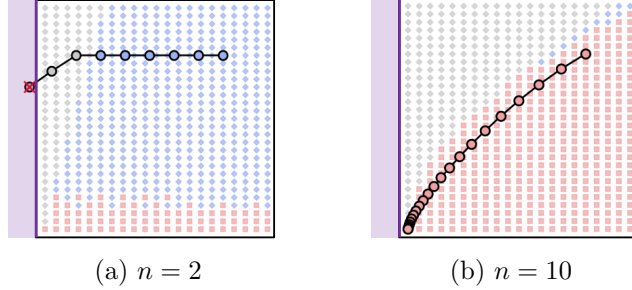


Figure 2: State trajectories and state feasibility of MPC under pointwise constraints.

4. Feasibility and Feasible Regions

We have seen that the improper selection of virtual-time constraints would result in the notorious infeasibility phenomenon. Now let us move to the formal definition of feasibility, which allows us to analyze what kind of virtual-time constraint can guarantee safety. The aforementioned example of pointwise constraint enlightens us that depending on the choice of constraint, virtual-time OCPs may suddenly lose feasibility at some time points or keep feasibility forever. These two results bring about the need to distinguish between (1) initial feasibility and (2) endless feasibility. As suggested by their names, initial feasibility may be only temporary, while endless feasibility always comes with all virtual-time OCPs in the everlasting future.

4.1. Initial feasibility

The existence of infeasibility phenomenon indicates that a state with feasible virtual-time OCP may inevitably evolve into a state with infeasible virtual-time OCP. As a result, the property of a state's yielding feasible OCP at the current step is termed as initial feasibility. This property is just a shortsighted feature and may not give any guarantee in the future (even just one step later!). To have an intuitive understanding of initial feasibility, recall the evolution of infeasibility phenomenon in Figure 1, where x_t is a feasible state but still evolves into a state x_{t+2} that is infeasible. The definition of initial feasibility is as follows.

Definition 4.1 (Initial feasibility of a state).

1. A state x is initially feasible if there exists a policy that satisfies the virtual-time constraint starting from x , i.e., $\exists \pi$, s.t. $g(x_{i|t}) \leq 0, i = 0, 1, 2, \dots, n$, where $x_{0|t} = x$.

2. *The initially feasible region (shortened as IFR), denoted as X_{init}^g , is the set of all states that are initially feasible.*

The subscript “init” of initial feasible region X_{init}^g stands for “initial”, and the superscript “g” emphasizes that it is related to the virtual-time constraint. For a state to be initially feasible, the only requirement is that the virtual-time OCP starting from it has a solution. Outside the IFR, there is no policy satisfying the virtual-time constraint, and hence the virtual-time OCP has no solution. In other words, IFR is a region restricted by the chosen virtual-time constraint, identifying those states that are meaningful under the virtual-time constraint. In this definition, the policy π can be arbitrarily chosen to meet the existence statement. Supposing that we have already been given a policy, it is also a natural need to check whether its output satisfies the virtual-time constraint, which leads to the definition of initial feasibility of a policy.

Definition 4.2 (Initial feasibility of a policy).

1. *A policy π is initially feasible in a state x if π satisfies the virtual-time constraint starting from x , i.e., $g(x_{i|t}) \leq 0, i = 0, 1, 2, \dots, n$, where $x_{0|t} = x$.*
2. *The set of all policies that are initially feasible in x is denoted as $\Pi_{\text{init}}^g(x)$.*
3. *The initially feasible region of a policy π , denoted as $X_{\text{init}}^g(\pi)$, is the set of all states in which π is initially feasible.*

The notation $X_{\text{init}}^g(\pi)$ means that the IFR is not only dependent on the constraint g but also a function of policy π . The initial feasibility of a policy is closely related to the initial feasibility of a state. If a state is initially feasible, then there must exist a policy that is initially feasible in this state. On the other hand, if a policy is initially feasible in a state, then there exists at least one solution to virtual-time OCP starting from this state, which indicates that this state is initially feasible. The IFR of a policy $X_{\text{init}}^g(\pi)$ is the region where one is approved to take policy π under the virtual-time constraint. Similar to the initial feasibility of a state, the initial feasibility of a policy only describes the property of a single virtual-time OCP starting from the current real-time step. It does not provide any guarantee on the virtual-time OCPs in future real-time steps. A policy that is initially feasible at the current time step may become infeasible after it takes a step forward in

the real-time domain. Examples of IFR of a policy are the union of blue and red regions in Figure 1 and Figure 2. Within these regions, the virtual-time constraints are satisfied by the policy, but there is no guarantee that they will still be satisfied in successive states.

4.2. Endless feasibility

Having the definition of initial feasibility at hand, we are able to describe what we actually care about, i.e., a long-term property taking infinite horizon constraints into account. This property is termed as endless feasibility. Intuitively, endless feasibility gives a guarantee of not only the feasibility of the current virtual-time OCP but also that of all future virtual-time OCPs. Here is an example to give a sense of why there is a distinction between initial and endless feasibility. In a car-following scenario, the ego vehicle is required to follow a leading vehicle while keeping a safe distance from it. Suppose that the virtual-time constraint requires a positive distance between vehicles only at the next time step. Then, a state with an overly high speed may be initially feasible since the vehicles will not collide immediately. However, due to the limited braking capability, the car will inevitably reach a state where collision is doomed at the next time step no matter what action is taken. In other words, the virtual-time OCP becomes infeasible after certain steps. Unlike initial feasibility, endless feasibility guarantees the feasibility of all future virtual-time OCPs by its definition. Just like the case for initial feasibility, it can also be defined for a state or a policy.

Definition 4.3 (Endless feasibility of a state).

1. A state x is endlessly feasible if it is initially feasible and its successive states under any initially feasible policy in each time step are initially feasible, i.e., $x_t \in X_{\text{init}}^g$ and $\forall \pi_{t+i} \in \Pi_{\text{init}}^g(x_{t+i}), x_{t+i+1} \in X_{\text{init}}^g, i = 0, 1, 2, \dots, \infty$, where $x_t = x$.
2. The endlessly feasible region (shortened as EFR), denoted as X_{edls}^g , is the set of all states that are endlessly feasible.

The subscript “edls” in the above symbols is an abbreviation of “endless”, and the superscript “g” emphasizes that it is related to the virtual-time constraint. The endless feasibility of a state is related to not only the current step in the real-time domain but also all future time steps. The states in future time steps can be obtained by arbitrary initially feasible policies in each time step. The arbitrariness of policy selection is critical to the definition

of endless feasibility. This is because, for any initially feasible state, there may exist more than one initially feasible policy in that state. Which one is the solution of the virtual-time OCP depends on the objective function. The arbitrariness of policy selection ensures that the solution can always ensure the successive states to be initially feasible, regardless of the objective function. As a result, the EFR identifies those states that can be definitely rendered safe by the virtual-time constraint. Similar to the initial feasibility of a policy, we can also define the endless feasibility of a policy.

Definition 4.4 (Endless feasibility of a policy).

1. A policy π is endlessly feasible in a state x if π is initially feasible in both x and the successive states of x under π , i.e., $\pi \in \Pi_{\text{init}}^g(x_{t+i}), i = 0, 1, 2, \dots, \infty$, where $x_t = x$.
2. The endlessly feasible region of a policy π , denoted as $X_{\text{edls}}^g(\pi)$, is the set of all states in which π is endlessly feasible.

The “ π ” in brackets serves the same purpose as in $X_{\text{init}}^g(\pi)$. Note that the definition of endless feasibility is built upon initial feasibility, requiring all successive states in a trajectory to be initially feasible. For endless feasibility of a policy, this trajectory is naturally induced by the given policy. For endless feasibility of a state, this trajectory can be induced by any initial feasible policy, i.e., it is required that at every step, taking any initially feasible policy (not just one of them), the next state is also initially feasible. With such a definition, it naturally holds that for an endlessly feasible state, there must also be an endlessly feasible policy in that state, i.e., EFR is a subset of the policy’s EFR. This conclusion will be formally stated and proved in Theorem 5.1. Examples of EFR of a policy are the red regions in Figure 1 and Figure 2. Within these regions, the policy is initially feasible in both the current and all successive states.

The above-mentioned four definitions are all specific to virtual-time constraints. Even for the same real-time OCP, a different choice of virtual-time constraint will lead to different feasible regions. A natural question is how to design a virtual-time constraint that induces an EFR as large as possible. To discuss this, we first need to define the maximum EFR.

Definition 4.5. The maximum endlessly feasible region (shortened as maximum EFR), denoted as X_{edls}^* , is the union of all endlessly feasible regions.

The above definition of maximum EFR is closely related to the notion of maximum control invariant set in MPC. The latter refers to the maximum set in which for any state, there exists a control sequence that keeps its successive states still in this set. The maximum EFR has one more requirement than the maximum control invariant set, i.e., all states in it must satisfy the real-time constraint. In other words, a constraint-satisfying maximum control invariant set is a maximum EFR. Either too strong or too weak a virtual-time constraint will induce an EFR X_{edls}^g smaller than X_{edls}^* . For the former case, less state is considered initially feasible and hence outside of X_{edls}^g . For the latter case, one may take too aggressive actions at the beginning of a trajectory due to the inadequate restriction of virtual-time constraint, leading to the occurrence of an infeasibility phenomenon. Examples of these two cases can be found in the experiment results in Section 8, where the pointwise constraint with $n = 2$ is too weak, and the control barrier function constraint with $k = 0.5$ is too strong. The maximum EFR is the largest area in which we can expect any policy to work safely. The following theorem gives an identical description of the maximum EFR, which explicitly relates endless feasibility with constraint satisfaction in real-time domain.

Theorem 4.1. *For an arbitrary state x , the following two statements 1) and 2) are equivalent:*

1. $x \in X_{\text{edls}}^*$.
2. $\exists \pi$, s.t. $h(x_{t+i}) \leq 0, i = 0, 1, 2, \dots, \infty, x_t = x$.

Proof. First, we prove $1) \Rightarrow 2)$. According to Definition 4.5,

$$\forall x \in X_{\text{edls}}^*, \exists \pi \text{ and } g, \text{ s.t. } x_{t+i} \in X_{\text{init}}^g, i = 0, 1, 2, \dots, \infty.$$

Since $X_{\text{init}}^g \subseteq X_{\text{cstr}}$, we have $x_{t+i} \in X_{\text{cstr}}$, i.e., $h(x_{t+i}) \leq 0$.

Next, we prove $2) \Rightarrow 1)$. For an arbitrary state x that satisfies 2), we can choose

$$g(x_{i|t}) = h(x_{i|t}), i = 0, 1, 2, \dots, \infty$$

as the virtual-time constraint. In this case, we have $x \in X_{\text{edls}}^g$. Since $X_{\text{edls}}^g \subseteq X_{\text{edls}}^*$, we conclude that $x \in X_{\text{edls}}^*$. $\square$

The above theorem justifies the importance of defining maximum EFR in that states in X_{edls}^* are exactly what we are eager to identify, which are the only states that can be rendered safe in the real-time domain. That is to

say, for any state $x \in X_{\text{edls}}^*$, there exists a policy π , such that all future states are in the constrained set X_{cstr} . Since 2) in Theorem 4.1 goes to infinity, the successive states are always in the maximum EFR.

Intuitively, if a state x is in the EFR of a policy π , it can be rendered safe by this policy. This result is stated formally in the following theorem, which theoretically justifies applying π in x and all successive states.

Theorem 4.2. *For any state x , statement 1) is a sufficient condition for statement 2):*

1. $\exists g, \text{ s.t. } x \in X_{\text{edls}}^g(\pi).$
2. $h(x_{t+i}) \leq 0, i = 0, 1, 2, \dots, \infty, x_t = x, u_t = \pi(x_t).$

Proof. According to Definition 4.4,

$$\forall x \in X_{\text{edls}}^g(\pi), x_{t+i} \in X_{\text{init}}^g, i = 0, 1, 2, \dots, \infty,$$

where x_{t+i} are obtained by π . Since $X_{\text{init}}^g \subseteq X_{\text{cstr}}$, we have $x_{t+i} \in X_{\text{cstr}}$, i.e., $h(x_{t+i}) \leq 0$. $\square$

Theorem 4.2 tells us that the EFR of a policy is where we can safely apply the policy in real-time domain. This is the reason why we need to find not only the policy itself but also its EFR when solving constrained OCPs. In literature, this EFR is found by identifying a control certificate related to virtual-time constraint, e.g., control barrier function (Ames et al., 2014), safety index (Liu and Tomizuka, 2014), and Hamilton-Jacobi reachability function (Mitchell et al., 2005). In fact, initial feasibility is not what we ultimately care about because it only describes whether constraints of the current virtual-time OCP are satisfied. The role of initial feasibility is to help us define endless feasibility, which describes whether constraints of all successive virtual-time OCPs are satisfied. Our main concern is endless feasibility because it is related to long-term safety in real-time domain. Therefore, we care about how to calculate the EFR of a policy, especially the policy obtained from solving a virtual-time OCP. We hope that the EFR of this policy equals the maximum EFR. Furthermore, the size of EFR of a policy is related to its performance. If we want a policy to be optimal, then its EFR must contain all optimal state trajectories. A larger EFR has a greater chance of containing optimal trajectories and therefore increases the probability of finding the optimal policy.

5. Some Properties of Feasible Regions

In this section, we formally explore the properties of feasible regions, as well as their relationships and how they are informative for virtual-time constraint design. Specifically, we will pay attention to containment relationships and equivalence relationships of different feasible regions.

5.1. Containment relationships of feasible regions

The definitions of feasible regions can be categorized by two dimensions: (a) initial ones or endless ones and (b) with a given policy or without a given policy. The four combinations correspond to (1) initially feasible region (IFR) X_{init}^g , (2) IFR of a policy $X_{\text{init}}^g(\pi)$, (3) endlessly feasible region (EFR) X_{edls}^g , and (4) EFR of a policy $X_{\text{edls}}^g(\pi)$. Besides these, there are two more regions we need to pay attention to, i.e., constrained set X_{cstr} and the maximum EFR X_{edls}^* . The constrained set comes from real-world constraints, and the maximum EFR is the feasible region that we always aim to identify because it gives the largest allowable working area of the policy. Among all of these six regions, X_{cstr} , X_{init}^g , and $X_{\text{init}}^g(\pi)$ are related only to the current step, while X_{edls}^g , $X_{\text{edls}}^g(\pi)$, and X_{edls}^* are our true ultimate goals. Since the latter is related to long-term feasibility, their identification is much more difficult than that of the former. The following theorem gives some containment relationships between these six regions so that we can let those easy-to-identify regions cast some light on the goal regions.

Theorem 5.1 (Feasible region containment).

1. $X_{\text{edls}}^g \subseteq X_{\text{init}}^g \subseteq X_{\text{cstr}}$.
2. $\forall \pi, X_{\text{edls}}^g(\pi) \subseteq X_{\text{init}}^g(\pi) \subseteq X_{\text{init}}^g$.
3. $\forall \pi, X_{\text{edls}}^g(\pi) \subseteq X_{\text{edls}}^*$.
4. $\forall \pi$, if $X_{\text{init}}^g(\pi) = X_{\text{init}}^g$, then $X_{\text{edls}}^g \subseteq X_{\text{edls}}^g(\pi)$.

Proof.

1. The first containment relationship $X_{\text{edls}}^g \subseteq X_{\text{init}}^g$ can be directly concluded from Definition 4.3. For the second one, according to Definition 4.1,

$$\forall x \in X_{\text{init}}^g, g(x) = g(x_{0|t}) \leq 0.$$

Since $g(x_{0|t})$ is not weaker than $h(x_t)$, we have $h(x) \leq 0$, i.e., $\forall x \in X_{\text{cstr}}$. Therefore, $X_{\text{init}}^g \subseteq X_{\text{cstr}}$.

2. This holds by Definition 4.4 and Definition 4.2.
3. As stated in Definition 4.4,

$$\forall x \in X_{\text{edls}}^g(\pi), \pi \in \Pi_{\text{init}}^g(x_{t+i}), i = 0, 1, 2, \dots, \infty,$$

where $x_t = x$. Thus, $x_{t+i} \in X_{\text{init}}^g(\pi)$. Since $X_{\text{init}}^g(\pi) \subseteq X_{\text{init}}^g$, it follows that $x_{t+i} \in X_{\text{init}}^g$. Thus, $x_{t+i} \in X_{\text{cstr}}$, i.e., $h(x_{t+i}) \leq 0$. Recall Theorem 4.1 and we reach $X_{\text{edls}}^g(\pi) \subseteq X_{\text{edls}}^*$.

4. Since $X_{\text{init}}^g = X_{\text{init}}^g(\pi)$, it holds that π is initially feasible in all states in X_{init}^g . That is to say, $\pi \in \Pi_{\text{init}}^g(x), \forall x \in X_{\text{init}}^g$. Starting from any $x \in X_{\text{edls}}^g$ and choosing

$$\pi_{t+i} = \pi \in \Pi_{\text{init}}^g(x_{t+i}), i = 0, 1, 2, \dots, \infty.$$

By definition of endlessly feasibility, we have

$$x_{t+i} \in X_{\text{init}}^g = X_{\text{init}}^g(\pi) \text{ i.e., } \pi \in \Pi_{\text{init}}^g(x_{t+i}).$$

This means π is endlessly feasible in x , i.e., $\forall x \in X_{\text{edls}}^g(\pi)$. Thus, $X_{\text{edls}}^g \subseteq X_{\text{edls}}^g(\pi)$.

□

Figure 3 illustrates the containment relationships in Theorem 5.1. Figure 3(a) is an illustration of $X_{\text{edls}}^g \subseteq X_{\text{init}}^g \subseteq X_{\text{cstr}}$. The light-colored circles correspond to virtual-time states, and the dark-colored ones stand for real-time states. The first containment relationship holds by definition since a state must first be initially feasible, then it could be endlessly feasible. This is also the case for the second one.

Figure 3(b) is an illustration of $X_{\text{init}}^g(\pi) \subseteq X_{\text{init}}^g$. For a state in X_{init}^g , there exists a policy satisfying the virtual-time constraint, but it is not necessarily the given policy π . Intuitively, given a policy π poses a greater restriction on the IFR, making the IFR of a policy become a subset of IFR.

Figure 3(c) is an illustration of $X_{\text{edls}}^g(\pi) \subseteq X_{\text{edls}}^*$. On the left are the zero-sublevel set of $h(x)$ and $g(x)$, they correspond to the maximum EFR X_{edls}^* and the EFR of a policy $X_{\text{edls}}^g(\pi)$ on the right, respectively. To see how $\{x | h(x) \leq 0\}$ is related to X_{edls}^* , recall Theorem 4.1 which states that $x \in X_{\text{edls}}^*$ is equivalent to the existence of a policy rendering x safe in real-time domain. Likewise, $x \in X_{\text{edls}}^g(\pi)$ means that the specific policy π renders x safe in virtual-time domain, without violating virtual-time constraint $g(x) \leq 0$.

Naturally, a larger valid state set ($\{x|h(x) \leq 0\}$ versus $\{x|g(x) \leq 0\}$) and no restriction on the policy yields a larger EFR.

Figure 3(d) is an illustration of the statement “If $X_{\text{init}}^g(\pi) = X_{\text{init}}^g$, then $X_{\text{edls}}^g \subseteq X_{\text{edls}}^g(\pi)$.” The left picture corresponds to the condition $X_{\text{init}}^g(\pi) = X_{\text{init}}^g$, with the red region representing both X_{init}^g and $X_{\text{init}}^g(\pi)$. This condition implies that $\pi \in \Pi_{\text{init}}^g(x), \forall x \in X_{\text{init}}^g$, where $\Pi_{\text{init}}^g(x)$ is the set of policies that are initially feasible under x . The right picture illustrates how X_{edls}^g is a subset of $X_{\text{edls}}^g(\pi)$. For a state in the purple region X_{edls}^g , every initially feasible policy at every time step must lead to an initially feasible state at the next time step. That is to say, the policy π' (in purple) and π'' (in red) are chosen arbitrarily from the initially feasible policy sets $\Pi_{\text{init}}^g(x_{t+1})$ and $\Pi_{\text{init}}^g(x_{t+2})$, respectively. So π is also a possible choice for π' and π'' (and all future policies). This guarantees that π renders any state in X_{edls}^g endlessly feasible under itself.

Talking about $X_{\text{init}}^g(\pi)$ and $X_{\text{edls}}^g(\pi)$, the previous theorem only discusses the containment relationships for an arbitrary policy π . This policy π may not come from the optimization of virtual-time OCP. In safe RL, besides trying to identify the maximum EFR, one also seeks to find the best policy, i.e., π^* . The optimal policy π^* , including its corresponding IFR and EFR, have special importance in building effective RL algorithms. As a special case, let us first talk about an interesting property of IFR at π^* , i.e., $X_{\text{init}}^g(\pi^*)$. For any state $\forall x \in X_{\text{init}}^g$, its corresponding virtual-time OCP must has a feasible solution, and one can construct an optimal policy π^* in this region by the mapping from this state to its feasible action. Therefore, x must be initially feasible under the policy π^* , i.e., $x \in X_{\text{init}}^g(\pi^*)$. This leads to an important condition that $X_{\text{init}}^g(\pi^*) = X_{\text{init}}^g$, and therefore a series of useful properties.

Corollary 5.1.1.

1. $X_{\text{edls}}^g \subseteq X_{\text{edls}}^g(\pi^*) \subseteq X_{\text{init}}^g(\pi^*) = X_{\text{init}}^g$.
2. $X_{\text{edls}}^g \subseteq X_{\text{edls}}^g(\pi^*) \subseteq X_{\text{edls}}^*$.
3. If $X_{\text{edls}}^g = X_{\text{init}}^g$, then $X_{\text{edls}}^g = X_{\text{edls}}^g(\pi^*) = X_{\text{init}}^g(\pi^*) = X_{\text{init}}^g \subseteq X_{\text{edls}}^*$.
4. If $X_{\text{edls}}^g = X_{\text{edls}}^*$, then $X_{\text{edls}}^g = X_{\text{edls}}^g(\pi^*) = X_{\text{edls}}^* \subseteq X_{\text{init}}^g(\pi^*) = X_{\text{init}}^g$.

Proof.

1. According to Theorem 5.1(2), it holds that $X_{\text{edls}}^g(\pi^*) \subseteq X_{\text{init}}^g(\pi^*) \subseteq X_{\text{init}}^g$. What left is to show that $X_{\text{init}}^g \subseteq X_{\text{init}}^g(\pi^*)$ and $X_{\text{edls}}^g \subseteq X_{\text{edls}}^g(\pi^*)$.

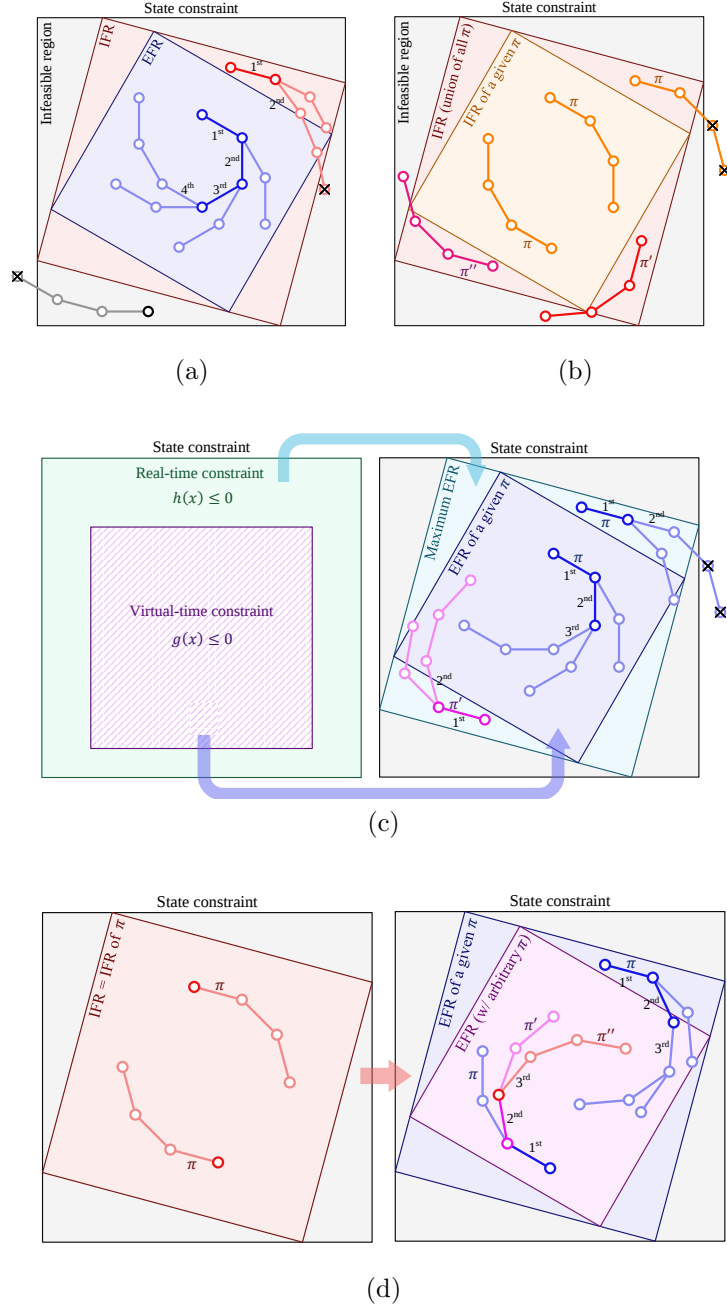


Figure 3: Illustration of containment relationships.

Since

$$\forall x \in X_{\text{init}}^g, \pi^* \in \Pi_{\text{init}}^g(x), \text{ i.e., } x \in X_{\text{init}}^g(\pi^*),$$

we have $X_{\text{init}}^g \subseteq X_{\text{init}}^g(\pi^*)$, which yields $X_{\text{init}}^g(\pi^*) = X_{\text{init}}^g$. Now we can choose $\pi = \pi^*$ in Theorem 5.1(4) and conclude that $X_{\text{edls}}^g \subseteq X_{\text{edls}}^g(\pi^*)$. Thus, we have

$$X_{\text{edls}}^g \subseteq X_{\text{edls}}^g(\pi^*) \subseteq X_{\text{init}}^g(\pi^*) = X_{\text{init}}^g.$$

2. This follows directly from Corollary 5.1.1(1) and Theorem 5.1(3).
3. This follows directly from Corollary 5.1.1(1) and (2).
4. This follows directly from Corollary 5.1.1(1) and (2).

□

Corollary 5.1.1(1) reveals the relationship between IFR, EFR, and those of the optimal policy. This relationship tells us a range of the optimal policy's EFR, i.e., it is lower bounded by EFR and upper bounded by the optimal policy's own IFR. Corollary 5.1.1(2) further clarifies the range of the optimal policy's EFR by revealing its relationship with the maximum EFR, i.e., it is a subset of the latter. Corollary 5.1.1(3) gives a method for obtaining the EFR of the optimal policy under a certain condition that IFR equals EFR. In this case, the EFR of the optimal policy can be easily obtained since it directly equals the IFR. Therefore, this condition also provides a rule for designing virtual-time constraints. Corollary 5.1.1(4) gives a condition when the EFR of the optimal policy equals the maximum EFR. The condition is that the EFR equals the maximum EFR. Combining with Corollary 5.1.1(3), we arrive at an important rule for designing virtual-time constraints: $X_{\text{edls}}^g = X_{\text{init}}^g = X_{\text{edls}}^*$, i.e., EFR, IFR, and the maximum EFR are equal. Following this rule, we can guarantee that the EFR of the optimal policy equals the maximum EFR.

5.2. Equivalence conditions of two feasible regions

The EFR is only determined by the dynamics model and its virtual-time constraint. The perfect design for a virtual-time constraint should result in an EFR that equals the maximum EFR, i.e., $X_{\text{edls}}^g = X_{\text{edls}}^*$. However, X_{edls}^g is very difficult to compute because it requires checking every initially feasible policy in every time step. To avoid this problem, we hope that $X_{\text{edls}}^g = X_{\text{init}}^g$ so that we only need to compute X_{init}^g , which is much easier than computing X_{edls}^g . In this case, if $X_{\text{init}}^g = X_{\text{edls}}^*$, then $X_{\text{edls}}^g(\pi^*) = X_{\text{edls}}^*$. The following

theorem helps us to have a sense of what kind of condition $X_{\text{edls}}^g = X_{\text{init}}^g$ needs.

Theorem 5.2 (Conditions for $X_{\text{edls}}^g = X_{\text{init}}^g$).

Necessary conditions:

1. $X_{\text{init}}^g \subseteq X_{\text{edls}}^*$.
2. $\forall x_t \in X_{\text{init}}^g, \exists u_t \in \mathcal{U}, \text{ s.t. } x_{t+1} \in X_{\text{init}}^g$.

Sufficient conditions (examples of virtual-time constraints):

1. $h(x_{i|t}) \leq 0, i = 0, 1, 2, \dots, \infty$.
2. $h(x_{0|t}) \leq 0, x_{1|t} \in X_{\text{edls}}^*$.

Proof. Necessary conditions:

1. This is obvious by noting Definition 4.5.
2. Since $x_t \in X_{\text{init}}^g$, it follows that $\Pi_{\text{init}}^g(x_t) \neq \emptyset$. Considering that we also have $x_t \in X_{\text{edls}}^g$, any $\pi \in \Pi_{\text{init}}^g(x_t)$ and $u_t = \pi(x_t)$ will induce an initially feasible successive state.

Sufficient conditions:

1. $\forall x_t \in X_{\text{init}}^g$ and $\pi \in \Pi_{\text{init}}^g(x_t)$, we have

$$h(x_{t+i}) = h(x_{i|t}) \leq 0, i = 0, 1, \dots, \infty.$$

Since this goes to infinity, π must also be a feasible policy in x_{t+1} , i.e., $x_{t+1} \in X_{\text{init}}^g$. By the arbitrariness of x_t and π , we can conclude that $x_t \in X_{\text{edls}}^g$ and hence $X_{\text{edls}}^g = X_{\text{init}}^g$.

2. $\forall x_t \in X_{\text{init}}^g$ and $\pi \in \Pi_{\text{init}}^g(x_t)$, we have $x_{t+1} = x_{1|t} \in X_{\text{edls}}^*$, and thus $x_{t+1} \in X_{\text{init}}^g$. By the arbitrariness of x_t and π , it holds that $\forall x \in X_{\text{edls}}^g$. We can conclude that $X_{\text{edls}}^g = X_{\text{init}}^g$.

□

Theorem 5.2 has important guiding significance for designing virtual-time constraints. The necessary conditions should always be satisfied in constraint design. These conditions require that IFR is a subset of the maximum EFR and a control invariant set at the same time. Checking these conditions only involves computing IFR, which is much easier than computing EFR. One may argue that the maximum EFR is unknown so the first necessary condition cannot be checked. In fact, we do not need to exactly know the maximum

EFR. Instead, we just need to check if all states in IFR are endlessly feasible for a given policy. If this is true, then the first necessary condition is satisfied. The sufficient conditions in Theorem 5.2 can be viewed as two examples of virtual-time constraints that can satisfy $X_{\text{edls}}^g = X_{\text{init}}^g$. When designing virtual-time constraints, one should try to satisfy a sufficient condition under the premise that the necessary conditions are already satisfied. Note that there still exist other forms of sufficient conditions that can ensure the equivalence of EFR and IFR. We can choose different ones according to the specific constrained OCP we aim to solve.

6. Feasibility Function and Constraint Types

The goal of safe RL is to find not only the optimal policy but also its EFR. Whether this EFR equals the maximum EFR depends on the choice of virtual-time constraints. Therefore, constructing a proper virtual-time constraint becomes an indispensable task in constrained optimal control. In this section, we first introduce a tool for representing EFR called feasibility function, which helps us construct proper constraints that satisfy the equivalence conditions and enable us to find the maximum EFR. Next, we review several commonly used virtual-time constraint formulations from the perspective of feasibility function. We point out that these virtual-time constraints are all constructed from some feasibility function and therefore satisfy some desirable properties. Moreover, we use the tools introduced in the previous section to analyze the containment relationships of feasible regions under these virtual-time constraints.

6.1. Feasibility function

Feasibility functions are used for constructing virtual-time constraints and representing EFRs. A feasibility function is defined as a mapping from the state space to a real number, i.e., $F : \mathcal{X} \rightarrow \mathbb{R}$. Through proper design, its zero-sublevel set can represent an EFR:

$$X_F = \{x | F(x) \leq 0\}.$$

The core of designing feasibility functions is to offer some kind of recursion properties so that by properly constructing the virtual-time constraint, X_F can become an EFR. There are two basic methods that lead to the design of feasibility functions. The first is defined through a control invariant set (CIS)

and the second is defined through constraint aggregation (CA). A control invariant set is a region in which there exists a policy that keeps all successive states still in this region. A constraint aggregation is a function that can replace the infinite-step real-time constraint with a single-step virtual-time constraint. These two kinds of feasibility functions result in two families of virtual-time constraints. The former type restricts the state in a control invariant set while the latter type constrains an aggregation of infinite-step real-time constraints.

For the first type, the zero-sublevel set of feasibility function is chosen as a control invariant set, i.e., the state is maintained within this set under certain control given that the previous state is in the set. This type of feasibility function is defined as follows.

Definition 6.1 (Control invariant set). *A function $F : \mathcal{X} \rightarrow \mathbb{R}$ is a feasibility function if $X_F \subseteq X_{\text{cstr}}$ is a control invariant set, i.e.,*

$$\forall x \in X_F, \exists u, \text{ s.t. } x' \in X_F.$$

In general, certain inequality conditions are constructed with F to serve as virtual-time constraints when using this type of feasibility function. These constraints require the policy to yield a next state that is still in X_F , starting from whatever state in X_F . That is to say, through the virtual-time constraint built upon it, the feasibility function F equips the resulting policy with the recursive property we are looking for. But does this always work? In other words, how can we be sure that such a policy that satisfies the constraint and thus has the recursive property always exists? The answer is through the special property of the zero-sublevel set of F , i.e., control invariance. Figure 4 gives an illustration of a feasibility function and its control invariant set. Any state x inside the control invariant set can be still kept in this set under some action u , while a state $\tilde{x}$ outside the control invariant set may not be able to enter this set.

For the second type, the feasibility function is chosen as an aggregation function whose one-step virtual-time constraint can represent the infinite-step real-time constraint. This type of feasibility function is defined as follows.

Definition 6.2 (Constraint aggregation). *A function $F : \mathcal{X} \rightarrow \mathbb{R}$ is a feasibility function if $\forall x \in \mathcal{X}$,*

$$F(x) \leq 0 \iff h(x_{t+i}) \leq 0, x_t = x, i = 0, 1, \dots, \infty.$$

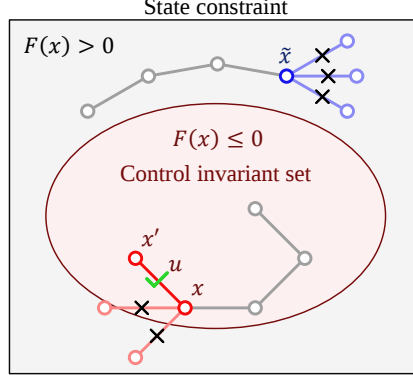


Figure 4: Feasibility function defined through control invariant set.

Figure 5 gives an illustration of a feasibility function defined through constraint aggregation. For states x_1 and x_3 , although they do not violate the state constraint at the current step, some states on their future trajectories leave the constrained set. Therefore, the feasibility function F is positive on these two states. For state x_2 , all of its successive states are inside the constrained set, making the feasibility function less than or equal to zero. Definition 6.2 means that F must be a function of a policy π . We can also denote it as F^π to show its connection with π . This equivalence compresses the infinitely many constraints into F and hence equips it with a natural recursive property, i.e., once $F(x_t) \leq 0$, it follows that $F(x_{t+1}) \leq 0$. However, this recursive property is not a free lunch. We have to make sure that the value of F is practically available before it can be used to construct virtual-time constraints. One may recall that the state-value function and the action-value function in RL also contain infinite future rewards, and we can calculate them iteratively by bootstrapping. Similarly, feasibility functions of constraint aggregation type are usually formulated in such ways that they satisfy a self-consistency condition and hence can be calculated iteratively.

6.2. Type I: control invariant set

This type of constraint restricts the state in a control invariant set, which is represented by the zero-sublevel set of feasibility function. There are mainly two kinds of feasibility functions of the control invariant set type: (a) control barrier function and (b) safety index. They differ in the inequal-

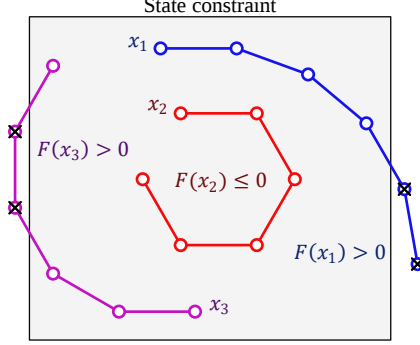


Figure 5: Feasibility function defined through constraint aggregation.

ity conditions that guarantee the invariant property of their zero-sublevel sets.

6.2.1. Control barrier function

Control barrier function (CBF) is a widely used feasibility function for synthesizing safe controllers in constrained OCPs (Ames et al., 2019, 2016; Cheng et al., 2019; Robey et al., 2020). The control invariance of the zero-sublevel set of CBF is guaranteed by the fact that the time derivative of CBF is always non-positive on the boundary of this set. As a state in the zero-sublevel set approaches the boundary, its CBF value may increase at first but will finally stop at some value not greater than zero. Therefore, the CBF will never take a value above zero as long as the initial state is in its zero-sublevel set. The formal definition of CBF in a discrete-time system is as follows.

Definition 6.3 (Control barrier function). *A function $B : \mathcal{X} \rightarrow \mathbb{R}$ is a control barrier function if it satisfies:*

1. $\forall x \in \mathcal{X}$, if $h(x) > 0$, then $B(x) > 0$,
2. $\forall x \in \mathcal{X}, \exists u \in \mathcal{U}$, s.t. $B(x') - B(x) \leq -\alpha(B(x))$,

where $x' = f(x, u)$ and $\alpha : \mathbb{R} \rightarrow \mathbb{R}$ is a strictly increasing function and $|\alpha(z)| \leq |z|$ for all $z \in \mathbb{R}$.

If a function satisfies the above two properties, it can be proved that its zero-sublevel set is a control invariant set (Ames et al., 2019). Therefore,

according to Definition 6.1, a CBF is a feasibility function itself, i.e.,

$$F(x) = B(x).$$

Property 2) in Definition 6.3 is the most critical element for ensuring control invariance of the zero-sublevel set. This property tells us that there exists an action such that the increment of CBF from the current step to the next step is upper bounded. When solving a constrained OCP, we must take it as a virtual-time constraint to ensure that the obtained action actually satisfies this property. This leads to a one-step constraint imposed on the second state in virtual-time domain, i.e., $x_{1|t}$. In addition, $B(\cdot) \leq 0$ is imposed on the first state in virtual-time domain, i.e., $x_{0|t}$. This is because we must ensure that the first state is in the control invariant set so that the successive states can be kept in this set. This first-step virtual-time constraint is not weaker than the real-time state constraint because property 1) in Definition 6.3 tells us that $h(\cdot) > 0$ implies $B(\cdot) > 0$, which is equivalent to the statement that $B(\cdot) \leq 0$ implies $h(\cdot) \leq 0$. Combining these two constraints, we have

$$\begin{aligned} g(x_{0|t}) &= B(x_{0|t}) \leq 0, \\ g(x_{1|t}) &= B(x_{1|t}) - B(x_{0|t}) + \alpha(B(x_{0|t})) \leq 0. \end{aligned} \tag{5}$$

The above constraint guarantees that the next state of $x_{0|t}$, which is $x_{1|t}$, is still in the zero-sublevel set of CBF. If all states in the zero-sublevel set satisfy this constraint, the set is forward invariant, i.e., the state is always kept in this set. Therefore, the state constraint in real-time domain is always satisfied.

In literature, there is a notion of high order control barrier function (HOCBF) (Xiao and Belta, 2019) that is used for high relative degree constraints. Here, the relative degree of a function refers to its lowest order of time derivative where the action explicitly appears. When the constraint function h has a high relative degree, directly choosing h as a CBF cannot effectively constrain the action because constraint (5) is irrelevant to the action. This necessitates HOCBF, which considers time derivatives of h where the action explicitly appears. When the time derivatives satisfy certain properties that ensure control invariance, h is an HOCBF (Xiao and Belta, 2019). Although not explicitly stated, Definition 6.3 is actually compatible with HOCBF because B does not have to be chosen as h and can be constructed as some function that makes the action explicitly appear in constraint (5).

Let us analyze the size of IFR and EFR under constraint (5). For IFR, we look for the state that if we take it as the starting point of virtual-time OCP with constraint (5), this OCP has a solution. In the first step, all states in the zero-sublevel set of B satisfy the virtual-time constraint. For notation simplicity, we denote this set as $X_B = \{x \in \mathcal{X} | B(x) \leq 0\}$. In the second step, according to property 2) in Definition 6.3, all states in the entire state space satisfy the virtual-time constraint. The IFR is the intersection of these two sets, which is X_B , i.e.,

$$X_{\text{init}}^g = X_B.$$

For EFR, we look for the state from which the OCP with constraint (5) has a solution in all successive states as long as an initially feasible policy is applied in every step. According to Theorem 5.1, EFR must be a subset of IFR. Since the IFR under constraint (5) is X_B , the EFR must be a subset of X_B . For any state in X_B , its successive state is still in X_B if we take an action such that the second constraint in (5) is satisfied. This requirement for the action is satisfied by any initially feasible policy. Thus, the state can always be kept in X_B as long as an initially feasible policy is applied at every time step. Moreover, since X_B is the IFR, the OCP with constraint (5) always has a solution in it. Therefore, we can conclude that the EFR equals X_B , i.e.,

$$X_{\text{edls}}^g = X_B.$$

We can see that EFR equals IFR under constraint (5). This shows the benefit of feasibility function for designing virtual-time constraints, i.e., it ensures the equivalence of EFR and IFR. This equivalence enables us to know the size of EFR by examining that of IFR. In a constrained OCP, we always aim to construct an EFR that equals the maximum EFR so that the policy obtained by solving the OCP has the largest working area. However, the size of EFR is difficult to obtain because it involves checking the constraint satisfaction in an infinite horizon. With the equivalence of EFR and IFR, we can avoid this difficulty by obtaining the size of IFR instead. If we construct an IFR that equals the maximum EFR, then the equivalence of EFR and the maximum EFR is also ensured. For a virtual-time constraint designed by CBF, its IFR is the zero-sublevel set of the CBF, i.e., X_B . The size of this set depends on the design of CBF. A better design results in a larger X_B , which is closer to the maximum EFR. In the best case, X_B equals the maximum EFR. For a general CBF, we can only say that X_B is a subset of the maximum EFR, i.e.,

$$X_B \subseteq X_{\text{edls}}^*.$$

Therefore, the relationship among the feasible regions is as follows:

$$X_{\text{init}}^g = X_{\text{edls}}^g \subseteq X_{\text{edls}}^*.$$

In conclusion, a CBF is a feasibility function that represents EFR using a control invariant set, which is its zero-sublevel set. A CBF must satisfy two properties, of which the first requires that the states in the control invariant set are constraint-satisfying, and the second requires that the increase rate of CBF is upper bounded under some action so that it will never exceed zero in the control invariant set. With a CBF, we can construct a two-step virtual-time constraint that restricts the state in the control invariant set in the first step and requires the action to satisfy the increase rate bound in the second step. Both IFR and EFR of this virtual-time constraint equal the control invariant set of CBF. However, this set may not equal the maximum EFR, i.e., it may be smaller than the maximum EFR. The size of control invariant set depends on the design of CBF. In systems with high-dimensional state and action spaces or systems without analytical models, it is usually difficult to handcraft a CBF whose control invariant set is equal to or close to the maximum EFR. Designing a CBF with the maximum control invariant set, i.e., the maximum EFR, in these systems remains a challenge.

6.2.2. Safety index

Safety index, originally proposed by Liu et al. (Liu and Tomizuka, 2014), is another feasibility function that represents a control invariant set by its zero-sublevel set, which is called safe set. Safety index ensures control invariance in a similar way to CBF. In particular, it is required that once the system deviates from the safe set, the control policy will pull it back. This is achieved by setting the time derivative of safety index to be negative outside the safe set. Safety index differs from CBF in that it explicitly considers the relative degree of system in construction. For systems with a high relative degree, the first-order time derivative is not enough to ensure control invariance because it does not constrain the action. To deal with this issue, safety index adopts a specific function form, which contains the constraint function or its nonlinear substitution, and a linear combination of its time derivatives. The formal definition of safety index is as follows.

Definition 6.4 (safety index). *A function $\phi : X \rightarrow \mathbb{R}$ is a safety index if it satisfies the following three properties:*

1. $\phi = h + k_1\dot{h} + \dots + k_n h^{(n)}$, where $k_1, \dots, k_n \in \mathbb{R}$,

2. All roots of $1 + k_1s + k_2s^2 + \dots + k_ns^n = 0$ are on the negative real line,
3. The relative degree from $h^{(n)}$ to u is one,

where $n + 1$ is the system order. Moreover, suppose h^* defines the same set as h does, i.e., $\{x \in \mathcal{X} | h^*(x) \leq 0\} = X_{\text{cstr}}$, then

$$\phi^* = \phi - h + h^*$$

is also a safety index.

Property 3) above ensures that the relative degree of safety index is one so that the action can be constrained by its first-order time derivative. With the above three properties, it can be proved that when the action u is unconstrained, there always exists u that satisfies $\dot{\phi} \leq 0$ when $\phi = 0$ (Liu and Tomizuka, 2014). When u is constrained in some space $\mathcal{U}$, a 4-th condition needs to be added: the parameters $k_i, i = 1, 2, \dots, n$ are chosen such that there exist $u \in \mathcal{U}$ for $\dot{\phi} \leq 0$ when $\phi = 0$. This is a necessary and sufficient condition for control invariance of zero-sublevel set of ϕ , i.e., the safe set. In Definition 6.4, the use of h^* is to introduce nonlinearity to the safety index. Since it defines the same set as h , it only shapes the boundary of the feasible region. With a safety index, we can construct the following feasibility function:

$$F(x) = \max\{\phi(x), h(x)\}.$$

Here, including h in the maximum operator is to ensure that the feasible region is a subset of the constrained set. When $h^* = h$ (i.e. the safety index is linearly defined), the feasible region of safety index can be represented by a set of inequalities similar to that of HOCBF (Xiao and Belta, 2019). When $h^* \neq h$ (i.e., the safety index is nonlinearly defined) and $n = 1$ (i.e., the dynamic system is second order), the feasible region of safety index equals the zero-sublevel set of F . For $h^* \neq h$ and $n > 1$, there is no conclusion regarding the exact representation of the feasible region with F .

The control invariance of safe set is guaranteed by the inequality $\dot{\phi} \leq 0$ when $\phi = 0$. This is a differential inequality and only applies to continuous-time systems. To make it applicable to discrete-time systems, we must convert it to a difference inequality. An intuitive conversion is to replace the derivative of ϕ with its discretization obtained by, for example, the forward Euler method, which results in the following inequality, $\phi(x_{1|t}) - \phi(x_{0|t}) \leq 0$ when $\phi(x_{0|t}) = 0$. However, this inequality only constrains the value of

ϕ when $\phi(x_{0|t})$ is exactly zero, which is not enough to guarantee the control invariance of safe set in discrete-time systems. This is because even if $\phi(x_{0|t}) < 0$, it is possible that ϕ becomes positive after only one step at $x_{1|t}$. Therefore, we must also constrain the value of ϕ when $\phi(x_{0|t}) < 0$. Specifically, we require that $\phi(x_{1|t}) < 0$ so that $x_{1|t}$ is still in the safe set. When $\phi(x_{0|t}) > 0$, $x_{0|t}$ is already out of the safe set. In this case, ϕ should decrease fast enough so that the state can return to the safe set quickly. Specifically, we require that $\phi(x_{1|t}) \leq \phi(x_{0|t}) - \eta$, where η is a positive real number. This inequality can be further relaxed to $\phi(x_{1|t}) \leq \max\{\phi(x_{0|t}) - \eta, 0\}$ because once $\phi(x_{1|t}) \leq 0$, the state already enters the safe set. The above two inequalities can be written as the following inequality,

$$\phi(x_{1|t}) - \max\{\phi(x_{0|t}) - \eta, 0\} \leq 0,$$

which acts as a virtual-time constraint of safety index. Similar to the virtual-time constraint of CBF, the safety index also needs a first-step constraint that restricts the state in the safe set. Combining these two constraints, we arrive at the virtual-time constraint of safety index,

$$\begin{aligned} g(x_{0|t}) &= \max\{\phi(x_{0|t}), h(x_{0|t})\} \leq 0, \\ g(x_{1|t}) &= \phi(x_{1|t}) - \max\{\phi(x_{0|t}) - \eta, 0\} \leq 0. \end{aligned} \tag{6}$$

The above constraint satisfies the requirement that the virtual-time constraint in the first time step must not be weaker than the real-time constraint. The second constraint in (6) requires that the safety index decreases every time step by at least a value of η when it is above zero. When the safety index is below zero, it is required not to become positive. With a policy satisfying constraint (6), the zero-sublevel set of safety index is forward invariant.

The relationship between feasible regions under safety index constraint is similar to that of CBF, so we omit the analysis here. One may discover that safety index is closely related to CBF in both feasibility function formulation and virtual-time constraint design. This relationship is not a coincidence but rather originates from the challenge of constructing a CBF. While CBF guarantees control invariance of its zero-sublevel set by definition, it does not provide any guidance for its construction. HOCBF provides one design rule for obtaining a valid CBF. Safety index can be considered as another approach to achieving this objective. However, we have seen that these methods, which define feasibility functions through control invariant sets, may not be able to obtain the maximum EFR. In contrast, the following feasibility function defined by constraint aggregation can lead to the maximum EFR.

6.3. Type II: constraint aggregation

This type of constraint replaces the infinite-step real-time constraints with a single-step virtual-time constraint by an aggregation function. Under this perspective, there are mainly two kinds of feasibility functions of this type: cost value function and Hamilton-Jacobi reachability function. The former uses a discounted summation function as the aggregation function while the latter uses a maximum function as the aggregation function. There are two types of virtual-time constraints constructed by this kind of feasibility function. One type uses the feasibility function of the current policy, which is to be optimized, to construct constraints. The other uses the feasibility function of a fixed policy, e.g., the policy from the last iteration, to construct constraints.

6.3.1. Cost value function

Cost value function (CVF) originates from a common formulation for safe RL, constrained Markov decision process (CMDP) (Altman, 1999), which augments a standard MDP with a cost function in parallel with the reward function. Being the expected cumulative costs, CVF shares the same mathematical form, and therefore the same properties, with state-value function $V^\pi(x)$, which greatly facilitates its usage. With a slight effort to define a cost function $c(x)$ with the constraint function $h(x)$, CVF can be well adapted to constrained OCPs. The definition of CVF is as follows.

Definition 6.5 (cost value function). *For a constrained OCP with $h(x)$ as a constraint function, $x_t, t = 0, 1, 2, \dots, \infty$ denoting the state trajectory starting from state x under policy π , the cost value function $F^\pi : \mathcal{X} \rightarrow \mathbb{R}$ is:*

$$F^\pi(x) = \sum_{t=0}^{\infty} \gamma^t c(x_t),$$

where $c(x) = \mathbf{1}[h(x) > 0]$ is the cost signal.

To see that CVF is a valid feasibility function, note that $c(x)$ is non-negative for all x , so that $F^\pi(x) \leq 0$ is sufficient and necessary for $c(x_t) = 0, \forall t = 0, 1, \dots, \infty$. The close relationship between CVF and state-value function implies some common properties. Actually, it is indeed the fact that self-consistency condition and Bellman equation, which play important roles in solving state-value functions, have their respective counterparts when

it comes to CVFs. For the sake of distinction, they are termed risky self-consistency condition:

$$F^\pi(x) = c(x) + \gamma F^\pi(x'),$$

and risky Bellman equation:

$$F^*(x) = c(x) + \gamma \min_u F^*(x').$$

Similar to the case for state-value function, the right-hand sides of the above two equations can also be viewed as contraction operators on a complete metric space. Hence, by fixed-point iteration, a CVF can be solved just like a state-value function.

Until now, we have defined a valid feasibility function with CVF and showed that it is practically available, the next thing is to further construct a virtual-time constraint with it. The first way of constructing a virtual-time constraint is

$$g(x_{0|t}) = F^\pi(x_{0|t}) \leq 0, \quad (7)$$

and $x_{i|t}, i = 1, \dots, n/\infty$ are unconstrained. Here, the superscript π of F is the policy to be optimized. Although $x_{0|t}$ is not affected by π , constraint (7) is related to π because F^π is a function of π . Technically speaking, for a given $x_{0|t}$, $F^\pi(x_{0|t})$ is a functional of π . In practice, F^π is difficult to compute directly and is usually approximated by importance sampling of trajectories collected by another policy, similar to the way of approximating value function in on-policy RL algorithms. This form of virtual-time constraint is compatible with direct RL/ADP methods for constrained OCPs so that it can serve as an equivalent substitute for the real-time constraint. Through the aggregation of constraints introduced by the summation function, the horizon of virtual-time constraint reduces to only one step, the current step. The IFR under constraint (7) is by definition simply

$$X_{\text{init}}^g = \{x | \exists \pi, \text{ s.t. } F^\pi(x) \leq 0\}.$$

And no matter which initially feasible policy we apply at the current step, we will always reach a state in the IFR. This is because this policy must still be initially feasible in the next state, which is guaranteed by the fact that the CVF's being less than or equal to 0 at a single state renders all future states safe. As a result, the EFR is still

$$X_{\text{edls}}^g = \{x | \exists \pi, \text{ s.t. } F^\pi(x) \leq 0\}.$$

What's more, an excellent property of constraint (7) is that

$$X_{\text{edls}}^g = X_{\text{init}}^g = X_{\text{edls}}^*.$$

The first equation is obvious from the above analysis. To see why the second equation holds, recall Theorem 4.1 and that $F^\pi(x) \leq 0$ is equivalent to $h(x_t) \leq 0, t = 0, 1, \dots, \infty$. The equivalence of these three regions is satisfying, for we can easily check the endless feasibility of a state by its initial feasibility, and the EFR is the maximum one that can ever be obtained.

The second way of constructing a virtual-time constraint is a little bit complex, involving both the current step and the next step:

$$g(x_{i|t}) = F^k(x_{i|t}) \leq 0, i = 0, 1, \quad (8)$$

and $x_{i|t}, i = 2, \dots, n/\infty$ are unconstrained. Unlike the former type, here the constraint function, which is the CVF of a fixed policy π_k , is irrelevant to the policy π to be optimized. Although $F^k(x_{0|t})$ is actually unrelated with π , $x_{1|t}$ is induced by π , which makes constraint (8) effective. Under this constraint, for a state x to be initially feasible, the first requirement is that $g(x_{0|t}) = F^k(x_{0|t}) \leq 0$, where $x_{0|t} = x$. This implies that if we take $\pi = \pi_k$, we will have $F^k(x_{1|t}) \leq 0$. Consequently, any x satisfying $F^k(x) \leq 0$ is an initially feasible state, with π_k being an initially feasible policy in it. That is,

$$X_{\text{init}}^g = \{x | F^k(x) \leq 0\}.$$

Because of the same reason analyzed above, we still have

$$X_{\text{edls}}^g = \{x | F^k(x) \leq 0\}.$$

Unfortunately, under this situation, we don't have as good equivalence to the maximum EFR as above. The IFR and EFR are related to a fixed policy which can be highly unsatisfying, rendering only a few states safe. So, we can only conclude that

$$X_{\text{edls}}^g = X_{\text{init}}^g \subseteq X_{\text{edls}}^*.$$

However, by properly choosing π_{k+1} under the guide of $F^{\pi_k}(x)$, the fact is that we can ensure the monotonic expansion of EFR and it will converge to the maximum EFR (Yang et al., 2023b).

6.3.2. Hamilton-Jacobi reachability function

The Hamilton-Jacobi (HJ) reachability analysis, a technique from robust optimal control theory, is used to guarantee constraint satisfaction in a rigorous way. HJ reachability computes a backward reachable set of a system, i.e., the set of states from which trajectories can reach some given target set (Mitchell et al., 2005; Bansal et al., 2017). For feasibility analysis, the target set is the unconstrained set, and the backward reachable set becomes the infeasible region. HJ reachability function aggregates infinite-step constraints by taking the maximum of them. Therefore, it considers the worst-case constraint violation in the entire trajectory, or the “closest distance” to the constraint boundary if no violation is going to happen. The definition of HJ reachability function is as follows.

Definition 6.6 (HJ reachability function). *For a constrained OCP with $h(x)$ as a constraint function, $x_t, t = 0, 1, 2, \dots, \infty$ donating the state trajectory starting from state x under policy π , the HJ reachability function $F^\pi : \mathcal{X} \rightarrow \mathbb{R}$ is:*

$$F^\pi(x) = \max_t h(x_t).$$

The constraint function $h(x)$ indicates the safety of the current state. As its extension to the whole temporal domain, $F^\pi(x)$ does not describe the risk level of the current state, but the worst case in the future. It is the maximizer that attempts to find the most dangerous constraint point along the whole state trajectory. Therefore, $F^\pi(x) \leq 0$ is equivalent to $h(x_t) \leq 0$ for $t = 0, 1, \dots, \infty$.

Similar to CVF, HJ reachability function naturally satisfies a risky self-consistency condition and the optimal HJ reachability function satisfies a risky Bellman equation. The risky self-consistency condition is

$$F^\pi(x) = \max\{h(x), F^\pi(x')\}.$$

The risky Bellman equation is

$$F^*(x) = \max\{h(x), \min_u F^*(x')\}.$$

In practical HJ reachability computation, a discount factor is needed on the right hand side of the above two equations so that they become contraction mappings, which enable the convergence of fixed point iteration (Fisac et al., 2019).

There are two ways to construct a virtual-time constraint with HJ reachability function. One way is to use a first-step constraint:

$$g(x_{0|t}) = F^\pi(x_{0|t}) \leq 0, \quad (9)$$

and $x_{i|t}, i = 1, \dots, n/\infty$ are unconstrained. By aggregating constraints using the maximum function, the virtual-time constraint is replaced by the first-step constraint of HJ reachability function. The other way is to use a two-step constraint:

$$g(x_{i|t}) = F^k(x_{i|t}) \leq 0, i = 0, 1, \quad (10)$$

and $x_{i|t}, i = 2, \dots, n/\infty$ are unconstrained. We omit the analysis of their corresponding feasible regions and containment relationships since it is similar to the case of CVF.

6.3.3. Constraint decay function

Constraint decay function (CDF) uses the remaining steps from the current state to constraint violation to aggregate the infinite-step constraints. Its basic idea is that if a state is infeasible, it will lead to a constraint violation in a finite number of steps. Otherwise, no constraint violation will happen in finite steps, i.e., the number of steps to constraint violation is infinite. The CDF is constructed as an exponential function with the number of steps to constraint violation as the exponent and a real number between zero and one as the base, which is formally defined as follows.

Definition 6.7 (constraint decay function). *For a constrained OCP, the constrained decay function $F^\pi : \mathcal{X} \rightarrow \mathbb{R}$ is*

$$F^\pi(x) = \gamma^N(x),$$

where $0 < \gamma < 1$ is the discount factor and N is the number of steps to constraint violation starting from x under π .

According to the property of exponential functions, as N increases from zero to infinity, $F^\pi(x)$ decreases from one to zero. A larger value of CDF indicates that the current state is more dangerous in the sense that it is closer to constraint violation. A value of one means that the current state already violates the constraint. A value of zero means that the constraint will never be violated in the infinite horizon. In other words, $F^\pi(x) \leq 0$ is a necessary and sufficient condition for infinite-horizon constraint satisfaction. Therefore,

CDF is a valid feasibility function of constraint aggregation type. The CDF given by Definition 6.7 is similar in form to the safety critic used in some safe RL literature, e.g., Thananjeyan et al. (Thananjeyan et al., 2021). Their difference is that they are used in different kinds of systems and for different purposes. The safety critic is used in stochastic systems for estimating the discounted probability of future constraint violation. The CDF is used in deterministic systems as a feasibility function for constructing virtual-time constraints and representing feasible regions.

Similar to other feasibility functions of constraint aggregation type, CDF naturally satisfies a risky self-consistency condition and the optimal CDF satisfies a risky Bellman equation. The risky self-consistency condition is

$$F^\pi(x) = c(x) + (1 - c(x))\gamma F^\pi(x').$$

The risky Bellman equation is

$$F^*(x) = c(x) + (1 - c(x))\gamma \min_u F^*(x').$$

There are two ways to construct a virtual-time constraint with CDF. The first way is to constrain the CDF of policy to optimize in the first time step,

$$g(x_{0|t}) = F^\pi(x_{0|t}) \leq 0, \quad (11)$$

and $x_{i|t}, i = 1, \dots, \infty$ is unconstrained. The second way is to constrain the CDF of a fixed policy in the first two time steps,

$$g(x_{i|t}) = F^k(x_{i|t}) \leq 0, i = 0, 1, \quad (12)$$

and $x_{i|t}, i = 2, \dots, \infty$ is unconstrained. We omit the analysis of their corresponding feasible regions and containment relationships since they are similar to the case of other feasibility functions of constraint aggregation type.

7. Review of Constraint Formulations

In this section, we review several kinds of commonly used virtual-time constraint formulations. In essence, these formulations are applications of different types of feasibility functions. Therefore, we categorize them according to the feasibility functions they use, i.e., control invariant set or constraint aggregation.

7.1. Control invariant set

7.1.1. Control barrier function

The concept of CBF originated from the field of control theory and was first proposed by Ames et al. (Ames et al., 2014) for continuous-time systems. It was later extended to discrete-time systems by Agrawal et al. (Agrawal and Sreenath, 2017). A large category of works solves OCPs with CBF constraints through online optimization, i.e., a single optimal action is computed each time the system arrives at a state. Ames et al. (Ames et al., 2016) unify CBF with control Lyapunov function (CLF) in the context of quadratic program (QP), in which performance objective is expressed by CLF and safety constraint is expressed CBF. The optimal control is obtained by solving the QP online and is ensured to keep the state in the invariant set of CBF. Nguyen et al. (Nguyen and Sreenath, 2016) propose exponential CBFs that enforce strict satisfaction of high relative degree safety constraints for nonlinear systems. They also develop a method for designing exponential CBFs based on techniques from linear control theory. Taylor et al. (Taylor et al., 2020) use a learning-based method to reduce model uncertainty in order to enhance the safety of a CBF-certified controller. Their approach iteratively collects data and updates the controller, ultimately achieving safe behavior.

Other works solve CBF-constrained OCPs in an offline manner using RL, i.e., they learn a policy that maps states to their corresponding optimal actions. Cheng et al. (Cheng et al., 2019) ensure safety of a model-free RL controller by combining it with a model-based CBF controller and online learning of unknown system dynamics. The CBF controller both guarantees safety and guides the learning process by constraining the set of explorable policies. Ohnishi et al. (Ohnishi et al., 2019) propose a barrier-certified adaptive RL algorithm, which constrains policy in the invariant set of CBF and optimizes the action-value function in this set. Their solutions to barrier-certified policy optimization are guaranteed to be globally optimal under mild conditions. Ma et al. (Ma et al., 2021) use a generalized CBF for systems with high relative degrees and mitigate the infeasibility of constrained policy optimization by an adaptive coefficient mechanism.

Most of the above works handcraft CBFs as functions with known forms, while some other works synthesize CBFs using neural networks. Robey et al. (Robey et al., 2020) leverage safe trajectories generated by an expert to optimize a CBF in control affine systems. The learned CBF enjoys provable safety guarantees under Lipschitz smoothness assumptions on system

dynamics. Qin et al. (Qin et al., 2020) jointly learn multi-agent control policies and CBFs in a decentralized framework. They propose a spontaneous policy refinement method to further enforce CBF conditions during testing. Yang et al. (Yang et al., 2023a) propose a safe RL algorithm that learns both policy and CBF in a model-free manner. They extend the CBF invariant loss to a multi-step version, which balances bias and variance and enhances both safety and performance of the policy.

7.1.2. Safety index

Safety index is commonly used as a safeguard for another (possibly unsafe) controller, e.g., an RL controller that solely maximizes reward performance. Zhao et al. (Zhao et al., 2021) propose a model-free RL algorithm that leverages a safety index to ensure zero constraint violation during training. This is achieved by an implicit safe set algorithm, which searches for safe control only by querying a black-box dynamic function. Ma et al. (Ma et al., 2022) simultaneously synthesize a safety index and learn a safe control policy with constrained RL. They learn the safety index by minimizing the occurrence of energy increases, which does not rely on knowledge about a prior controller.

In systems with control limits, there may be situations where it is impossible to find an action that satisfies the constraint of a safety index. Some works focus on how to synthesize a valid safety index under control limits. Wei et al. (Wei et al., 2022) propose a control-limits-aware safety index synthesis method for systems with bounded state-dependent uncertainties. They use convex semi-infinite programming to solve for a robust safe controller so that it is guaranteed to be realizable under control limits. Zhao et al. (Zhao et al., 2023c) propose a method for synthesizing safety index in general systems with control limits. They prove that ensuring the existence of safe control on a safe set boundary is equivalent to sum-of-squares programming. Zhao et al. (Zhao et al., 2023b) present an integrated dynamic model learning and safe control framework to safeguard any RL agent. They provide a design rule to construct a safety index under control limits and a probabilistic safety guarantee under stochastic dynamic models.

7.2. Constraint aggregation

7.2.1. Cost value function

Plenty of existing works, mostly concentrating on the setting of CMDP, have applied CVF to handle constraints. Generally, most works follow the

standard notion of CMDPs, where the constraints are imposed directly on CVF itself. Chow et al. (Chow et al., 2017) build a constraint based on the conditional value-at-risk of CVF and propose Lagrange multiplier methods for the constrained optimization problem. Ding et al. (Ding et al., 2020) also employ the primal-dual approach but update the primal variable via natural policy gradient and the dual variable via projected sub-gradient. This work is extended to an entropy-regularized case in (Ying et al., 2022). An equivalent linear formulation of the objective and the CVF-based constraint (Altman, 1999) is considered in (Bai et al., 2022) and solved by a stochastic primal-dual algorithm. To dampen the significant oscillation of state and cost value function during training when using Lagrange multiplier methods, Stooke et al. (Stooke et al., 2020) and Peng et al. (Peng et al., 2022) use PID control to update the Lagrange multiplier for a stabler intermediate performance. As et al. (As et al., 2022) use Bayesian world models to estimate an optimistic upper bound on task objective and pessimistic upper bounds on safety constraints. They use the augmented Lagrangian method to solve the constrained optimization problem based on these two bounds.

Besides Lagrangian-based methods, other approaches have also been proposed for solving OPCs with CVF constraints. Liu et al. (Liu et al., 2022) deal with instability issues of primal-dual style methods by introducing the Expectation-Maximization approach. By adopting a non-parametric variational distribution, the constrained optimization problem in the expectation step becomes convex and can be solved analytically. In (Liu et al., 2020), the authors introduce the interior-point method to augment the objective with logarithmic barrier functions composed with CVF. Chow et al. (Chow et al., 2018) propose a Lyapunov-based approach for transient MDPs which constructs Lyapunov functions w.r.t an undiscounted version of CVF. Achiam et al. (Achiam et al., 2017) construct constrained optimization problems on the basis of a novel bound on the difference in cumulative rewards or costs between two policies and solve them with trust region methods. However, these optimization problems may be infeasible, which undermines the theoretical monotonicity. Yang et al. (Yang et al., 2020) address this by first applying an unconstrained trust region method and then projecting the policy back onto the constrained set. Likewise, Zhang et al. (Zhang et al., 2020) also propose a two-stage algorithm that first searches for a solution of a constrained optimization problem in the non-parameterized policy space and then projects it back into the parametric one. Yu et al. (Yu et al., 2022b) propose a two-policy method where a safety editor, serving as an extension of

a safety shield, is trained to reconcile possible constraint violation with minimal influence on the objective. Zhao et al. (Zhao et al., 2023a) introduces the framework of maximum MDP, which is an extension of CMDP that constrains the expected maximum state-wise cost along a trajectory. Under this framework, they propose state-wise constrained policy optimization (SCPO) algorithm, which provides guarantees for state-wise constraint satisfaction in expectation.

7.2.2. *Hamilton-Jacobi reachability function*

HJ reachability analysis computes the backward reachable set of a constrained system, which is its infeasible region, and safeguards controllers from entering this set. Seo et al. (Seo et al., 2019) use the reachable set obtained by HJ reachability analysis to safeguard receding horizon planning against unknown bounded disturbances. They approximate the reachable set using ellipsoidal parameterization and plan a robust trajectory that avoids risky regions under disturbance. The exact computation of HJ reachability function requires solving an HJ partial differential equation (PDE) on a grid discretization of state space, resulting in an exponential computational complexity with respect to system dimension (Bansal et al., 2017). Many efforts have been made to reduce the computational burden of HJ reachability. Rubies et al. (Rubies-Royo et al., 2019) approximates the optimal controller of HJ reachability problem in control-affine systems as a sequence of simple binary classifiers, thus avoiding storing a representation of HJ reachability function. Herbert et al. (Herbert et al., 2021) propose several techniques, including decomposition, warm-starting, and adaptive grids, to speed up the computation of HJ reachability function. Their methods can update safe sets by one or more orders of magnitude faster than prior work.

Other works further accelerate computation by approximating HJ reachability function with neural networks. Fisac et al. (Fisac et al., 2019) introduce a time-discounted modification of HJ reachability function, which induces a contraction mapping and enables solving it by a fixed point iteration method. Their obtained reachability function approximates the maximum safe set and the safest policy. Based on this work, Yu et al. (Yu et al., 2022a) further consider policy performance optimization in safe RL. They use HJ reachability function to construct virtual-time constraints and solve for the optimal safe policy with the Lagrange method. Bansal et al. (Bansal and Tomlin, 2021) develop a neural PDE solver for high-dimensional reachability problems. The computational requirements of their method do not scale directly with the

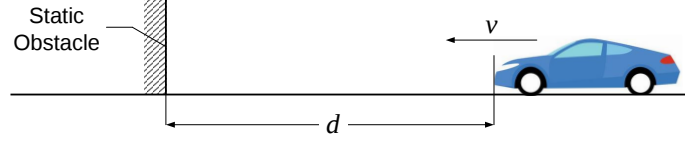


Figure 6: Emergency braking control scenario.

state dimension but rather with the complexity of the underlying reachable tube.

8. Example of Emergency Braking Control

In this section, we illustrate the proposed concepts, including feasible regions, containment relationships between them, and feasibility function with an emergency braking control problem, where a vehicle is supposed to avoid crash with minimum effort. Fig. 6 gives a schematic of this task.

The vehicle has a two-dimensional state $x_t = [d_t, v_t]^\top$ and a one-dimensional action $u_t = a_t$, where d is the distance to the static obstacle, v is the longitudinal velocity and a is the longitudinal acceleration. It follows a simplified longitudinal dynamics:

$$\begin{bmatrix} d_{t+1} \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_t \\ v_t \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} a_t, \quad (13)$$

where Δt is the time step size, assigned with 0.1s. We assume a maximum braking deceleration $a_{\text{brk}} = -10\text{m/s}^2$ so that the action is bounded in $[a_{\text{brk}}, 0]$. The reward function (or oppositely in control terminology, the utility function) is

$$r(x_t, u_t) = -u_t^2 = -a_t^2.$$

The overall objective to be maximized is

$$J = \sum_{i=0}^{\infty} \gamma^i r(x_{t+i}, u_{t+i}) = - \sum_{i=0}^{\infty} \gamma^i a_{t+i}^2. \quad (14)$$

The safety constraint requires that the vehicle-to-obstacle distance is above zero at all times, requiring collision-free control:

$$h(x_{t+i}) = -d_{t+i} \leq 0, i = 0, 1, 2, \dots, \infty. \quad (15)$$

The objective (14) follows a general RL setting. To show the applicability of our proposed theoretical framework to both MPC and RL, we also involve an MPC controller in the experiments. The objective of the MPC controller is limited to a finite horizon with length $N = 10$ and has no discounting, i.e., $\gamma = 1$.

In the following analysis, we build four types of virtual-time constraints: (1) pointwise constraint, (2) CBF constraint, (3) safety index constraint, and (4) HJ reachability constraint. Constraint (1) is a non-feasibility-function-based virtual-time constraint, constraints (2) and (3) are CIS-based, and constraint (4) is CA-based.

Pointwise constraint is composed of a finite number of real-time constraints, i.e.,

$$h(x_{i|t}) = -d_{i|t} \leq 0, i = 0, 1, 2, \dots, n, \quad (16)$$

where $n < \infty$. Since this constraint is not constructed by a feasibility function, its feasible regions are not readily accessible, and its IFR and EFR may not be equivalent.

CBF is handcrafted in the following form:

$$B(x_{i|t}) = -d_{i|t} + kv_{i|t}^2, \quad (17)$$

where k is a tunable parameter. This form is obtained by considering laws of kinematics. Specifically, when braking with a constant acceleration, the distance-to-go is a quadratic function of the current velocity. This form can also be understood from the perspective of HOCBF. If directly choosing the constraint function h as the CBF, its relative degree is two and we should use an HOCBF that considers the first order time derivative of h , which is exactly what (17) does. The corresponding CBF constraint applies to the first two steps in virtual-time domain:

$$\begin{aligned} B(x_{0|t}) &\leq 0, \\ B(x_{1|t}) - (1 - \alpha)B(x_{0|t}) &\leq 0, \end{aligned} \quad (18)$$

where α is a constant assigned with 0.1. Strictly speaking, the choice of α is not arbitrary when the action is constrained. It should be chosen such that property 2) in Definition 6.3 is satisfied. Otherwise, B may not be a valid CBF and its zero-sublevel set may not be control invariant. Here, we choose the value of α empirically without verifying property 2).

Safety index follows the form used by Zhao et al. (Zhao et al., 2021):

$$\phi(x_{i|t}) = \sigma + d_{\min}^n - d_{i|t}^n + kv_{i|t}, \quad (19)$$

where $d_{\min}$ is the minimum allowable distance to obstacle and σ, n, k are tunable parameters. In our experiments, we fix $d_{\min} = 0, \sigma = 0.12$ and adjust the values of n, k to see their effects. σ is set slightly larger than zero to avoid small amounts of constraint violation caused by numerical issues in optimization. The safety index constraint applies to the first two steps in virtual-time domain:

$$\begin{aligned}\phi(x_{0|t}) &\leq 0, \\ \phi(x_{1|t}) - \max\{\phi(x) - \eta, 0\} &\leq 0,\end{aligned}\tag{20}$$

where η is assigned with 0. Zhao et al. (Zhao et al., 2021) point out that for collision-avoidance tasks with action limits, the following design rule ensures forward invariance of the safe set:

$$\frac{n(\sigma + d_{\min}^n + kv_{\max})^{\frac{n-1}{n}}}{k} \leq -\frac{a_{\text{brk}}}{v_{\max}},\tag{21}$$

where $v_{\max}$ is the maximum velocity.

HJ reachability function must be defined with a policy. Here, we consider a safety-oriented policy that takes the maximum braking deceleration at every time step. Its corresponding HJ reachability function is the negative minimum future vehicle-to-obstacle distance starting from the current state:

$$F(x_{i|t}) = -d_{i|t} - \frac{v_{i|t}^2}{2a_{\text{brk}}}.\tag{22}$$

Its corresponding constraint also applies to the first two steps in virtual-time domain:

$$F(x_{i|t}) \leq 0, i = 0, 1.\tag{23}$$

Note that the CBF (17) is very similar in form to the HJ reachability function (22). In fact, they are equivalent when taking $k = -1/2 \cdot a_{\text{brk}}$ in (17). However, the equivalence of the two feasibility functions does not imply the equivalence of their virtual-time constraints. Specifically, the CBF constraint is more restrictive than HJ reachability because of its second constraint for set invariance. This will be clearly shown in the following experiment results.

We solve MPC and RL controllers under these four virtual-time constraints, resulting in eight combinations. For MPC, we use IPOPT to solve the constrained OCPs and compute the optimal actions online. For RL, we adopt the approximate dynamics programming (Li, 2023) framework and

use a recently proposed region-wise policy update method called feasible policy improvement (Yang et al., 2023b) to deal with safety constraints. This method solely minimizes constraint violations outside the current feasible region and maximizes rewards under constraints with the interior point method inside the feasible region. It is proved that (Yang et al., 2023b) this method ensures monotonic expansion and convergence of the feasible region.

To demonstrate the feasibility and control performance of different policies under different constraints, we visualize state trajectories and feasible regions of MPC and RL policies under the four virtual-time constraints. For MPC, we adjust the parameters of each virtual-time constraint to study their impact on the feasibility of the optimal policy. For RL, we fix the parameters and visualize trajectories and regions of intermediate policies at different iterations to study how they change. This demonstrates the capability of our theory to describe the feasibility of not only the optimal policy but also intermediate non-optimal policies during RL training.

Figure 7 shows state trajectories of MPC under pointwise constraints with different horizons. Each state trajectory is represented with a solid point followed by a series of hollow points, indicating the initial and successive states, respectively. The hollow point at the end of a trajectory is substituted with a cross if that state violates the real-time constraint indicated by the grey region. The red dashed line is the boundary of the maximum EFR computed analytically by considering the most cautious policy which always brakes with the maximum deceleration. When the constraint horizon n is short, although starting from a state inside the maximum EFR, some trajectories still end up crashing due to the short-sighted control policy under pointwise constraints. This means that the EFR of the optimal policy under pointwise constraints is only a subset of the maximum EFR. As the horizon becomes longer, more states become endlessly feasible under the optimal policy, which coincides with the fact that the maximum EFR actually corresponds to a pointwise constraint with $n = \infty$, as stated in Theorem 4.1.

Figure 8 shows feasible regions of MPC under pointwise constraints with different horizons. Since MPC can be viewed as an optimal policy, its EFR equals that of the optimal policy, and its IFR equals that of the constrained OCP (unrelated to any policy), i.e., $X_{\text{init}}^g(\pi^*) = X_{\text{init}}^g$. The feasible regions are obtained by running an episode from every state to test its feasibility. Results suggest that short-sighted pointwise constraints fail to recognize the inevitable collision at a distance and, hence, over-optimistically treat some states outside the maximum EFR as initially feasible. At the same time,

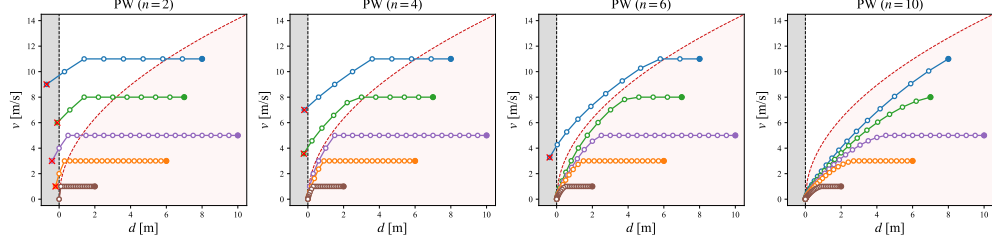


Figure 7: State trajectories of MPC under pointwise constraints with different horizons. “PW” stands for “pointwise”.

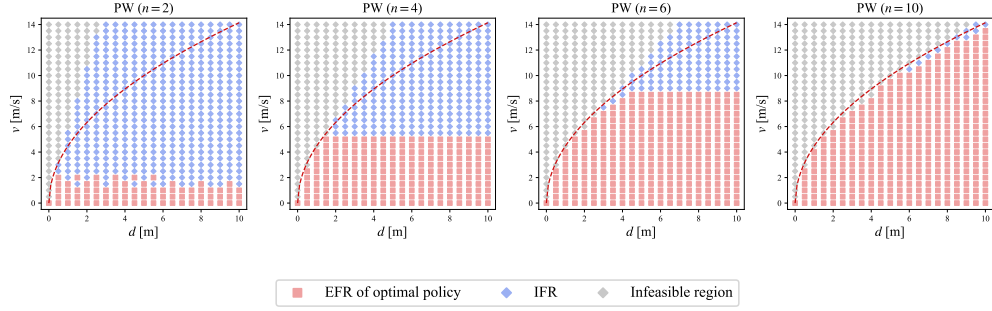


Figure 8: Feasible regions of MPC under pointwise constraints with different horizons.

pointwise constraints are weaker than real-time constraints, so the policy may take too aggressive actions early in the episode, making the EFR of the optimal policy smaller than the maximum EFR. The results also show that as the constraint horizon becomes longer, the EFR of the optimal policy enlarges while the IFR shrinks. When the horizon is long enough, e.g., $n = 10$, both regions are almost the same as the maximum EFR. This again supports that with infinite-horizon pointwise constraints, which essentially equal real-time constraints, IFR becomes EFR, and they both equal the maximum EFR.

Figure 9 and 10 show state trajectories and feasible regions of RL under the same pointwise constraint at different iterations. At iteration 10, the feasible regions of the policy are small, and states with high velocities break out of the maximum EFR, resulting in constraint violations. Note that the EFR is smaller than the IFR because, in some states, constraint violation does not happen in the finite virtual horizon (10 steps) but is inevitable in the long run. Here, this infeasibility phenomenon is mainly due to inadequate policy

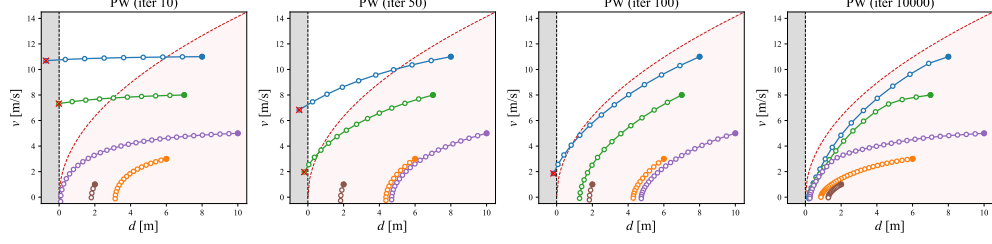


Figure 9: State trajectories of RL under the same pointwise constraint ($n = 10$) at different iterations.

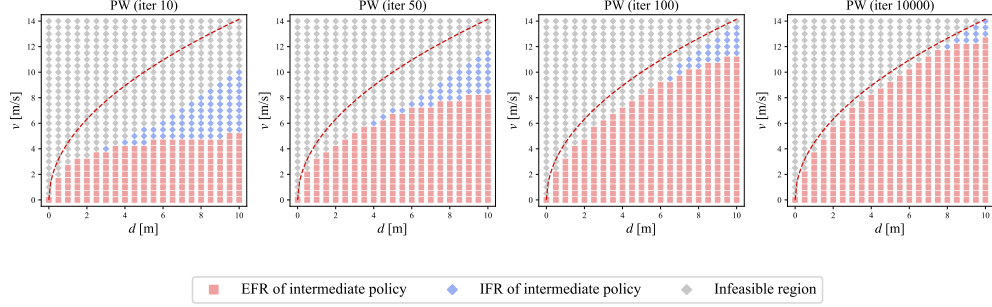


Figure 10: Feasible regions of RL under the same pointwise constraint ($n = 10$) at different iterations.

training. As training proceeds, the feasible regions monotonically expand, and more state trajectories are included in the maximum EFR. At iteration 10000, the feasible regions are close to those of the optimal policy of MPC. These results indicate the effectiveness of feasible policy improvement (Yang et al., 2023b), whose theoretical foundation is the feasibility analysis tools for intermediate non-optimal policies proposed in this paper.

Figure 11 and 12 show state trajectories and feasible regions of MPC under CBF constraints with different parameters. We observe that the EFR of the optimal policy is always identical to the IFR, which is a common feature of CIS-based virtual-time constraints and is consistent with the analysis in Section 6.2.1 and Corollary 5.1.1(3). A smaller value of k results in a larger feasible region, rendering more states safe in the long run. When $k = 0.05$, the CBF actually equals the HJ reachability function (22), and its zero-sublevel set equals the maximum EFR. Another phenomenon is that as the feasible regions expand, restrictions on trajectories inside the feasible regions

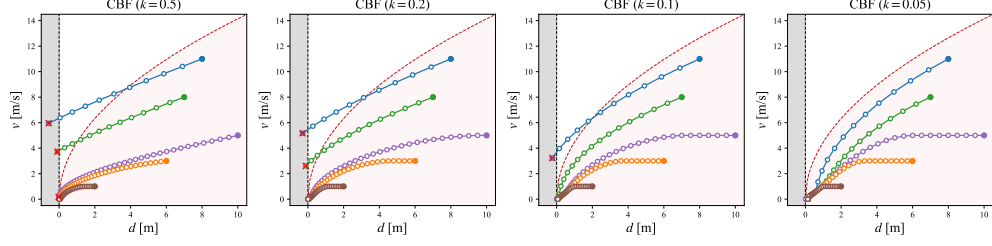


Figure 11: State trajectories of MPC under CBF constraints with different parameters.

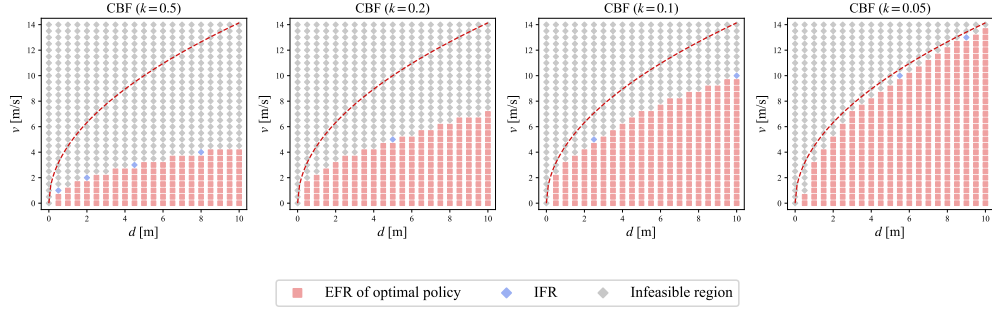


Figure 12: Feasible regions of MPC under CBF constraints with different parameters.

reduce. This can be seen by comparing the purple or orange trajectory when $k = 0.5$ and $k = 0.05$, in which the former decelerates immediately while the latter keeps its velocity for some time before decelerating. In other words, trajectories inside the feasible region become less conservative as k decreases.

Figure 13 and 14 show state trajectories and feasible regions of RL under the same CBF constraint at different iterations. The feasible regions monotonically expand during training and are close to the maximum EFR at iteration 10000. The state trajectories are either very conservative or infeasible at the beginning and gradually converge to the optimal behavior, i.e., drive the state to the lower left corner. It's worth noting that the EFR and IFR are nearly always the same during training (we believe their mismatch at iteration 10 is due to inadequate training), which is very different from the case of the safety index and HJ reachability introduced below. This phenomenon is due to the second constraint in (18), which not only requires the state to stay in the feasible region but also restricts the increasing rate of CBF.

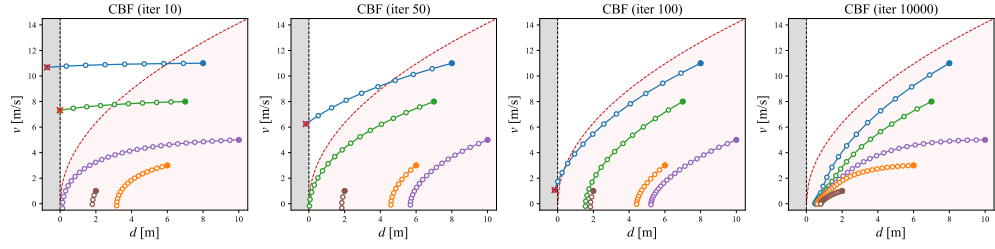


Figure 13: State trajectories of RL under the same CBF constraint ($k = 0.05$) at different iterations.

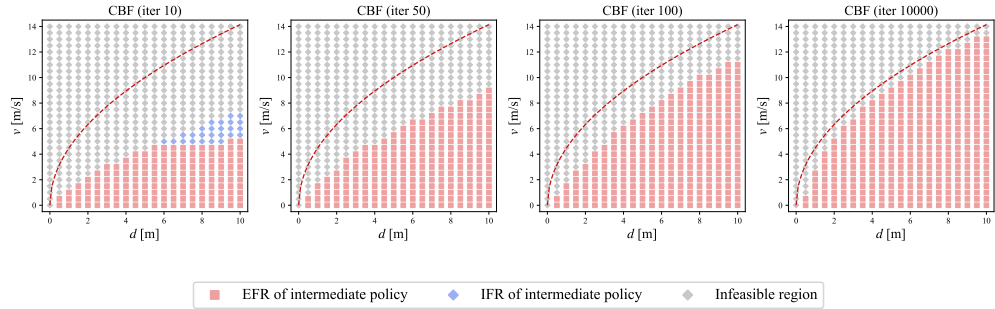


Figure 14: Feasible regions of RL under the same CBF constraint ($k = 0.05$) at different iterations.

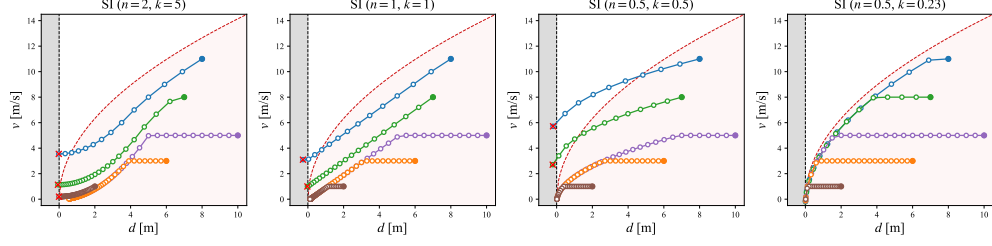


Figure 15: State trajectories of MPC under SI constraints with different parameters. “SI” stands for “safety index”.

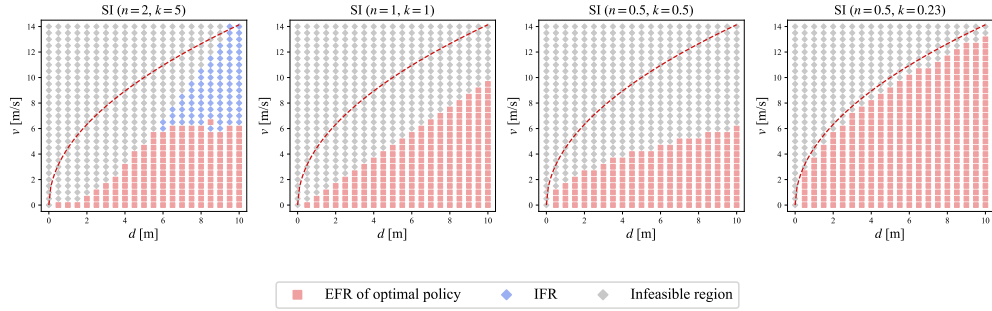


Figure 16: Feasible regions of MPC under SI constraints with different parameters.

Figure 15 and 16 show state trajectories and feasible regions of MPC under safety index constraints with different parameters. The feasible regions’ boundary shape accurately reflects the safety index’s function form. When $n = 2$, the boundary is a quadratic function with respect to d . It becomes a linear function when $n = 1$ and a square root function when $n = 0.5$. When $n = 0.5, k = 0.23$, the zero-sublevel set of the safety index equals the maximum EFR. Note that when $n = 2, k = 5$, the EFR is smaller than the IFR because with these parameters, the design rule (21) is violated, and thus, forward invariance of the safe set is not guaranteed. Comparing Figure 15 with Figure 11, we observe a difference between safety index and CBF: under the safety index constraint, the policy does not decelerate until it reaches the boundary of EFR while under the CBF constraint, the policy starts to decelerate before coming close the boundary. This is because of the difference in their constraint design: the safety index only requires the next state to stay in the EFR, while CBF further restricts its value-increasing rate.

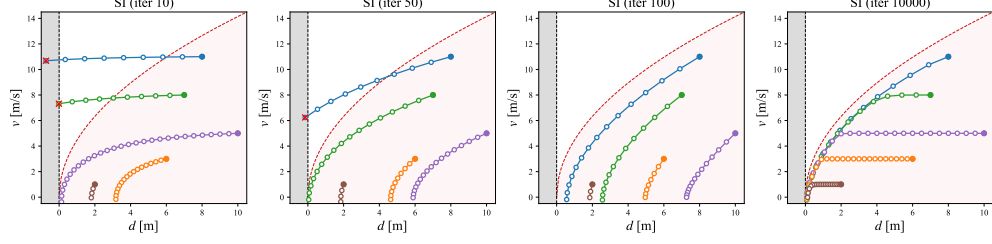


Figure 17: State trajectories of RL under the same SI constraint ($n = 0.5, k = 0.23$) at different iterations.

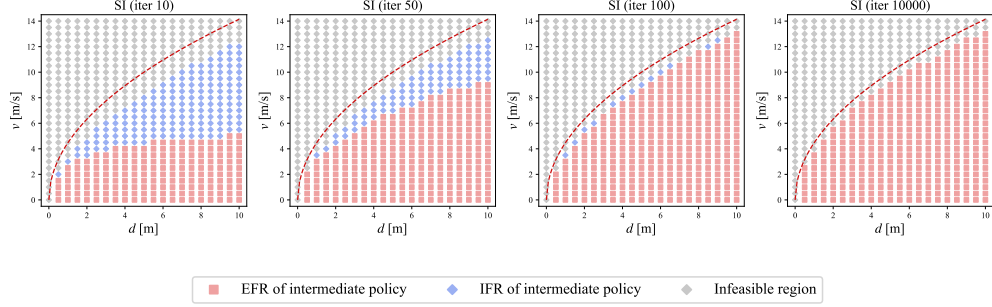


Figure 18: Feasible regions of RL under the same SI constraint ($n = 0.5, k = 0.23$) at different iterations.

Figure 17 and 18 show state trajectories and feasible regions of RL under the same safety index constraint at different iterations. The EFR monotonically expands and becomes close to the maximum EFR at iteration 10000, while the state trajectories also gradually converge to those of the optimal policy. We observe that at an early stage of training, the IFR under safety index constraint is already quite large, much larger than the EFR. This is different from the case of CBF constraint shown in Figure 14, where EFRs are close to IFRs. The reason is that the safety index constraint only requires the next state to be in the safe set without restricting its value-increasing rate as CBF does. At iteration 10 in Figure 17, the states with high velocities are initially feasible as long as they do not leave the safe set in one step. Accordingly, states close to the safe set boundary are initially infeasible because their next states will leave the safe set.

Figure 19 shows state trajectories and feasible regions of MPC under HJ reachability constraint. Being a CA-based virtual-time constraint, the HJ

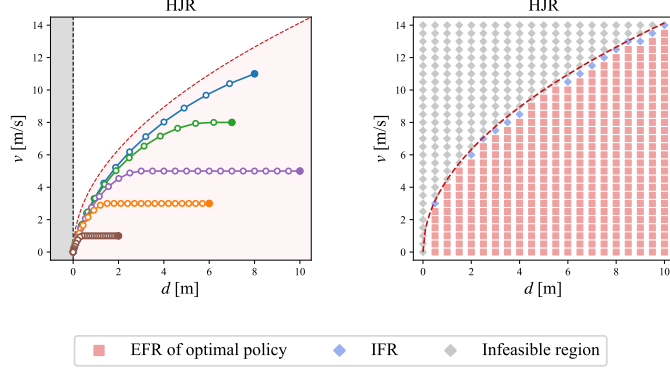


Figure 19: State trajectories and feasible regions of MPC under HJ reachability constraint. “HJR” stands for “HJ reachability”.

reachability constraint yields identical IFR and EFR of the optimal policy. In addition, since this constraint is built upon the reachability function of a safety-oriented policy, both its IFR and EFR equal the maximum EFR, which coincides with the analysis in Section 6.3.2. This indicates that CA-based virtual-time constraints result in the maximum EFR as long as the optimal feasibility function is found. Comparing the right figure in Figure 19 and the last figure in Figure 11, we observe that the policy under HJ reachability constraint is less conservative than that under CBF constraint, even though the feasible regions of the two constraints are identical. The reason also lies in the fact that CBF constraint restricts the value-increasing rate while HJ reachability constraint does not.

Figure 20 and 21 show state trajectories and feasible regions of RL under HJ reachability constraint at different iterations. As is the case in other constraints, the EFR monotonically expands to the maximum one, and the state trajectories converge to the optimal ones. Like safety index, the IFR under HJ reachability constraint is much larger than the EFR at an early stage of training, which is also because of the minimum restrictiveness on the next state.

9. Conclusion

This paper proposes a feasibility theory that applies to both MPC and RL. Compared with existing theories built for MPC, our theory fills in the

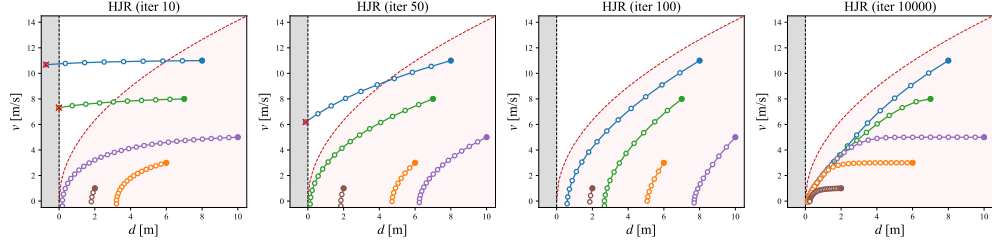


Figure 20: State trajectories of RL under HJR constraint at different iterations.

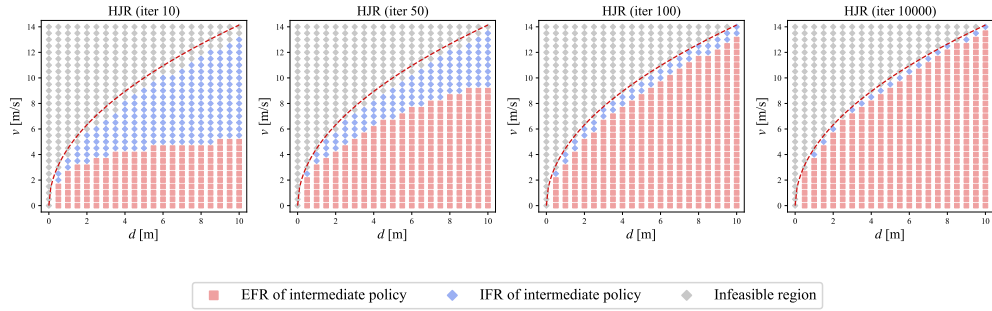


Figure 21: Feasible regions of RL under HJR constraint at different iterations.

missing part of feasibility analysis for an arbitrary policy. Based on a decoupling view of virtual-time domain and real-time domain, we separately define initial and endless, state and policy feasibility, and their corresponding feasible regions. We further analyze the containment relationships between different feasible regions, which enables us to analyze the feasibility of an arbitrary policy. After that, we provide virtual-time constraint design rules along with a practical design tool called feasibility function that helps achieve the maximum feasible region. The feasibility function is categorized into control invariant set and constraint aggregation, which lead to different kinds of virtual-time constraints. We review most of existing constraint formulations and point out that they are essentially applications of feasibility functions in different forms. Finally, we demonstrate our feasibility theory in an emergency braking control task by visualizing initially and endlessly feasible regions of MPC and RL policies under different kinds of virtual-time constraints.

References

- Achiam, J., Held, D., Tamar, A., Abbeel, P., 2017. Constrained policy optimization, in: International Conference on Machine Learning, PMLR. pp. 22–31.
- Agrawal, A., Sreenath, K., 2017. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation., in: Robotics: Science and Systems, Cambridge, MA, USA. pp. 1–10.
- Altman, E., 1999. Constrained Markov decision processes. volume 7. CRC press.
- Ames, A.D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., Tabuada, P., 2019. Control barrier functions: Theory and applications, in: 2019 18th European control conference (ECC), IEEE. pp. 3420–3431.
- Ames, A.D., Grizzle, J.W., Tabuada, P., 2014. Control barrier function based quadratic programs with application to adaptive cruise control, in: 53rd IEEE Conference on Decision and Control, IEEE. pp. 6271–6278.

- Ames, A.D., Xu, X., Grizzle, J.W., Tabuada, P., 2016. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control* 62, 3861–3876.
- As, Y., Usmanova, I., Curi, S., Krause, A., 2022. Constrained policy optimization via bayesian world models, in: *International Conference on Learning Representations*.
- Bai, Q., Bedi, A.S., Agarwal, M., Koppel, A., Aggarwal, V., 2022. Achieving zero constraint violation for constrained reinforcement learning via primal-dual approach, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3682–3689.
- Bansal, S., Chen, M., Herbert, S., Tomlin, C.J., 2017. Hamilton-jacobi reachability: A brief overview and recent advances, in: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE. pp. 2242–2253.
- Bansal, S., Tomlin, C.J., 2021. Deepreach: A deep learning approach to high-dimensional reachability, in: *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE. pp. 1817–1824.
- Boccia, A., Grüne, L., Worthmann, K., 2014. Stability and feasibility of state constrained mpc without stabilizing terminal constraints. *Systems & control letters* 72, 14–21.
- Borrelli, F., Bemporad, A., Morari, M., 2017. *Predictive control for linear and hybrid systems*. Cambridge University Press.
- Brunke, L., Greeff, M., Hall, A.W., Yuan, Z., Zhou, S., Panerati, J., Schoellig, A.P., 2022. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems* 5, 411–444.
- Cheng, R., Orosz, G., Murray, R.M., Burdick, J.W., 2019. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks, in: *Proceedings of the AAAI conference on artificial intelligence*, pp. 3387–3395.
- Chow, Y., Ghavamzadeh, M., Janson, L., Pavone, M., 2017. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research* 18, 6070–6120.

- Chow, Y., Nachum, O., Duenez-Guzman, E., Ghavamzadeh, M., 2018. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems* 31.
- Ding, D., Zhang, K., Basar, T., Jovanovic, M., 2020. Natural policy gradient primal-dual method for constrained markov decision processes. *Advances in Neural Information Processing Systems* 33, 8378–8390.
- Fisac, J.F., Lugovoy, N.F., Rubies-Royo, V., Ghosh, S., Tomlin, C.J., 2019. Bridging hamilton-jacobi safety analysis and reinforcement learning, in: *2019 International Conference on Robotics and Automation (ICRA)*, IEEE. pp. 8550–8556.
- Gondhalekar, R., Imura, J.i., Kashima, K., 2009. Controlled invariant feasibility—a general approach to enforcing strong feasibility in mpc applied to move-blocking. *Automatica* 45, 2869–2875.
- Gondhalekar, R., Jones, C.N., 2011. Mpc of constrained discrete-time linear periodic systems—a framework for asynchronous control: Strong feasibility, stability and optimality via periodic invariance. *Automatica* 47, 326–333.
- Guan, Y., Ren, Y., Sun, Q., Li, S.E., Ma, H., Duan, J., Dai, Y., Cheng, B., 2022. Integrated decision and control: Toward interpretable and computationally efficient driving intelligence. *IEEE transactions on cybernetics* 53, 859–873.
- Herbert, S., Choi, J.J., Sanjeev, S., Gibson, M., Sreenath, K., Tomlin, C.J., 2021. Scalable learning of safety guarantees for autonomous systems using hamilton-jacobi reachability, in: *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE. pp. 5914–5920.
- Li, S.E., 2023. *Reinforcement learning for sequential decision and optimal control*. Springer.
- Liu, C., Tomizuka, M., 2014. Control in a safe set: Addressing safety in human-robot interactions, in: *Dynamic Systems and Control Conference*, American Society of Mechanical Engineers. p. V003T42A003.

- Liu, Y., Ding, J., Liu, X., 2020. Ipo: Interior-point policy optimization under constraints, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4940–4947.
- Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, S., Li, B., Zhao, D., 2022. Constrained variational policy optimization for safe reinforcement learning, in: International Conference on Machine Learning, PMLR. pp. 13644–13668.
- Löfberg, J., 2012. Oops! i cannot do it again: Testing for recursive feasibility in mpc. *Automatica* 48, 550–555.
- Ma, H., Chen, J., Eben, S., Lin, Z., Guan, Y., Ren, Y., Zheng, S., 2021. Model-based constrained reinforcement learning using generalized control barrier function, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 4552–4559.
- Ma, H., Liu, C., Li, S.E., Zheng, S., Chen, J., 2022. Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning, in: Learning for Dynamics and Control Conference, PMLR. pp. 97–109.
- Mitchell, I.M., Bayen, A.M., Tomlin, C.J., 2005. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control* 50, 947–957.
- Nguyen, Q., Sreenath, K., 2016. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints, in: 2016 American Control Conference (ACC), IEEE. pp. 322–328.
- Ohnishi, M., Wang, L., Notomista, G., Egerstedt, M., 2019. Barrier-certified adaptive reinforcement learning with applications to brushbot navigation. *IEEE Transactions on robotics* 35, 1186–1205.
- Peng, B., Duan, J., Chen, J., Li, S.E., Xie, G., Zhang, C., Guan, Y., Mu, Y., Sun, E., 2022. Model-based chance-constrained reinforcement learning via separated proportional-integral lagrangian. *IEEE Transactions on Neural Networks and Learning Systems* .
- Qin, Z., Zhang, K., Chen, Y., Chen, J., Fan, C., 2020. Learning safe multi-agent control with decentralized neural barrier certificates, in: International Conference on Learning Representations.

- Ravaioli, U.J., Cunningham, J., McCarroll, J., Gangal, V., Dunlap, K., Hobbs, K.L., 2022. Safe reinforcement learning benchmark environments for aerospace control systems, in: 2022 IEEE Aerospace Conference (AERO), IEEE. pp. 1–20.
- Robey, A., Hu, H., Lindemann, L., Zhang, H., Dimarogonas, D.V., Tu, S., Matni, N., 2020. Learning control barrier functions from expert demonstrations, in: 2020 59th IEEE Conference on Decision and Control (CDC), IEEE. pp. 3717–3724.
- Rubies-Royo, V., Fridovich-Keil, D., Herbert, S., Tomlin, C.J., 2019. A classification-based approach for approximate reachability, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE. pp. 7697–7704.
- Seo, H., Lee, D., Son, C.Y., Tomlin, C.J., Kim, H.J., 2019. Robust trajectory planning for a multirotor against disturbance based on hamilton-jacobi reachability analysis, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 3150–3157.
- Stooke, A., Achiam, J., Abbeel, P., 2020. Responsive safety in reinforcement learning by pid lagrangian methods, in: International Conference on Machine Learning, PMLR. pp. 9133–9143.
- Taylor, A., Singletary, A., Yue, Y., Ames, A., 2020. Learning for safety-critical control with control barrier functions, in: Learning for Dynamics and Control, PMLR. pp. 708–717.
- Thananjeyan, B., Balakrishna, A., Nair, S., Luo, M., Srinivasan, K., Hwang, M., Gonzalez, J.E., Ibarz, J., Finn, C., Goldberg, K., 2021. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters* 6, 4915–4922.
- Wei, T., Kang, S., Zhao, W., Liu, C., 2022. Persistently feasible robust safe control by safety index synthesis and convex semi-infinite programming. *IEEE Control Systems Letters* 7, 1213–1218.
- Xiao, W., Belta, C., 2019. Control barrier functions for systems with high relative degree, in: 2019 IEEE 58th conference on decision and control (CDC), IEEE. pp. 474–479.

- Yang, T.Y., Rosca, J., Narasimhan, K., Ramadge, P.J., 2020. Projection-based constrained policy optimization, in: International Conference on Learning Representations.
- Yang, Y., Jiang, Y., Liu, Y., Chen, J., Li, S.E., 2023a. Model-free safe reinforcement learning through neural barrier certificate. *IEEE Robotics and Automation Letters* 8, 1295–1302.
- Yang, Y., Zheng, Z., Li, S.E., 2023b. Feasible policy iteration. *arXiv preprint arXiv:2304.08845* .
- Ying, D., Ding, Y., Laveai, J., 2022. A dual approach to constrained markov decision processes with entropy regularization, in: International Conference on Artificial Intelligence and Statistics, PMLR. pp. 1887–1909.
- Yu, D., Ma, H., Li, S., Chen, J., 2022a. Reachability constrained reinforcement learning, in: International Conference on Machine Learning, PMLR. pp. 25636–25655.
- Yu, H., Xu, W., Zhang, H., 2022b. Towards safe reinforcement learning with a safety editor policy. *Advances in Neural Information Processing Systems* 35, 2608–2621.
- Zhang, L., Zhuang, S., Braatz, R.D., 2016. Switched model predictive control of switched linear systems: Feasibility, stability and robustness. *Automatica* 67, 8–21.
- Zhang, Y., Vuong, Q., Ross, K., 2020. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems* 33, 15338–15349.
- Zhao, W., Chen, R., Sun, Y., Wei, T., Liu, C., 2023a. State-wise constrained policy optimization. *arXiv preprint arXiv:2306.12594* .
- Zhao, W., He, T., Liu, C., 2021. Model-free safe control for zero-violation reinforcement learning, in: 5th Annual Conference on Robot Learning.
- Zhao, W., He, T., Liu, C., 2023b. Probabilistic safeguard for reinforcement learning using safety index guided gaussian process models, in: Learning for Dynamics and Control Conference, PMLR. pp. 783–796.

Zhao, W., He, T., Wei, T., Liu, S., Liu, C., 2023c. Safety index synthesis via sum-of-squares programming, in: 2023 American Control Conference (ACC), IEEE. pp. 732–737.