2024 IEEE Real-Time Systems Symposium (RTSS) | 979-8-3315-4026-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/RTSS62706.2024.00018

Deadline-Safe Reach-Avoid Control Synthesis for Cyber-Physical Systems with Reinforcement Learning

Mengyu Liu [§], Pengyuan Lu [†], Xin Chen[‡], Oleg Sokolsky[†], Insup Lee[†], Fanxin Kong[§]
[§]*Department of Computer Science and Engineering, University of Notre Dame, Notre Dame IN*
[†]*Department of Computer and Information Science, University of Pennsylvania, Philadelphia PA*
[‡]*Department of Computer Science, University of New Mexico, Albuquerque NM*
mliu9@nd.edu, pelu@seas.upenn.edu, chenxin@unm.edu, sokolsky@cis.upenn.edu, lee@cis.upenn.edu, fkong@nd.edu

***Abstract*—Meeting deadlines is a fundamental requirement of cyber-physical systems (CPS) in real-time applications to consolidate their reliability and effectiveness in executing time-critical tasks. Recent research have focused on applying reinforcement learning to synthesize controllers for real-time systems, particularly in terms of achieving fast reach-avoid. However, achieving fast behavior does not necessarily equate to meeting deadlines. Sometimes reinforcement learning agents are trying to maximize the total reward by exploiting the reward function, and thus performing unwanted behavior, known as reward hacking. Therefore, depending on the deadlines, it is possible to have fast controllers that miss the deadlines and slow controllers that meet the deadlines. To address the misalignment between fast and meeting deadlines, we investigate the relationship between as soon as possible (ASAP) and deadline-safe. Additionally, we formulate the problem into a new Markov decision process R-MDP including time to avoid non-Markovian rewards when considering deadlines. Furthermore, we have designed new reward functions that encourage the agent to meet the deadlines. Moreover, we evaluate our method on various benchmarks. The experiment results show the effectiveness of our method in ensuring deadline compliance without compromising safety.**

***Index Terms*—cyber-physical systems, reinforcement learning, control synthesis**

I. Introduction

Cyber-physical systems (CPS) are integrations of computation, networking, and physical processes [1]. Among the numerous challenges in CPS design, real-time considerations stand out as particularly crucial. In time-critical scenarios, the system must respond to changes in the environment within strict time constraints to ensure safety, reliability, and efficiency [2]. For instance, in autonomous vehicular systems, any delay in processing and responding to sensor data can lead to catastrophic failures [3]–[6], This exemplifies the critical nature of real-time computing in ensuring the safety and reliability of CPS.

The reach-avoid problem is a fundamental challenge in control theory and CPS, particularly relevant to autonomous systems and robotics [7], [8]. It involves synthesizing a control strategy for a system so that it can reach a target set (the ”reach” condition) while simultaneously avoiding a set of undesirable states (the ”avoid” condition). In addition, it is important to reach the target set by the deadline for real-time applications to achieve ”deadline-safe” behavior.

To address the reach-avoid problem, especially in situations where the system model is unknown, researchers have turned to reinforcement learning (RL) [9]–[11]. An important problem to tackle in the design and training of RL agents is the formulation of reward functions. A well-designed reward function should lead to a policy that exactly encourages completing the intended task the agent was supposed to accomplish. However, many researchers have overlooked the misalignment between the “as soon as possible” (ASAP) and deadline-safe in RL when designing reward functions. They design reward functions which encourage finishing the task ASAP instead of deadline-safe for tasks with a fixed deadline. This will cause reward hacking [12] that the agent seeks to optimize the total reward by exploiting the reward function, which leads to unwanted and unsafe behaviors.

While numerous studies have focused on achieving reach-avoid tasks rapidly using RL, the specific challenge of meeting deadlines within these tasks has received significantly less attention. Researchers in [13] consider a finite horizon Markov Decision Process (MDP) is an infinite horizon MDP with time in the state space. Researchers in [14] argue time should be part of the environment and thus it should be included in the state space to avoid violating the Markov property [15]. They also provide a novel discount factor distribution to prioritize the long-term reward to make the agent time-aware. Unfortunately, none of them systematically consider deadlines from a problem level. It is important to design novel reinforcement learning methods for CPS that encourage deadline-safe.

In response to these gaps, this work targets the design of deadline-safe reinforcement learning method for CPS in reach-avoid tasks. Encouraging deadline-safe behavior is challenging due to several key issues: First, the ambiguous definition of deadline-safe makes it hard to design reward functions. Some researchers attempt to achieve this temporal property using RL but the reward functions they designed are encouraging ASAP. Second, there is a trade-off between the curse of dimensionality and the flexibility of including more information in the state space. Third, it is challenging to provide a guarantee of the deadline-safe behavior for time-critical applications in CPS due to the unknown system dynamics and the probabilistic nature of machine learning.

To address the above issues, we propose a novel RL method to achieve deadline-safe. Specifically, the contributions of this work are as follows:

1) We systematically elaborate the relationship between ASAP and deadline-safe on a problem level. We formally define what is deadline-safe and ASAP policies and investigate the corresponding propositions of them. This provides theoretical foundations of reward design for deadline-safe.
2) We propose a new formulation of MDP, R-MDP, which is feasible for applications with deadline-safe requirements. This new formulation partially mitigates the curse of dimensionality compared to existing methods.
3) We propose a new reward function design method to encourage deadline-safe behavior. Additionally, we prove there is a probabilistic guarantee for the deadline-safe behavior under some assumptions.
4) We evaluate the proposed method on various benchmarks. The results show that the proposed reward design method can efficiently train a deadline-safe policy and outperform baselines.

The rest of this paper is organized as follows: Section II discusses related papers. Section III presents preliminaries. Section IV elaborates on the relationship between ASAP and deadline-safe. Section V presents how to design deadline-safe rewards and the proof of the probabilistic guarantee of deadline-safe behavior. Section VI evaluates our algorithm. Section VII concludes the paper.

II. Related Works

Traditional model-based control relies on accurate mathematical models of the system, which may not be readily available for complex systems [4], [16], [17]. In contrast, data-driven methods are motivated by the availability of large datasets and the desire to extract control policies or models directly from them [18]. Data-driven control methods attempt to identify system models from the data and design controllers to meet specific objectives. Researchers have applied data-driven methods to solve the reach-avoid problems under various scenarios and assumptions [19], [20]. However, an explicit system model identified from the data is still required for data-driven control which may not be feasible for some systems.

To tackle the above problems, some researchers attempt to solve control problems using model-free RL to find optimal control policies [21]–[23]. However, training can be challenging due to sparse rewards, making convergence difficult. Researchers in [24] try to design appropriate episodic reward functions for RL. They investigate the conditions of modifying the reward function and preserving the optimal policy to the original MDP. The importance of designing dense reward functions is well-established in the RL literature [25]. Moreover, it provides theoretical foundations for potential-based reward-shaping (PBRS) and provides basic guidance to avoid some reward hacking cases.

To address the timing requirements of the task, some researchers have attempted integrating temporal logic with RL [26]–[29]. This integration of formal methods and RL has provided some insights into the reward design phase since temporal logic allows for formal specifications of desired behavior over time. Researchers in [26], [27] utilize temporal logic to construct tasks as logical formulas, creating rewards that encourage the agent to satisfy these formulas. To solve problems with continuous signals, researchers adapt linear temporal logic to signal temporal logic [30]. Researchers in [28], [29] provide a dense reward implementation monitoring the satisfaction of signal temporal logic formulas on continuous problems. However, the MDP property is violated in [28] and can result in unstable performance.

Some researchers have noted that time should be considered as part of the environment state space for tasks with timing constraints [13], [14]. Researchers in [13] study how to design special efficient training algorithms for finite-horizon MDP instead of directly using training algorithms for infinite-horizon MDP. However, it does not consider rewards are time-dependent in more general cases. Researchers in [14] considering time in more general cases of time-dependent MDP, they design a novel time-dependent reward-distributing method by assigning higher coefficients for rewards of later time steps in the value function. However, none of them consider special reward functions to encourage deadline-safe.

For critical applications using machine learning, having guarantees is desirable but challenging, especially when the system model is unknown. Researchers have attempted to provide probabilistic guarantees using Gaussian process. A Gaussian process (GP) is a probabilistic model used in machine learning and statistics. It provides a way to generalize the Gaussian distribution to functions, making it applicable to a variety of problems. GPs offer significant advantages for RL due to their inherent ability to model uncertainty and they are highly sample-efficient which often requires fewer training samples to build accurate models [31]. Researchers have attempted integrating GP with delayed Q-learning [32], [33] to provide probabilistic guarantees for reinforcement learning. Researchers in [34] have attempted to provide RL model-based algorithms with a probabilistic guarantee. Researchers in [35] propose Delayed Gaussian process Q-learning (DGPQ), a novel sample efficient model-free RL algorithm using GP with probabilistic guarantee.

III. Preliminaries

In this section, we introduce basic terms and concepts for later sections and our problem statement.

A. Reinforcement Learning(RL)

A MDP is a stochastic control process that operates in discrete time [36]. It offers a mathematical framework for decision-making scenarios where outcomes are influenced both by randomness and the actions of a decision maker. Generally, a MDP can be formalized as:

$$\text{MDP} = (S, A, \mathcal{P}, R, \gamma)$$

where S is the state space, A is the action space, $\mathcal{P}$ is the transition probability, R is the reward function and γ is the discount factor.

For RL, the goal of learning algorithms is to find an optimal policy π with maximized cumulative rewards when converged by performing actions on a MDP. Specifically, it can be formulated as maximizing the return G_t for an infinite-horizon MDP:

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k} \tag{1}$$

The discount factor for infinite-horizon MDP is usually set to $0 < \gamma < 1$. When considering deadlines, for a finite-horizon MDP with a fixed episode length T, the return G_t is:

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=1}^{T} \gamma^{k-1} R_{t+k} \tag{2}$$

At this time, the discount factor can be set to 1 since G_t is upper-bounded.

The Actor-Critic algorithm is an advanced reinforcement learning method that optimizes a policy in an off-policy way. This approach aims to balance exploration and exploitation by making a trade-off between long-term reward and short-term reward. Specifically, the Bellman Equation of Actor-Critic algorithm can be formalized as:

$$V(s) = \mathbb{E}_{a \sim \pi}\left[R(s, a) + \gamma V\left(s'\right)\right] \tag{3}$$

Equation (3) describes how the critic evaluates the action taken by the actor by estimating the state-value function. $V(s)$ is the state-value function, $R(s, a)$ is the immediate reward if taking action a at state s. Soft Actor-Critic (SAC) is a variation of the Actor-Critic algorithm which introduces entropy into the value function that encourages exploitation, SAC is designed for continuous action spaces and is particularly noted for its sample efficiency and stability relative to other reinforcement learning algorithms.

B. Probably approximately correct learning

Probably Approximately Correct (PAC) learning is a theoretical framework in machine learning that helps understand the limits of what computers can learn from data [37]–[39]. “Probably” means that the learning model should make accurate predictions with high probability, and “approximately” indicates that the predictions should be close to the actual results within a small margin of error. This framework is crucial for assessing whether learning problems can be solved efficiently and how much data is needed to build accurate models.

Formally, the PAC learning framework can provide a guarantee on the learned concept:

$$P(e(h) < \epsilon) < \delta \tag{4}$$

We can interpret the PAC bound in Equation (4) as ”for a specific learning algorithm, the probability that a concept h it learns will have an error e bound by ϵ is δ”. The concept h usually is a hypothesis on learning.

C. Signal Temporal Logic

Signal Temporal Logic (STL) is a formal language that is motivated by monitoring real-time properties over signals (e.g., state trajectories). It enables precise specification of how signals should behave over time, using logical operators and temporal operators like ”eventually,” ”always,” and ”until.” STL can define constraints and patterns that the signals should follow, allowing engineers to express complex timing requirements for system behaviors. This is particularly useful in real-time and embedded systems to ensure the system adheres to specified safety and reliability requirements. A STL formula can be defined using the following syntax:

$$\varphi ::= \top \mid f(\bar{s}) < 0 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 U_{[t_1,t_2]} \varphi_2$$

where $\top$ is tautology, and f is a function that maps a trajectory $\bar{s}$ to a real number. U is the until operator, e.g., $\varphi_1 U_{[t_1,t_2]} \varphi_2$ indicates that φ_2 must hold at some time t between t_1 and t_2. Additionally, φ_1 must always hold before time t. The time interval $[t_1, t_2]$ is interpreted as $\{t_1, t_1 + 1, \ldots, t_2\}$. We can induce other temporal operators from the U operator such as F (finally) and G (globally). Specifically, $F_{[0,t]}\varphi$ means φ must hold at *one* time step between 0 and t, and $G_{[0,t]}\varphi$ means φ must hold at *all* time steps between 0 and t.

Researchers have attempted to construct reward functions for RL based on the formal specification. They map the degree of the robustness of the specification to a real value, this mapping also known as quantitative semantics. In general, the quantitative semantics can be formalized as:

$$\begin{aligned}
&\rho(\bar{s}, t, (f(\bar{s}) < d)) = d - f(s_t)\\
&\rho(\bar{s}, t, \neg\varphi) = -\rho(\bar{s}, \varphi, t)\\
&\rho(\bar{s}, t, \varphi_1 \wedge \varphi_2) = \min(\rho(\bar{s}, t, \varphi_1), \rho(\bar{s}, t, \varphi_2))\\
&\rho(\bar{s}, t, \varphi_1 \vee \varphi_2) = \max(\rho(\bar{s}, t, \varphi_1), \rho(\bar{s}, t, \varphi_2))\\
&\rho(\bar{s}, t, F_{[t_1,t_2]}\varphi) = \max_{t' \in [t'+t_1, t'+t_2]} \rho(\bar{s}, t', \varphi)\\
&\rho(\bar{s}, t, G_{[t_1,t_2]}\varphi) = \min_{t' \in [t'+t_1, t'+t_2]} \rho(\bar{s}, t', \varphi)\\
&\rho(\bar{s}, t, \varphi_1 U_{[t_1,t_2]} \varphi_2)\\
&= \max_{t' \in [t+t_1, t+t_2]} \Big(\min(\rho(\bar{s}, t', \varphi_2), \min_{t'' \in [t,t')} \rho(\bar{s}, t'', \varphi_1))\Big)
\end{aligned} \tag{5}$$

However, since the input of the signal temporal logic is a trajectory of states $\bar{s}$ instead of a state at one time step. Therefore, some researchers have introduced new formats of MDP to make STL tractable. A τ-MDP is an adaption of the original MDP where the state space is a Cartesian product of the original state space and the set of trajectories $\Sigma^{\bar{s}}$ over the horizon [40].

$$\tau\text{-MDP} = (S \times \Sigma^{\bar{s}}, A, \mathcal{P}, R, \gamma)$$

This formulation suffers from the exponential state space growth with horizon T, which is also known as the *curse* *of history*. Instead, researchers try to compress the history information to a series binary states to form the so called F-MDP where the state space is a Cartisian product of the original state space and the sets of these binary states.

$$F\text{-MDP} = (S \times \prod_{i=1}^{n} \mathfrak{F}_i, A, \mathcal{P}, R, \gamma)$$

These binary states $\mathfrak{F}_i$ are flags to test for the satisfaction of each STL sub-formula.

D. Problem Statement

In this work, we try to find a deadline-safe control strategy for a finite-horizon MDP of reach-avoid using RL. Specifically, we are trying to find a policy that has the highest probability to achieve deadline-safe (deadline-safe probability). Formally, the problem is trying to find an optimal policy π_{op}:

$$\begin{gathered}\pi_{op} = \arg\max_{\pi} P^{\pi}(s_0, t) \\ P^{\pi}(s,t) = \sum_{a \in A} \pi(a \mid s) \sum_{s' \in S} \mathcal{P}(s' \mid s, a) \cdot P^{\pi}(s', t-1)\end{gathered} \tag{6}$$

$P^{\pi}(s,t)$ represents the probability that the agent reaches the target from state s within the deadline t controlled by policy π. $\mathcal{P}(s' \mid s, a)$ is the probability of transitioning from state s to s' if taking action a. s_0 is an arbitrary initial state in the initial set, S is the state space, A is the action space.

IV. Deadline-Safe vs As soon as possible

In this section, we formally define what is deadline-safe and as soon as possible. We introduce ASAP to represent "fast" reach-avoid behaviors since "fast" is imprecise. We elaborate on the difference between deadline-safe and ASAP on the problem level. Moreover, we provide insights about the relationship between these two behaviors to provide theoretical foundations for further reward design.

A. Definitions of deadline-Safe and ASAP

Before discussing the relationship between deadline-safe and as soon as possible, it is important to formally define them. This formalization will provide a clear framework for comparing these concepts and understanding their interactions, particularly in systems where timing and safety are crucial.

Definition IV.1. A deadline-safe policy π_{ds} for a reach task can drive the system from any initial point in the initial set to the target within the given deadline T. Formally, it can be written as follows:

$$\begin{gathered}\forall s_0 \in S_{init}, S_{traj} \cap S_{target} \neq \emptyset \\ S_{traj} = \{s_1, s_2, ..., s_T\}, s_{i+1} = f(s_i, a_i), a_i = \pi_{ds}(s_i)\end{gathered} \tag{7}$$

f is the dynamics of the system, a_i is the action at i^{th} time step. A deadline T is the maximum acceptable time by which a task must be completed after its initiation. S_{init} is the initial set and S_{target} is the target set. S_{traj} stands for state trajectory set which contains the states on each time step before the deadline.

Definition IV.1 provides a formal concept defines the deadline-safe property of a policy for reach tasks. Similarly, for deadline-safe policy of reach&avoid tasks, we can adapt Definition IV.1 to :

$$\begin{gathered}\forall s_0 \in S_{init}, S_{traj} \cap S_{target} \neq \emptyset \wedge S_{traj} \cap S_{unsafe} = \emptyset \\ S_{traj} = \{s_1, s_2, ..., s_T\}, s_{i+1} = f(s_i, a_i), a_i = \pi_{ds}(s_i)\end{gathered} \tag{8}$$

where S_{unsafe} stands for unsafe that has to be avoided all the time.

Definition IV.2. ASAP policy π_{sp} for reach tasks can drive the system from any initial point in the initial set to the target in the shortest time. Formally, it can be written as follows:

$$\begin{gathered}\forall s_0 \in S_{init}, s_t \cap S_{target} \neq \emptyset \\ \nexists \pi_{op} : s'_q \cap S_{target} \neq \emptyset \wedge q < t \\ s_{i+1} = f(s_i, a_i), a_i = \pi_{sp}(s_i) \\ s'_{i+1} = f(s'_i, a'_i), a'_i = \pi_{op}(s'_i)\end{gathered} \tag{9}$$

where s'_q and s_t are the earliest time step on a trajectory that reaches the target. s_i and s'_i represent the states of the system at the i^{th} time step controlled by π_{sp} and π_{op}, respectively. Definition IV.2 explains the temporal property of ASAP policy for reach tasks. Specifically, there is no other policy that can drive the system to the target in a shorter time than a ASAP policy. Similarly, it can be extended to reach-avoid tasks:

$$\begin{gathered}\forall s_0 \in S_{init}, s_t \cap S_{target} \neq \emptyset \wedge S_{traj} \cap S_{unsafe} = \emptyset \\ S_{traj} = \{s_1, s_2, ..., s_t\}, s_{i+1} = f(s_i, a_i), a_i = \pi_{sp}(s_i) \\ S'_{traj} = \{s'_1, s'_2, ..., s'_t\}, s'_{i+1} = f(s'_i, a'_i), a'_i = \pi_{op}(s'_i) \\ \nexists \pi_{op} : s'_q \cap S_{target} \neq \emptyset \wedge q < t \wedge S'_{traj} \cap S_{unsafe} = \emptyset \\ s_{i+1} = f(s_i, a_i), a_i = \pi_{sp}(s_i) \\ s'_{i+1} = f(s'_i, a'_i), a'_i = \pi_{op}(s'_i)\end{gathered} \tag{10}$$

In this work, we consider different reward functions to solve the same problem. However, since the reward function is part of the definition of the MDP, to formally define what is a problem, we have to introduce a new concept: Markovian decision process without rewards.

Definition IV.3. Markovian decision process without rewards (MDP$\backslash R$) is an adaption of Markovian decision process which ignores the reward aspects. It can be formulated as

$$\text{MDP} \setminus \mathcal{R} = (S, A, \mathcal{P}, \_, \gamma)$$

MDP$\backslash R$ focuses solely on the state and action dynamics. It excludes the aspects of reward that usually guide the optimization of policies towards specific objectives.

Fig. 1 shows the topology of policies we defined for the same MDP$\backslash R$. We can see the big red eclipse is the set of all the policies for the MDP$\backslash R$, the green eclipse is the set of all deadline-safe policies, the blue eclipse is the set of all ASAP policies. We have two major observations from this figure: First, the ASAP policies set is a subset of deadline-safe policies set. Second, the ASAP policies set are not always

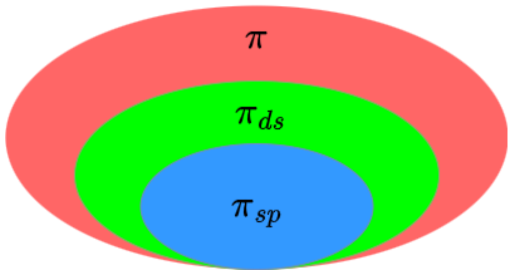


Fig. 1: Topology of policies for the same MDP $\setminus R$, the red set π is the set of all the policies for a MDP $\setminus R$, the green set π_{ds} is the set of all deadline-safe policies, the blue set π_{sp} is the set of all ASAP policies.

identical to the deadline-safe policies set. We will formally prove these two statements in the following subsection.

B. Relationships between deadline-safe and ASAP

In this subsection, we will discuss the relationship between the deadline-safe policies and ASAP policies.

Proposition IV.1. The ASAP policies set is not always identical to the deadline-safe policies set. This can formalized as:

$$S_{\pi_{ds}} \not\equiv S_{\pi_{sp}} \ if \ S_{\pi_{ds}} \neq \emptyset \tag{11}$$

Proposition IV.1 claims the relationship between deadline-safe and ASAP is not an identity. This claim seems a little bit trivial but it has been overlooked when designing applications with deadline-safe requirements. We can prove this proposition by contradiction.

Proof. Assume if $S_{\pi_{ds}} \equiv S_{\pi_{sp}}$ and $S_{\pi_{ds}} \neq \emptyset$, the set of deadline policies is identical to the set of ASAP policies for a MDP$\setminus R$. This implies any deadline-safe policy is also an ASAP policy. However, we can easily find a counterexample. Consider a simple discrete MDP$\setminus R$ with only 2 states, A and B as shown in Fig. 2:

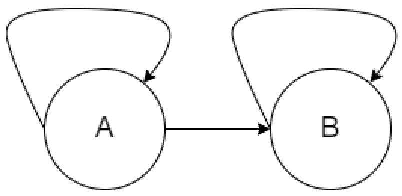


Fig. 2: A counterexample of $S_{\pi_{ds}} \equiv S_{\pi_{sp}}$, there are only 2 states A and B in this MDP$\setminus R$, the goal is to reach B in 2 steps from A.

The initial state of this MDP$\setminus R$ is A, and the goal is to reach B in 2 steps, in other words, the deadline is 2 steps, the time horizon is finite. There are 2 actions available for the agent at A, the first is staying, and the second is moving to next as the arrows in Fig. 2. There is only one action for for agent at B which is staying. We can easily find two policies:

1) Policy π_1: The agent chooses to move to the next immediately, it reaches B in 1 step.
2) Policy π_2: The agent chooses to stay at A then moving to the next, it reaches B in 2 steps.

We can see both policy π_1 and policy π_2 are deadline-safe, but policy π_1 is ASAP but policy π_2 is not. This counterexample conflicts with the assumption that the set of deadline policies is a subset of ASAP policies for a MDP$\setminus R$. Therefore, there is a contradiction, and proposition IV.1 holds. □

Proposition IV.1 has provided some insights about the potential reward hacking vulnerability if we design a reward function encouraging ASAP for deadline-safe tasks. However, in fact, a reward function encouraging ASAP works relatively well on some deadline-safe tasks. To further elaborate on the reason, we need more insights of the relationship between these two behaviors.

Proposition IV.2. The set of ASAP policies is a subset of deadline-safe policies for a MDP$\setminus R$. Formally, we have:

$$S_{\pi_{sp}} \subseteq S_{\pi_{ds}} \ if \ S_{\pi_{ds}} \neq \emptyset \tag{12}$$

Proof. Assume the set of ASAP policies is not a subset of deadline-safe policies for a MDP$\setminus R$. This implies that there exists a policy that is ASAP but not deadline-safe. Given the definition IV.1, the ASAP policy cannot drive the system to the target by the given deadline. According to the definition IV.2, if the ASAP policy cannot drive the system to the target by the deadline, there does not exist a policy that can drive the system to the target earlier than the deadline. Therefore, the set of deadline-safe policies is empty, then there is an contradiction. □

Note that the condition of proposition IV.2 is important. If $S_{\pi_{ds}} = \emptyset$, proposition IV.2 does not hold since there might be ASAP policies that can not meet the deadlines. In other words, in that case, there are no deadline-safe policies even they are as soon as possible.

There is an inconsistency between the definition of ASAP policies when considering deadlines. We have to adapt definition IV.2 to the following:

$$\begin{gathered}
\forall s_0 \in S_{init}, s_t \cap S_{target} \neq \emptyset \wedge S_{traj} \cap S_{unsafe} = \emptyset \\
S_{traj} = \{s_1, s_2, ..., s_T\}, s_{i+1} = f(s_i, a_i), a_i = \pi_{sp}(s_i) \\
S'_{traj} = \{s'_1, s'_2, ..., s'_T\}, s'_{i+1} = f(s'_i, a'_i), a'_i = \pi_{op}(s'_i) \\
\nexists \pi_{op} : s'_q \cap S_{target} \neq \emptyset \wedge q < t \wedge S'_{traj} \cap S_{unsafe} = \emptyset \\
s_{i+1} = f(s_i, a_i), a_i = \pi_{sp}(s_i) \\
s'_{i+1} = f(s'_i, a'_i), a'_i = \pi_{op}(s'_i)
\end{gathered} \tag{13}$$

where the agent should avoid the unsafe set until the deadline ends instead of just avoid it before reach.

The relationship between ASAP and deadline-safe has been elaborated by propositions IV.1 and IV.2. These two propositions indicate that there is a difference between deadline-safe and ASAP from the problem level so that we can not treat them identically. This message is important for researchers to formally design rewards to achieve deadline-safe to avoid potential reward hacking. Additionally, deadline-safe requires a predefined deadline for training but it is not required for ASAP.

V. Deadline-Safe Training

In this section, we will introduce how to design reward functions to encourage deadline-safe behaviors instead of ASAP. Additionally, we describe the details of the new MDP formulation, R-MDP, to avoid violating Markovian property when using the reward function we introduced. Moreover, we want to provide a probabilistic guarantee for our method under certain conditions to make it suitable for time-critical applications in CPS. We provide a formal PAC bound on the deadline-safe probability and prove its correctness.

A. *deadline-safe reward design*

When designing a reward function, the most important consideration is ensuring that the function aligns with the desired goals and behaviors of the system, this process is known as *reward alignment*. Let's start with that simple discrete example again which is shown in Fig.2. But the insights we provide is also feasible for continuous space problems. How do we design a deadline-safe reward function for this simple problem?

The most intuitive idea is to design an episodic reward function to encourage deadline-safe:

$$R_{epi}(t) = \begin{cases} 10 & \text{if } t = T \text{ and the state is B} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

This reward design encourages deadline-safe by only setting rewards for the last time step of the episode and give zero reward for other time steps. Deadline-safe is encouraged since the optimal policy of this MDP is reaching B by the deadline T. Note that this reward design does not encourage ASAP since an earlier reach will not grant additional rewards for the agent compared to an agent reaching it at the last time step in the episode.

This reward function is good enough for this simple example since there are only 2 states, and it is easy to have a sample trajectory that reaches B during training even if there is no immediate reward at every time step except the last time step. However, it is going to be harder to find a sample trajectory that reaches B if there are more states between A and B as Fig. 3 shown.

It is possible to apply PBRS to resolve this sparse reward issue by adapting the reward function in Equation (14):

$$R_{pbrs}(t) = \begin{cases} 10 & \text{if } t = T \text{ and the state is B} \\ d(s_{t-1}) - d(s_t) & \text{otherwise} \end{cases} \tag{15}$$

We define $d(s)$ as the length of the shortest path from s to B. This reward shaping provides dense rewards at each time step which encourages the agent to reach B. Unfortunately, this shaping will make the reward function encouraging ASAP instead of deadline-safe. For the example in Fig. 3, assuming

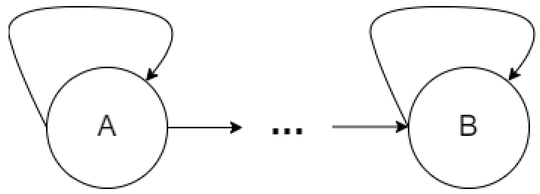


Fig. 3: If there are multiple states between A and B, the agent has 2 actions at each state: moving to next or staying, then it is harder and harder to get motivated to reach B since there is no immediate rewards. In other words, the agent will get immediate zero rewards no matter choose to stay or move to next.

the deadline is 5 and there are 2 states between A and B, the global optimal policy is moving to the next state at every state. This policy will grant a positive immediate reward (which is 1 since the length of the shortest path is reduced by 1) and maximize the global return as introduced in Equation (1). However, this reward is not encouraging deadline-safe since the different deadline-safe policies will have different returns. In other words, under R_{pbrs}, we think the ASAP policy (which is also deadline-safe) is better than other deadline-safe policies since it grants higher cumulative rewards. This is a phenomenon of reward hacking and we want to avoid this problem. Now the question for us is "How to design a reward function that grants equal cumulative rewards for every deadline-safe policy?"

Researcher have attempted to apply STL and quantitative semantics to resolve this problem. If the pure quantitative semantics in Equation (5) are applied for training, then the reward function is episodic and the training suffers from the sparse reward again. Researchers in [28], [41] have made the rewards dense by evaluating the degree of robustness of the STL formula at every time step. However, these shaping methods either suffer from the curse of dimensionality or violate the Markovian property. Additionally, the rewards in these works are still encouraging ASAP instead of deadline-safe. To the best of our knowledge, there is no existing work that has answered this question properly.

It is challenging to answer this question since there is a key conflict between the dense rewards for encouraging reach and the attached encouragement for ASAP instead of deadline-safe. To resolve this conflict, we propose a novel reward function which assigns rewards for the steps after reaching to "balance" the reward granted before reaching:

$$R_{ds}(t) = \begin{cases} 10 & \text{if } (s_t \in B) \wedge (C(B) = 1) \wedge (t \neq T) \\ 10 - \sum r_{br} & \text{if } (s_t \in B) \wedge (C(B) = 1) \wedge (t = T) \\ d(s_{t-1}) - d(s_t) & \text{if } C(B) = 0 \\ \frac{-\sum r_{br}}{T-t+1} & \text{if } C(B) > 1 \vee ((s_t \notin B) \wedge (C(B) = 1)) \end{cases} \tag{16}$$

$C(s)$ represents the number of times this state s has been visited including this time step. There are three differences between the new deadline-safe reward function in Equation (16) and the PBRS reward function in Equation (15):

1) If the goal is achieved (in the example, the goal is to reach B), the reward will be granted at that time step instead of the last time step. However, this reward is only granting once which is the first time.
2) We separate the time steps to two parts: before reaching B and after reaching B. Before reaching B, the reward is the same as the reward designed in Equation (15). We store these rewards r_{br} before reaching B into a buffer and calculate the sum of them. For every time step after reaching B, the reward is the additive inverse of that sum divided by the remaining time steps.
3) There is a special case that the first time the agent reach the goal is at the last time step. In this case, there is no time step remaining and the additive inverse of the reward sum is added to include in the immediate reward at this time step, too.

The main idea in the design of the reward function in Equation (16) is using the time steps after reaching to compensate the rewards granted before reaching. The goal of this design is to encourage deadline-safe instead of ASAP. To be noticed, we have not included avoid in the reward yet. It is important to consider avoiding behavior as well as considering deadline-safe. We can trivially consider collision behavior is as bad as not achieving deadline-safe. Therefore, if there is a collision at one time step, it will get the same reward for not reaching at the last time step. We can formally define when the reward function is encouraging deadline-safe:

Definition V.1. A reward function R_{ds} is encouraging deadline-safe if every deadline-safe policy grants equal cumulative rewards for starting from any initial point in the initial set. Formally, we have:

$$\begin{gathered} V^{\pi_1}(s_0) = V^{\pi_2}(s_0) \ \forall \pi_1, \pi_2 \in \pi_{ds}, \gamma = 1, s_0 \in I \\ V^{\pi}(s) = \mathop{\mathbb{E}}_{a \sim \pi}\left[\sum_{k=0}^{T-1} \gamma^k R_{ds}(k+1) \mid s_0 = s\right] \end{gathered} \tag{17}$$

Lemma V.1. Reward function in Equation (16) is encouraging deadline-safe.

Proof. Since the horizon is finite, it is possible to set the discount factor γ to 1. Based on this simplification, the value of the initial point is the expectation of the sum of the reward of the episode.

We assume there is a single case that there are two deadline-safe policies π_1 and π_2 and they grant different cumulative rewards such that $V^{\pi_1}(s_0) \neq V^{\pi_2}(s_0)$. Since both of them are deadline-safe, then it must reach the goal before the deadline. However, according to Equation (16), the sum of rewards of any episode if reach is 10. Therefore, there is a contradiction. □

In this subsection, we have introduced how to design reward functions to encourage deadline-safe. Although we only prove Lemma V.1, this is not confined to the special instance in Equation (16). In fact, it is more like a template since we can replace $d(s_{t-1}) - d(s_t)$ with any reward shaping that encourages reach behavior (such as quantitative semantics) and the reward function is still encouraging deadline-safe.

However, the reward depends on both the remaining time steps and the previous rewards which violates the Markovian property. Therefore, we need a new formulation of the MDP problem to restore Markovian property.

B. R-MDP formulation

In this subsection, we introduce R-MDP for deadline-safe rewards and compare them to existing formulations.

In R-MDP, the state space is a Cartesian product of the original state space S, the set of remaining time steps T_r, the set of reward history Σ^r, and a set of a binary flag state K indicates whether the goal has been achieved:

$$R\text{-MDP} = (S \times T_r \times \Sigma^r \times K, A, \mathcal{P}, R, \gamma) \tag{18}$$

We can assign the state space of R-MDP as Σ^R where we have $\Sigma^R \in (S \times T_r \times \Sigma^r \times K)$. In this formulation, the deadline-safe reward function does not violate the Markovian property. The format of the reward history is an array with size T which is the deadline, the initial values are set to 0.

R-MDP does not suffer from the curse of history as much as τ-MDP since the reward is a single value instead of vectors. This is a very important property when the dimension of the system model is high. Also, the binary states in F-MDP are not able to include the reward history and remaining time steps in the MDP since these are real values.

For time-critical applications in CPS, it is important to have guarantees on the performance of the proposed method. In the next subsection, we introduce how to obtain a probabilistic guarantee on deadline-safe behavior.

C. deadline-safe PAC guarantee

We define the probability of deadline-safe as the probability of reaching the target by the deadline from arbitrary initial point s_0 in the initial set under policy π is $P^{\pi}(s_0)$ in the problem statement . We want to find the an optimal policy which has the highest deadline-safe probability within the given deadline T from arbitrary initial point in the initial set. However, there is an inconsistency since the RL algorithms are maximizing the value formulated in Bellman Equation instead of probabilities. Therefore, it is important to show the monotonicity between deadline-safe probability and value. Before we show how to obtain the probabilistic guarantee, we have to make certain adaptation to the reward function in Equation (16):

$$R_{ds}(t) = \begin{cases} 10 \quad \text{if } (s_t = B) \wedge (C(B) = 1) \wedge (t \neq T) \\ 10 - \sum r_{br} \text{ if } (s_t = B) \wedge (C(B) = 1) \wedge (t = T) \\ -10 - \sum r_{br} \text{ if } (C(B) = 0) \wedge (t = T) \\ \mu - d(s_t) \ \text{ if } C(B) = 0 \wedge (t \neq T) \\ \frac{-\sum r_{br}}{T-t+1} \ \text{ if } C(B) > 0 \end{cases} \tag{19}$$

μ is a constant represents the threshold for reach, in discrete settings, μ can be set to 0, in continuous settings, usually μ is setting to the radius of the target set if we assume target set is a ball. Therefore, the rewards before reach are negative since $d(s)$ is greater or equal to 0. This adaption does not violate the deadline-safe property from Lemma V.1. The values of 10 and -10 are constants and can vary depending on the problem.

Lemma V.2 (Monotonicity between deadline-safe probability and value)**.** With the reward in Equation (19), starting from a s_0 in the initial set, for any two control policies π_1 and π_2 with values $v^{\pi_1}(s_0)$, $v^{\pi_2}(s_0)$ and deadline-safe probability $P^{\pi_1}(s_0)$, $P^{\pi_2}(s_0)$, we have monotonicity

$$P^{\pi_1}(s_0) \leq P^{\pi_2}(s_0) \Longleftrightarrow V^{\pi_1}(s_0) \leq V^{\pi_2}(s_0) \tag{20}$$

Proof. To be noticed, π_1 and π_2 are not guaranteed to be deadline-safe. For policies π_1, we denote $r_t^{\pi_1}$ as the reward received at time t, $r_t^{\pi_1} = R_{ds}(t)$. Similarly, we have $r_t^{\pi_2}$. To prove $\Rightarrow$, we have:

$$\begin{gathered} V^{\pi_1}(s_0) = E_1^{\pi_1} + E_2^{\pi_1} \\ E_1^{\pi_1} = P^{\pi_1}(s_0) * \Sigma^{ds} = \mathbb{E}_{ds}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t^{\pi_1}\right] \\ E_2^{\pi_1} = (1 - P^{\pi_1}(s_0)) * \Sigma^{nds} = \mathbb{E}_{nds}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t^{\pi_1}\right] \end{gathered} \tag{21}$$

E_1 is the expectation of deadline-safe cases, E_2 is the expectation of other cases. The sum of rewards of any episode if deadline-safe is 10, also, this is the highest cumulative rewards it can obtain, therefore $\Sigma^{ds} = 10$. On the other hand, any non-deadline-safe episode will grant cumulative reward -10. then we have $\Sigma^{nds} < 10$, $\Sigma^{ds} > \Sigma^{nds}$ Therefore, $V^{\pi_1}(s_0) - V^{\pi_2}(s_0) = (E_1^{\pi_1} - E_1^{\pi_2}) + (E_2^{\pi_1} - E_2^{\pi_2}) = (P^{\pi_1}(s_0) - P^{\pi_2}(s_0)) * \Sigma^{ds} + (P^{\pi_2}(s_0) - P^{\pi_1}(s_0)) * \Sigma^{nds} = (P^{\pi_1}(s_0) - P^{\pi_2}(s_0)) * (\Sigma^{ds} - \Sigma^{nds})$. Since $P^{\pi_1}(s_0) \leq P^{\pi_2}(s_0)$, we have $(P^{\pi_1}(s_0) - P^{\pi_2}(s_0)) \leq 0$. And we also have $\Sigma^{ds} > \Sigma^{nds}$, then $(P^{\pi_1}(s_0) - P^{\pi_2}(s_0)) * (\Sigma^{ds} - \Sigma^{nds}) \leq 0$.

To prove $\Leftarrow$, same as above, we have $V^{\pi_1}(s_0) - V^{\pi_2}(s_0) = (P^{\pi_1}(s_0) - P^{\pi_2}(s_0)) * (\Sigma^{ds} - \Sigma^{nds})$. Because $(\Sigma^{ds} - \Sigma^{nds}) = 20 > 0$, if $V^{\pi_1}(s_0) - V^{\pi_2}(s_0) \leq 0$, we can obtain $(P^{\pi_1}(s_0) - P^{\pi_2}(s_0)) \leq 0$. ☐

Lemma V.2 claims that a greater probability of deadline-safe is equivalent to a higher value. Therefore, the optimal control policy with the largest value must reach the target within the greatest probability of deadline-safe. Therefore, we can obtain this PAC bound on deadline-safe probability:

Lemma V.3. Under a model-free RL algorithm that optimizes a control policy π to reach a target in a deadline T, if we use the reward in Equation (19), with an error $\epsilon \geq 0$, false probability of the hypothesis $\delta \in [0, 1]$, the reward of first time reaching $r_{fr} > 0$ and reward of not reaching at the end $r_{nr} < 0$, and discount factor $\gamma = 1$, we have:

$$\begin{gathered} P[V^{\pi^*}(s_0) - V^{\pi}(s_0) \leq \epsilon] \geq 1 - \delta \\ \Longrightarrow P[P^{\pi^*}(s_0) - P^{\pi}(s_0) \leq \epsilon_{ds}] \geq 1 - \delta \\ \epsilon_{ds} \leq \frac{\epsilon}{r_{fr} - r_{nr}} \end{gathered} \tag{22}$$

π^* is the optimal policy

Proof. From Lemma V.2, we know $P^{\pi^*}(s_0) - P^{\pi}(s_0) \geq 0$, let $P^{\pi^*}(s_0) = P^{\pi}(s_0) + \Delta p$, $\Delta p \geq 0$, then we have a probability of $1-\delta$ that $V^{\pi^*}(s_0) - V^{\pi}(s_0) = (P^{\pi^*}(s_0) - P^{\pi}(s_0)) * (\Sigma^{ds} - \Sigma^{nds}) \leq \epsilon$. To be noticed, in the settings of Equation (19), we set $r_{fr} = 10$ and $r_{nr} = -10$. In general, we have $(\Sigma^{ds} - \Sigma^{nds}) = r_{fr} - r_{nr}$. Therefore, $\Delta p \leq= \epsilon/(r_{fr} - r_{nr})$. ☐

Lemma V.3 provides a guarantee of the probability of deadline-safe under a model-free RL using the reward function in the format of Equation (19) based on the guarantee of the value obtained. For example, assuming we are using the reward function in Equation (19), and we have $\delta = 0.05$ and $\epsilon = 1$. If there is 95% probability that the value at an initial point is at most 1 less than the value of optimal policy we have at least 95% probability that the deadline-safe probability is no more than 5% less than the deadline-safe probability of the optimal policy.

To obtain this bound efficiently during the training, we need to apply PAC-MDP (sample efficient) [33] algorithms with polynomial number of training samples. We choose DGPQ [35] which is available for continuous state space and discrete action space and proved to be PAC-MDP.

Theorem V.4. With our reward design in Equation (19), within $m = \text{poly}(\mathcal{N}_S, 1/\epsilon, 1/\delta, r_{nr} + r_{fr} + (d_{max} - \mu) * (t-1))$, if the maximum distance from an original state to the target is d_{max}, we can achieve:

$$P[P^{\pi_*}(s_0) - P^{\pi}(s_0) \leq \frac{\epsilon}{r_{fr} - r_{nr}}] \geq 1 - \delta \tag{23}$$

$\mathcal{N}_S$ is a covering number of state space S.

Proof. Since DGPQ is PAC-MDP, it satisfies:

$$P[V^{\pi^*}(s_0) - V^{\pi}(s_0) \leq \epsilon] \geq 1 - \delta$$

with a number of samples $m = \text{poly}(\mathcal{N}_S, 1/\epsilon, 1/\delta, 1/(1-\gamma)))$ for an infinite-horizon MDP where the range of the value is $[0, 1/(1-\gamma)]$ [35]. However, under the reward function in Equation (19) with a finite-horizon MDP, the value range is $[r_{nr}, r_{fr} + (d_{max} - \mu) * (t-1)]$ since the the lower bound of the value is r_{nr}, and the highest value is $r_{fr} + (d_{max} - \mu) * (t-1)$ at the case if the agent reaches the target at the last step, and it keeps the maximum distance to the target at every time step before. Then with Lemma V.3, we can obtain the PAC bound on the deadline-safe probability. ☐

We have proved that using DGPQ can efficiently obtain a PAC bound on the probability of deadline-safe. Based on Proposition IV.1 and Proposition IV.2, we are motivated to provide standards to design deadline-safe reward and we deliver Lemma V.1. Since the optimal policy of MDP is obtained by solving Bellman Equation which maximizes the value, we derive Lemma V.2 to connect the probability of deadline-safe with the value. Based on Lemma V.2, we deliver Lemma V.3 to provide a guarantee of the probability of deadline-safe based on the guarantee of the value. Since PAC learning can obtain a guarantee of the value, based on Lemma V.3, we derive Theorem V.4 that shows a probabilistic guarantee of deadline-safe can be obtained by PAC learning.

VI. Evaluation

In this section, we evaluate the proposed deadline-safe control synthesis methods. First, we introduced the baselines and metrics. Then, we compared it with existing baselines to show its effectiveness in two case studies on a low-dimensional linear system and a high-dimensional non-linear system. Additionally, we make a special case study to validate the PAC bound we proved in Theorem V.4.

A. Baselines and Metrics

In this work, we consider two baselines:

1) STL-reward: The reward function introduced in [30] with dense reward settings [28]. This reward is based on the quantitative semantics of STL in equation 5. This format of reward functions is designed to satisfy STL specifications and has been applied in existing works [27], [28].
2) Distance-based reward: Distance-based reward functions in RL primarily leverage the spatial or state space distance between an agent's current state and a goal state to guide its learning process. This approach simplifies the reward structure by focusing on reducing the distance to the goal without requiring additional domain expertise [42], [43]. In our experiments, we use Euclidean distance to measure the distance between the agent to the target and the obstacle and generate corresponding rewards.

We use the rate of deadline-safe as the metric for evaluation. No matter the agent has a collision with the unsafe set or has not reached the target by the deadline, we consider this test case is not deadline-safe.

B. Case Study: DC Motor Position

A DC motor is a crucial component in various applications where precise motion control is required, such as robotics, automotive systems, and industrial machinery. The ability to control the position of a DC motor is vital for tasks that demand high accuracy in short deadlines. Position control in DC motors is typically achieved through the use of feedback mechanisms, such as encoders, which provide real-time data on the motor's position. This allows for precise adjustments to be made to the motor's speed and direction, ensuring that the motor reaches and maintains the desired position. In our experiments, DC motor position benchmark is a linear system that uses the current as control input to manipulate the system that drives the motor angle to a target position. For the details of the system model, please check [4]. The initial set of the DC motor position benchmark is $[0, -1, -10]$ to $[\pi, 1, 10]$, the target set is a ball with radius 0.5 centered at $[\pi/2, 0, 0]$, the unsafe set is set as a ball with radius 0.2 centered at $[\pi/4, 0, 0]$,

Fig. 4 shows the results of the experiments on the DC motor position benchmark for reach-avoid tasks. There are two main observations from Fig. 4:

T	15			20			30		
	DS	STL	DIS	DS	STL	DIS	DS	STL	DIS
A	**86.5**	76.1	5.9	**96.7**	94.8	9.4	**100.0**	99.9	16.0

TABLE I: The deadline-safe rate of deadline-safe RL, STL-reward and distance-based reward on DC motor position benchmark with different deadlines. T stands for deadline, A stands for reach-avoid tasks, DS stands for deadline-safe, STL stands for STL-reward, and DIS stands for distance-based reward.

1) Deadline-Safe RL outperforms baselines when the deadline is 15, it has similar results with STL-reward when the deadline is 20 and 30. This observation shows the effectiveness of deadline-safe RL compared to the baselines. The performance of the distance-based reward is shaking due to the unawareness of the deadline.
2) STL-based reward converges slightly faster than the deadline-safe RL when the deadline is 30. This observation may be due to the expansion of the state space in deadline-safe. Since R-MDP includes reward history to the state space which is greater than the original state space which only has three states.

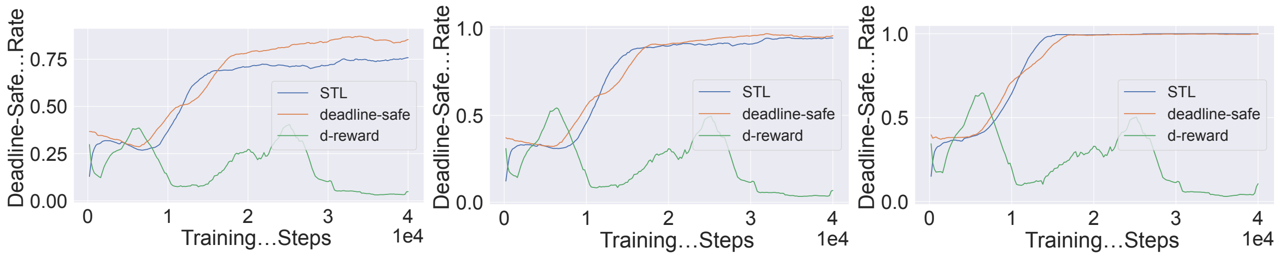
Table I shows the quantitative results on the deadline-safe rate on the DC motor position benchmark using SAC for training. Deadline-Safe RL outperforms the baselines on various deadlines for reach-avoid tasks.

C. Case Study: Attitude Control

Attitude control in real-time is essential for maintaining the stability and orientation of various CPS, such as satellites, aircraft, and spacecraft. Accurate and timely attitude control ensures that these systems can perform their intended functions, such as communication, navigation, or scientific observation, without deviation from their desired orientations. Real-time attitude control allows for immediate adjustments in response to disturbances or changes in environmental conditions, enhancing the reliability and effectiveness of the system. Moreover, it is critical for ensuring safety, as improper attitude control can lead to mission failures and collisions.

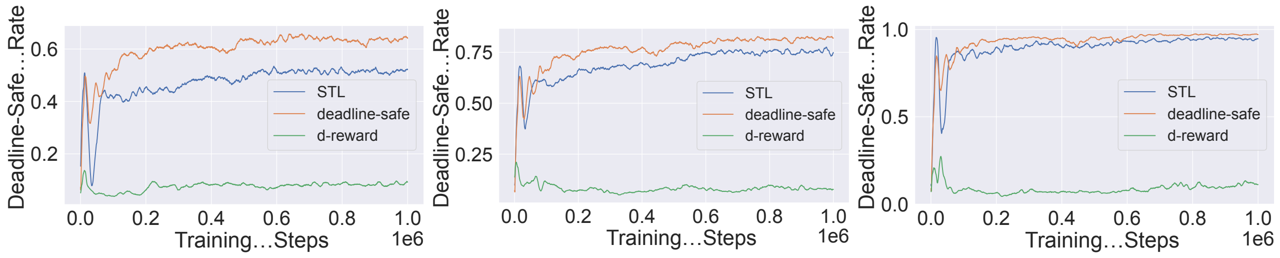
In our experiment, we consider a non-linear model of attitude control for a rigid body which is introduced in the competition of the workshop of Applied Verification of Continuous and Hybrid Systems (ARCH'2022), please check the details of system model at [44]. This benchmark is more complex than the DC motor position benchmark due to the non-linearity and the increasing of the dimension of the states. To better illustrate the effectiveness of deadline-safe method we propose, we first start with the reach tasks without considering collision. The initial set of the attitude control benchmark is $[-1, -1, -1, -1, -1, -1]$ to $[1, 1, 1, 1, 1, 1]$, the target set is a ball with a radius 0.8 centered at $[0, 0, 0, 0, 0, 0]$, the unsafe set is set as a ball with radius 0.3 centered at $[0, 0, 0.2, 0, 0, 0]$, the time step of this benchmark is 0.1 seconds

Fig. 5 shows the deadline-safe rate of the proposed method with two baselines over the training process in 1 million steps using SAC algorithm. The number of training steps is the product of the episode length with the number of episodes. The x-



(a) deadline=15 (b) deadline=20 (c) deadline=30

Fig. 4: Deadline-safe rate comparison between deadline-safe RL, STL-reward and distance-based reward for reach-avoid tasks on DC motor position benchmark. Deadline-Safe rate is smoothed with a moving window of size 20 for better visualization.



(a) deadline=15 (b) deadline=20 (c) deadline=30

Fig. 5: Deadline-safe rate comparison between deadline-safe RL, STL-reward and distance-based reward for reach tasks on attitude control benchmark. Deadline-Safe rate is smoothed with a moving window of size 20 for better visualization.

axis is the number of training steps, the y-axis is the deadline-safe rate. For better visualization, the deadline-safe rate is smoothed by a moving window of size 20. The orange curve represents the deadline-safe rate of deadline-safe RL proposed in this work, the blue curve represents the deadline-safe rate of STL-reward, the green curve represents the deadline-safe rate of the distance-based reward. The deadline-safe rate is calculated over 1000 testing initial points in the initial set. We conduct three groups of experiments where we set the deadline to 15,20 and 30 for reach tasks without considering collision.

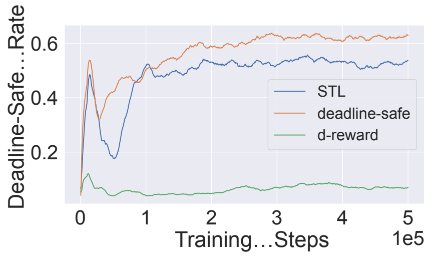
There are three main observations from Fig. 5:

1) Deadline-safe RL outperforms baselines on different deadlines in terms of deadline-safe rate. When the deadline is set to 15, STL-reward achieves around 50% deadline-safe rate at the end of the training, deadline-safe RL achieves over 60% deadline-safe rate. The distance-based reward performs not well under the model-free scenario if there is a deadline.
2) Deadline-safe RL converges faster than STL-reward. The gap between the deadline-safe rate of deadline-safe RL and that of STL-reward is getting smaller with the increment of the training steps. This observation shows deadline-safe RL is efficient for the training of time-critical tasks with a fixed deadline. With more training steps, STL-reward agents gradually improve their performance and become better at completing tasks on time, even though their primary motivation is to finish ASAP.
3) The gap between the deadline-safe rate of deadline-safe RL and that of STL-reward is getting smaller with a longer deadline. In other words, deadline-safe RL is more efficient compared with the baselines when the deadline is short. This observation conforms to the relationship between ASAP and deadline-safe. Deadline-safe RL is designed to ensure task completion within the specified deadline, prioritizing reliability and adherence to time constraints. On the other hand, STL-reward encourages ASAP policies, encouraging the agent to complete tasks as quickly as possible. As the deadline increases, the aggressive nature of the STL-reward mechanism means that the agent, already motivated to finish early, finds it progressively easier to meet the deadline. When the deadline is getting greater, it is becoming harder and harder for it to miss the deadline due to its aggressive behavior.

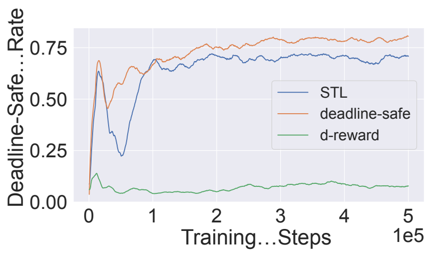
We also conduct experiments for reach-avoid tasks which we add an unsafe set in ball shape in the way between the initial set and the target set.

Fig. 6 shows the results for reach-avoid tasks. We can see similar trends to the reach tasks. Deadline-Safe RL can efficiently complete the reach-avoid tasks by the given deadline.

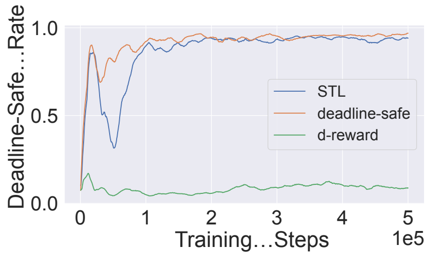
Quantitative results for experiments on attitude control benchmarks can be found in Table II. There is a minor observation that the gap between the deadline-safe rate of deadline-safe RL and that of STL-reward for reach-avoid is slightly smaller than that for reach tasks when the deadline is 15, but it is bigger when the deadline is 20. This may caused by various reasons such as the position and the size of the unsafe set. This may also be the reason for the slight difference between the performance of deadline-safe RL on



(a) deadline=15



(b) deadline=20



(c) deadline=30

Fig. 6: Deadline-safe rate comparison between deadline-safe RL, STL-reward and distance-based reward for reach-avoid tasks. Deadline-Safe rate is smoothed with a moving window of size 20 for better visualization.

T	15			20			30		
	DS	STL	DIS	DS	STL	DIS	DS	STL	DIS
R	**63.6**	51.5	8.8	**81.6**	76.1	7.8	**96.9**	95.2	10.2
A	**62.7**	54.0	7.0	**81.0**	70.4	7.9	**97.2**	93.8	9.0

TABLE II: The deadline-safe rate of deadline-safe RL, STL-reward and distance-based reward on attitude control benchmark with different deadlines. T stands for deadline, R stands for reach tasks, A stands for reach-avoid tasks, DS stands for deadline-safe, STL stands for STL-reward, DIS stands for distance-based reward.

reach and reach-avoid tasks.

D. *Case Study: PAC deadline-safe*

In this subsection, we will validate the PAC bound of the probability of deadline-safe on a discretized model of DC motor position since DGPQ can only be applied to the system with discrete action space.

The optimal policy can be obtained by brute-force or other optimization-based methods since the deadline is fixed. For each initial point, we can do a brute-force on the set of possible combinations of actions to see whether it can reach the target by the deadline, and this is guaranteed to find a solution if there exists one. As long as there is a single case that a series of actions can drive the agent from the initial point to the target by the deadline, the optimal policy should be able to find it. We need to randomly select several initial points from the initial set to estimate the probability of the deadline-safe of the optimal policy. We choose 50 initial points since the minimum requirements for the law of large numbers to make a statistically sound conclusion about a population is 30.

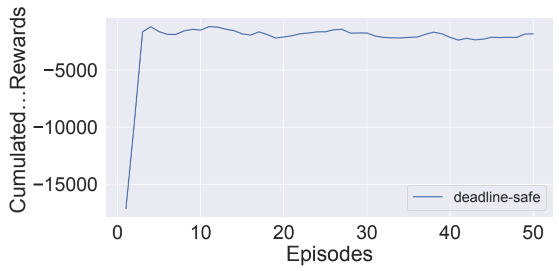


Fig. 7: The cumulative rewards of deadline-safe RL using DGPQ, the training takes 50 episodes

Fig. 7 shows the cumulative rewards of deadline-safe RL using DGPQ on the discretized model of the DC motor position. The cumulative rewards is the sum of the rewards on each step of the testing trajectories from 50 random initial points after each 2 episodes during training. The sum is negative since we early stop the testing whenever it reaches the target. The curve shows it converges after the 7^{th} episode.

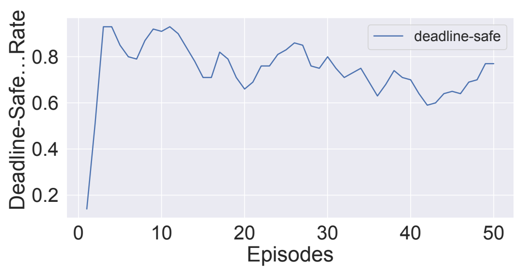


Fig. 8: The deadline-safe rate of deadline-safe RL using DGPQ, the training takes 50 episodes

Fig. 8 shows the deadline-safe rate of deadline-safe RL using DGPQ on the discretized model of DC motor position. The initial set of the discretized DC motor position benchmark is $[0, -1, -1]$ to $[\pi, 1, 1]$, the target set is a ball with a radius 0.5 centered at $[\pi/2, 0, 0]$, the unsafe set is set as a ball with radius 0.2 centered at $[\pi/4, 0, 0]$, the time step of this benchmark is 0.05 seconds and the deadline is 100 time steps. The control input is discretized as $\{-5, -2, -1, 1, 2, 5\}$, we can see the performance reaches the peak after the 7^{th} episode.

Since $\delta = 0.05$, $\epsilon/(r_{fr} - r_{nr}) = 0.05$, and the deadline-safe rate is 98% at the peak, and the optimal policy cannot reach an deadline-safe rate greater than 100%, the experiment results validate the statement in Lemma V.4 and the efficiency of DGPQ for deadline-safe tasks. There is a minor observation that the performance is shaking after reaching the peak. This is due to the over-fitting and can be improved by regularization and other existing methods. Similar phenomenons are found in the experiments from [35].

VII. Conclusion

In conclusion, this work addresses the challenges inherent in real-time CPS by proposing an innovative integration of deadline-safe with RL. By redefining the reward structure within a newly formulated reward-based Markov Decision Process (R-MDP), we have mitigated issues associated with non-Markovian rewards and reward hacking. Our reward functions are specifically designed to encourage deadline-safe, thereby enhancing the reliability and safety of operations within time-critical applications. The empirical evaluations across diverse benchmarks demonstrate that our method not only fosters the development of deadline-safe policies but also significantly outperforms existing baselines. In the future, we are interested in designing efficient deadline-safe training algorithms. Another interesting direction is to extend our method to multi-agent scenarios.

VIII. Acknowledgment

This work was supported in part by ARO W911NF-20-1-0080, NSF CNS-2143274 and CNS-2333980.

References

[1] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Proceedings of the 47th design automation conference*, 2010, pp. 731–736.

[2] P. Lu, L. Zhang, M. Liu, K. Sridhar, O. Sokolsky, F. Kong, and I. Lee, "Recovery from adversarial attacks in cyber-physical systems: Shallow, deep, and exploratory works," *ACM Computing Surveys*, vol. 56, no. 8, pp. 1–31, 2024.

[3] M. Liu, L. Zhang, V. V. Phoha, and F. Kong, "Learn-to-respond: Sequence-predictive recovery from sensor attacks in cyber-physical systems," in *2023 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2023, pp. 78–91.

[4] L. Zhang, P. Lu, F. Kong, X. Chen, O. Sokolsky, and I. Lee, "Real-time attack-recovery for cyber-physical systems using linear-quadratic regulator," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–24, 2021.

[5] F. Kong, M. Xu, J. Weimer, O. Sokolsky, and I. Lee, "Cyber-physical system checkpointing and recovery," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 22–31.

[6] L. Zhang, K. Sridhar, M. Liu, P. Lu, X. Chen, F. Kong, O. Sokolsky, and I. Lee, "Real-time data-predictive attack-recovery for complex cyber-physical systems," in *2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2023, pp. 209–222.

[7] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 174–179.

[8] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, pp. 3–27, 2015.

[9] V. Pong, S. Gu, M. Dalal, and S. Levine, "Temporal difference models: Model-free deep rl for model-based control," *arXiv preprint arXiv:1802.09081*, 2018.

[10] N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis, "Learning model-free robot control by a monte carlo em algorithm," *Autonomous Robots*, vol. 27, pp. 123–130, 2009.

[11] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[12] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[13] D. Harada, "Reinforcement learning with time," in *AAAI/IAAI*, 1997, pp. 577–582.

[14] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4045–4054.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[16] C. Fan, Z. Qin, U. Mathur, Q. Ning, S. Mitra, and M. Viswanathan, "Controller synthesis for linear system with reach-avoid specifications," *IEEE Transactions on Automatic Control*, vol. 67, no. 4, pp. 1713–1727, 2021.

[17] Z. Zhou, J. Ding, H. Huang, R. Takei, and C. Tomlin, "Efficient path planning algorithms in reach-avoid problems," *Automatica*, vol. 89, pp. 28–36, 2018.

[18] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.

[19] T. Badings, L. Romao, A. Abate, and N. Jansen, "A stability-based abstraction framework for reach-avoid control of stochastic dynamical systems with unknown noise distributions," *arXiv preprint arXiv:2404.01726*, 2024.

[20] A. Alanwar, Y. Stürz, and K. H. Johansson, "Robust data-driven predictive control using reachability analysis," *European Journal of Control*, vol. 68, p. 100666, 2022.

[21] G. P. Kontoudis and K. G. Vamvoudakis, "Kinodynamic motion planning with continuous-time q-learning: An online, model-free, and safe navigation framework," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 12, pp. 3803–3817, 2019.

[22] I. Koryakovskiy, M. Kudruss, R. Babuška, W. Caarls, C. Kirches, K. Mombaur, J. P. Schlöder, and H. Vallery, "Benchmarking model-free and model-based optimal control," *Robotics and Autonomous Systems*, vol. 92, pp. 81–90, 2017.

[23] Y. Zhang, B. Zhao, and D. Liu, "Deterministic policy gradient adaptive dynamic programming for model-free optimal control," *Neurocomputing*, vol. 387, pp. 40–50, 2020.

[24] M. Grzes, "Reward shaping in episodic reinforcement learning," 2017.

[25] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287.

[26] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3834–3839.

[27] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 349–10 355.

[28] N. Hamilton, P. K. Robinette, and T. T. Johnson, "Training agents to satisfy timed and untimed signal temporal logic specifications with reinforcement learning," in *International Conference on Software Engineering and Formal Methods*. Springer, 2022, pp. 190–206.

[29] N. K. Singh and I. Saha, "Stl-based synthesis of feedback controllers using reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 15 118–15 126.

[30] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[31] Y. Engel, S. Mannor, and R. Meir, "Reinforcement learning with gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 201–208.

[32] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "Pac model-free reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 881–888.

[33] A. L. Strehl, L. Li, and M. L. Littman, "Reinforcement learning in finite mdps: Pac analysis." *Journal of Machine Learning Research*, vol. 10, no. 11, 2009.

[34] L. Li, M. L. Littman, and T. J. Walsh, "Knows what it knows: a framework for self-aware learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 568–575.

[35] R. Grande, T. Walsh, and J. How, "Sample efficient reinforcement learning with gaussian processes," in *International Conference on Machine Learning*. PMLR, 2014, pp. 1332–1340.

[36] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[37] F. Denis, "Pac learning from positive statistical queries," in *International conference on algorithmic learning theory*. Springer, 1998, pp. 112–126.

[38] J. Pazis and R. Parr, "Pac optimal exploration in continuous space markov decision processes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, no. 1, 2013, pp. 774–781.

[39] D. Haussler and M. Warmuth, "The probably approximately correct (pac) and other learning models," *The Mathematics of Generalization*, pp. 17–36, 2018.

[40] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6565–6570.

[41] M. Liu, P. Lu, X. Chen, F. Kong, O. Sokolsky, and I. Lee, "Fulfilling formal specifications asap by model-free reinforcement learning," *arXiv preprint arXiv:2304.12508*, 2023.

[42] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2011, pp. 761–768.

[43] L. P. Kaelbling, "Learning to achieve goals," in *IJCAI*, vol. 2. Citeseer, 1993, pp. 1094–8.

[44] D. M. Lopez, M. Althoff, L. Benet, X. Chen, J. Fan, M. Forets, C. Huang, T. T. Johnson, T. Ladner, W. Li *et al.*, "Arch-comp22 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants," in *9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22)*. EasyChair, 2022, pp. 142–184.