2024 22nd ACM-IEEE International Symposium on Formal Methods and Models for System Design (MEMOCODE) | 979-8-3503-7802-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/MEMOCODE63347.2024.00009

Model-free PAC Time-Optimal Control Synthesis 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*—Reaching a target safely and quickly is a control goal pursued by various applications, such as post-disaster rescue robots and industrial shipment. However, it is hard to formally guarantee safety and time-optimality under unknown dynamics via model-free controller synthesis algorithms. As a response, we propose a model-free reinforcement learning (RL) algorithm that synthesize a controller to reach a predefined target set of states with a probabilistic guarantee of time optimality, i.e., the actual reaching time is bounded close to the shortest time possible with high probability, and the bound becomes tighter when more training data is sampled. Our algorithm leverages a reward function that based on signal temporal logic (STL) robustness to reward fast reaching. With this reward function, we prove that Probably Approximately Correct (PAC) optimality in the state-value function implies PAC optimality in reach time. Then, we build our algorithm by extending Deplayed Gaussian Process Q learning (DGPQ) algorithm with a safety margin to protect the controlled agent. Consequently, our algorithm guarantees safety and a PAC bound in recovery time. Experiments show our method can achieve 97.7% success rate to reach the target with in the maximum time tolerance and outperform baselines.**

I. Introduction

Time is life, and time is money — this saying applies to various controller synthesis scenarios. For example, a post-disaster rescue robot needs to find survivors within the shortest time possible [1], [2], and any delay in industrial shipping may impose a large cost on manufacturing [3]. Therefore, researchers have discussed time-optimal control [4], [5] since 1950s. Recent progress in controller synthesis has also targeted safe and fast task accomplishment, such as by setting a maximum tolerable time [6], [7], [8] or encouraging high speed by rewards [9], [10].

Consider a motivating example of a post-disaster rescue robot [1]. The robot navigates through ruins after an earthquake, aiming to reach a location where trapped survivors are detected. This scenario presents numerous challenges: the robot must traverse highly irregular terrains, avoid obstacles, and contend with varying road surface conditions and different loads. These factors make the system dynamics complex and hard to be identified in an explicit form, such as ordinary differential equations (ODE). Traditional control methods, which rely on precise mathematical models, struggle in such unpredictable environments. In scenarios with such unknown and complex dynamics, our objective is for the robot to reach survivors in the shortest possible time. However, if achieving the absolute shortest time is infeasible due to unforeseen obstacles and unknown dynamics, the robot's reaching time should not significantly exceed this shortest time.

To address this problem, we find the concept of Probably Approximately Correct (PAC) bounds [11] relevant. Specifically, a PAC bound is a probabilistic bound on a quantity's deviation from its optimal value, that is achievable with small computational overhead. This concept is particularly useful in control systems where exact optimality is difficult to guarantee due to uncertainty and complexity. Our major challenge is to guarantee a PAC-bounded gap between the actual target reaching time and the shortest time possible with dynamics unknown. Controller synthesis for constraint satisfaction without knowledge of the dynamics still remains an open problem [12]. So far, controllers with guaranteed time-optimality in reaching targets require knowing the dynamics explicitly. To tackle time-optimality in black-box dynamics [9], [10], scientists have leveraged model-free reinforcement learning (RL) [13], which generates adaptive control policies and bypasses dynamics identification. Indeed, with RL, we can train the rescue robot in similar environments, and its control policies are able to adapt to the actual dynamics [14], [15], [16]. However, existing works cannot provide a guarantee in a bounded reaching time gap in RL-based solutions so far.

To address this limitation, we propose *Safe and PAC Time-Optimal DGPQ (SPT-DGPQ)*, a model-free RL algorithm that guarantees PAC time-optimality in target reaching tasks on top of safety. SPT-DGPQ is extended from a PAC-MDP algorithm, Delayed Gaussian Process Q-learning (DGPQ)[17], added with (1) a specific reward function that aims for time-optimality and (2) safety margins during exploration and inference time. We prove that as long as the synthesized control policy has its state-value function PAC-bounded, it also has the target reaching time PAC-bounded. That is, starting from any initial state in the initial set, the gap between the actual reaching time and the shortest reaching time possible is bounded with high probability, and the bound becomes tighter with more training data.

In summary, we have the following contributions:

1) We propose a model-free RL algorithm that produces controllers safely reaching a target fast in unknown dynamics. To the best of our knowledge, this is the

* denotes equal contribution.

2832-6520/24/$31.00 ©2024 IEEE
DOI 10.1109/MEMOCODE63347.2024.00009

Methods	Model -free	Time-optimality guaranteed	Safety considered
Model-based e.g. [4], [18], [19], [20], [21], [22]		✓	
RL-based e.g. [9], [10], [23], [24]	✓		✓
SPT-DGPQ (ours)	✓	✓(PAC)	✓

TABLE I: SPT-DGPQ vs. existing time-optimal control methods.

first model-free RL algorithm provides a probabilistic guarantee on the reaching time.

2) We prove that the synthesized control policies guarantee PAC time-optimality with enough training samples. Our method is the first to integrate all three essential features: robustness to uncertainty, temporal logic satisfaction, and optimal control performance for systems operating in a continuous state space.
3) We run extensive experiments on various benchmarks including linear systems and non-linear systems. We provide a case study that validates the PAC time-optimality of SPT-DGPQ.

The rest of this paper is organized as follows: Section II discusses related papers. Section III presents preliminaries. Section IV presents the problem statement and assumptions. Section V presents the time-optimality proof. Section VI evaluates our method. Section VII concludes the paper.

II. Related Works

The problem of time-optimal control has been well-established since the 1950s. The motivation is to achieve the fastest trajectory to achieve a given objective. The first known time-optimal control is developed in bang-bang systems, i.e., the systems that switch between two discrete states like power on/off for a water heater [4]. In the following decades, online optimization-based time-optimal control has been synthesized for planning the fastest routes in robots [20] and identifying the fastest quantum state transition using magnetic pulses [21]. In 2000s, MPC-based time-optimal control has been leveraged in both linear [22], [18] and nonlinear dynamical systems [19].

The assumption for these methods above is a known system dynamics, and therefore analytical time-optimal control can be derived. Additionally, the above literature have not considered obstacles that have to be avoided during the control synthesis. When the system model is unknown, data-driven control synthesis takes place. For fully unknown dynamics, state-of-the-art techniques leverage model-free RL [25]. Recently, researchers have addressed the issue of achieving maximal reward within the time limit using RL, where these time limits can be set by either the environment or for training purposes [24]. More research in RL has shown time-optimality can be encouraged by reward function designs in various applications, including autonomous racing and quadcopter drones [9], [10], [23].

However, one shortcoming of these RL-based time-optimal control is the lack of guarantees. Researchers have categorized control synthesis methods into three levels of guarantees: (level 1) no guarantee, only empirical supports available, (level 2) probabilistic guarantee and (level 3) analytical guarantee [12]. So far, all model-free time-optimal control algorithms belong to level 1, and our goal is to develop the first algorithm in level 2. PAC RL has been studied in scenarios such as simulation-to-reality transfer [26]. To obtain a probabilistic guarantee of time-optimality, we consider probably approximately correct (PAC) RL [27], [11], [28], which provides probabilistic guarantees in achieving maximal total reward with sample efficiency.

Unfortunately, the majority of PAC RL algorithms assume finite state and action spaces, which are not applicable to continuous systems. We have identified one method that assumes continuous state space, namely delayed Gaussian process Q learning (DGPQ) [17]. This approach provides a PAC bound on training sample efficiency to achieve the maximum expected reward by using Gaussian processes to represent value functions. In this paper, we modify DGPQ so that its PAC guarantee on value is converted into PAC guarantee on time-optimality.

A comprehensive comparison between our method and existing time-optimal control methods is listed in Table I. We categorize existing works into model-based methods and RL-based methods. We compare these methods on three features: model-free, guarantee of time-optimality and safety consideration. Model-based methods have time-optimality guarantee since the control is obtained by solving optimization problems based on the system model. This guarantee is generally unavailable in RL-based methods. Our method is the first work that includes all three features.

III. Preliminaries

A. Signal Temporal Logic (STL) and Robustness

Signal temporal logic (STL) is a logical formalization to specify properties of continuous signals [29], such as trajectories in a physical state space. On a trajectory denoted as $\bar{s} := s_0 s_1 \dots s_T$, $\forall s_t \in S$, the grammar of STL is defined as

$$\varphi ::= \top \mid g(\bar{s}) < 0 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 U_{[t_1,t_2]} \varphi_2. \quad (1)$$

Here, $\top$ is tautology and $g : S^T \mapsto \mathbb{R}$ is a real-valued function on signals. Temporal operator $U_{[t_1,t_2]}$ denotes the until operator, and $\varphi_1 U_{[t_1,t_2]} \varphi_2$ means φ_2 must hold at some time $t \in [t_1, t_2]$ and φ_1 must always hold before t. The until operator can be converted into two additional temporal operators (1) eventually/finally operator $F_{[t_1,t_2]}$ and (2) always/globally operator $G_{[t_1,t_2]}$. Please refer to [29] for more details.

Researchers have defined robustness scores on STL, also known as quantitative semantics [30], [31], [32]. Specifically, a robustness score is a real-valued function $\rho : S^T \times \mathbb{N}_{\geq 0} \times \Phi \mapsto$

$\mathbb{R}$ that evaluates $\rho(\bar{s}, t, \varphi)$ on three inputs: a trajectory $\bar{s}$ of length T, a starting time step t of the trajectory, and a STL task $\varphi \in$ task space Φ. The function ρ evaluates how much a trajectory satisfies a specification, with $\rho \geq 0$ equivalent to task accomplished, and $\rho < 0$ equivalent to failure. Please refer to [30] for more details.

B. Model-free reinforcement learning (RL)

Model-free RL is a machine learning paradigm that learns control policies in black-box dynamics without system identification [13], [33], [34]. Specifically, RL models a system state's evolution with a Markov decision process (MDP) $\mathcal{M} = (S, A, f, r, \gamma)$. Here, S and A are the state space and action space. In addition, $r : S \mapsto \mathbb{R}$ is a real-valued reward at each state, and $\gamma \in (0, 1)$ is a discount factor.

The transition distribution $f : S \times A \mapsto \Delta(S)$ models a stochastic dynamics. The notation $\Delta(\cdot)$ denotes the space of all probability distributions over a set. Moreover, a stochastic control policy is a function $\pi : S \mapsto \Delta(A)$. Based on this formalization, from an initial state $s_0 \in S$, a trajectory of states is sampled by: (i) at a discrete time step t, action $a_t \sim \pi(s_t)$ and (ii) the next state $s_{t+1} \sim f(s_t, a_t)$. Notice that deterministic dynamics, control policies and initial states are special cases of their stochastic versions, with a probability of obtaining a state or action being either 0 or 1.

Under the model-free assumption, f is unknown. The goal for model-free RL is to find a control policy that maximizes the value function for all $s \in S$, i.e.,

$$\text{maximize}_\pi \, v(s) := \mathbb{E}_{f,\pi}[\sum_{t=1}^{\infty} \gamma^t r(s_t) \mid s_0 = s]. \tag{2}$$

C. PAC-MDP RL

A RL algorithm $\mathcal{A}$ is PAC-MDP [11], [27] iff it synthesizes a control policy π within a polynomial sample size $m = \text{poly}(1/\epsilon, 1/\delta)$ and satisfy

$$\forall s \in S, \Pr_{f,\pi,\mathcal{A}}[v^*(s) - v^\pi(s) \leq \epsilon] \geq 1 - \delta, \tag{3}$$

where v^* and v^π are the value function under optimal control policy π^* and the synthesized control policy π, respectively, $\epsilon \geq 0$ is an error rate, and $\delta \in (0, 1)$ is the probability of failure. Notice that besides f and π, randomness also sources from the algorithm $\mathcal{A}$, such as sampling training trajectories.

D. Q Learning

Q learning is a standard value-based model-free RL technique [35], [36], [37]. The idea is to estimate an action-value function, also known as Q-value function $q : S \times A \mapsto \mathbb{R}$ and represent it as learnable parameters. The Q-value function is defined as

$$q(s_t, a_t) := \mathbb{E}_{f,\pi}[\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}) \mid s_t, a_t]. \tag{4}$$

That is, the expected cumulative reward that can be received if we take an action a_t at a state s_t. From this definition, we can see the relationship between Q-value function and the state-value function (also known as the V-value function) is $v(s_t) = \mathbb{E}_\pi[q(s_t, a_t)]$, i.e., the V-value is the expected Q-value over all actions according to the policy π. Q-learning can be implemented in various ways. The most naive method is to represent the Q-value function as a table, updating and caching the estimated Q-value at each state, action tuple [38], [39], [40]. However, this table representation has overhead proportional to the size of discrete state and action spaces, and is not feasible for continuous spaces. Therefore, researchers have investigated deep Q learning, where the Q-value function is represented by a deep neural network [41]. State-of-the-art deep Q learning algorithms include prioritized experience replay [42], error reduction by double representations [43], and injecting noises into learnable parameters for exploration tuning [44].

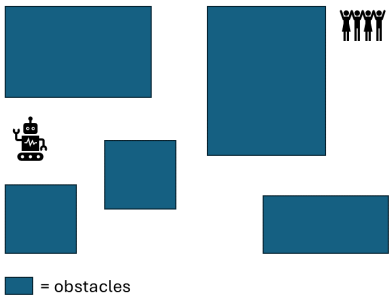


Fig. 1: A rescue robot is learning how to reach people in shortest time possible in a training field.

E. Gaussian Process and DGPQ

An alternative learnable parameter representation in model-free RL is Gaussian Processes (GP) [45]. A GP represents the joint distribution between an input x and a target y as $y \mid x \sim \mathcal{N}(\mu, k)$, where μ is a mean living in the same space of the target y and k is a covariance function, also known as a kernel. In plain words, the parameters to be updated throughout learning are μ and k. The motivation behind using GP for RL is to handle continuous state spaces while modeling the uncertainty behind the decision process. Example model-free RL using GP representation include GP-Sarsa [46] and iGP-Sarsa [47].

Delayed Gaussain Process Q-learning (DGPQ) is a PAC-MDP model-free RL algorithm that leverages GP representation in Q learning[17]. It is able to achieve Equation (3) by sampling $m = \text{poly}(\mathcal{N}_S, 1/\epsilon, 1/\delta, 1/(1-\gamma))$ data points. Here, $\mathcal{N}_S$ is a covering number of state space S. This method assumes a continuous state space and a finite action space, and parameterizes one GP per action. The idea is to maintain both GP representations and a tabular representation, and the table is updated only when the GP converges to a significantly different Q value. Therefore, the table's learning is delayed.

IV. Problem Formulation

A. Motivating Example

There are plenty of cases where we prefer an agent to complete a task as fast as possible. Let us consider a motivating example of a rescue robot [1], which is required to

reach the location of injured people, and time-optimality is desired. As illustrated in Figure 1, the robot must identify a route that takes as little time as possible while avoiding obstacles. In various post-disaster scenarios, the ground-truth dynamics would be too complex to be captured, and model-free RL can be leveraged to train the robot in a similar training environment. Then, transfer learning shall be adopted at runtime [48].

It would be promising that the time-optimality of a rescue robot, or any other applications that ask for speediness, can be theoretically guaranteed. Although a deterministic guarantee remains an open problem, we hope to reach the low-hanging fruit of probabilistic guarantee. Specifically, we would like to develop a learning algorithm that provides a PAC bound on time-optimality. That is, by using this algorithm, with high probability and low sample complexity, the learned policy would produce a trajectory that takes not much longer time than the shortest time possible.

B. Assumptions

Let $t = 0, 1, 2, \ldots, T$ be discrete time steps, with T being the maximum tolerable time of target reaching. We denote a continuous, compact and bounded physical state space as S, and a finite action space as A. Both S and A are metric spaces. A control policy is abstracted as a deterministic function $\pi : S \mapsto A$, i.e., the action $a_t = \pi(s_t)$. A system dynamics is abstracted as another deterministic function $f : S \times A \mapsto S$, i.e., the next state $s_{t+1} = f(s_t, a_t)$. We assume that the explicit model of f is unknown, but we can sample trajectories for training purposes. Moreover, we are given a safe set of states $S_{safe} \subset S$ and a target set of states $S_{target} \subset S_{safe}$ to be reached. We always start from some $s_0 \in S_{safe} \setminus S_{target}$. Finally, a trajectory's length will not exceed the tolerable time T. That is, if a trajectory successfully reaches S_{target} within T steps, we call an early stop. If it enters an unsafe state or does not reach the target within T, the task is aborted.

C. Problem Statement

We first define the target reaching time.

Definition 1 (Target reaching time)**.** *Given dynamics f and target state set S_{target}, the target reaching time $t^{\pi}_{rec} : S \mapsto \{0, 1, \ldots, T, \infty\}$ evaluates the number of time steps needed for reaching the target. That is, $t^{\pi}_{rec}(s) = \arg\min_{t \in \{0,\ldots,T\}} s_t \in S_{target}$ if S_{target} is reached from $s_0 = s$ within T steps under control policy π, and $t^{\pi}_{rec}(s) = \infty$ otherwise.*

The target reaching time is a way to measure how quickly the system can reach one of the goal states starting from an initial state, following a specific strategy, and within a given time limit. The target reaching time tells us the number of steps needed to get to the goal state from the starting point if it is possible to do so within the allowed time. If the system reaches the goal state within the allowed time, the target reaching time will be a specific number of steps it took to get there. If the system cannot reach the goal state within the allowed time, the target reaching time will indicate that it's not achievable in the given time horizon.

Next, we define a safe and time-optimal reaching problem as follows.

Definition 2 (Safe and Time-Optimal Reaching Problem)**.** *Given dynamics f and $S_{target} \subset S_{safe} \subset S$, this problem aims to solve*

$$\forall s_0 \in S_{safe} \setminus S_{target}, \textit{minimize}_{\pi}\, t^{\pi}_{rec}(s_0) \quad \textit{subject to } \forall t \in \{0, \ldots, T\},\ s_t \in S_{safe}, \tag{5}$$

With the dynamics being unknown, solving for the optimal solution of Equation (5) becomes undecidable for an arbitrary initial state s_0 in the initial set. This is because reachability problems under unknown dynamics are already proven to be undecidable [49], and if we can decide whether there exists a solution for (5), we can also decide whether S_{target} is reachable. This is primarily because the lack of complete knowledge about the system's behavior makes it impossible to exhaustively explore all possible state transitions and outcomes. These undecidability results highlight the inherent limitations of algorithmic methods in dealing with complex systems where the dynamics are not fully known or are highly complex. Therefore, we aim for a sub-optimal solution as follows.

Definition 3 (Safe and PAC Time-Optimal Reaching Problem)**.** *This problem aims to solve the problem in Definition 2 by model-free RL. Instead of finding a controller π^* that minimizes the reaching time $t_{rec}(s_0)$ for all $s_0 \in S$, it seeks a RL-based synthesis algorithm $\mathcal{A}$ for a sub-optimal controller π, such that, within $m = poly(\mathcal{N}_S, 1/\epsilon, 1/\delta)$ samples, we can achieve*

$$\forall s_0 \in S_{safe} \setminus S_{target},\ \Pr_{\mathcal{A}}[t^{\pi}_{rec}(s_0) - t^{*}_{rec}(s_0) \leq \epsilon_{rec}] \geq 1 - \delta, \tag{6}$$

where $t^{\pi}_{rec}(s_0)$ and $t^{*}_{rec}(s_0)$ denote the reaching time by π and π^* from a state s_0, respectively, $\epsilon_{rec} \geq 0$ is a gap in reaching time and $\delta \in [0, 1]$ is a failure probability.

We aim to solve a problem where we want to minimize the time it takes to reach a goal state from any starting point, but we do this using a learning approach that doesn't rely on the model of the system's dynamics. Instead of aiming for a perfect controller that optimally minimizes the reaching time from any starting state, this problem focuses on developing a sub-optimal controller through an RL-based synthesis algorithm. This approach ensures that, with a reasonable number of samples, the resulting controller can achieve nearly optimal reaching times with a high probability, while allowing for a small acceptable difference in performance compared to the optimal controller.

V. Safe and PAC Time-Optimal DGPQ (SPT-DGPQ)

A. Reward Function Design for PAC Time-Optimality

Our goal in this subsection is to design a reward function, so that a PAC bound on value function implies a PAC bound

on reaching time. To do so, at time t, we start from a STL formula on trajectory segment $\bar{s} = s_0 \dots s_t$

$$\varphi(t) := F_{t' \in [0,t]} d_{target}(s_{t'}) \leq 0, \tag{7}$$

where $d_{target} : S \mapsto \mathbb{R}_{\geq 0}$ evaluates the smallest distance from $s_{t'}$ to a point on S_{target}'s boundary. Since S is bounded, we have a maximum distance $\forall s \in S,\ d_{target}(s) \leq d_{max}$. Then, we design the following reward function on $\bar{s} = s_0 \dots s_t$.

$$\tilde{r}(\bar{s}) := \begin{cases} \rho(\bar{s}, 0, \varphi(t)) = \max_{t' \in [0,t]}(-d_{target}(s_{t'})) \text{ if } s_t \notin S_{target} \\ r_{max} \geq (d_{max} \sum_{t'=0}^{T} \gamma^{t'})/\gamma^T \text{ otherwise.} \end{cases} \tag{8}$$

That is, if the trajectory segment has not reached S_{target}, we give it a negative reward, which is the negative of its closest distance to S_{target} so far, i.e., the STL robustness of $\varphi(t)$. Once S_{target} os reached, we offer a large positive reward r_{max} and terminate the trajectory. Notice that this $\tilde{r}$ is not Markovian, since it requires a trajectory segment as input instead of just the current state. We can turn it into a Markovian r by:

$$r(s_t, r_{prev}) := \begin{cases} \rho(\bar{s}, 0, \varphi(t)) = \max(r_{prev}, (-d_{target}(s_t))) \text{ if } s_t \notin S_{target} \\ r_{max} \geq (d_{max} \sum_{t'=0}^{T} \gamma^{t'})/\gamma^T \text{ otherwise.} \end{cases} \tag{9}$$

This reward caches the previous robustness into r_{prev}, which is initialized to $-\infty$. Therefore, we have Markovian $r : S \times \mathbb{R} \mapsto \mathbb{R}$, with the state space augmented by a one-dimensional reward r_{prev}.

To support our forthcoming claims, we present the following Lemma:

Lemma 1 (Monotonicity between Reaching Time and Value)**.** *With the reward in Equation* (9)*, starting from a $s_0 \in S_{safe} \setminus S_{target}$, for any two control policies π_1 and π_2 with values $v^{\pi_1}(s_0)$, $v^{\pi_2}(s_0)$ and reaching time $t^{\pi_1}_{rec}(s_0)$, $t^{\pi_2}_{rec}(s_0)$, we have monotonicity*

$$\begin{aligned} &\forall s_0 \in S_{safe} \setminus S_{target}, \\ &\left(t^{\pi_1}_{rec}(s_0) \leq t^{\pi_2}_{rec}(s_0) \iff v^{\pi_1}(s_0) \geq v^{\pi_2}(s_0)\right) \end{aligned} \tag{10}$$

Proof. For policies π_1, we denote $r^{\pi_1}_t$ as the reward received at time t. Likewise, we have $r^{\pi_2}_t$. To prove $\Rightarrow$, the value under a policy π_i is $v^{\pi_i}(s_0) = \gamma^{t^{\pi_i}_{rec}(s_0)} r_{max} + \sum_{t=0}^{t^{\pi_i}_{rec}(s_0)} \gamma^t r^{\pi_i}_t$. That is, a large discounted positive reward upon reaching the target, plus the sum of negative discounted rewards before. Therefore,

$$\begin{aligned} &v^{\pi_1}(s_0) - v^{\pi_2}(s_0) \\ &= \underbrace{(\gamma^{t^{\pi_1}_{rec}(s_0)} - \gamma^{t^{\pi_2}_{rec}(s_0)})}_{\geq \gamma^T} r_{max} + \underbrace{\sum_{t=0}^{t^{\pi_1}_{rec}(s_0)} \gamma^t r^{\pi_1}_t}_{\geq -d_{max} \sum_{t=0}^{T} \gamma^t} \\ &\quad \underbrace{- \sum_{t=0}^{t^{\pi_2}_{rec}(s_0)} \gamma^t r^{\pi_2}_t}_{\leq 0} \\ &\geq \gamma^T r_{max} - d_{max} \sum_{t=0}^{T} \gamma^t \geq 0 \text{ by definition of } r_{max} \end{aligned} \tag{11}$$

To prove $\Leftarrow$, we can use the same bounds above to show the contrapositive holds, i.e., $t^{\pi_1}_{rec}(s_0) > t^{\pi_2}_{rec}(s_0) \Rightarrow v^{\pi_1}(s_0) < v^{\pi_2}(s_0)$. □

Lemma 1 establishes a connection between the value and the concept of reaching time. The value function represents the expected total reward an agent can obtain, starting from a given state and following a particular policy. Usually, the RL algorithm uses the estimated value function to improve the policy. The optimal policy is one that maximizes the value function, ensuring the agent accumulates the highest possible reward over time. Lemma 1 claims that shorter reaching time is equivalent to a higher value. Therefore, the optimal control policy with largest value must reach the target within shortest time possible.

In light of this relationship between reaching time and value, it is crucial to explore how the optimization of the value function translates to practical performance metrics. By maximizing the value function, the agent is effectively seeking the policy that yields the highest expected cumulative rewards. Consequently, the pursuit of maximizing the value function aligns with minimizing the reaching time, as the agent efficiently navigates toward states that yield maximal rewards. While maximizing the value function inherently guides the agent towards strategies that reduce reaching time, it is not sufficient to only establish this qualitative relationship. For practical applications, especially in scenarios requiring performance guarantees, a more rigorous quantitative analysis is necessary.

Therefore, to bridge this gap, we employ bounding techniques to derive a Probably Approximately Correct (PAC) guarantee on the reaching time. This approach allows us to leverage the PAC guarantee on the value function to obtain a corresponding bound on the reaching time, thus providing a more concrete and actionable performance metric for the optimal policy.

Lemma 2 (PAC Time-Optimality)**.** *Under a model-free RL algorithm $\mathcal{A}$ that synthesizes a control policy π to reach S_{target}, if we use the reward in Equation* (9)*, with an error $\epsilon \geq 0$, failure probability $\delta \in [0, 1]$, maximum trajectory length*

$T > 0$, *maximum distance from a state to* S_{target} *as* $d_{max} > 0$, *and discount factor* $\gamma \in [0, 1]$, *we have*

$$\begin{aligned}
&\forall s_0 \in S_{safe} \setminus S_{target}, \Big(\Pr_{\mathcal{A}}[v^*(s_0) - v^\pi(s_0) \leq \epsilon] \geq 1 - \delta \\
&\implies \Pr_{\mathcal{A}}[t^\pi_{rec}(s_0) - t^*_{rec}(s_0) \leq \epsilon_{rec}] \geq 1 - \delta \Big), \\
&\textit{with } \epsilon_{rec} = \log_\gamma \Big(1 - \frac{\epsilon + d_{max} \sum_{t=0}^{T} \gamma^t}{\gamma^T r_{max}}\Big).
\end{aligned} \tag{12}$$

Proof. From Lemma 1 we know $t^\pi_{rec}(s_0) \geq t^*_{rec}(s_0)$ on all $s_0 \in S_{safe} \setminus S_{target}$, so let $t^\pi_{rec}(s_0) = t^*_{rec}(s_0) + \Delta t$, with $\Delta t \geq 0$. We have

$$\begin{aligned}
&v^*(s_0) - v^\pi(s_0) \leq \epsilon \\
&\implies (\gamma^{t^*_{rec}(s_0)} r_{max} + \sum_{t=0}^{t^*_{rec}(s_0)-1} \gamma^t r^*_t) \\
&\qquad - (\gamma^{t^*_{rec}(s_0)+\Delta t} r_{max} + \sum_{t=0}^{t^*_{rec}(s_0)+\Delta t-1} \gamma^t r^\pi_t) \leq \epsilon \\
&\implies \gamma^{t^*_{rec}(s_0)}(1 - \gamma^{\Delta t}) r_{max} \leq \epsilon + \underbrace{\sum_{t=0}^{t^*_{rec}(s_0)+\Delta t-1} \gamma^t r^\pi_t}_{\leq 0} \\
&\qquad - \underbrace{\sum_{t=0}^{t^*_{rec}(s_0)-1} \gamma^t r^*_t}_{\geq -d_{max} \sum_{t=0}^{T} \gamma^t} \\
&\implies \gamma^{t_{rec}(s_0)^*}(1 - \gamma^{\Delta t}) r_{max} \leq \epsilon + d_{max} \sum_{t=0}^{T} \gamma^t \\
&\implies \Delta t \leq \log_\gamma \Big(1 - \frac{\epsilon + d_{max} \sum_{t=0}^{T} \gamma^t}{\gamma^{t^*_{rec}(s_0)} r_{max}}\Big) \\
&\qquad \leq \underbrace{\log_\gamma \Big(1 - \frac{\epsilon + d_{max} \sum_{t=0}^{T} \gamma^t}{\gamma^T r_{max}}\Big)}_{\epsilon_{rec}}.
\end{aligned} \tag{13}$$

Therefore, the event $\Delta t \leq \epsilon_{rec}$ must occur with probability at least that of $v^*(s_0) - v^\pi(s_0) \leq \epsilon$, i.e., $\geq 1 - \delta$. □

The idea behind Lemma 2 is that when the positive reward r_{max} received upon reaching is large, a small delay Δt will lead to r_{max} being discounted, causing a large difference in value. Therefore, to maintain a small gap in value, we need to maintain a small gap in reaching time. Based on the connection established by Lemma 1, Lemma 2 maps the PAC bound on value to the PAC bound on reaching time.

We can plug in some numbers to check the tightness of ϵ_{rec}. For example, when $\epsilon = 0.1$, $\gamma = 0.95$, $T = 100$ and $d_{max} = 10$, we have $r_{max} \geq 33590.8$ by Equation (9). If we choose $r_{max} = 10^5$, then $\epsilon_{rec} = 7.99$. This means we have a high probability of the reaching time is no more than 7.99 steps from the shortest time possible.

B. Main Algorithm with Safety Margin

With Lemma 2, any model-free PAC-MDP algorithm on continuous state space can achieve PAC guarantee in time optimality when equipped with our reward. We choose DGPQ [17], which is one of the few algorithms that satisfy the above criteria. The training algorithm is modified with the following differences.

1) The reward function is modified to Equation (9).
2) Any training episode terminates upon S_{target} is reached or after T steps.
3) Upon exploration and inference, we encourage safety with a safety margin. That is, if we enter a state within a distance $d_{safe} \geq 0$ to S_{unsafe}, the task is aborted.

That is, safety constraint is encouraged by terminating the trajectory upon reaching within a margin of S_{unsafe}. Termination provides clear feedback about the consequences of the agent's actions. If the agent's policy leads to frequent terminations due to collisions, the agent will quickly learn to recognize and avoid states and actions that increase the risk of collisions. This feedback loop is crucial for refining the agent's policy to improve performance

With the modified SPT-DGPQ, we have our main theorem.

Theorem 1 (PAC Time-Optimality of SPT-DGPQ)**.** *With our algorithm SPT-DGPQ, we achieve PAC time-optimality is Definition 3 if we have maximum tolerable trajectory length* $T > 0$ *and maximum distance to* S_{target} *as* $d_{max} > 0$.

Proof. As mentioned in Section III-B, DGPQ is PAC-MDP and satisfies

$$\forall s_0 \in S_{safe} \setminus S_{target}, \Pr_{\mathcal{A}}[v^*(s_0) - v^\pi(s_0) \leq \epsilon] \geq 1 - \delta, \tag{14}$$

with $\epsilon \geq 0$ and $\delta \in [0, 1]$, achievable with number of training samples $m = \text{poly}(\mathcal{N}_S, 1/\epsilon, 1/\delta, 1/(1-\gamma))$. Therefore, by Lemma 2, we have

$$\forall s_0 \in S_{safe} \setminus S_{target}, \Pr_{\mathcal{A}}[t^\pi_{rec}(s_0) - t^*_{rec}(s_0) \leq \epsilon_{rec}] \geq 1 - \delta, \tag{15}$$

with $\epsilon_{rec} = \log_\gamma \Big(1 - \frac{\epsilon + d_{max} \sum_{t=0}^{T} \gamma^t}{\gamma^T r_{max}}\Big)$ achievable in m training samples. □

Theorem 1 states the probability that our algorithm's performance is close to the optimal performance is very high. This is achievable with a certain number of training samples, which depends on various factors like the number of states, the accuracy we want, and the confidence level we desire.

Then, Lemma 2 helps translate this value performance into time performance. Specifically, it tells us that the time it takes for our algorithm to reach the target state from any safe state is very close to the optimal time, within a small error margin. This error margin can be achieved with enough training samples.

To be noticed, the time PAC bound is dependent to the maximum of the reward and the ϵ. However, increasing the magnitude of the reward does not necessarily lead to better reaching performance due to various reasons. When the maximum reward is very large, the estimated value of states and

actions can become exaggerated. This can lead to instability in the learning process as the algorithm might become overly optimistic about certain states or actions. Furthermore, large rewards can cause the learning algorithm to converge more slowly. The agent may take longer to learn the optimal policy because the value function updates become larger, causing more fluctuation in the estimates. Additionally, modifying the value of ϵ will also change the number of training examples to achieve the PAC bound in Theorem 1.

Algorithm 1 SPT-DGPQ (single target, *reach&avoid* task)

Input: GP kernel $k(\cdot,\cdot)$, state space S, action space A, STL property φ, dynamics f that is unknown but allows trajectory sampling, target set S_{target}, sampled trajectory length T, discount factor γ, thresholds σ_{tol} and ϵ

Output: Estimated Q-value function as a table $\hat{q}$

1: **for** each action $a \in A$ **do**
2: $GP_a \leftarrow \mathcal{N}(r_{max}/(1-\gamma), k)$
3: Table $\hat{q}_a \leftarrow$ empty table
4: **end for**
5: **while** $m_{train} < m$ **do**
6: $s_0 \leftarrow$ uniform sampling from S
7: $r_{prev} \leftarrow -\infty$
8: **for** $t = 0, \ldots, T$ **do**
9: $a_t \leftarrow \arg\max_{a \in A}(\hat{q}_a(s_t)$ using the upper bound method in [17]
10: $s_{t+1} \leftarrow$ sampling from $f(\cdot|s_t, a_t)$
11: $r_t \leftarrow r(s_t, r_{prev})$ as in Equation (9)
12: $\sigma_1^2 \leftarrow$ variance of $GP_{a_t}(s_t)$
13: **if** $\sigma_1^2 > \sigma_{tol}^2$ **then**
14: GP_{a_t}.fit$(s_t, r_t + \gamma \hat{q}_{a_t}(s_{t+1}))$
15: **end if**
16: $\sigma_2^2 \leftarrow$ variance of $GP_{a_t}(s_t)$
17: **if** $\sigma_2^2 \leq \sigma_{tol}^2 < \sigma_1^2$ and $\hat{q}_{a_t}(s_t) -$ mean of $GP_{a_t}(s_t) > \epsilon$ **then**
18: Update table entry $\hat{q}_{a_t}(s_t)$ to $GP_{a_t}(s_t) + \epsilon/2$
19: Reinitialize GP_{a_t} to $\mathcal{N}(\hat{q}_{a_t}, k)$
20: **end if**
21: $r_{prev} \leftarrow r_t$
22: $m_{train} \leftarrow m_{train} + 1$
23: **end for**
24: **end while**
25: Concatenate tables by $\hat{q} \leftarrow \{\hat{q}_a \mid a \in A\}$

Delay Gaussian Process Q-Learning (DGPQ) is an algorithm that combines delay Q-Learning with Gaussian processes to address the challenges of model-free learning and maintain the PAC-MDP property when the state-space is continuous. DGPQ maintains two representations of the estimated Q value function: a set of tables $\hat{q}_a$ and a set of GPs GP_a for all $a \in A$. The idea is to only cache values of converged GPs into the table. The control policy is therefore taking the action that maximizes the estimated Q value at each state.

As shown in Algorithm 1, we extend DGPQ into SPT-DGPQ, which produces a control policy with PAC bound on time-optimality. From lines 1 to 4, the final Q table is initialized to empty, and the intermediate GPs are initialized to Gaussians the same way as in DGPQ. Then, from lines 5 to 24, we train the GPs as well as the Q table. At line 7 and 11, we generate and cache our designed reward as in Equation (9), which ensures the equivalence between optimality in time and value based on Theorem 1. The update rule of the Q table is the same as in DGPQ. First, the GP of an action is updated when the current Q value at a state has a large variance (from lines 12 to 15). Then, the Q table is updated by caching the value of a GP when the GP converges (line 16 to 20).

VI. Experiments

In this section, we evaluate SPT-DGPQ on various benchmarks and compare it with other baselines. This section is organized as follows: We first introduce the benchmarks that are used for evaluation, then the baselines and settings for comparison. After that, we explain the metrics we use and show the experiment results.

A. Benchmarks

We have done experiments on two benchmarks with linear dynamics: DC motor position and aircraft pitch, as well as one nonlinear dynamics: inverted pendulum.

The DC motor position benchmark involves a comprehensive assessment of the performance characteristics and capabilities of direct current (DC) motors in terms of their ability to accurately and efficiently reach, maintain, and adjust to specific positions. This benchmark is particularly relevant in fields like robotics, automated control systems, and precision machinery, where exact motor positioning is crucial. Key aspects of this benchmark include evaluating the positional accuracy of the motor, which is the measure of how precisely and how efficiently the motor can achieve a given position, often quantified as the deviation from the target position. DC motor position benchmark uses the current as control input to control the system to drive the motor angle to a target position, for more details please refer to [50].

The aircraft pitch benchmark is governed by longitudinal dynamics. We assume that the aircraft maintains a steady cruising state at a fixed altitude and velocity, ensuring that the forces of thrust, drag, weight, and lift are in equilibrium both horizontally (x-direction) and vertically (y-direction). Additionally, we assume that variations in the aircraft's pitch angle will not affect its speed under any conditions. We want to control the angle of attack using the elevator deflection angle as control input, for more details please refer to [51].

The inverted pendulum swingup benchmark consists of a pendulum attached at one end to a fixed point, and the other end being free. The goal is to make the system balance the free end at an upright position, with its center of gravity right above the fixed point. The control input of this benchmark is the torque which is binary, it is either clockwise or counterclockwise. For more details, please refer to the documentation on the official website of Openai Gym [52] and its implementation.

B. Baselines

We have implemented two baselines to compare with SPT-DGPQ:

1) φ-reward deploys standard quantitative semantics as reward function for training, the details of the quantitative semantics of STL can be found in [53]. Additionally, we apply the dense rewards settings introduced in [32] to make the training more efficient. STL control synthesis aims to ensure that an agent satisfies temporal logic formulas efficiently and effectively. By leveraging optimization techniques and robustness metrics, it is possible to encourage the agent to meet these specifications in a short time,
2) d-reward deploys heuristic distance-based reward function for training. In the baseline, the reward function is crafted from a heuristic combination of two distinct reward elements. The former is designed to encourage the agent's movement toward the target, and the latter aims to discourage proximity to obstacles. The format is the same as the settings in [33]. These two elements are merged into a single reward via a linear combination, with each element weighted with a coefficient. To make a fair comparison with other methods, these coefficients have been left untuned.

These two baselines are trained with the DGPQ method for a fair comparison to SPT-DGPQ.

C. Settings

1) **Benchmark Settings:** The benchmark settings include the settings for the initial set, the target set, the obstacle position and other settings related to the benchmark. The initial set of 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 maximum time tolerance is 100 control steps. The initial set of the aircraft pitch benchmark is $[-1, 0, 0]$ to $[1, 0, 0]$, the target set is a ball with a radius 0.5 centered at $[0, 0, 0]$, the unsafe set is set as a ball with a radius 0.1 centered at $[0.7, 0, 0]$, the time step of this benchmark is 0.05 seconds and the maximum time tolerance is 30 control steps. The initial set of the inverted pendulum is $[sin(\pi/4), 0, 0]$, the target set is a ball with a radius 0.5 centered at $[1, 0, 0]$ which is the balance position, the unsafe set is set as a ball with a radius 0.1 centered at $[0, 1, 3]$, the time step of this benchmark is 0.05 seconds and the maximum time tolerance is 20 control steps.
2) **Training Settings:** The training settings include the parameters of DGPQ and other baselines. The training steps for the three benchmarks are set to 40,000 control steps, the delta tolerance parameter of DGPQ is 0.5, and the epsilion of DGPQ is 0.05. The action of the DC motor position benchmark is discretized to $\{-10, -5, -2, 2, 5, 10\}$, the action of the aircraft pitch benchmark is discretized to$\{-10, 0, 10\}$, the action of the inverted pendulum is $\{-1, 1\}$, the discount factor of the value function is set to 0.95.

The target sets and unsafe sets are set to ball shape in the experiments to simplify the distance calculation. They can also be set to rectangles like in Fig. 1 or other shapes as users prefer as long as we can get the distance between the state to the sets to measure the degree of the robustness of satisfying these STL formulas. The experiments are deployed on a PC with Intel(R) Core(TM) i7-10700KF CPU @ 3.80 GHz, a Nvidia GeForce GTX 3080 GPU and 64 GB RAM and a laptop with Intel(R) Core(TM) i7-13700HX @ 2.10 GHz, Nvidia GeForce RTX 4060 GPU and 32 GB RAM.

For the training and the implementation details of DGPQ, we use the same settings from [17].

D. Metrics

We evaluate our experimental results on the following metrics.

1) **Success rate**. This is the percentage of the testing trajectories that successfully reach the predefined target set within maximum tolerable time without touching the unsafe region (obstacles).
2) **Reaching time.** This is the average reaching time of all testing trajectories. If a trajectory fails to reach due to collision into obstacles or take time longer than maximum tolerable time, we assign it with reaching time = maximum tolerable time + 1.
3) **Collision rate.** This is the percentage of the testing trajectories that failed due to touching the unsafe region, i.e., colliding into the obstacles.
4) **Stable rate (for inverted pendulum only).** For inverted pendulum, we are also interested in whether we are able to stay within a stable set of states after reaching the target. Therefore, we record the percentage of testing trajectories that not only reach the target but also stay stable.

E. Results

Table **??** shows the main results of SPT-DGPQ performance on the three benchmarks compared with the two reward baselines. There are three main observations from this table:

1) SPT-DGPQ has the highest success rate on three benchmarks compared with the other two baselines. SPT-DGPQ achieves 85.2% success rate on the DC motor position benchmark, 74.7% success rate on the aircraft pitch benchmark and 97.7% success rate on the inverted pendulum benchmark. In comparison, φ-reward achieves 85.0% success rate on the DC motor position benchmark, 68.5% success rate on aircraft pitch benchmark and 90.8% success rate on the inverted pendulum benchmark. d-reward achieves 72.7% success rate on the DC motor position benchmark, 69.3% success rate on aircraft pitch benchmark and 92.3% success rate on the inverted pendulum benchmark. Since SPT-SGPQ

	DC Motor Position			Aircraft Pitch			Inverted Pendulum			
	Success	Collision	Time	Success	Collision	Time	Success	Collision	Stable	Time
φ-reward	85.0	5.2	37.06	68.5	12.1	8.20	90.8	0	15.5	4.33
d-reward	72.7	3.2	50.58	69.3	9.1	6.97	92.3	0	18.3	3.88
SPT-DGPQ	**85.2**	**0.9**	**36.63**	**74.7**	**12.2**	4.30	**97.7**	0	**24.4**	**2.89**

TABLE II: Performance of SPT-DGPQ and baselines. The success rate and collision rate are in percentage, the unit of reach time is number of time steps.

has a reward mechanism that guarantees the PAC time-optimality by design, therefore, SPT-SGPQ will have a higher chance to reach the target set before the maximum time tolerance.

2) SPT-DGPQ has the shortest reach time on three benchmarks compared with the other two baselines. SPT-DGPQ needs 36.63 time steps on average to reach the target set on the DC motor position benchmark, 4.30 time steps on average to reach the target set on the aircraft pitch benchmark and 2.89 time steps on average to reach the target set on the inverted pendulum benchmark. In comparison, φ-reward needs 37.06 time steps on average to reach the target set on the DC motor position benchmark, 8.20 time steps on average to reach the target set on the aircraft pitch benchmark and 4.33 time steps on average to reach the target set on the inverted pendulum benchmark. d-reward needs 50.58 time steps on average to reach the target set on the DC motor position benchmark, 6.97 time steps on average to reach the target set on the aircraft pitch benchmark and 3.88 time steps on average to reach the target set on the inverted pendulum benchmark. Similar to the above insights, SPT-DGPQ has encouraged the policy to achieve PAC time-optimality, therefore, it will have short reach time compared to baselines.
3) Sometimes SPT-DGPQ can have a slightly higher collision rate compared to the two baselines. We notice SPT-DGPQ has 12.2% collision rate on aircraft pitch benchmark, in comparison, φ-reward has 12.1% collision rate and d-reward has 9.1% collision rate. This is due to that the policy choose to take the risk of collision but intends to achieve a shorter reach time and a higher success rate since safety is encouraged but not enforced. Moreover, this does not conflict with Theorem 1 since there is no explicit guarantee on the collision probability.

There are also some minor observations from this table. There is an additional metric for the inverted pendulum benchmark which is the stay rate. This metric is evaluating whether the system can stay in the target set after reaching it within the maximum time tolerance. SPT-DGPQ achieves highest stay rate compared with baselines. However, since the focus of our work is reachability control instead of stability control, the stay rate is not explicitly encouraged during training, and the stay rate is not as high as reach rate. In the future, we plan to make the control policy that satisfies fast reaching and stability at the same time.

The reaching success rate of φ-reward is fairly close to the reach rate of SPT-DGPQ. Since the only difference between SPT-DGPQ and the φ-reward baseline is the reward shaping after reaching, therefore, it is possible for them to have similar performance on some tasks, but SPT-DGPQ has a higher reach rate compared to the φ-reward baseline.

F. Case Study: PAC Time-Optimality Validation

In this subsection, we will implement experiments on the DC Motor Position benchmark with some adaption to validate the PAC time-optimality of SPT-DGPQ.

We have to compare the reach time of SPT-DGPQ with the reach time of the optimal policy on the DC Motor Position benchmark. However, it is challenging to make this comparison since it is hard to find the optimal policy that takes the shortest time to reach the target. Although the system is linear and the action space is discrete, it is challenging to search for a time-optimal policy.

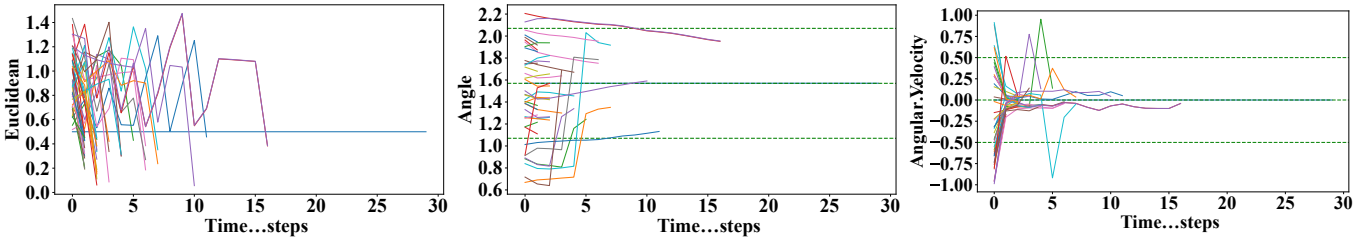
To tackle with this problem, we make a compromise to find the time-optimal policy for the set of test cases by traversing the combinations of the actions over the time horizon. Unfortunately, the complexity of traversing is exponential to the length of the time horizon. For example, if there are 4 actions for the benchmark, and the time horizon is 20, we have to traverse $4^{20} \approx 1.09 * 10^{12}$ number of action combinations to find the time-optimal control from a single initial point.

Therefore, we have to find efficient ways that compress the size of search space to reduce computational cost. An intuitive solution to compress the search space by reducing the action space and the time horizon. For this case study, we reduce the maximum time tolerance to 30, and we adapt the action space to $\{-2, -1, 1, 2\}$. Furthermore, the search progress can be accelerated by referring the solutions of SPT-DGPQ. For example, assuming at one arbitrary initial point, the control generated by SPT-DGPQ takes 12 time steps to reach the target, we can search all the combinations of the policies that take at most 11 time steps. In this case, we only have to traverse $4^{11} \approx 4.19 * 10^6$ action combinations which significantly reduces the search space. This process can be further accelerated if we dynamically change the reference to the current best solution instead of the solution generated by SPT-SGPQ. For some special cases, such as linear systems with binary control inputs, the time-optimal solution can be obtained by optimization[4].

Fig. 2 shows qualitative results of testing trajectories for the adapted DC Motor Position benchmark. We have included the history of Euclidean distance, angle, and angular velocity of 40 trajectories that reach the target by the maximum time tolerance from random initial points in the initial set.

There are three main observations from Fig. 2:

1) Euclidean distance varies over time for different trajectories. The distances fluctuate significantly and are not in a



(a) Euclidean Distance (b) Angle (c) Angular Velocity

Fig. 2: Qualitative results of testing. Each curve represents a trajectory from a random initial point in the initial set.

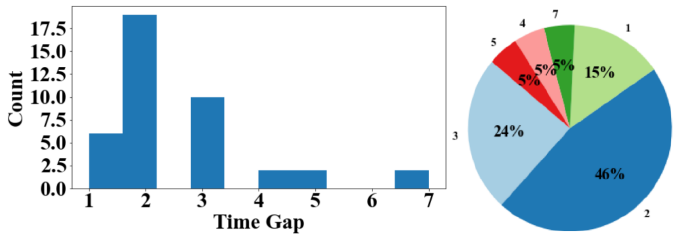


Fig. 3: Time gaps on 40 testing initial points from the initial set. The left is a histogram chart showing the distribution, the right is a pie chart showing detailed percentage of occurrences.

monopoly pattern. This does not conflict with the time-optimality of SPT-DGPQ since the system dynamics and the control strategies employed will jointly determine the state transitions. The observed non-monotonic Euclidean distances imply that the control strategy is adaptive to the randomized initial points, responding to real-time feedback from the system. This adaptability is crucial for achieving time-optimality when the initial point is not static.

2) Most angles of the trajectories stabilized in a short time. Initially, the angles vary widely, but after around 5 time steps, most trajectories tend to stabilize. This rapid stabilization highlights the efficiency and responsiveness of the SPT-DGPQ approach. There are a few trajectories with noticeable deviations, indicating some variability in the angles over time. This observation partially validates the probabilistic guarantee of the time-optimality of SPT-DGPQ. The initial wide variability in angles suggests that the system undergoes rapid adjustments in the early stages. These adjustments reflect the system's response to the control inputs aimed at steering it toward the desired state.
3) Most angular velocities of the trajectories reduce to zero in a short time. This is due to the design of reward function in Equation 9 since the optimal policy is reach the target in the shortest time and stay in it. The rapid reduction of angular velocities to zero ensures that the system remains stable at the target state. This stability is crucial for applications requiring precise positioning or steady-state maintenance.

Then we compare the control generated by SPT-DGPQ and the time-optimal control policy.

By Theorem 1, the time gap should be less than $\epsilon_{rec} = \log_\gamma \left(1 - \frac{\epsilon + d_{max} \sum_{t=0}^{T} \gamma^t}{\gamma^T r_{max}}\right)$, the maximum reward is set to 20 and $d_{max} = 1.5$, then $\epsilon_{rec} \approx 1.2$, this should be rounded to 2 to make an overapproximation without violating Theorem 1.

Fig. 3 shows the time gap between SPT-DGPQ and the time-optimal control policy by traversal. We use a histogram chart and a pie chart to show the distribution of the time gap. There are two observations:

1) The minimum time gap is one time step and the maximum time gap is seven time steps. From the histogram chart We can see in most cases, the time gap is less or equal to two time steps, only a few cases that the time gap is greater than three time steps.
2) From the pie chart, we can see 15% of the total occurrences take only one more time step to reach the targets compared with the time-optimal policy, 46% of the total occurrences take two more time steps to reach the targets compared with the time-optimal policy. This observation validates Theorem 1 since there are 61% occurrences take less than or equal to two more time steps to reach the target compared with the time-optimal policy, which is greater than $1 - \delta$.

VII. Conclusion

In conclusion, we propose a safe and PAC time-optimal model-free learning method for reinforcement learning. The results demonstrate that the proposed method, SPT-DGPQ, can efficiently and safely control the system to a pre-defined target set while outperforming baselines on three benchmarks. Additionally, SPT-DGPQ offers a probabilistic guarantee on time-optimality, which underscores its robustness and effectiveness in various scenarios. In the future, we are interested in designing new methods that can achieve stability alongside the PAC time-optimality, thus addressing a broader range of practical applications. Another compelling direction for future research is to extend SPT-DGPQ to multi-agent scenarios, which could open up new possibilities for collaborative and competitive environments in reinforcement learning.

VIII. Acknowledgment

This work was supported in part by NSF CNS-2333980. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the National Science Foundation (NSF).

REFERENCES

[1] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano, *et al.*, "The current state and future outlook of rescue robotics," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.

[2] A. H. Reddy, B. Kalyan, and C. S. Murthy, "Mine rescue robot system–a review," *Procedia Earth and Planetary Science*, vol. 11, pp. 457–462, 2015.

[3] M. D. Roy, S. S. Sana, and K. Chaudhuri, "An optimal shipment strategy for imperfect items in a stock-out situation," *Mathematical and Computer Modelling*, vol. 54, no. 9-10, pp. 2528–2543, 2011.

[4] J. LaSalle, "Time optimal control systems," *Proceedings of the National Academy of Sciences*, vol. 45, no. 4, pp. 573–577, 1959.

[5] J. P. LaSalle *et al.*, "The time optimal control problem," *Contributions to the theory of nonlinear oscillations*, vol. 5, pp. 1–24, 2016.

[6] 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.

[7] 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)*, pp. 209–222, IEEE, 2023.

[8] F. A. T. Abad, R. Mancuso, S. Bak, O. Dantsker, and M. Caccamo, "Reset-based recovery for real-time cyber-physical systems with temporal safety constraints," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, 2016.

[9] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[10] W. Zhu and M. Hayashibe, "A hierarchical deep reinforcement learning framework with high efficiency and generalization for fast and safe navigation," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 5, pp. 4962–4971, 2022.

[11] 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*, pp. 881–888, 2006.

[12] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[13] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[14] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[15] S. G. Khan, G. Herrmann, F. L. Lewis, T. Pipe, and C. Melhuish, "Reinforcement learning and optimal adaptive control: An overview and implementation examples," *Annual reviews in control*, vol. 36, no. 1, pp. 42–59, 2012.

[16] S. Padakandla, "A survey of reinforcement learning algorithms for dynamically varying environments," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–25, 2021.

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

[18] L. Van den Broeck, M. Diehl, and J. Swevers, "Time optimal mpc for mechatronic applications," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 8040–8045, IEEE, 2009.

[19] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear mpc in real-time," in *53rd IEEE conference on decision and control*, pp. 2505–2510, IEEE, 2014.

[20] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

[21] N. Khaneja, R. Brockett, and S. J. Glaser, "Time optimal control in spin systems," *Physical Review A*, vol. 63, no. 3, p. 032308, 2001.

[22] D. Q. Mayne and W. Schroeder, "Robust time-optimal control of constrained linear systems," *Automatica*, vol. 33, no. 12, pp. 2103–2118, 1997.

[23] F. Fei, Z. Tu, D. Xu, and X. Deng, "Learn-to-recover: Retrofitting uavs with reinforcement learning-assisted flight control under cyber-physical attacks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7358–7364, 2020.

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

[25] S. Çalışır and M. K. Pehlivanoğlu, "Model-free reinforcement learning algorithms: A survey," in *2019 27th signal processing and communications applications conference (SIU)*, pp. 1–4, IEEE, 2019.

[26] N. Jiang, "Pac reinforcement learning with an imperfect model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[27] 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.

[28] A. Krishnamurthy, A. Agarwal, and J. Langford, "Pac reinforcement learning with rich observations," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[29] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pp. 152–166, Springer, 2004.

[30] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[31] Y. Gilpin, V. Kurtz, and H. Lin, "A smooth robustness measure of signal temporal logic for symbolic control," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 241–246, 2020.

[32] 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*, pp. 190–206, Springer, 2022.

[33] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[34] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[35] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[36] J. Clifton and E. Laber, "Q-learning: Theory and applications," *Annual Review of Statistics and Its Application*, vol. 7, pp. 279–301, 2020.

[37] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE access*, vol. 7, pp. 133653–133667, 2019.

[38] H. Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, 2010.

[39] A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A deterministic improved q-learning for path planning of a mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1141–1153, 2013.

[40] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is q-learning provably efficient?," *Advances in neural information processing systems*, vol. 31, 2018.

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

[42] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proceedings of the AAAI conference on artificial intelligence*, 2016.

[43] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[44] M. Fortunato, M. Gheshlaghi Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Deep reinforcement learning with double q-learning," in *Proceedings of the International Conference of Learning Representations*, 2018.

[45] M. Kuss and C. Rasmussen, "Gaussian processes in reinforcement learning," *Advances in neural information processing systems*, vol. 16, 2003.

[46] Y. Engel, S. Mannor, and R. Meir, "Reinforcement learning with gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*, pp. 201–208, 2005.
[47] J. J. Chung, N. R. Lawrance, and S. Sukkarieh, "Gaussian processes for informative exploration in reinforcement learning," in *2013 IEEE international conference on robotics and automation*, pp. 2633–2639, IEEE, 2013.
[48] Y. Guo, R. Jena, D. Hughes, M. Lewis, and K. Sycara, "Transfer learning for human navigation and triage strategies prediction in a simulated urban search and rescue task," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pp. 784–791, IEEE, 2021.
[49] D. S. Graça, J. Buescu, and M. L. Campagnolo, "Boundedness of the domain of definition is undecidable for polynomial odes," *Electronic Notes in Theoretical Computer Science*, vol. 202, pp. 49–57, 2008.
[50] L. Zhang, X. Chen, F. Kong, and A. A. Cardenas, "Real-time recovery for cyber-physical systems using linear approximations," in *41st IEEE Real-Time Systems Symposium (RTSS)*, IEEE, 2020.
[51] F. L. Lewis, "Aircraft control and simulation, brian l. stevens,"
[52] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
[53] 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*, pp. 92–106, Springer, 2010.