Modular Deep Reinforcement Learning for Continuous Motion Planning With Temporal Logic

Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan

***Abstract*—This letter investigates the motion planning of autonomous dynamical systems modeled by Markov decision processes (MDP) with unknown transition probabilities over continuous state and action spaces. Linear temporal logic (LTL) is used to specify high-level tasks over infinite horizon, which can be converted into a limit deterministic generalized Büchi automaton (LDGBA) with several accepting sets. The novelty is to design an embedded product MDP (EP-MDP) between the LDGBA and the MDP by incorporating a synchronous tracking-frontier function to record unvisited accepting sets of the automaton, and to facilitate the satisfaction of the accepting conditions. The proposed LDGBA-based reward shaping and discounting schemes for the model-free reinforcement learning (RL) only depend on the EP-MDP states and can overcome the issues of sparse rewards. Rigorous analysis shows that any RL method that optimizes the expected discounted return is guaranteed to find an optimal policy whose traces maximize the satisfaction probability. A modular deep deterministic policy gradient (DDPG) is then developed to generate such policies over continuous state and action spaces. The performance of our framework is evaluated via an array of OpenAI gym environments.**

***Index Terms*—Deep Reinforcement Learning, Linear Temporal Logic, Motion Planning.**

I. Introduction

THE goal of motion planning is to generate valid configurations such that robotic systems can complete pre-specified tasks. Markov decision processes (MDPs) are often employed to model uncertainties of dynamic systems. Growing research has been devoted to studying the motion planning modelled as an MDP when the transition probabilities are initially unknown. Reinforcement learning (RL) is a sequential decision-making process that learns optimal action policies for an unknown MDP via gathering experience samples from the MDP [1]. RL has achieved impressive results over the past few years, but often the learned solution is difficult to understand and examine by humans. Two main challenges existing in many RL applications are: (i) the design of an appropriate reward shaping mechanism to ensure correct mission specification, and (ii) the increasing sample complexity when considering continuous state and action spaces.

Temporal logics offer rich expressivity in describing complex tasks beyond traditional go-to-goal navigation for robotic systems [2]. Specifically, motion planning under linear temporal logic (LTL) constraints attracted growing research attention in the past few years [3], [4]. Under the assumption of full knowledge of the MDP model, one common objective is to maximize the probability of accomplishing the given LTL task [5]–[7]. Once this assumption is relaxed, model-based RL is employed [8]–[10] by treating LTL specifications as reward shaping schemes to generate policies that satisfy LTL tasks by explicitly learning unknown transition probabilities of the MDP. This means that a model of the MDP is inferred over which an optimal policy is synthesized. However, scalability is a pressing issue for applying model-based approaches due to the need to store the learned model. On the other hand, by relaxing the need to construct an MDP model, model-free RL is recently adopted where appropriate reward shaping schemes are proposed [11]–[15]. Signal temporal logic is also proposed in [16] where a task-guided reward function is proposed based on the robustness degrees. Despite the recent progresses, the aforementioned works can not handle many real-world applications that perform in high-dimensional continuous state and action spaces. In a pioneer work [17], Deep Q Network (DQN) addressed high-dimensional state space and is capable of human-level performance on many Atari video games. However, DQN can only handle discrete and low-dimensional action spaces. By leveraging actor-critic methods, deep networks, and the policy gradient methods, deep deterministic policy gradient (DDPG) was proposed to approximate optimal policies over a continuous action space to improve the learning performance [18].

In this letter, we consider motion planning under LTL task specifications in continuous state and action spaces when the MDP is fully unknown. An unsupervised one-shot and on-the-fly DDPG-based motion planning framework is developed to learn the state of an underlying structure without explicitly constructing the model. The high-level LTL task over infinite horizon is converted to a limit deterministic generalized Büchi automaton (LDGBA) [19] acting as task-guided reward shaping scheme and decomposing complex tasks into low-level and achievable modules.

Related works: When considering deep RL with formal methods, deterministic finite automata (DFA) were applied as

Manuscript received February 23, 2021; accepted July 15, 2021. Date of publication August 4, 2021; date of current version August 20, 2021. This letter was recommended for publication by Associate Editor K. N. Kaipa and Editor L. Pallottino upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant U2013601. *(Corresponding author: Zhen Kan.)*

Mingyu Cai and Shaoping Xiao are with the Department of Mechanical Engineering, University of Iowa, Iowa City 52242, IA USA (e-mail: mingyu-cai@uiowa.edu; shaoping-xiao@uiowa.edu).

Mohammadhosein Hasanbeig and Alessandro Abate are with the Department of Computer Science, University of Oxford, Oxford OX1 2JD, U.K. (e-mail: hosein.hasanbeig@icloud.com; a.abate@tudelft.nl).

Zhen Kan is with the Department of Automation, University of Science and Technology of China, Hefei 230026, Anhui, China (e-mail: zkan@ustc.edu.cn).

Digital Object Identifier 10.1109/LRA.2021.3101544

reward machines in [20], [21]. In [22], a truncated linear temporal logic was considered and its robustness degree was used as the reward signal to facilitate learning. However, [20]–[22] only consider tasks over finite horizons. In contrast, this work extends previous research to tasks over the infinite horizon, where finite horizon motion planning can be regarded as a special case of the infinite horizon setting. Along this line of research, the most relevant works include [23]–[26]. In [23], LTL constraints were translated to Deterministic Rabin Automata (DRA), which might fail to find policies that maximize LTL satisfaction probability [13]. The work in [26] proposed a binary vector to record the visited accepting sets of LDGBA and designed a varying reward function to improve the satisfaction of LTL specifications. However, the maximum probability of task satisfaction cannot be guaranteed and the standard DDPG algorithm cannot distinguish the sub-task module in [26], resulting in an unsatisfactory success rate. Modular DDPG that jointly optimizes LTL sub-policies was first introduced in [24], [25]. However, [24], [25] do not explicitly record visited or unvisited accepting sets of LDGBA in each round of repetitive pattern over the infinite horizon, which might be essential for synthesising a deterministic policy [15].

Contributions: The contributions of this work are multifold. In contrast to most existing works that consider either a discrete state space or a discrete action space, this paper proposes a modular DDPG architecture integrated with potential functions [27]. This allows our method to generate optimal policies that efficiently solve LTL motion planning of an unknown MDP over continuous state and action spaces. The novelty is to construct an embedded product MDP (EP-MDP) to record unvisited accepting sets of the automaton at each round of the repeated visiting pattern, by introducing a tracking-frontier function. Such a design ensures task completion by encouraging the satisfaction of LDGBA accepting conditions. To facilitate learning of optimal policies, the designed reward is enhanced with potential functions that effectively guide the agent toward task satisfaction without adding extra hyper-parameters to the algorithm. Unlike [15], rigorous analysis shows that the maximum probability of task satisfaction can be guaranteed. Compared to approaches based on limit deterministic Büchi automata (LDBA), e.g., [13], [14], LDGBA has several accepting sets while LDBA only has one accepting set which can result in sparse rewards during training. In summary, our approach can find the optimal policy in continuous state and action spaces to satisfy LTL specifications over infinite horizon with maximum probability.

II. Preliminaries

A. Continuous Labeled MDP and Reinforcement Learning

A continuous labeled MDP is a tuple $\mathcal{M} = (S, A, p_S, \Pi, L, \Lambda)$, where $S \subseteq \mathbb{R}^n$ is a continuous state space, $A \subseteq \mathbb{R}^m$ is a continuous action space, Π is a set of atomic propositions, $L : S \to 2^\Pi$ is a labeling function, and $p_S : \mathfrak{B}(\mathbb{R}^n) \times A \times S \to [0, 1]$ is a Borel-measurable conditional transition kernel such that $p_S(\cdot|s, a)$ is a probability measure of $s \in S$ and $a \in A$ over the Borel space $(\mathbb{R}^n, \mathfrak{B}(\mathbb{R}^n))$, where $\mathfrak{B}(\mathbb{R}^n)$ is the set of all Borel sets on $\mathbb{R}^n$. The transition probability p_S captures the motion uncertainties of the agent. It is assumed that p_S is not known *a priori*, and the agent can only observe its current state and the associated label of the current state.

A deterministic policy $\boldsymbol{\xi}$ of continuous MDP is a function $\boldsymbol{\xi} : S \to A$ that maps each state to an action over the action space A. The MDP $\mathcal{M}$ evolves by taking an action ξ_i at each stage i, and thus the control policy $\boldsymbol{\xi} = \xi_0\xi_1 \ldots$ is a sequence of actions, which yields a path $\boldsymbol{s} = s_0s_1s_2 \ldots$ over $\mathcal{M}$ with the transition $s_i \xrightarrow{a_i} s_{i+1}$ exists, i.e., s_{i+1} belongs to the smallest Borel set B such that $p_S(B|s_i, a_i) = 1$. The control policy $\boldsymbol{\xi}$ is memoryless if each ξ_i only depends on its current state, and $\boldsymbol{\xi}$ is a finite memory policy if ξ_i depends on its past history.

Let $\Lambda : S \times A \times S \to \mathbb{R}$ denote a reward function. Given a discounting function $\gamma : S \times A \times S \to \mathbb{R}$, the expected discounted return under policy $\boldsymbol{\xi}$ starting from $s \in S$ is

$$U^{\boldsymbol{\xi}}(s) = \mathbb{E}^{\boldsymbol{\xi}}\left[\sum_{i=0}^{\infty} \gamma^i (s_i, a_i, s_{i+1}) \cdot \Lambda(s_i, a_i, s_{i+1}) \,|s_0 = s\right].$$

An optimal policy $\boldsymbol{\xi}^*$ maximizes the expected return for each state $s \in S$, i.e., $\boldsymbol{\xi}^* = \arg\max_{\boldsymbol{\xi}} U^{\boldsymbol{\xi}}(s)$. If the MDP is not fully known, but the state and action spaces are countably finite, tabular approaches are usually employed [28]. However, traditional tabular RL methods are not applicable to MDPs with continuous state and action spaces. In this work, we propose a policy gradient method that relies on deep neural networks to parameterize the policy model.

B. LTL and Limit-Deterministic Generalized Büchi Automaton

Linear temporal logic is a formal language that is widely used to describe complex mission tasks. Detailed descriptions of the syntax and semantics of LTL can be found in [2]. Given an LTL specification, its satisfaction can be evaluated by an LDGBA [19]. An LDGBA is a sub-class of generalized Büchi automata (GBA) that can express the set of words of an LTL formula.

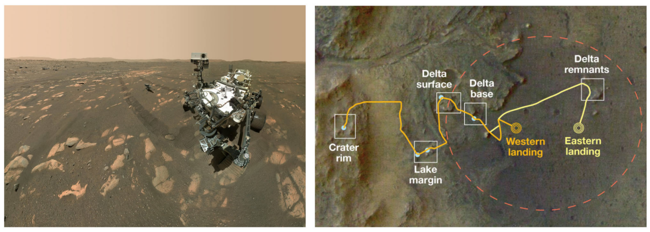
Definition 1: A GBA is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, $\Sigma = 2^\Pi$ is a finite alphabet, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F = \{F_1, F_2, \ldots, F_f\}$ is the set of accepting sets where $F_i \subseteq Q, \forall i \in \{1, \ldots f\}$.

Denote by $\boldsymbol{q} = q_0q_1 \ldots$ a run of a GBA, where $q_i \in Q$, $i = 0, 1, \ldots$. The run $\boldsymbol{q}$ is accepted by the GBA, if it satisfies the generalized Büchi acceptance condition, i.e., $\inf(\boldsymbol{q}) \cap F_i \neq \emptyset$, $\forall i \in \{1, \ldots f\}$, where $\inf(\boldsymbol{q})$ denotes the infinite part of $\boldsymbol{q}$.

Definition 2: A GBA is an LDGBA if the transition function δ is extended to $Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$, and the state set Q is partitioned into a deterministic set Q_D and a non-deterministic set Q_N, i.e., $Q_D \cup Q_N = Q$ and $Q_D \cap Q_N = \emptyset$, where

- the state transitions in Q_D are total and restricted within it, i.e., $\left|\delta(q, \alpha)\right| = 1$ and $\delta(q, \alpha) \subseteq Q_D$ for every state $q \in Q_D$ and $\alpha \in \Sigma$,
- An ϵ-transition is not allowed in the deterministic set, i.e., for any $q \in Q_D$, $\delta(q, \epsilon) = \emptyset$, and
- the accepting states are only in the deterministic set, i.e., $F_i \subseteq Q_D$ for every $F_i \in F$.

In Definition 2, ϵ-transitions are only for state transitions from Q_N to Q_D, without reading an alphabet. In the following



(a) (b)

Fig. 1. Example of Mars exploration (courtesy of NASA).

analysis, we use $\mathcal{A}_\phi$ to denote the LDGBA corresponding to an LTL formula ϕ.

Definition 3: A non-accepting sink component $Q_{sink} \subseteq Q$ of an LDGBA is a strongly connected directed graph induced by a set of states, such that the accepting condition can not be satisfied if starting from any state in Q_{sink}.

III. Problem Statement

Consider a robot that performs a mission described by an LTL formula ϕ. For instance, as shown in Fig. 1, a helicopter (or satellite) provides the map consisting of areas of interest, based on which the ground Mars rover is tasked to visit all regions marked with rectangles. Due to the complex terrain of Mars surface, there exist motion uncertainties affecting the rover movement. The interaction of the robot with the environment is modeled by a continuous MDP $\mathcal{M}$, which can be used to model decision-making problems for general robotic systems. Under a policy $\boldsymbol{\xi} = \xi_0\xi_1\ldots$, the induced path over $\mathcal{M}$ is $\boldsymbol{s}_\infty^{\boldsymbol{\xi}} = s_0 \ldots s_i s_{i+1} \ldots$. Let $L(\boldsymbol{s}_\infty^{\boldsymbol{\xi}}) = l_0 l_1 \ldots$ be the sequence of labels associated with $\boldsymbol{s}_\infty^{\boldsymbol{\xi}}$ such that $l_i \in L(s_i)$. Denote the satisfaction relation by $L(\boldsymbol{s}_\infty^{\boldsymbol{\xi}}) \models \phi$ if the induced trace satisfies ϕ. The probabilistic satisfaction under the policy ξ from an initial state s_0 can then be defined as

$$\Pr{}_M^{\boldsymbol{\xi}}(\phi) = \Pr{}_M^{\boldsymbol{\xi}}\left(L\left(\boldsymbol{s}_\infty^{\boldsymbol{\xi}}\right) \models \phi \middle| \boldsymbol{s}_\infty^{\boldsymbol{\xi}} \in \boldsymbol{S}_\infty^{\boldsymbol{\xi}}\right), \tag{1}$$

where $\boldsymbol{S}_\infty^{\boldsymbol{\xi}}$ is a set of admissible paths from the initial state under the policy $\boldsymbol{\xi}$.

Assumption 1: It is assumed that there exists at least one policy whose induced traces satisfy the task ϕ with non-zero probability.

Problem 1: Given an LTL-specified task ϕ and a continuous-state continuous-action labeled MDP $\mathcal{M}$ with unknown transition probabilities, the goal is to learn a policy $\boldsymbol{\xi}^*$ that maximizes the satisfaction probability in the limit, i.e., $\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\arg\max} \Pr{}_M^{\boldsymbol{\xi}}(\phi)$.

IV. Automaton Design

To address Problem 1, Section IV-A presents the construction of the EP-MDP. The advantages of incorporating EP-MDP are discussed in Section IV-B.

A. Embedded Product MDP

Given an LDGBA $\mathcal{A}_\phi = (Q, \Sigma, \delta, q_0, F)$, a tracking-frontier set T is designed to keep track of unvisited accepting sets. Particularly, T is initialized as $T_0 = F$ and $\mathcal{B}$ is a Boolean variable, and $(T, \mathcal{B})$ can be then updated according to the following rule:

$$f_V(q,T) = \begin{cases} (T \setminus F_j, \text{False}), & \text{if } q \in F_j \text{ and } F_j \in T, \\ (F \setminus F_j, \text{True}) & \text{if } q \in F_j \text{ and } T = \emptyset, \\ (T, \text{False}), & \text{otherwise.} \end{cases} \tag{2}$$

Once a state $q \in F_j$ is visited, F_j will be removed from T by rendering function $f_V(q, T)$. If T becomes empty before visiting set F_j, it will be reset as $F \setminus F_j$. Since the acceptance condition of LDGBA requires to infinitely visit all accepting sets, we call it one round if all accepting sets have been visited (i.e., a round ends if T becomes empty). The second output of $f_V(q, T)$ is to indicate whether all accepting sets have been visited in current round, which is applied in Section V-C to design the potential function. Based on (2), the EP-MDP is constructed as follows.

Definition 4: Given an MDP $\mathcal{M}$ and an LDGBA $\mathcal{A}_\phi$, the EP-MDP is defined as $\mathcal{P} = \mathcal{M} \times \mathcal{A}_\phi = (X, U^{\mathcal{P}}, p^{\mathcal{P}}, x_0, F^{\mathcal{P}}, T, f_V, \mathcal{B})$, where $X = S \times Q \times 2^F$ is the set of product states and 2^F denotes all subsets of F for $\mathcal{A}_\phi$, i.e., $x = (s, q, T) \in X$; $U^{\mathcal{P}} = A \cup \{\epsilon\}$ is the set of actions, where the ϵ-actions are only allowed for transitions from Q_N to Q_D; $x_0 = (s_0, q_0, T_0)$ is the initial state; $F^{\mathcal{P}} = \{F_1^{\mathcal{P}}, F_2^{\mathcal{P}} \ldots F_f^{\mathcal{P}}\}$ where $F_j^{\mathcal{P}} = \{(s, q, T) \in X \big| q \in F_j \wedge F_j \subseteq T\}$, $j = 1, \ldots f$, is a set of accepting states; $p^{\mathcal{P}}$ is the transition kernel for any transition $p^{\mathcal{P}}(x, u^{\mathcal{P}}, x')$ with $x = (s, q, T)$ and $x' = (s,' q,' T)$ such that : (1) $p^{\mathcal{P}}(x, u^{\mathcal{P}}, x') = p_S(s'|s, a)$ if $s' \backsim p_S(\cdot|s, a)$, $\delta(q, L(s)) = q'$ where $u^{\mathcal{P}} = a \in A$ (2) $p^{\mathcal{P}}(x, u^{\mathcal{P}}, x') = 1$ if $u^{\mathcal{P}} \in \{\epsilon\}$, $q' \in \delta(q, \epsilon)$ and $s' = s$; and (3) $p^{\mathcal{P}}(x, u^{\mathcal{P}}, x') = 0$ otherwise. After completing each transition $q' = \delta(q, \alpha)$ based on $p^{\mathcal{P}}$, T is synchronously updated as $(T, \mathcal{B}) = f_V(q,' T)$ by (2).

The state-space is embedded with the tracking-frontier set T that can be practically represented via one-hot encoding based on the indices of the accepting set. Compared to the standard construction of product MDPs, any state of the accepting set in EP-MDP, e.g., $(s, q, T) \in F_j^{\mathcal{P}}$, requires the automaton state to satisfy $q \in F_j \wedge F_j \subseteq T$, and embedded tracking frontier set T is updated based on (2) after each transition. Consequently, to satisfy the accepting condition, the agent is encouraged to visit all accepting sets.

The EP-MDP captures the intersections between all feasible paths over $\mathcal{M}$ and all words accepted to $\mathcal{A}_\phi$, facilitating the identification of admissible agent actions that satisfy task ϕ. The procedure of obtaining a valid run $\boldsymbol{x}_{\mathcal{P}}$ on-the-fly within EP-MDP by randomly selecting an action can be found in Alg.1 of [29].

Let $\boldsymbol{\pi}$ denote a policy over $\mathcal{P}$ and denote by $\boldsymbol{x}_\infty^{\boldsymbol{\pi}} = x_0 \ldots x_i x_{i+1} \ldots$ the infinite path generated by $\boldsymbol{\pi}$. A path $\boldsymbol{x}_\infty^{\boldsymbol{\pi}}$ is accepted if $\inf(\boldsymbol{x}_\infty^{\boldsymbol{\pi}}) \cap F_i^{\mathcal{P}} \neq \emptyset$, $\forall i \in \{1, \ldots f\}$. We denote $\Pr^{\boldsymbol{\pi}}[x \models \text{Acc}_p]$ as the probability of satisfying the accepting condition of $\mathcal{P}$ under policy $\boldsymbol{\pi}$, and denote $\Pr_{max}[x \models \text{Acc}_p] = \underset{\boldsymbol{\pi}}{\max} \Pr_M^{\boldsymbol{\pi}}(\text{Acc}_p)$ as the maximum probability of satisfying the accepting condition of $\mathcal{P}$. Let $\boldsymbol{\pi}^*$ denote an optimal policy that maximizes the expected discounted return over $\mathcal{P}$, i.e., $\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}}{\arg\max} U^{\boldsymbol{\pi}}(s)$. Note that the memory-less policy $\boldsymbol{\pi}^*$ over $\mathcal{P}$ yields a finite-memory policy $\boldsymbol{\xi}^*$ over $\mathcal{M}$, allowing us to reformulate Problem 1 as:

Problem 2: Given a user-specified LTL task ϕ and a general and unknown continuous-state continuous-action labeled MDP, the goal is to asymptotically find a policy $\boldsymbol{\pi}^*$ satisfying the acceptance condition of $\mathcal{P}$ with maximum probability, i.e., $\Pr^{\boldsymbol{\pi}^*}[x \models \mathrm{Acc}_p] = \Pr_{max}[x \models \mathrm{Acc}_p]$.

B. Properties of EP-MDP

Consider a sub-EP-MDP $\mathcal{P}'_{(X,'U')}$, where $X' \subseteq X$ and $U' \subseteq U^{\mathcal{P}}$. If $\mathcal{P}'_{(X,'U')}$ is a maximum end component (MEC) of $\mathcal{P}$ and $X' \cap F_i^{\mathcal{P}} \neq \emptyset$, $\forall i \in \{1, \ldots f\}$, then $\mathcal{P}'_{(X,'U')}$ is called an accepting maximum end component (AMEC) of $\mathcal{P}$. Once a path enters an AMEC, the subsequent path will stay within it by taking restricted actions from U'. There exist policies such that any state $x \in X'$ can be visited infinitely often. As a result, satisfying task ϕ is equivalent to reaching an AMEC. Moreover, a MEC that does not contain any accepting sets is called a rejecting accepting component (RMEC) and a MEC with only partial accepting sets is called a neutral maximum end component (NMEC) [2].

Definition 5: Let $MC_{\mathcal{P}}^{\boldsymbol{\pi}}$ denote the Markov chain induced by a policy $\boldsymbol{\pi}$ on $\mathcal{P}$, whose states can be represented by a disjoint union of a transient class $\mathcal{T}_{\boldsymbol{\pi}}$ and n_R closed irreducible recurrent classes $\mathcal{R}_{\boldsymbol{\pi}}^j$, $j \in \{1, \ldots, n_R\}$ [30].

Lemma 1: Given an EP-MDP $\mathcal{P} = \mathcal{M} \times \mathcal{A}_\phi$, the recurrent class $R_{\boldsymbol{\pi}}^j$ of $MC_{\mathcal{P}}^{\boldsymbol{\pi}}$, $\forall j \in \{1, \ldots, n_R\}$, induced by π satisfies one of the following conditions: $R_{\boldsymbol{\pi}}^j \cap F_i^{\mathcal{P}} \neq \emptyset, \forall i \in \{1, \ldots f\}$, or $R_{\boldsymbol{\pi}}^j \cap F_i^{\mathcal{P}} = \emptyset, \forall i \in \{1, \ldots f\}$.

Proof: The strategy of the following proof is based on contradiction. Assume there exists a policy such that $R_{\boldsymbol{\pi}}^j \cap F_k^{\mathcal{P}} \neq \emptyset$, $\forall k \in K$, where K is a subset of $2^{\{1, \ldots f\}} \setminus \{\{1, \ldots f\}, \emptyset\}$. As discussed in [31], for each state in recurrent class, it holds that $\sum_{n=0}^{\infty} p^n(x,x) = \infty$, where $x \in R_{\boldsymbol{\pi}}^j \cap F_k^{\mathcal{P}}$ and $p^n(x,x)$ denotes the probability of returning from a transient state x to itself in n steps. This means that each state in the recurrent class occurs infinitely often. However, based on the embedded tracking-frontier function of EP-MDP, once $x_k = (s, q_k, T) \in R_{\boldsymbol{\pi}}^j \cap F_k^{\mathcal{P}}$ is visited, the corresponding F_k of $F_k^{\mathcal{P}}$ is removed from T, and the tracking set T will not be reset until all accepting sets have been visited. As a result, neither $q_k \in F_k$ nor x_k will occur infinitely, which contradicts the property $\sum_{n=0}^{\infty} p^n(x_k, x_k) = \infty$. ■

Lemma 1 indicates for any policy, all accepting sets will be placed either in the transient class or in the recurrent classes.

V. Learning-Based Control Synthesis

In the following, we discuss a base reward design and present rigorous analysis to show how such a design can guide the RL-agent over the EP-MDP to find an optimal policy whose traces satisfies the LTL task with maximum probability. In order to improve the reward density, Section V-B proposes a reward shaping process via integrating a potential function under which the optimal policies remain invariant. Finally, Section V-C shows how to apply the shaped reward function with DDPG to construct a modular DDPG architecture and effectively solve Problem 2.

A. Base Reward

Let $F_U^{\mathcal{P}}$ denote the union of accepting states, i.e., $F_U^{\mathcal{P}} = \{x \in X \big| x \in F_i^{\mathcal{P}}, \forall i \in \{1, \ldots f\}\}$. For each transition $(x, u^{\mathcal{P}}, x')$ in the EP-MDP, the reward and discounting function only depend on current state x, i.e., $R(x, u^{\mathcal{P}}, x') = R(x)$ and $\gamma(x, u^{\mathcal{P}}, x') = \gamma(x)$.

Inspired by [14], we propose a reward function as:

$$R(x) = \begin{cases} 1 - r_F, & \text{if } x \in F_U^{\mathcal{P}}, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

and a discounting function as

$$\gamma(x) = \begin{cases} r_F, & \text{if } x \in F_U^{\mathcal{P}}, \\ \gamma_F, & \text{otherwise}, \end{cases} \tag{4}$$

where $r_F(\gamma_F)$ is a function of γ_F satisfying $\lim_{\gamma_F \to 1^-} r_F(\gamma_F) = 1$ and $\lim_{\gamma_F \to 1^-} \frac{1-\gamma_F}{1-r_F(\gamma_F)} = 0$.

Given a path $\boldsymbol{x}_t = x_t x_{t+1} \ldots$ starting from x_t, the return is denoted by

$$\mathcal{D}(\boldsymbol{x}_t) := \sum_{i=0}^{\infty} \left(\prod_{j=0}^{i-1} \gamma(\boldsymbol{x}_t[t+j]) \cdot R(\boldsymbol{x}_t[t+i]) \right), \tag{5}$$

where $\prod_{j=0}^{-1} := 1$ and $\boldsymbol{x}_t[t+i]$ denotes the $(i+1)$th state in $\boldsymbol{x}_t$. Based on (5), the expected return of any state $x \in X$ under policy π can be defined as

$$U^{\boldsymbol{\pi}}(x) = \mathbb{E}^{\boldsymbol{\pi}}[\mathcal{D}(\boldsymbol{x}_t) | \boldsymbol{x}_t[t] = x|]. \tag{6}$$

Even though we adopt the reward desgin of the work [14], we can not obtain the same results. This is because we apply the LDGBA with several accepting sets resulting more complicated situations, e.g., AMEC, NMEC and RMEC. Fortunately, we can still establish the following theorem as one of the main contributions.

Theorem 1: Given the EP-MDP $\mathcal{P} = \mathcal{M} \times \mathcal{A}_\phi$, for any state $x \in X$, the expected return under any policy π satisfies

$$\exists i \in \{1, \ldots f\}, \lim_{\gamma_F \to 1^-} U^{\boldsymbol{\pi}}(x) = \Pr{}^{\boldsymbol{\pi}}\left[\Diamond F_i^{\mathcal{P}}\right], \tag{7}$$

where $\Pr^{\boldsymbol{\pi}}[\Diamond F_i^{\mathcal{P}}]$ is the probability that the paths starting from state x will eventually intersect a $F_i^{\mathcal{P}} \in F^{\mathcal{P}}$.

Details of the proof can be found in [29]. Theorem 1 builds the connection between the probability of visiting the accepting sets and the expected return. Next, we will show in the following sections how Lemma 1 and Theorem 1 can be leveraged to ensure that the RL-agent satisfies the accepting condition of $\mathcal{P}$ with the maximum probability.

Theorem 2: Consider an MDP $\mathcal{M}$ and an LDGBA $\mathcal{A}_\phi$ corresponding to an LTL formula ϕ. Based on Assumption 1, there exists a discount factor $\underline{\gamma}$, with which any optimization method for (6) with $\gamma_F > \underline{\gamma}$ and $r_F > \underline{\gamma}$ can obtain a policy $\bar{\boldsymbol{\pi}}$, such that the induced run $r_{\mathcal{P}}^{\bar{\boldsymbol{\pi}}}$ satisfies the accepting condition $\mathcal{P}$ with non-zero probability in the limit.

The detailed proof of Theorem 2 can be found in [29]. Theorem 2 proves that by selecting $\gamma_F > \underline{\gamma}$ and $r_F > \underline{\gamma}$, optimizing the expected return in (6) can find a policy satisfying the given task ϕ with non-zero probability.

Theorem 3: Given an MDP $\mathcal{M}$ and an LDGBA $\mathcal{A}_\phi$, by selecting $\gamma_F \to 1^-$, the optimal policy in the limit $\boldsymbol{\pi}^*$ that maximizes the expected return (6) of the corresponding EP-MDP also maximizes the probability of satisfying ϕ, i.e., $\Pr^{\boldsymbol{\pi}^*}[x \models Acc_{\mathcal{P}}] = \Pr_{max}[x \models Acc_{\mathcal{P}}]$.

Proof: Since $\gamma_F \to 1^-$, we have $\gamma_F > \underline{\gamma}$ and $r_F > \underline{\gamma}$ from Theorem 2. There exists an induced run $r_{\mathcal{P}}^{\boldsymbol{\pi}^*}$ satisfying the accepting condition of $\mathcal{P}$. According to Lemma 1, $\lim_{\gamma_F \to 1^-} U^{\boldsymbol{\pi}^*}(x)$ is exactly equal to the probability of visiting the accepting sets of an AMEC. Optimizing $\lim_{\gamma_F \to 1^-} U^{\boldsymbol{\pi}^*}(x)$ is equal to optimizing the probability of entering AMECs. ■

B. Reward Shaping

Since the base reward function in Section V-A is always zero for $x \notin F_U^{\mathcal{P}}$, the reward signal might become sparse. To resolve this, we propose a potential function $\Phi : X \to \mathbb{R}$, and transform the reward as follows:

$$R'\left(x, u^{\mathcal{P}}, x'\right) = R(x) + \gamma(x) \cdot \Phi\left(x'\right) - \Phi(x) \qquad (8)$$

As shown in [27], given any MDP model, e.g., EP-MDP $\mathcal{P}$, transforming the reward function using a potential function $\Phi(x)$ as in $R'(x, u^{\mathcal{P}}, x')$ will not change the set of optimal policies. Thus, a real-valued function $\Phi(x)$ will improve the learning performance while guaranteeing that the resulted policies via $R'(x, u^{\mathcal{P}}, x')$ are still optimal with respect to the base reward function $R(x)$.

Given $\mathcal{P} = \mathcal{M} \times \mathcal{A}_\phi = (X, U^{\mathcal{P}}, p^{\mathcal{P}}, x_0, F^{\mathcal{P}}, T, f_V, \mathcal{B})$ with $\mathcal{A}_\phi = (Q, \Sigma, \delta, q_0, F)$, let $F_U = \{q \in Q \big| q \in F_i, \forall i \in \{1, \ldots f\}\}$ denote the union of automaton accepting states. For the states of $\mathcal{P}$ whose automaton states belong to $Q \setminus (F_U \cup q_0 \cup Q_{sink})$, it is desirable to assign positive rewards when the agent first visits them and assign large value of reward to the accepting states to enhance the convergence of neural network. This is because starting from the automaton initial state, any automaton state that can reach any of the accepting sets has to be explored. To this end, an automaton tracking-frontier set T_Φ is designed to keep track of unvisited automaton components $Q \setminus (q_0 \cup Q_{sink})$, and T_Φ is initialized as $T_{\Phi 0} = Q \setminus (q_0 \cup Q_{sink})$. The set $T_{\Phi 0}$ is then updated after each transition $((s, q, T), u^{\mathcal{P}}, (s,' q,' T))$ of $\mathcal{P}$ as:

$$f_\Phi\left(q,' T_\Phi\right) = \begin{cases} T_\Phi \setminus q,' & \text{if } q \in T_\Phi, \\ T_{\Phi 0} \setminus q' & \text{if } \mathcal{B} = \text{True}, \\ T_\Phi, & \text{otherwise}. \end{cases} \qquad (9)$$

The set T_Φ will only be reset when $\mathcal{B}$ in f_V becomes True, indicating that all accepting sets in the current round have been visited. Based on (9), the potential function $\Phi(x)$ for $x = (s, q, T)$ is constructed as:

$$\Phi(x) = \begin{cases} \eta_\Phi \cdot (1 - r_F), & \text{if } q \in T_\Phi, \\ 0, & \text{otherwise}. \end{cases} \qquad (10)$$

Intuitively, the value of potential function for unvisited and visited states in $T_{\Phi 0}$ is equal to $1 - r_F$ and 0 respectively, in each round, which will guide the system finally reach any of the accepting sets. To illustrate the idea, an example of applying the shaped reward can be found in [29].

C. Modular Deep Deterministic Policy Gradient

To deal with continuous-state and continuous-action MDPs, deep deterministic policy gradient (DDPG) [18] is adopted in this work to approximate the current deterministic policy via a parameterized function $\boldsymbol{\pi}(x|\theta^u)$ called actor. The actor is a deep neural network whose set of weights are θ^u. The critic function also applies a deep neural network with parameters θ^Q to approximate action-value function $Q(x, u^{\mathcal{P}}|\theta^Q)$, which is updated by minimizing the following loss function:

$$L\left(\theta^Q\right) = \mathbb{E}^{\boldsymbol{\pi}}_{s \sim \rho^\beta}\left[\left(y - Q\left(x, \boldsymbol{\pi}\left(x \left| \theta^u\right.\right) \left| \theta^Q\right.\right)\right)^2\right], \qquad (12)$$

where ρ^β is the state distribution under any arbitrary policy β, and $y = R'(x, u^{\mathcal{P}}, x') + \gamma(x) Q(x,' u^{\mathcal{P}'}|\theta^Q)$ with $u^{\mathcal{P}'} = \boldsymbol{\pi}(x'|\theta^u)$. The actor can be updated by applying the chain rule to the expected return with respect to actor parameters θ^u as the following policy gradient theorem [18]:

$$\begin{aligned} \nabla_{\theta^u} U^u(x) &\approx \mathbb{E}^{\boldsymbol{\pi}}_{s \sim \rho^\beta}\left[\nabla_{\theta^u} Q\left(x, \boldsymbol{\pi}\left(x \left|\theta^u\right.\right) \left|\theta^Q\right.\right)\right] \\ &= \mathbb{E}^{\boldsymbol{\pi}}_{s \sim \rho^\beta}\left[\nabla_{u^{\mathcal{P}}} Q\left(x, u^{\mathcal{P}} \left|\theta^Q\right.\right) \big|_{u^{\mathcal{P}} = \boldsymbol{\pi}(x|\theta^u)} \nabla_{\theta^u} \boldsymbol{\pi}\left(x \left|\theta^u\right.\right)\right]. \end{aligned} \qquad (13)$$

Inspired by [24] and [25], the complex LTL task ϕ can be divided into simple composable modules. Each state of the automaton in the LDGBA is module and each transition between these automaton states is a "task divider". In particular, given ϕ and its LDGBA $\mathcal{A}_\phi$ that has a set of states Q, we propose a modular architecture with $|Q|$ DDPG structures, i.e., $\boldsymbol{\pi}_{q_i}(x|\theta^u)$ and $Q_{q_i}(x, u^{\mathcal{P}}|\theta^Q)$ with $q_i \in Q$, along with their own replay buffer. Experience samples are stored in each replay modular buffer B_{q_i} in the form of $(x, u^{\mathcal{P}}, R(x), \gamma(x), x')$. By dividing the LTL task into sub-stages, the set of neural nets acts in a global modular DDPG architecture, which allows the agent to jump from one module to another by switching between the set of neural nets based on transition relations of $\mathcal{A}_\phi$.

The proposed method to solve a continuous MDP with LTL specifications is summarized in Algorithm 1, and the product states of EP-MDP are synchronized on-the-fly (line 9-12). We assign each DDPG an individual replay buffer B_{q_i} and a random process noise N_{q_i}. The corresponding weights of modular networks, i.e., $Q_{q_i}(x, u^{\mathcal{P}}|\theta^{Q_{q_i}})$ and $\boldsymbol{\pi}_{q_i}(x|\theta^{u_{q_i}})$, are also updated at each iteration (line 15-20). All neural networks are trained using their own replay buffer, which is a finite-sized cache that stores transitions sampled from exploring the environment. Since the direct implementation of updating target networks can be unstable and divergent [17], the soft update (11) is employed, where target networks are slowly updated via relaxation (line 20). Note that for each iteration we first observe the output of the shaped reward function R', then execute the update process via f_V and f_Φ (line 10-12). Note that since DDPG leverages a deep neural network to approximate the action-value function, and that in practice we have to stop the training after finite number of steps the synthesised policy might be sub-optimal with respect to the true $\boldsymbol{\pi}^*$ as in Theorem 3. This is due to the nature of DDPG algorithm and non-linear function approximation in deep architectures whose study is out of the scope of this paper.

VI. Case Studies

The developed modular DDPG-based control synthesis is implemented in Python, and all simulation videos and source codes can be found in our Github repository.[1] Owl [32] is used to convert LTL specifications to LDGBA. We test the algorithm in 4 different environments in OpenAI gym and with 6 LTL tasks. In particular, we first assign complex tasks, formulated via LTL, to Ball-Pass and CartPole problems. To show the advantages of EP-MDP with modular DDPG, we compare the method with (i) the standard product MDP using modular DDPG and (ii) the EP-MDP using standard DDPG. Then, we test the scalability of our algorithm in pixel-based Mars exploration environments.

[1]https://github.com/mingyucai/Modular_Deep_RL_E-LDGBA

Algorithm 1: Modular DDPG.

```
1:  procedure INPUT: (MDP $\mathcal{M}$, LDGBA $\mathcal{A}_\phi$)
      Output: modular DDPG for optimal policy $\boldsymbol{\pi}^*$
      Initialization: $|Q|$ actor $\boldsymbol{\pi}_{q_i}(x|\theta^{u_{q_i}})$ and critic
    networks $Q_{q_i}(x, u^{\mathcal{P}}|\theta^{Q_{q_i}})$ with arbitrary weights $\theta^{u_{q_i}}$
    and $\theta^{\mathbf{Q}_{q_i}}$ for all $q_i \in Q$; $|Q|$ corresponding target
    networks $\boldsymbol{\pi}'_{q_i}(x|\theta^{u'_{q_i}})$ and $Q'_{q_i}(x, u^{\mathcal{P}}|\theta^{Q'_{q_i}})$ with
    weights $\theta^{u'_{q_i}}$ and $\theta^{Q'_{q_i}}$ for each $q_i \in Q$, respectively;
    $|Q|$ replay buffers $B_{q_i}$; $|Q|$ random processes
    noise $N_{q_i}$
2:    set $r_F = 0.99$ and $\gamma_F = 0.9999$ to determine $R(x)$
      and $\gamma(x)$
3:    set maximum episodes $E$ and iteration number $\tau$
4:    for each episode in $E$ do
5:      set $t = 0$
6:      sample an initial state $s_0$ of $\mathcal{M}$ and $q_0$ of $\mathcal{A}_\phi$ as
        $s_t, q_t$
7:      set $t = 0$ and construct an initial product state
        $x_t = (s_t, q_t, T)$, where $T = T_0$
8:      while $t \leq \tau$ do
9:        select action $u_t^{\mathcal{P}} = \boldsymbol{\pi}_{q_t}(x|\theta^{u_{q_t}}) + R_{q_t}$ based on
          exploitation versus exploration noise
10:       execute $u_t^{\mathcal{P}}$, $\delta$ and observe
          $x_{t+1} = (s_{t+1}, q_{t+1}, T)$, $R(x_{t+1})$, $\Phi(x_{t+1})$,
          $\gamma(x_{t+1})$
11:       obtain $\Phi(x_t)$ based on current $T_\Phi$ and calculate
          $R'(x_t, u_t^{\mathcal{P}}, x_{t+1})$
12:       execute the updates via $f_V(q_{t+1}, T)$ and
          $f_\Phi(q_{t+1}, T_\Phi)$
13:       store the sample
          $(x_t, u_t^{\mathcal{P}}, R'(x_t, u_t^{\mathcal{P}}, x_{t+1}), \gamma(x_{t+1}), x_{t+1})$ in
          replay buffers $B_{q_t}$
14:       mini-batch sampling $N$ data from the replay
          buffers $B_{q_t}$
15:       calculate target values for each $i \in N$ as:
            $y_i = R'\left(x_i, u_i^{\mathcal{P}}, x_{i+1}\right) + \gamma\left(x_i\right) \cdot Q'_{q_{i+1}}\left(x_{i+1}, u_{i+1}^{\mathcal{P}} \middle| \theta^{Q'_{q_{i+1}}}\right)$
16:       update weights $\theta^{Q_{q_t}}$ of critic neural network
          $Q_{q_t}(x, u^{\mathcal{P}}|\theta^{Q_{q_t}})$ by minimizing the loss
          function:
            $L = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q_{q_t}\left(x_i, u_i^{\mathcal{P}}\left|\theta^{Q_{q_t}}\right.\right)\right)^2$
17:       update weights $\theta^{u_{q_t}}$ of actor neural network
          $\boldsymbol{\pi}_{q_t}(x|\theta^{u_{q_t}})$ by maximizing the policy gradient:
            $\nabla_{\theta^{u_{q_t}}} U^{q_t} \approx \frac{1}{N}\sum_{i=1}^{N}\left(\nabla_{u^{\mathcal{P}}} Q_{q_t}\left(x_i, u^{\mathcal{P}}\left|\theta^{Q_{q_t}}\right.\right)\Big|_{u^{\mathcal{P}}=\boldsymbol{\pi}_{q_t}(x_i|\theta^{u_{q_t}})} \cdot \nabla_{\theta^{u_{q_t}}} \boldsymbol{\pi}_{q_t}\left(x_i\left|\theta^{u_{q_t}}\right.\right)\right)$
18:       soft update of target networks:
            $\theta^{u'_{q_t}} \leftarrow \tau\theta^{u_{q_t}} + (1-\tau)\,\theta^{u'_{q_t}}$
            $\theta^{Q'_{q_t}} \leftarrow \tau\theta^{Q_{q_t}} + (1-\tau)\,\theta^{Q'_{q_t}}$    (11)
19:       $x_t \leftarrow x_{t+1}$ and $t{+}{+}$
20:     end while
21:   end for
22: end procedure
```

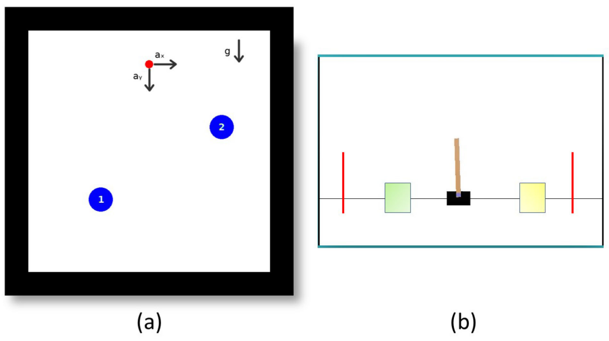


Fig. 2. (a) Ball-Pass and (b) Cart-Pole.

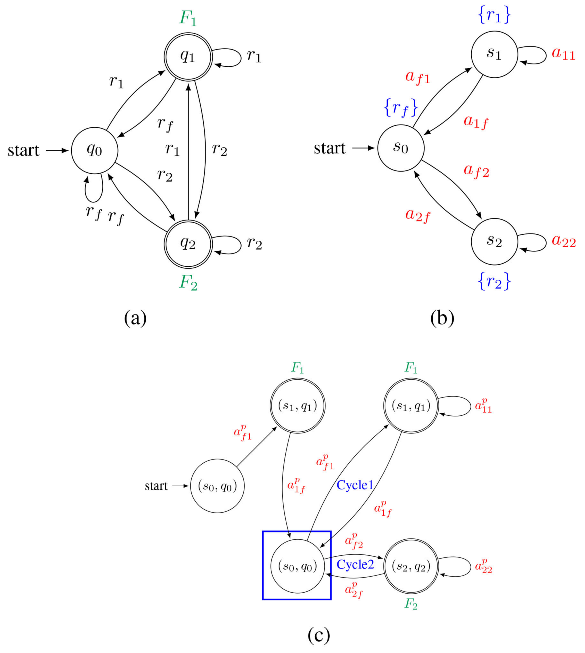


Fig. 3. (a) LDGBA of LTL formula φ_{B1}. (b) Generalized MDP model for Ball-pass case. (c) The standard product MDP.

Finally, we analyze and compare the performances of the modular and standard DDPG algorithms via success rates of task accomplishments.

(1). Ball-Pass: We first demonstrate the developed control synthesis in a Ball-Pass environment (600 m × 600 m). Consider a red ball moving according to the following dynamics: $\ddot{x} = a_x$, $\ddot{y} = a_y + g$ in Fig. 2(a), where (x, y) is the planar position of the ball, a_x, a_y represent accelerations along x and y induced by an external force (i.e., the control input), respectively, and g is the gravitational acceleration. The simulation step size is $\Delta t = 0.05$ s, and the acceleration range is $a_x, a_y \in [0, 1] m/s$. We consider two LTL tasks, e.g., $\varphi_{B1} = (\square\lozenge\text{Region1}) \wedge (\square\lozenge\text{Region2})$ and $\varphi_{B2} = \lozenge(\text{Region1} \wedge \lozenge\text{Region2})$. φ_{B1} is is a task over the infinite horizon and requires the agent repetitively visits Region1 and Region2, and φ_{B2} is a task over the finite horizon and requires the agent first to visit Region1 and then Region2.

For comparison of method (i), Fig. 3(a) shows the corresponding LDGBA of φ_{B1} with two accepting sets $F = \{\{q_1\}, \{q_2\}\}$, and r_1, r_2 and r_f represent Region 1, Region 2 and free space

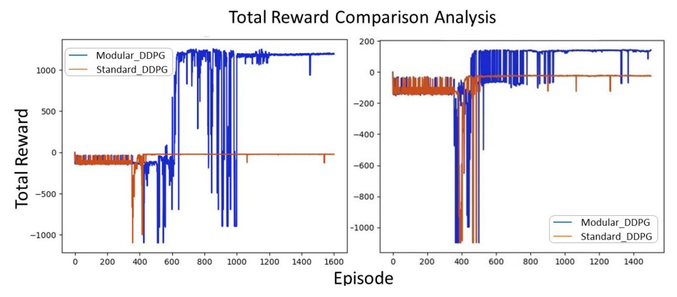


Fig. 4. Total reward analysis for Ball-Pass compared with standard DDPG (left) φ_{B1} and (right) φ_{B2}.

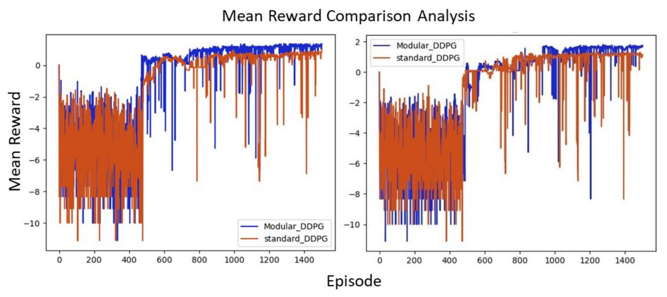


Fig. 5. Mean reward analysis for CartPole compared with standard DDPG (left) φ_{C1} and (right) φ_{C2}.

being visited, respectively. Fig. 3(b) shows the Ball-pass MDP model, where a_{fi} (or a_{if}) represent a sequence of continuous actions that drive the ball from free space to Region i (or inverse). In addition, Fig. 3(c) shows the resulting standard product MDP. By Definition 2, the policy that satisfies φ_{B1} should enforce the repetitive trajectories, i.e., cycle 1 and cycle 2 in Fig. 3(c). However, there doesn't exist deterministic policy that can periodically select two actions a^P_{f1} and a^P_{f2} at state (s_f, q_0) (marked with a blue rectangle) in Fig. 3(c). As a result, applying standard product MDP cannot generate a pure deterministic optimal policy to complete task φ_{B1}. In contrast, the tracking-frontier set of EP-MDP developed in this work can resolve this issue by recording unvisited acceptance sets and being embedded with each state at every time-step via one-hot encoding. We simulate 10 runs for tasks φ_{B1} and φ_{B2} and select the worst case of applying standard product MDP as comparison. For comparison of method (ii), we conduct 1600 episodes for each task (φ_{B1} and φ_{B2}) of Ball-pass, which are shown in Fig. 4.

(2). CartPole: We also test our control framework for the Cart-pole[2] in Fig. 2(b). The pendulum starts upright with an initial angle between -0.05 and 0.05 rads. The horizontal force exerted on the cart is defined over a continuous space (action-space) with a range $(-10\,N, 10\,N)$. The green and yellow regions range from -1.44 to -0.96 m and from 0.96 to 1.44 m, respectively. The objective is to prevent the pendulum from falling over while moving the cart between the yellow and green regions. Similarly, we consider two tasks $\varphi_{C1} = (\Box\Diamond\text{Green}) \wedge (\Box\Diamond\text{Yellow}) \wedge \neg\Box\text{Unsafe}$ and $\varphi_{C2} = \Diamond(\text{Green} \wedge \Diamond\text{Yellow}) \wedge \neg\Box\text{Unsafe}$, where Unsafe represents the condition that the pendulum falls over or exceeds the red line, and Green, Yellow represent colored areas. Method (i) has the similar issue as shown in Fig. 3 for task φ_{C1} over infinite horizon. To compare the method (ii), we conduct mean reward collections over 1500 episodes for each task (φ_{C1} and φ_{C2}) of CartPole shown in Fig. 5.

The reward collections and videos of comparison for tasks φ_{B1}, φ_{B2}, φ_{C1}, φ_{C2} of these environments are shown in [29] and our Github repository. We can conclude the modular DDPG has a better performance of collecting positive rewards during learning. The figures also illustrate that the modular DDPG has a better performance than the standard DDPG for tasks over both finite and infinite horizons.

(3). Mars exploration: To test our algorithm in large scale continuous environments, we conduct motion planning via two

[2]https://gym.openai.com/envs/CartPole-v0/#barto83

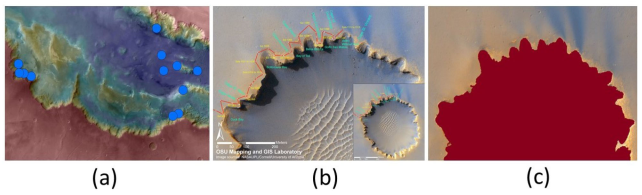


Fig. 6. (a) Melas Chasma with possible location of water (blue dots). (b) Victoria Crater with exploration path (red line). (c) Processed image of Victoria Crater.

TABLE I
COMPARISON OF STANDARD AND MODULAR DDPG WITH EP-MDP. STATISTICS ARE TAKEN OVER 200 RUNS

LTL Task	DDPG	Success rate
φ_{B1}	Modular DDPG	100%
	Standard DDPG	0%
φ_{B2}	Modular DDPG	100%
	Standard DDPG	77.5%
φ_{C1}	Modular DDPG	100%
	Standard DDPG	23.5%
φ_{C2}	Modular DDPG	100%
	Standard DDPG	61%
φ_{Melas}	Modular DDPG	100%
	Standard DDPG	0%
$\varphi_{Victoria}$	Modular DDPG	100%
	Standard DDPG	0%

Mars exploration missions using satellite images under two infinite-horizon TL tasks ($\varphi_{Melas}, \varphi_{Victoria}$) as shown in Fig. 6. Due to space limitation, More details of Mars exploration cases can be found in [29].

(4). Training time and performance evaluation: We take 200 runs and analyze the success rate for all aforementioned tasks in Table I. We can conclude that the standard DDPG with a recording method as [26] might yield poor performance results for tasks with repetitive pattern (infinite horizon), whereas the modular DDPG has better performance from the perspective of training and success rate. It should also be noted that this paper has mainly considered surveillance tasks, in view of its applications. However, the temporal operators in LTL, such as "next" and "until," enable the modular DDPG algorithm to solve problems with other types of complex tasks, as further reported in [29]. In those cases, LTL formulas can be converted to LDGBAs [19], and the same framework can be applied.

At last, although several actor-critic neural network pairs are adopted in the modular architecture, they are synchronously trained online, and each of them is only responsible for a sub-task. This setup is more effective to complete LTL complex tasks. The training time for each task with two different algorithms (the standard DDPG and the modular DDPG) is shown in Table 2 of the extended version [29]. It can be seen that the complexity is not increased in the modular DDPG.

VII. Conclusion

In this letter, a automatic model-free deep RL learning is developed to synthesize control policies in continuous-state and continuous-action MDPs. The LTL specifications are applied to express complex objectives which can be converted into LDGBA, and the designed EP-MDP can enforce the task satisfaction. A modular DDPG framework is proposed to decompose the complex tasks into interrelated sub-tasks based on automaton structure such that the resulted optimal policies are shown to satisfy the LTL specifications with a higher success rate compared to the standard DDPG.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT press, 2018.

[2] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[3] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.

[4] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *Int. J. Robot. Res.*, vol. 39, No. 8, pp. 1002–1028, 2020.

[5] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of Markov decision processes with linear temporal logic constraints," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1244–1257, May 2014.

[6] B. Lacerda, F. Faruq, D. Parker, and N. Hawes, "Probabilistic planning with formal performance guarantees for mobile service robots," *Int. J. Robot. Res.*, vol. 38, no. 9, pp. 1098–1123, 2019.

[7] M. Kloetzer and C. Mahulea, "LTL-based planning in environments with probabilistic observations," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 4, pp. 1407–1420, Oct. 2015.

[8] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications," in *Proc. IEEE Conf. Decis. Control.*, 2014, pp. 1091–1096.

[9] J. Fu, and U. Topcu, "Probably approximately correct MDP learning and control with temporal logic constraints," 2014, arXiv preprint arXiv:1404.7073.

[10] J. Wang, X. Ding, M. Lahijanian, I. C. Paschalidis, and C. A. Belta, "Temporal logic motion control using actor-critic methods," *Int. J. Robot. Res.*, vol. 34, no. 10, pp. 1329–1344, 2015.

[11] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic LTL motion planning with environment and motion uncertainties," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2386–2392, May 2021.

[12] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in *Proc. IEEE Conf. Decis. Control*, 2019, pp. 5338–5343.

[13] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak, "Omega-regular objectives in model-free reinforcement learning," in *Proc. Int. Conf. Tools Alg. Constr. Anal. Syst.*, Springer, 2019, pp. 395–412.

[14] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *Proc. Int. Conf. Robot. Autom.*, 2020, pp. 10 349–10355.

[15] R. Oura, A. Sakakibara, and T. Ushio, "Reinforcement learning of control policy for linear temporal logic specifications using limit-deterministic generalized Büchi automata," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 761–766, Jul. 2020.

[16] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *Proc. IEEE Conf. Decis. Control*, 2016, pp. 6565–6570.

[17] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[19] S. Sickert, J. Esparza, S. Jaax, and J. Křetínský, "Limit-deterministic Büchi automata for linear temporal logic," in *Proc. Int. Conf. Comput. Aided Verif.*, Springer, 2016, pp. 312–332.

[20] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2107–2116.

[21] A. Camacho, R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith, "LTL and beyond: Formal languages for reward function specification in reinforcement learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, vol. 19, pp. 6065–6073.

[22] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," *Sci. Robot.*, vol. 4, no. 37, 2019.

[23] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. M. Zavlanos, "Reduced variance deep reinforcement learning with temporal logic specifications," in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, 2019, pp. 237–248.

[24] L. Z. Yuan, M. Hasanbeig, A. Abate, and D. Kroening, "Modular deep reinforcement learning with temporal logic specifications," 2019, *arXiv:1909.11591*.

[25] M. Hasanbeig, D. Kroening, and A. Abate, "Deep reinforcement learning with temporal logics," in *Proc. Int. Conf. Formal Model. Anal. Timed Syst.*, Springer, 2020, pp. 1–22.

[26] C. Wang, Y. Li, S. L. Smith, and J. Liu, "Continuous motion planning with temporal logic specifications using deep neural networks," 2020, *arXiv:2004.02610*.

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

[28] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, 1992.

[29] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," 2021, *arXiv:2102.12855*.

[30] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Mach. Learn.*, vol. 49, no. 2-3, pp. 209–232, 2002.

[31] R. Durrett and R. Durrett, *Essentials of Stochastic Processes*, vol. 1. Berlin, Germany: Springer, 1999.

[32] J. Kretínský, T. Meggendorfer, and S. Sickert, "Owl: A library for ω-words, automata, and LTL," in *Proc. Autom. Tech. Verif. Anal.*, Springer, 2018, pp. 543–550.