T4NMTD: Transition-Centric Reinforcement Learning for Non-Markovian Task Decomposition

Ruixuan Miao[1*], Xu Lu[1*], Cong Tian[1†], Bin Yu[1], Zhenhua Duan[1]

[1]Institute of Computing Theory and Technology and State Key Laboratory of ISN, Xidian University, PR China
22031212106@stu.xidian.edu.cn, {xlu, byu}@xidian.edu.cn, {ctian, zhhduan}@mail.xidian.edu.cn

Abstract

Non-Markovian Tasks (NMTs) are distinguished by their dependence on long-term memory and state-dependent dynamics, setting them apart from the traditional Markovian models typically employed in Reinforcement Learning (RL). NMTs not only suffer from reward sparseness but also rely on historical information, making their resolution considerably more challenging. In this paper, we propose a novel RL framework T4NMTD (Transition-centric framework for NMT Decomposition), designed specifically for learning NMTs which are specified by temporal logic. The core of T4NMTD is a task decomposition mechanism along with a parallel training approach for NMTs. An NMT is first decomposed as basic units based on the transitions of the automata which are derived from temporal logic formulae. The units are then modularized into sub-tasks according to their semantic similarity under logical interpretation. The training strategy of T4NMTD adopts a dual-level structure: the high-level learns to shape the boundaries and coordinate arrangement of the sub-tasks from a global perspective, while the low-level learns those sub-tasks in parallel. In addition, we invent a dynamic policy intervention scheme to mitigate the policy myopic issue during parallel training. A comprehensive evaluation is conducted on benchmark problems with respect to various metrics. The experimental results demonstrate that T4NMTD effectively addresses NMTs, achieving significant performance improvements compared with related studies.

Code — https://github.com/syemichel/T4NMTD

Introduction

Reinforcement Learning (RL) is a well-known paradigm for autonomous decision-making in complex and unknown environments (Sutton and Barto 2018). It is used to help agents learn a policy in pursuit of goals, from trial-and-error experiences towards maximizing long-term rewards. In most RL settings, Markov Decision Processes (MDPs) serve as the mathematical foundation, where the reward at each time relies only on the current state and action. Nevertheless, in the real-world scenario, there are many complex tasks whose behaviors are deeply reflected over historical sequences of states and actions. Such tasks are known as Non-Markovian Tasks (NMTs). As an example, an autonomous taxi may pick up passengers and subsequently deliver them to their respective destinations. Most of the state-of-the-art RL algorithms that attempt to learn NMTs without realizing that they are non-Markovian will display sub-optimal behaviors since there is little guidance for their relatively myopic lookahead (Camacho et al. 2017).

Applying RL to NMTs has gained a lot of attractions recently. Most studies employ Linear Temporal Logic (LTL) (Pnueli 1977) or its variants as high-level languages to specify temporal dependencies in NMTs. Researchers typically derive reward functions from automata structures through LTL-to-automaton translation, thereby providing suitable guidance for RL agents. However, challenges such as sparse reward signals and long-term decision dependencies still hinder effective exploration of future states, leading to prolonged training times or even task failure.

In order to overcome the above problems, we propose a novel RL framework for dealing with NMTs specified by temporal logic formulae, referred to as T4NMTD (Transition-centric framework for NMT Decomposition). The major contribution of T4NMTD is to introduce a task-decomposition module and a parallel-training module in traditional RL settings. The former module is responsible for decomposing an NMT into simpler sub-tasks, while the latter one is used to learn the sub-tasks in a parallel manner. What we mean by "transition-centric" is that the task-decomposition module is based on automata transitions constructed from temporal logic specifications. The parallel-training module adopts a hierarchical learning strategy, including the coordination of sub-tasks (high-level) with their realizations (low-level).

Related Work

The work in (Camacho et al. 2018) introduces a means of specifying non-Markovian rewards, expressed in LTL_f (Linear Temporal Logic over finite traces) (De Giacomo and Vardi 2013). Non-Markovian reward functions are encoded as automata representations, and off-the-shelf MDP planners leverage these reshaped automata-based rewards to guide their search process. This approach is quite limited, as it lacks extensibility to continuous domains. (Icarte et al. 2022) propose a more generalized form of NMTs, referred

*These authors contributed equally.
†Corresponding author.

to as reward machines, which are inherently a special kind of finite state automaton. Various methodologies based on Q-learning and Hierarchical RL (HRL) are leveraged to cooperate with reward machines, including automated reward shaping, task decomposition, and counterfactual reasoning for data augmentation. However, the authors acknowledge a primary drawback that reward shaping in their approach does not help in continuous domains. Using a similar formal specification to LTL$_f$, the Logical Specifications-guided dynamic Task Sampling (LSTS) (Shukla et al. 2024) employs an adaptive teacher-student learning approach to address tasks with temporal dependencies. LSTS makes steady advances by systematically progressing through each subtask, with every step bringing it closer to the final objectives defined in the DAG (Directed Acyclic Graph) structure corresponding to the specification. DIRL (Jothimurugan et al. 2021) is a compositional RL approach to learn NMTs by leveraging the DAG structure. DIRL utilizes Dijkstra's algorithm to prioritize sub-tasks (edges in the DAG) for exploration, aiming to learn policies that maximize success rates in reaching specific nodes. It depends on manually specified interaction limits for each sub-task, requiring prior knowledge of task complexity.

A model-free RL algorithm is presented in (Hasanbeig, Kroening, and Abate 2023), that enables the use of LTL to specify a goal for unknown continuous-state/action MDPs. An LTL specification is converted to a Limit Deterministic Büchi Automaton (LDBA) and synchronised on-the-fly with the agent/environment. The authors develop a modular Deep Deterministic Policy Gradient (DDPG) framework designed to produce a control policy that maximizes the probability of the given LTL specification. This synchronisation process automatically decomposes a complex global task into subtasks, effectively mitigating the problem of reward sparsity. The literature (Voloshin et al. 2022) studies the problem of policy optimization with LTL constraints. A learning algorithm is derived based on a reduction from the product of MDP and LDBA to a reachability problem. This approach enjoys a sample complexity analysis for ensuring task satisfaction and cost optimality. Analogously, an RL framework is presented in (Bozkurt et al. 2020) to synthesise a control policy from a product of MDP and LDBA. The novelty of the framework is to introduce a novel rewarding and discounting scheme based on the Büchi acceptance condition.

Preliminaries

Linear Temporal Logic over Finite Traces

LTL$_f$ is a variant of LTL concentrating on expressing temporal properties over finite traces. In this paper, LTL$_f$ is employed to specify NMTs. Let AP be a set of atomic propositions. The syntax of LTL$_f$ is defined as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathrm{U} \varphi_2$$

where $p \in AP$ is an atomic proposition, $\bigcirc$(next) and U(until) are temporal operators. Propositional binary connectives $\vee, \rightarrow, \leftrightarrow$ and boolean values $\top$ (true), $\bot$ (false) can be derived in terms of basic operators. Other temporal operators can also be expressed. For example, $\Diamond$(eventually) and $\Box$(always) are defined by: $\Diamond\varphi \equiv \top \mathrm{U} \varphi, \Box\varphi \equiv \neg\Diamond\neg\varphi$. $\bigcirc\varphi$ denotes that φ holds in the next state which must exist. $\varphi_1 \mathrm{U} \varphi_2$ indicates that φ_1 holds until φ_2 is true. $\Diamond\varphi$ means that φ will eventually hold before or right in the last state. $\Box\varphi$ represents that φ holds along the whole trace.

A state s is a subset of AP that is true, while other propositions in $AP \backslash s$ are assumed to be false. The semantics of an LTL$_f$ formula is interpreted over a finite trace $\sigma = s_0 \dots s_n$. We say that σ satisfies φ, written as $\sigma \models \varphi$, when $\sigma, 0 \models \varphi$.

- $\sigma, i \models p$ **iff** $p \in s_i$.
- $\sigma, i \models \neg\varphi$ **iff** $\sigma, i \not\models \varphi$.
- $\sigma, i \models \varphi_1 \wedge \varphi_2$ **iff** $\sigma, i \models \varphi_1$ and $\sigma, i \models \varphi_2$.
- $\sigma, i \models \bigcirc\varphi$ **iff** $i < n$ and $\sigma, (i+1) \models \varphi$.
- $\sigma, i \models \varphi_1 \mathrm{U} \varphi_2$ **iff** there exists $i \leq j \leq n$ such that $\sigma, j \models \varphi_2$, and $\sigma, k \models \varphi_1$ for each $i \leq k < j$.

Theoretically, every LTL$_f$ formula can be transformed to a Deterministic Finite Automaton (DFA) that recognizes the same language (De Giacomo and Favorito 2021). A DFA is a tuple $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, where Q is a finite set of states, Σ is a finite set of input alphabet, $\delta : Q \times \Sigma \rightarrow Q$ is a transition function, $q_0 \in Q$ is an initial state, $F \subseteq Q$ is a set of accepting states. For a transition $\tau = (q, l, q') \in \delta$, l is called the label of τ. Let $\neg AP = \{\neg p \mid p \in AP\}$ be the set of negations of the propositions in AP. The transition dynamics of a DFA is defined as finite words or traces over the alphabet $\Sigma = 2^{AP \cup \neg AP}$. We say that $\mathcal{A}$ accepts $\sigma = s_1 \dots s_n$ if there exists $\rho = q_0 \dots q_n$ such that $q_{i+1} = \delta(q_i, l_{i+1})$ for $0 \leq i < n$, s_{i+1} satisfies l_{i+1} and $q_n \in F$. σ (or ρ) is called an accepting *state trace* (or *DFA trace*). The states $E \subseteq Q$ that can never reach F are regarded as the *error states*. To make it more clear, we distinguish s an environment state and q a DFA state in the sequel.

MDP and NMDP

An MDP M is defined as a tuple $\langle S, A, R, P, \gamma, s_0 \rangle$. Here, S is a set of states, A is a set of actions, $R : S \times A \times S \longrightarrow \mathbb{R}$ is a reward function, $P(s_{t+1}|s_t, a_t) \in [0, 1]$ is a transition probability distribution over the set of next states, given that the agent takes action a_t in state s_t at step t and reaches state s_{t+1}, γ is the discounted factor, $s_0 \in S$ is the initial state.

MDPs are limited in expressiveness due to its memoryless feature, while Non-Markovian Decision Processes (NMDPs) are more powerful by extending MDPs with non-Markovian rewards, which is suitable for modeling NMTs (Thiébaux et al. 2006). An NMDP is a tuple $NM = \langle S, A, R, P, \gamma, s_0 \rangle$, where S, A, P, γ, s_0 are the same as those in an MDP. The only difference is the reward function R, which is defined as $(S \times A)^* \rightarrow \mathbb{R}$. This function indicates that non-Markovian rewards are governed by a finite sequence of states and actions (memory). Consequently, LTL$_f$ is a perfect match with non-Markovian rewards, which is defined in terms of a pair $R = \langle \varphi, r \rangle$, where φ is an LTL$_f$ formula (reward formula) and $r \in \mathbb{R}$ is the associated reward.

Since the standard formulation of RL is based on MDP, advanced RL algorithms cannot be directly used to learn NMTs modeled by NMDP. To address this problem, analogous to priori researches, we apply a synchronisation

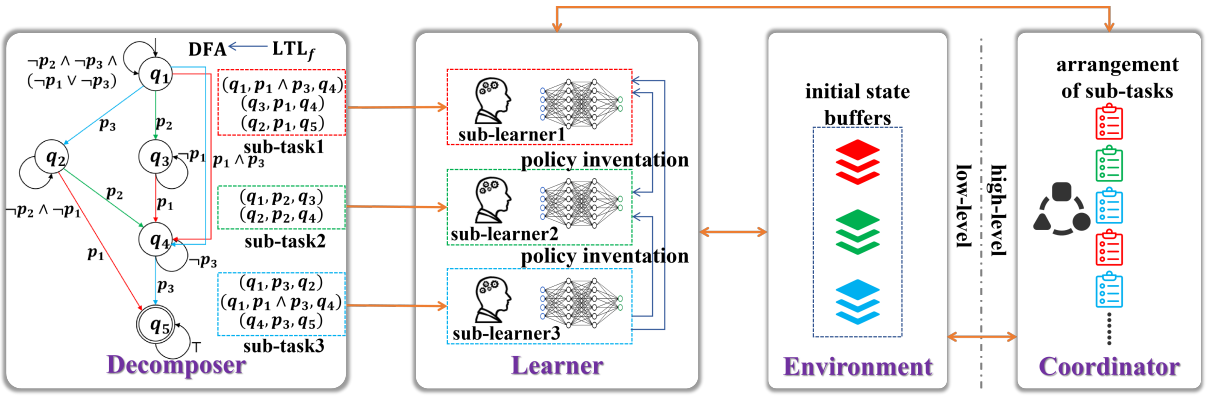

Figure 1: An overview of T4NMTD framework

technique to generate a product of an NMDP and an LTL$_f$ formula (in fact a DFA) in a on-the-fly fashion. Given an NMDP $NM = \langle S, A, R = \langle \varphi, r \rangle, P, \gamma, s_0 \rangle$ and a DFA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ with respect to the reward formula φ, the converted MDP is a product $M^{\otimes} = \langle S^{\otimes}, A, R^{\otimes}, P^{\otimes}, \gamma, s_0^{\otimes} \rangle$, where $S^{\otimes} = S \times Q$ and $s_0^{\otimes} = \langle s_0, q_0 \rangle$. For $s^{\otimes} = \langle s, q \rangle \in S^{\otimes}$, s is the *state ingredient* of $s^{\otimes}$, and q is the *DFA ingredient* of $s^{\otimes}$. The reward function is defined as:

$$R^{\otimes}(s_t^{\otimes}, a_t, s_{t+1}^{\otimes}) = \begin{cases} r, & s_{t+1}^{\otimes} = \langle s_{t+1}, q_{t+1} \rangle, q_{t+1} \in F \\ 0, & \text{otherwise} \end{cases}$$

which means that the agent receives a reward r only when the DFA ingredient of $s_{t+1}^{\otimes}$ is an accepting state. The transition probability $P^{\otimes}(s_{t+1}^{\otimes} \mid s_t^{\otimes}, a_t)$ equals $P(s_{t+1} \mid s_t, a_t)$ if there exists a possible DFA transition enabled by s_t from the DFA ingredient of $s_t^{\otimes}$ to that of $s_{t+1}^{\otimes}$, and 0 otherwise, which is formalized below:

$$P^{\otimes}(s_{t+1}^{\otimes} \mid s_t^{\otimes}, a_t) = \begin{cases} P(s_{t+1} \mid s_t, a_t), & s_t^{\otimes} = \langle s_t, q_t \rangle, s_{t+1}^{\otimes} = \langle s_{t+1}, q_{t+1} \rangle, \\ & \exists l \in \Sigma : \langle q_t, l, q_{t+1} \rangle \in \delta, s_t \text{ satisfies } l \\ 0, & \text{otherwise} \end{cases}$$

Transition-Centric RL Framework for Learning Non-Markovian Tasks

Figure 1 depicts the overall workflow of T4NMTD, which consists of four key components: Decomposer, Coordinator, Learner, and Environment.

The Decomposer is responsible for modularizing NMTs into a group of sub-tasks, which enables the framework to tackle intricate temporal dependencies through a divide-and-conquer paradigm. Intuitively, it clusters semantically similar transitions into individual sub-tasks.

The Coordinator and the Learner together form a hierarchical learning structure. At the lower level, the Learner assigns a dedicated sub-learner to each sub-task. Each sub-learner maintains its own environment. A policy intervention mechanism is introduced during the parallel training process in order to mitigate the issue of myopic learning. At the higher level, the Coordinator serves as a global decision-maker, orchestrating the scheduling of the sub-tasks.

Unlike conventional RL environment setup, the Environment is uniquely characterized by its initial state buffers. They are used to store environment states that designated as initial states for sub-learners, allowing them to initiate interactions from appropriate starting points rather than always from the environment's default initial state.

Task Decomposition

The foremost step of T4NMTD involves task decomposition. Given an NMT specified by an LTL$_f$ formula, we first translate it into a DFA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ with E denoting the set of error states. The intuition behind the decomposition is that the transitions of $\mathcal{A}$ that share a common element in their labels are categorized into a single sub-task. The number of sub-task categories stems from the total number of *positive and negative* propositions appearing in the labels of δ without self loops. Formally, the set of decomposed sub-tasks is given by $\mathcal{T} = \{\mathcal{T}_\iota \mid \exists \tau \in \delta : \tau = (q, l, q'), q \neq q', \iota \in l\}$, and ι is called the *pid* of $\mathcal{T}_\iota$. For example, as illustrated by the DFA in Figure 1, there are three sub-tasks whose pids are p_1, p_2 and p_3 respectively.

Each transition is classified by the decomposing mapping $\mathcal{D} : \delta \longrightarrow \mathcal{T}$, defined as $\mathcal{D}(\tau) = \{\mathcal{T}_\iota \mid \exists \iota : \tau = (q, l, q'), \iota \in l\}$. The sub-task $\mathcal{T}_{p_1}$ in Figure 1 contains the transitions $(q_1, p_1 \wedge p_3, q_4)$, (q_2, p_1, q_5) and (q_3, p_1, q_4) that share the proposition p_1. In the worst case, decomposing an NMT may result in a number of sub-tasks that is up to twice that of the propositions in the corresponding LTL$_f$ formula. The complexity grows linear in the number of propositions, and thus computationally manageable. Note that a transition may be associated with different sub-tasks. In the DFA of Figure 1, the transition $(q_1, p_1 \wedge p_3, q_4)$ has dual memberships in the sub-tasks $\mathcal{T}_{p_1}$ and $\mathcal{T}_{p_3}$. Through this categorization, transitions within each sub-task possess a common objective, i.e., manipulating the truth value of a specific proposition. Therefore, this objective reflects semantic similarity, which brings two advantages. On the one hand, it improves sample efficiency when learning a sub-task. One the other hand, it facilitates better transfer performance when adapting to new NMTs.

Algorithm 1: Pseudo-code for T4NMTD

Input: $\mathcal{T}, M^{\otimes}, \mathcal{A}$

1: $\left\{\Theta_\iota(a \mid s^{\otimes}), \Omega_\iota(s^{\otimes}, a), \mathcal{B}_\iota = \emptyset, \mathcal{R}_\iota = \emptyset \mid \mathcal{T}_\iota \in \mathcal{T}\right\}$
2: $\mathcal{Q} = Q \times \mathcal{T} \times \{-\infty\}$
3: **for** $\mathcal{T}_\iota$ **in** $\mathcal{T}$ **do**
4: create and start a thread **SL**($\mathcal{T}_\iota, M^{\otimes}, \mathcal{A}$)
5: **end for**
6: **return** $\langle \bigcup_{\mathcal{T}_\iota \in \mathcal{T}} \Theta_\iota, \mathcal{Q} \rangle$

Algorithm 2: Pseudo-code for Sub-task Learning (**SL**)

Input: $\mathcal{T}_\iota, M^{\otimes}, \mathcal{A}$

1: $s_1^{\otimes} \leftarrow \langle s_1, q_1 \rangle \in \mathcal{B}_\iota$
2: **for** $t \leftarrow 1$ **to** $total\_steps$ **do**
3: choose action $a_t \sim \Theta_\iota(s_t^{\otimes})$
4: take a_t, observe $s_{t+1}^{\otimes} = \langle s_{t+1}, q_{t+1} \rangle$ and receive r_t
5: reshape reward $r_t \leftarrow (r_t + \gamma\rho(q_{t+1}) - \rho(q_t)) \cdot \mathbf{1}_{q_{t+1} \in Q_{target}(q_t, \iota)}$
6: save experience $\langle s_t^{\otimes}, a_t, r_t, s_{t+1}^{\otimes} \rangle$ in $\mathcal{R}_\iota$
7: **if** $q_{t+1} \in Q_{target}(q_t, \mathcal{T}_\iota)$ and $q_{t+1} \notin F$ **then** save $s_{t+1}^{\otimes}$ in $\mathcal{B}_{\iota'}$ with $\mathcal{T}_{\iota'} \sim \mathcal{E}(\mathcal{Q}, q_{t+1})$
8: **if** $q_t \neq q_{t+1}$ **then** $\mathcal{Q}(q_t, \iota) \leftarrow \mathcal{Q}(q_t, \mathcal{T}_\iota) + \alpha(r_t + \gamma \max_{\mathcal{T}'_\iota} \mathcal{Q}(q_{t+1}, \mathcal{T}'_\iota) - \mathcal{Q}(q_t, \mathcal{T}_\iota))$
9: **if** $q_{t+1} \in Q_{target}(q_t, \mathcal{T}_\iota)$ or $q_{t+1} \in E$ **then** $s_{t+1}^{\otimes} \leftarrow \langle s_{t+1}, q_{t+1} \rangle \in \mathcal{B}_\iota$
10: **if** $t \bmod update\_interval = 0$ **then**
11: sample experience batch $\left\{\langle s_i^{\otimes}, a_i, r_i, s_{i+1}^{\otimes} \rangle\right\}_{i=1\ldots n}$ from $\mathcal{R}_\iota$
12: calculate critic loss $L_{\Omega_\iota} = \frac{1}{n}\sum_i^n (\Omega_\iota(s_i^{\otimes}, a_i) - (r_i + \gamma\Omega_{\iota'}(s_{i+1}^{\otimes}, a_{i+1})))^2$ with $\mathcal{T}_{\iota'} \sim \mathcal{E}(\mathcal{Q}, q_{t+1})$ **if** $q_i \neq q_{i+1}$ and $q_{i+1} \notin F \cup E$ **else** $\iota' = \iota$
13: calculate actor loss $L_{\Theta_\iota} = -\frac{1}{n}\sum_i^n \log(\Theta_\iota(a_i | s_i^{\otimes})) \cdot \Omega_\iota(s_i^{\otimes}, a_i)$
14: update networks Θ_ι and Ω_ι
15: **end if**
16: **end for**

Hierarchical Learning

In the next phase, sub-tasks are learned in parallel. The pseudo-code for T4NMTD is shown in Algorithm 1, which is built on an actor-critic RL procedure.

The input comprises a set of decomposed sub-tasks $\mathcal{T}$, a product MDP $M^{\otimes} = \left\langle S^{\otimes}, A, R^{\otimes}, P^{\otimes}, \gamma, s_0^{\otimes} \right\rangle$, and a DFA $\mathcal{A}$ corresponding to a given NMT. At the beginning, line 1 initializes an actor and a critic networks $\Theta_\iota, \Omega_\iota$, an initial state buffer $\mathcal{B}_\iota$ and an experience buffer $\mathcal{R}_\iota$ for each sub-task $\mathcal{T}_\iota \in \mathcal{T}$. In addition, a dictionary $\mathcal{Q}$ is initialized representing the Q-values by mapping a pair $(q, \mathcal{T}_\iota)$ to an associated value, where $q \in Q$, $\mathcal{T}_\iota \in \mathcal{T}$. In short, $\mathcal{Q}$ is used to evaluate the expected return of performing a sub-task from certain DFA state. Details on how $\mathcal{Q}$ is computed will be provided later. Lines 3–5 launch parallel learning for all sub-tasks, each running in its own thread. Finally, the global policy can be adaptively synthesized from sub-policies, guided by high-level decisions, i.e., $\mathcal{Q}$.

The notation $Q_{target}(q, \mathcal{T}_\iota)$ denotes the target DFA states that refer to the possible successor DFA states from q, reached after enabling the transitions in $\mathcal{T}_\iota$ during training. More precisely, $Q_{target}(q, \mathcal{T}_\iota)$ is formalized as $\{q' \mid \exists \tau \in \mathcal{T}_\iota : \tau = (q, l, q')\}$. Algorithm 2 describes a single sub-task learning thread for $\mathcal{T}_\iota$. Whenever resetting (lines 1 and 9), a state is sampled from the initial state buffer $\mathcal{B}_\iota$ to initialize the environment.

In line 5, we adopt the idea of reward shaping from (Miao et al. 2024) to augment the learning process with additional reward signals. Concretely, the reshaped reward function is defined as:

$$R^{\otimes'}(s_t^{\otimes}, a_t, s_{t+1}^{\otimes}) = R(s_t^{\otimes}, a_t, s_{t+1}^{\otimes}) + (\gamma\rho(q_{t+1}) - \rho(q_t)) \cdot \mathbf{1}_{q_{t+1} \in Q_{target}(q_t, \mathcal{T}_\iota)}$$

where $\rho(q) = C/(dist(q) + 1)$ is the potential function for $q \in Q \setminus F \setminus E$, C is a positive constant, $dist(q)$ is the average distance from q to accepting states (each transition is counted as unit distance) and $\mathbf{1}_*$ is an indicator function which equals 1 if $*$ is true and 0 otherwise. The potential value of $q \in F$ and $q \in E$ is set to C and $C/(D_{max} + 1)$ respectively, where D_{max} denotes the maximum such distance across all states. Intuitively, the smaller $dist(q)$ is, the higher the completion degree of the NMT is roughly considered.

If the current DFA state transitions to a target DFA state (line 7), the next environment state is stored in a selected initial state buffer $\mathcal{B}_{\iota'}$, which is sampled using the ϵ-greedy strategy:

$$\mathcal{T}_{q_{t+1}}^{\mathcal{Q}} = \{\mathcal{T}_{\iota'} \mid \exists \tau \in \mathcal{T}_{\iota'}, \mathcal{T}_{\iota'} \in \mathcal{T} : \tau = (q_{t+1}, l, q), \iota' \in l\}$$

$$\mathcal{T}_{\iota'} \sim \mathcal{E}(\mathcal{Q}, q_{t+1}) = \begin{cases} \arg\max_{\mathcal{T}_{\iota'} \in \mathcal{T}_{q_{t+1}}^{\mathcal{Q}}} \mathcal{Q}(q_{t+1}, \mathcal{T}_{\iota'}), 1 - \epsilon \\ \text{random sampling from } \mathcal{T}_{q_{t+1}}^{\mathcal{Q}}, \epsilon \end{cases}$$

$\mathcal{B}_{\iota'}$ satisfies the condition that performing its corresponding sub-task $\mathcal{T}_{\iota'}$ from q_{t+1} yields the highest Q-value. The selection of $\mathcal{B}_{\iota'}$ is crucial to ensuring that each sub-learner operates within its own isolated environment. Again, when a transition occurs, regardless of which sub-task it belongs to, the Q-value of $\mathcal{Q}(q_t, \mathcal{T}_\iota)$ is updated according to the standard Q-learning update rule in line 8. The sub-learner is able to progressively refine its understanding of which sub-tasks are most valuable to perform in different DFA states, ultimately leading to more effective hierarchical decision making.

Lines 10-15 implement the actor-critic training procedure for $\mathcal{T}_\iota$. It is worth noting that the loss calculation for the critic networks fundamentally differs from that of standard actor–critic algorithms. The key innovation lies in the introduction of dynamic policy intervention, where action-value estimation integrates values from other sub-task networks. Without loss of generality, given an experience $\langle s_t^{\otimes} = \langle s_t, q_t \rangle, a_t, r_t, s_{t+1}^{\otimes} = \langle s_{t+1}, q_{t+1} \rangle \rangle$, the temporal difference error is:

$$\Omega_\iota(s_t^{\otimes}, a_t) - (r_t + \gamma\Omega_{\iota'}(s_{t+1}^{\otimes}, a_{t+1}))$$

where ι' is dynamically determined by $\mathcal{E}(\mathcal{Q}, q_{t+1})$ when a DFA state transition in $\mathcal{T}_\iota$ occurs, and set to ι otherwise. We aim to address a major limitation: each sub-learner can only collect experiences within its own horizon and cannot accurately evaluate action-values in the contexts of other sub-tasks, resulting in myopic policies. The policy intervention

propagates action-values from closely related sub-learners, thereby assisting each sub-learner in training a more far-sighted policy. Consider the DFA in Figure 1, the transition (q_1, p_3, q_2) is part of the sub-task $\mathcal{T}_{p_3}$. Suppose that the thread of $\mathcal{T}_{p_3}$ has just triggered that transition and generated an experience $\langle s_t^{\otimes} = \langle s_t, q_1\rangle, a_t, r_t, s_{t+1}^{\otimes} = \langle s_{t+1}, q_2\rangle\rangle$. Before calculating the loss, we do the sampling $\mathcal{T}_{\iota'} \sim \mathcal{E}(\mathcal{Q}, q_2)$. Since there are two transitions from q_2, i.e., $(q_2, p_1, q_5) \in \mathcal{T}_{p_1}$ and $(q_2, p_2, q_4) \in \mathcal{T}_{p_2}$, it follows that the loss update of Ω_{p_3} depends on either Ω_{p_1} or Ω_{p_2} at this step. The larger one between $\mathcal{Q}(q_2, p_1)$ and $\mathcal{Q}(q_2, p_2)$ determines the subscript of the network.

Evaluation

We implement T4NMTD as an open source tool built on top of Stable-Baselines3 (SB3) library (Raffin et al. 2021). Concretely, we utilize SAC (Soft Actor-Critic) (Haarnoja et al. 2018) and PPO (Proximal Policy Optimization) (Schulman et al. 2017) from SB3 to perform the evaluations, with SAC applied to continuous environments and PPO to discrete ones. The hyperparameters of SAC and PPO are configured by default in SB3.

Experiment Setup

Empirical evaluations were performed on four benchmark problems: *Waterworld*, *Halfcheetah*, *Racecar*, and *Frozenlake*. The first three environments have continuous spaces, whereas *Frozenlake* features discrete ones.

We randomly generate maps for *Waterworld* within a 30×30 bounded square, where each map contains balls of different colors positioned at various locations with different velocities. *Halfcheetah* features a 2D robot with 9 links connected by 8 joints, where the agent controls joint forces to move the robot forward or backward along a track of length 30. *Racecar* involves navigating a race car around a circular track with radius 50, where the car must reach different targets in a specified sequence. *Frozenlake* presents an 8×8 grid world where the agent must transport appropriate passengers to designated locations.

The problems are augmented with several sophisticated NMTs, which are listed in Table 1 (see a more complete version in the Technical Appendix). We take Task1 and Task7 as examples to explain the meanings of the tasks. Task1 requires the agent to touch the colored balls in two strict sequential orders: first red (r), then blue (b), then green (g), and first black (B), then white (W), then grey (G). Task7 requires guiding one of the three passengers (p_1, p_2 or p_3) to the goal, without guiding any two of them simultaneously. Note that the major difference between Tasks 1-4 and Tasks 5-6 is whether the subgoals are finished in a strict order or not (strict then/then). We exploit the open source tool LTL$_f$2DFA (Fuggitti 2019) to translate LTL$_f$ formulae into DFAs. The last column shows the number of DFA states (#S) and transitions (#T) corresponding to NMTs.

A reward of 100 will be received if an NMT is completed. Following the standard training process, a policy is evaluated periodically at fixed intervals. To better reflect the optimization trend of the policy, we modify the reward function **only** in the validation phase:

$$R(s_t^{\otimes}, a_t, s_{t+1}^{\otimes}) = (\rho(q_{t+1}) - \rho(q_t)) \cdot \mathbf{1}_{q_{t+1} \notin E}$$

where $q_{t+1} \notin E$ is a condition used to check whether an error state is reached. In this way, a higher reward indicates a higher degree of task completion.

Environment	NMT Description	#S/#T
Waterworld (continuous) (Karpathy 2015)	**Task1:** (r strict-then b strict-then g) and (B strict-then W strict-then G)	17/65
	Task2: (r then b then g then p) and (B then W then G then Y)	26/106
Halfcheetah (continuous) (Icarte et al. 2022)	**Task3:** h_4 strict-then h_3 strict-then h_2	5/11
	Task4: (h_4 strict-then h_3 strict-then h_2) or (h_2 strict-then h_3 strict-then h_4)	9/25
Racecar (continuous) (IPPC2023)	**Task5:** g_1 then g_2 then g_3 then g_4	5/16
	Task6: (g_1 then g_2 then g_3 then g_4) or (g_4 then g_3 then g_2 then g_1)	18/69
Frozenlake (discrete) (Brockman et al. 2016)	**Task7:** guide one of $\{p_1, p_2, p_3\}$ to g, one at a time	6/26
	Task8: guide one of $\{p_1, p_2\}$ to g_1 and p_3 to g_2, one at a time	11/47

Table 1: Descriptions of the NMTs for *Waterworld*, *Halfcheetah*, *Racecar*, and *Frozenlake*

Experimental Results

In the **first experiment**, we compare the performance of T4NMTD with six relevant works (baselines), i.e., Mod, QRM, HRM, HDQN, LSTS and DIRL. Mod denotes Modular DDPG, which is the modular architecture in (Hasanbeig, Kroening, and Abate 2023). QRM and HRM refer to Q-learning for Reward Machines and Hierarchical RL for Reward Machines proposed in (Icarte et al. 2022). HDQN is a typical and well-established HRL algorithm (Kulkarni et al. 2016). LSTS is also an HRL approach specifically designed for NMTs (Shukla et al. 2024), and is the one closest to ours. DIRL is a compositional learning approach for NMTs using DAG structures (Jothimurugan et al. 2021). We train each NMT with the above methods, repeating 10 times and collecting the average performance. Note that the reward shaping technique of T4NMTD is applied to the baselines as well for fair comparison.

Figure 2 shows the comparison results on convergence rate. The horizontal axis represents the training time, and the vertical axis denotes the average rewards per episode. In summary, the reward curves of T4NMTD are the best for all NMTs, substantially outperforming the baselines. This highlights its superior performance in both discrete and continuous environments. In comparison, although the baselines exhibit gradual upward trends in their reward curves, they fail to converge within the time limit on most NMTs.

The performance of QRM and HRM are unsatisfactory. This is attributed to the counterfactual reasoning employed

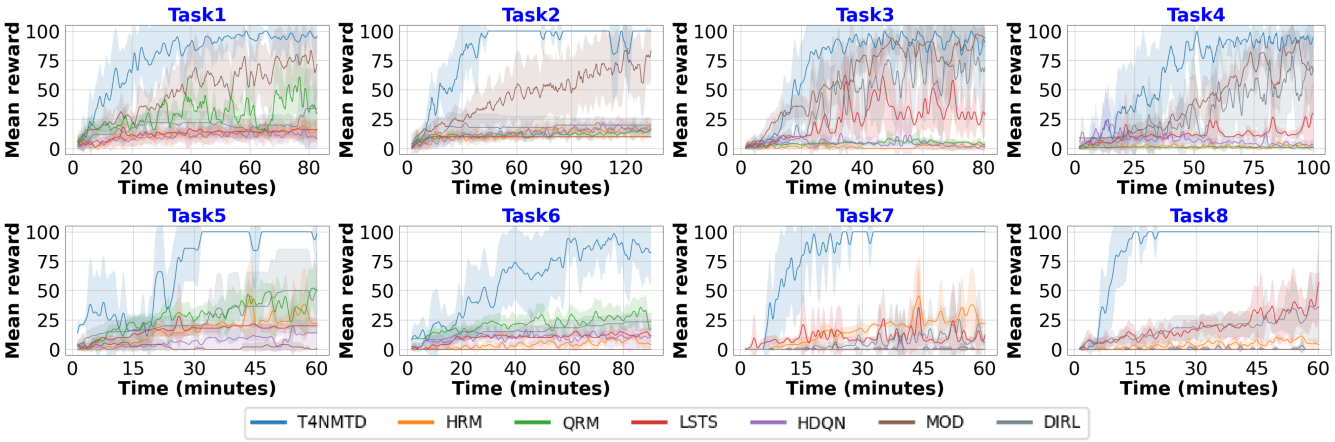

Figure 2: Comparison with baselines on convergence rate

in both methods, which generates inaccurate experiences and reward signals. As a result, these inaccuracies are detrimental to guiding the training process and ultimately impair the learning performance. HDQN adopts a two-level hierarchical structure similar to ours, in which the high-level network learns a policy over sub-goals and the low-level network learns policies to accomplish each sub-goal. The difference is that we have multiple learners at low-level, whereas HDQN uses only one. The decomposed sub-tasks are regarded as the sub-goals in HDQN. Obviously, the low-level network of HDQN must handle many sub-tasks which is not a trivial work. The results show that HDQN is prone to producing suboptimal policies across different sub-tasks. LSTS displays consistently poor performance for all NMTs. The underlying reason is that LSTS, which is based on the teacher-student architecture, treats each DAG edge as a separate sub-task. The NMTs in the experiments are too complex for LSTS, featuring an excessive number of transitions. Moreover, LSTS focuses on learning policies for promising sub-tasks which are thought to be easier to take a step forward, but lacks an effective coordination mechanism among sub-policies. Similar to LSTS, an DAG edge is considered as a sub-task in DIRL, but does not employ hierarchical learning technique. DIRL suffers from the same challenge of excessive transitions when dealing with complex NMTs. Mod only achieves much better performance than other baselines on Tasks 1-4, benefiting from its modularized architecture which is well-suited to NMTs with automata structures. Mod decomposes NMTs based on automata states, each of which corresponds to a sub-learner. There are two main factors why T4NMTD outperforms Mod. (1) Despite being modularized, Mod does not develop a parallel training process. (2) Mod encounters a similar issue to that faced by LSTS and DIRL, i.e., the number of sub-tasks is so large that it inevitably leads to computational bottlenecks.

Furthermore, although T4NMTD adopts a parallel training style, it does not necessarily imply that T4NMTD requires more experiences to learn NMTs, contrary to common intuition. To assess T4NMTD's performance in this regard, we compare its sample efficiency with that of the baselines, as reported in Table 2. Surprisingly, T4NMTD learns successful policies with the fewest interactions across all NMTs, while maintaining a perfect 100% success rate compared to baselines. Each sub-learner maintains relatively independent environment states, avoiding exploration of irrelevant states from other sub-tasks. This eliminates unnecessary interactions and allows it to focus exclusively on task-relevant state spaces. Additionally, the policy intervention mechanism facilitates timely value propagation, preventing excessive policy updates that may cause training instability.

Approach	Task1	Task2	Task3	Task4
T4NMTD	**0.8±0.1/100**	**1.5±0.1/100**	**1.3±0.1/100**	**1.5±0.2/100**
HRM	2.0/0	3.0/0	4.0/0	4.0/0
QRM	1.3±0.1/**100**	3.0/0	4.0/0	4.0/0
LSTS	2.0/0	3.0/0	**1.3±0.1/100**	3.1±0.6/**100**
HDQN	1.8±0.1/30	3.0/0	4.0/0	2.5±0.7/60
MOD	**0.8±0.1/100**	1.9±0.2/**100**	1.4±0.2/**100**	2.0±0.1/**100**
DIRL	2.9±0.1/20	3.0/0	3.5±0.7/**100**	3.7±0.9/**100**
	Task5	Task6	Task7	Task8
T4NMTD	**1.1±0.3/100**	**2.5±0.4/100**	**4.4±0.8/100**	**4.9±0.3/100**
HRM	2.3±0.5/40	4.0/0	19.1±0.9/10	20.0/0
QRM	2.5±0.4/16	4.0/0	20.0/0	20.0/0
LSTS	3.0/0	4.0/0	20.0/0	20.0/0
HDQN	3.0/0	4.0/0	20.0/0	20.0/0
MOD	3.0/0	4.0/0	20.0/0	20.0/0
DIRL	2.8±0.2/33	4.0/0	20.0/0	20.0/0

Table 2: Comparison with baselines on sample efficiency. Values represent #interactions ($\times 10^6$) / success rate (%).

The **second experiment** is conducted to demonstrate the effectiveness of the task decomposition mechanism. We introduce a baseline that randomly partitions DFA transitions into different sub-tasks, referred to as "random", while keeping the number of sub-tasks unchanged. The comparison results are presented in Figure 3. The task decomposition indeed helps improve the performance of T4NMTD and achieves higher convergence efficiency. This is due to the

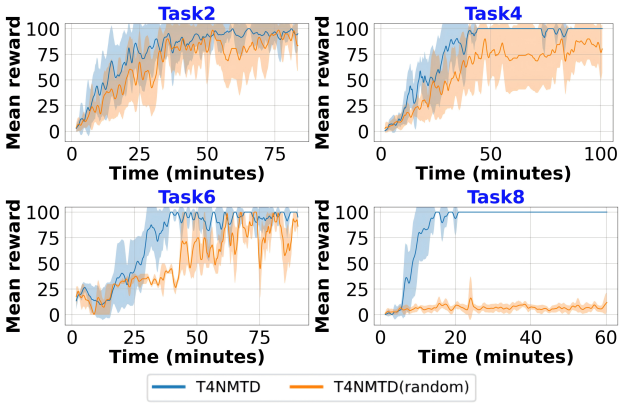

Figure 3: Comparison between proposition-based and random decomposition of NMTs

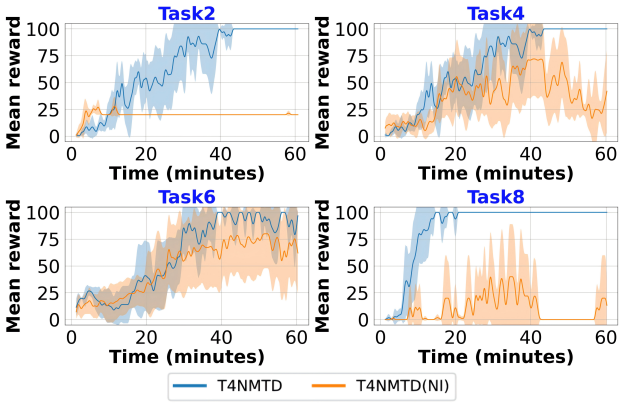

Figure 4: Comparison between T4NMTD with and without policy intervention

fact that our classification is inherently semantic, i.e., transitions within a sub-task share a similar or an identical objective. The boundaries between sub-tasks are relatively well-defined and narrow. The effect is particularly pronounced in Task8, where the reward curve drops sharply when sub-tasks are randomly grouped. In addition to the aforementioned reason, we speculate that forcing transitions with significant differences into the same sub-task may produce conflicting optimal policies. In this case, an environment state could satisfy contradictory truth values for a proposition, thereby substantially amplifying the learning difficulty.

The **third experiment** aims to answer the question: Does the policy intervention effectively mitigate the myopic issue? No intervention (named NI) indicates that each sub-task relies on itself during policy updates. We compare it with T4NMTD in Figure 4, and the results provide a positive answer. The observed performance degradation across all NMTs in the absence of policy intervention validates the effectiveness of our approach. Without policy intervention, each sub-learner optimizes its own sub-task policy independently, disregarding global considerations. The environment states achieved by finishing a transition in a sub-task may be unsuitable or even infeasible for initializing subsequent sub-

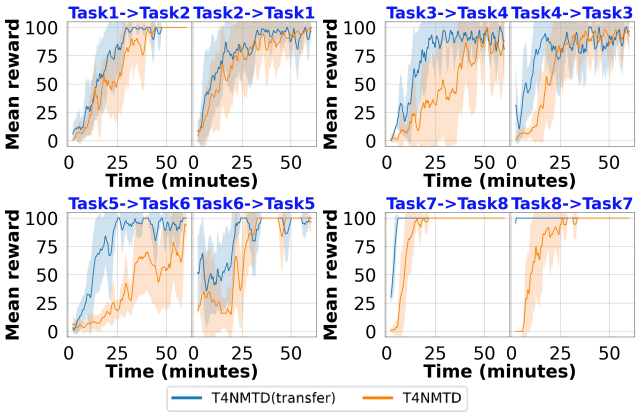

Figure 5: Transfer learning vs. From-scratch learning on convergence rate

tasks, creating a misalignment between sub-task boundaries.

The **final experiment** is designed to test the transfer learning performance of T4NMTD. In fact, propositions in states reflect the status of an environment, and they play a central role in expressing the state-level semantics of NMTs. Therefore, T4NMTD may adapt to task transfer more efficiently by leveraging the previously trained low-level sub-learners from prior sub-tasks. To evaluate this hypothesis, we examine four scenarios in which the NMTs are altered after their previous counterparts have converged (still in the same domain, e.g., Task1 transfers to Task2 and vice versa). Only the low-level networks remain unchanged, while the initial state buffers, replay buffers and the high-level $\mathcal{Q}$ are reinitialized. The results are shown in Figure 5, where "transfer" refers to the transfer reward curve. Not surprisingly, we observe that T4NMTD achieves better transfer learning performance in all scenarios, with learning curves surpassing training from scratch. The convergence rate improves by over 2.67×, where convergence is rigorously defined as achieving 100% success rate with stable policy lengths (std $< 10\%$) over five consecutive runs.

Summary and Discussion

Learning an NMT is challenging since an agent receives increasingly sparse rewards for complex, temporally-extended behaviors. In this paper, we propose a novel RL framework T4NMTD for parallel and modularized learning of NMTs. We specify NMTs using the formal language LTL_f. Based on the DFA structure transformed from LTL_f, T4NMTD decomposes an NMT into multiple sub-tasks which are effectively learned in a well-designed parallel training paradigm. The experimental results on benchmark problems demonstrate the effectiveness and feasibility of T4NMTD, showing significant improvements in convergence rate and sample efficiency versus related studies. In addition, preliminary results show that our approach has good transferability. Future work will focus on further exploiting the transferability of T4NMTD to enhance learning efficiency across loosely related NMTs, as well as applying it to a wider range of scenarios and domains to demonstrate the robustness.

Acknowledgments

The authors really appreciate for the reviewing efforts of the editors and the reviewers. This research is supported by National Natural Science Foundation of China (62372347, 62192734, 62202361, 62172322); Fundamental Research Funds for the Central Universities (ZYTS25076, YJSJ25012); Innovation Fund of Xidian University (YJSJ25012).

References

Bozkurt, A. K.; Wang, Y.; Zavlanos, M. M.; and Pajic, M. 2020. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 10349–10355. IEEE.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Camacho, A.; Chen, O.; Sanner, S.; and McIlraith, S. 2017. Non-markovian rewards expressed in LTL: guiding search via reward shaping. In *Proceedings of the International Symposium on Combinatorial Search*, volume 8, 159–160.

Camacho, A.; Chen, O.; Sanner, S.; and McIlraith, S. A. 2018. Non-Markovian rewards expressed in LTL: Guiding search via reward shaping (extended version). In *GoalsRL, a workshop collocated with ICML/IJCAI/AAMAS*.

De Giacomo, G.; and Favorito, M. 2021. Compositional approach to translate LTLf/LDLf into deterministic finite automata. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, 122–130.

De Giacomo, G.; and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 854–860. Association for Computing Machinery.

Fuggitti, F. 2019. LTLf2DFA. https://github.com/whitemech/LTLf2DFA. Accessed: 2025-11-18.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

Hasanbeig, H.; Kroening, D.; and Abate, A. 2023. Certified reinforcement learning with logic guidance. *Artificial Intelligence*, 322: 103949.

Icarte, R. T.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2022. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73: 173–208.

Jothimurugan, K.; Bansal, S.; Bastani, O.; and Alur, R. 2021. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 34: 10026–10039.

Karpathy, A. 2015. Reinforcejs: Waterworld demo. http://cs.stanford.edu/people/karpathy/reinforcejs/waterworld.html. Accessed: 2025-11-18.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29.

Miao, R.; Lu, X.; Tian, C.; Yu, B.; Cui, J.; and Duan, Z. 2024. Using experience classification for training non-Markovian tasks. *Expert Systems with Applications*, 124649.

Pnueli, A. 1977. The temporal logic of programs. In *18th annual symposium on foundations of computer science (sfcs 1977)*, 46–57. ieee.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1): 12348–12355.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shukla, Y.; Burman, T.; Kulkarni, A. N.; Wright, R.; Velasquez, A.; and Sinapov, J. 2024. Logical specifications-guided dynamic task sampling for reinforcement learning agents. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 532–540.

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

Thiébaux, S.; Gretton, C.; Slaney, J.; Price, D.; and Kabanza, F. 2006. Decision-theoretic planning with non-Markovian rewards. *Journal of Artificial Intelligence Research*, 25: 17–74.

Voloshin, C.; Le, H.; Chaudhuri, S.; and Yue, Y. 2022. Policy optimization with linear temporal logic constraints. *Advances in Neural Information Processing Systems*, 35: 17690–17702.