Reward-Machine-Guided, Self-Paced Reinforcement Learning

Cevahir Koprulu
Ufuk Topcu
The University of Texas at Austin, USA

Abstract

Self-paced reinforcement learning (RL) aims to improve the data efficiency of learning by automatically creating sequences, namely curricula, of probability distributions over contexts. However, existing techniques for self-paced RL fail in long-horizon planning tasks that involve temporally extended behaviors. We hypothesize that taking advantage of prior knowledge about the underlying task structure can improve the effectiveness of self-paced RL. We develop a self-paced RL algorithm guided by reward machines, i.e., a type of finite-state machine that encodes the underlying task structure. The algorithm integrates reward machines in 1) the update of the policy and value functions obtained by any RL algorithm of choice, and 2) the update of the automated curriculum that generates context distributions. Our empirical results evidence that the proposed algorithm achieves optimal behavior reliably even in cases in which existing baselines cannot make any meaningful progress. It also decreases the curriculum length and reduces the variance in the curriculum generation process by up to one-fourth and four orders of magnitude, respectively.

1 INTRODUCTION

The design of task sequences, i.e., curricula [Bengio et al., 2009], aims to reduce the sample complexity of teaching reinforcement learning (RL) agents complex behaviors. Given a target task, a common curriculum design approach is to begin with easier tasks and increase the difficulty in a gradual manner, which requires domain expertise to define what is easy or hard [Narvekar et al., 2020]. To eliminate the need for manual curriculum design, many studies such as Baranes and Oudeyer [2010], Svetlik et al. [2017], Andrychowicz et al. [2017] focus on automating the process of curriculum generation. Klink et al. [2020a] adopt self-paced learning [Kumar et al., 2010] in RL by developing an algorithm that creates a sequence of probability distributions over contexts [Hallak et al., 2015]. The dynamics, the reward function, and the initial state distribution of an environment change with respect to the context. Given a target context distribution, a self-paced RL algorithm iteratively generates context distributions that maximizes the expected discounted return, regularized by the Kullback-Leibler (KL) divergence from the target context distribution.

Although empirical evidence by Klink et al. [2021] suggests that self-paced RL outperforms the state-of-the-art curriculum learning methods [Florensa et al., 2018; Portelas et al., 2020], existing self-paced RL approaches work poorly in long-horizon planning tasks, which involve temporally extended behaviors. We focus on tasks where the reward depends on the history of states and actions. In other words, the reward function of such a long-horizon planning task is non-Markovian. A remedy to tasks that require temporally extended behaviors is to expose the high-level structural relationships to the agent [Singh, 1992; Parr and Russell, 1997]. Icarte et al. [2018a] use a type of finite-state machine, called reward machines, as the high-level structural...
knowledge to encode non-Markovian reward functions in RL.

We claim that exploiting the high-level structural knowledge about a long-horizon planning task can improve self-paced RL. To this end, we study self-paced RL for long-horizon planning tasks in which such knowledge is available a priori to the agent in the form of reward machines. Specifically, we focus on contextual long-horizon planning tasks, where the context parameterizes the dynamics and the non-Markovian reward function. The underlying temporal task structure remains the same irrespective of the context, hence a reward machine can encode all possible non-Markovian reward functions. We define a labeled contextual Markov decision process (MDP) to model such long-horizon planning tasks (see Figure 1).

Contribution. Our contribution is three-fold. 1) We propose an intermediate self-paced RL algorithm that combines a labeled contextual MDP and its reward machine in a product contextual MDP to update the policy and value functions of an RL agent. 2) We establish a mapping that, given a transition in the reward machine, outputs the smallest set of context parameters, that determine whether the transition, namely a high-level event, occurs or not. 3) We develop a reward-machine-guided, self-paced RL algorithm that exploits reward machines not only to update the policy and value functions but also to navigate the generation of curriculum via the proposed mapping (see Figure 1).

Our experiments conclude that, first, proposed reward-machine-guided and intermediate self-paced RL algorithms enable RL agents to accomplish long-horizon planning tasks by encoding non-Markovian reward functions as reward machines, whereas state-of-the-art automated curriculum generation methods fail to do so; and, second, guiding curriculum generation via a reward-machine-context mapping not only boosts learning speed reliably but also stabilizes the curriculum generation process by reducing curricula variance by up to four orders of magnitude, and thus avoid inefficient exploration of the curriculum space.

2 RELATED WORK

We propose an automated curriculum generation method, that exploits high-level structural knowledge about long-horizon planning tasks. Our work falls under two subjects.

Curriculum learning for RL. Automatically generating curriculum in RL modifies the configuration of the environment iteratively to accelerate convergence to optimal policies. As we do, many studies in the literature consider a curriculum as a sequence of distributions over environment configurations. Florensa et al. [2017] proposes the generation of distributions over initial states that iteratively get further away from goal states. Others focus on goal-conditioned RL where a curriculum is a sequence of distributions over goal states that optimize value disagreement [Zhang et al., 2020], feasibility and coverage of goal states [Racaniere et al., 2020], intrinsic motivation [Baranes and Oudeyer, 2010, Portelas et al., 2020], and intermediate goal difficulty [Florensa et al., 2018]. For procedural content generation environments, curricula prioritize levels with higher learning potential [Jiang et al., 2021]. In comparison, self-paced RL is adopted from supervised learning where training samples are automatically ordered in increasing complexity [Kumar et al., 2010, Jiang et al., 2015, Ren et al., 2018] consider curricula as a sequence of environment interactions and proposes a self-paced mechanism that minimizes coverage penalty. Fimer et al. [2021]’s work generates a sequence of contexts, not distributions, with respect to their capacity of value improvement. Klink et al. [2020a|b|2021, 2022, Koprulu et al., 2023 formulate the generation of curricula as interpolations between distributions over contexts. Similarly, Chen et al. [2021] study interpolations between task distributions, but not under the self-paced RL framework.

Incorporating high-level structural knowledge into RL.

Singh [1992], Parr and Russell [1997], Sutton et al. [1999], Dietterich [2000] propose the idea of incorporating high-level structural knowledge to decompose a task into a hierarchy of subtasks. The proposed hierarchy allows the agent to learn a meta-controller that choices between subtasks to pursue, and a low-level controller that acts in the chosen subtask. Another way to incorporate such knowledge is to capture temporal abstractions in long-horizon planning tasks via temporal logic [Bacchus et al., 1996, Li et al., 2017, Littman et al., 2017], or reward machines [Icarte et al., 2018a, Camacho et al., 2019], which address MDPs with non-Markovian structures. We investigate a multi-task setting with non-Markovian reward functions and propose an automated curriculum generation approach that uses reward machines, 1) to encode non-Markovian reward functions; and 2) to guide the curriculum generation process. Similar to curriculum learning, Icarte et al. [2018b], Xu and Topcu [2019], Kuo et al. [2020], Zheng et al. [2022], Velasquez et al. [2021] study the use of temporal logic and reward machines in topics such as generalization, transfer learning, and multi-task learning.

3 PRELIMINARIES

In this section, we provide the background for our problem of interest. We illustrate a two-door environment, (see Figure 2), which we will revisit throughout the paper. The agent has to complete 4 subtasks in the following order: (1) Passing through Door 1, (2) getting a key from Box, (3) opening Door 2 with the key, and (4) arriving at Goal. The agent has to avoid hitting the walls that separate the rooms.
A policy $\pi$ is a function mapping states in $S$ to a probability distribution over actions in $A$. At state $s \in S$, an agent using policy $\pi$ picks an action $a$ with probability $\pi(s, a)$, and the new state $s'$ is chosen with probability $p(s, a, s')$.

For a fixed context, we can model the two-door environment as a labeled MDP. The states are the coordinates of the agent and the actions are moving in the four cardinal directions, whereas the transitions are deterministic. The agent receives the labels $\{d1\}, \{b\}, \{d2\}, \{g\}$, and $\{w\}$ when it moves onto Door 1, Box, Door 2, Goal, and the walls, respectively.

**3.1 LABELED MDPS AND REWARD MACHINES**

**Definition 1.** A labeled Markov decision process (Xu et al. [2020]) is a tuple $M = (S, A, p, R, \phi, \gamma, \mathcal{P}, L)$ consisting of a state space $S$, an action space $A$, a probabilistic transition function $p: S \times A \times S \to [0, 1]$, and an initial state distribution $\phi: S \to [0, 1]$. A reward function $R: (S \times A)^* \times S \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1]$ specify the returns to the agent. A finite set $\mathcal{P}$ of propositional variables, and a labeling function $L: S \times A \times S \to 2^P$ determine the set of high-level events that the agent sees in the environment.

A policy $\pi$ is a function mapping states in $S$ to a probability distribution over actions in $A$. At state $s \in S$, an agent using policy $\pi$ picks an action $a$ with probability $\pi(s, a)$, and the new state $s'$ is chosen with probability $p(s, a, s')$.

For a fixed context, we can model the two-door environment as a labeled MDP. The states are the coordinates of the agent and the actions are moving in the four cardinal directions, whereas the transitions are deterministic. The agent receives the labels $\{d1\}, \{b\}, \{d2\}, \{g\}$, and $\{w\}$ when it moves onto Door 1, Box, Door 2, Goal, and the walls, respectively.

**Definition 2.** A reward machine (Carte et al. [2018a]) $R = (Q, q_0, 2^P, O, \delta_q, \delta_r)$ consists of a finite, nonempty set $Q$ of states, an initial state $q_0 \in Q$, an input alphabet $2^P$, an output alphabet $O \subseteq \mathbb{R}$, a deterministic transition function $\delta_q: Q \times 2^P \to Q$, and an output function $\delta_r: Q \times 2^P \to O$.

Reward machines encode non-Markovian reward functions. The run $q_0(\ell_1, r_1)q_1(\ell_2, r_2)\ldots(\ell_k, r_k)q_{k+1}$ of a reward machine $R$ on a label sequence $\ell_1\ldots\ell_k \in 2^P$ is a sequence of states and label-reward pairs such that $q_0 = q_1$, $\delta_q(\ell_1, r_1) = q_{i+1}$, and $\delta_r(\ell_i, r_i) = r_i$ for all $i \in \{1, \ldots, k\}$. The reward machine $R$ produces a sequence of rewards from an input label sequence as $\mathcal{R}(\ell_1\ldots\ell_k) = r_1\ldotsr_k$. We say that a reward machine $\mathcal{R}$ implements the reward function $R$ of an MDP if for every trajectory $s_0a_0\ldots s_ka_k s_{k+1}$ and the corresponding label sequence $\ell_1\ldots\ell_k$, the reward sequence equals $\mathcal{R}(\ell_1\ldots\ell_k)$.

The reward machine of the two-door environment is in Figure 3. Every node is a state in the reward machine. An edge $(q_i, q_j)$ with tuple $(\rho, \tau)$ indicates that given a label $\ell \in 2^P$ satisfying propositional formula $\rho$, the transition from state $q_i$ to $q_j$ yields reward $\tau = \delta_r(q_i, \ell)$. For instance, the transition $(q_1, q_5)$ with $(w \vee d_2, 0)$ in Figure 3 occurs if a label $\ell$ satisfies $\rho = w \vee d_2$, i.e., if the agent moves into a wall $w$ or passes the second door $d_2$, yielding a reward of 0. The agent receives rewards of 1, 2, 3, and 4 upon completing the first, second, third, and fourth subtasks, respectively. We note that there can be multiple reward machines that encode an MDP’s reward function, and reward machines may differ on a label sequence that does not correspond to any trajectory of the MDP.

**3.2 CONTEXTUAL MDPS**

**Definition 3.** A contextual Markov decision process (CMDP) (Hallak et al. [2015]) $M = (S, A, C, M)$ is defined by a state space $S$, an action space $A$, a context space $C$ and a mapping $M$ from $C$ to MDP parameters. $M$ represents a family of MDPs parameterized by contexts $c \in C \subseteq \mathbb{R}^n$, $n \in \mathbb{Z}^+$. An MDP in this family is a tuple $M(c) = (S, A, p_c, R_c, \phi_c, \gamma)$ that shares the same state space $S$ and action space $A$ with other members, but its probabilistic transition function $p_c: S \times A \times S \to [0, 1]$, reward function $R_c: S \times A \to [0, 1]$ and initial state distribution $\phi_c: S \to [0, 1]$ depend on $c$.

CMDPs appear in the multi-task RL literature to model tasks, where transition functions, reward functions, and initial state distributions are parameterized via contexts. Although the two-door environment in Figure 2 has a context...
that determines the door positions, which in return affects the transition and reward functions, the CMDP framework fails to model such an environment as the two-door environment has a non-Markovian reward function. We present a new MDP formulation to address this limitation in Section 4.

Given a CMDP $\mathcal{M}$, contextual RL [Hallak et al. 2015] aims to learn a policy that maximizes the expected value of an initial state in context $c$ sampled from a target context distribution $\varphi : C \rightarrow [0, 1]$, namely, $\max_{\nu} \mathbb{E}_{\varphi(c), \nu(c)}[V_\nu(s, c)]$, where $V_\nu(s, c)$ is the value of state $s$ in context $c$ and encodes the expected discounted return obtained by following the policy $\pi_\nu(a|s, c)$ as $V_\nu(s, c) = \mathbb{E}_{\pi_\nu(a|s, c)}[R_\tau(s, a) + \gamma \mathbb{E}_{\pi_\nu(s'|s, c)}[V_\nu(s', c)]]$. $\mathbb{E}_{\pi_\nu(s'|s, c)}[V_\nu(s', c)]$.

3.3 SELF-PACED REINFORCEMENT LEARNING

Klink et al. [2020a,b, 2021] propose algorithms under the Self-paced RL framework to address the contextual RL problem. The self-paced RL aims to iteratively generate a sequence of context distributions by maximizing the expected performance with respect to the current distribution, which is regularized by the KL divergence from the target context distribution, namely,

$$\max_{\nu, \omega} \mathbb{E}_{\varphi(c), \nu(c)}[V_\nu(s, c)] - \alpha D_{\text{KL}}(\nu(c) \mid \varphi(c)) \quad \text{s.t.} \quad D_{\text{KL}}(\nu(c) \mid \varphi(c)) \leq \epsilon,$$

where $\varphi(c)$ and $\epsilon$ are the current context distribution parameterized by $\nu$ and the regularization coefficient, respectively. Klink et al. [2020a] introduce the constraint in (1) to restrict the divergence of the current context distribution from the previous context distribution parameterized by $\nu_{\text{prev}}$. Klink et al. [2020b] propose a way to estimate the expectation in (1). By following policy $\pi_\omega$, they collect a set $D = \{(c_1, \tau_1)\}_{i=1} \sim \varphi(c), \nu(c)\}_{i=1} \sim \varphi(c)$ of trajectories $\tau_i = (s_{i,0}, a_{i,0}, r_{i,1}, s_{i,1}), \cdots, (s_{i,T_i-1}, a_{i,T_i-1}, r_{i,T_i}, s_{i,T_i})$, where $r_{i,t+1} = R_{\tau_i}(s_{i,t}, a_{i,t}, s_{i,t+1})$ is the reward received at time $t + 1$ in trajectory $\tau_i$. Then, they use the cumulative sum of discounted rewards collected in trajectories to obtain an unbiased estimator of the expectation as

$$\frac{1}{N} \sum_{i=1}^N \mathbb{E}_{\nu_{\text{prev}}}[\mathbb{E}_{\varphi(c)}[\mathbb{E}_{\nu_{\text{prev}}}[V_\nu(s, c)]]] - \frac{\mathbb{E}_{\nu_{\text{prev}}}[\mathbb{E}_{\varphi(c)}[\mathbb{E}_{\nu_{\text{prev}}}[V_\nu(s, c)]]]}{\mathbb{E}_{\nu_{\text{prev}}}[\mathbb{E}_{\varphi(c)}[\mathbb{E}_{\nu_{\text{prev}}}[V_\nu(s, c)]]]}$$

where $\mathbb{E}_{\nu_{\text{prev}}}[\mathbb{E}_{\varphi(c)}[\mathbb{E}_{\nu_{\text{prev}}}[V_\nu(s, c)]]]$ is an importance weight used to estimate the value of state $s_{i,0}$ in context $c_i$ with respect to the current context distribution $\nu(c_i)$, as $c_i$ is sampled from $\varphi(c)$. $\mathbb{E}_{\nu_{\text{prev}}}[\mathbb{E}_{\varphi(c)}[\mathbb{E}_{\nu_{\text{prev}}}[V_\nu(s, c)]]].$

4 PROBLEM FORMULATION

We begin with integrating a labeling function into a CMDP to propose a labeled CMDP. We use labeled CMDPs to model long-horizon planning tasks, described via contexts.

Definition 4. A labeled CMDP $\mathcal{M}_L = (S, A, C, M^L)$ consists of a CMDP $\mathcal{M}$ and a labeling function $L_c : S \times A \times S \rightarrow 2^C$. A member of a labeled CMDP $\mathcal{M}_L$ is a labeled MDP $M^L(c) = (S, A, p_c, R^L_c, \phi_c, \gamma, P, L^c_c)$ parameterized by a context $c \in C$. $\mathcal{M}_L$. $\mathcal{M}_L$. $\mathcal{M}_L$. $\mathcal{M}_L$. $\mathcal{M}_L$. $\mathcal{M}_L$. $\mathcal{M}_L$. $\mathcal{M}_L$. $\mathcal{M}_L$. $\mathcal{M}_L$.

A labeled MDP $M^L(c)$ differs from a labeled MDP $\mathcal{M}$, from Definition 1, as the former depends on a context $c \in C$. However, every labeled MDP $M^L(c)$ obtained in $\mathcal{M}_L$ can use the same reward machine, that encodes the underlying task structure. Throughout this paper, we make Assumption 1 on the context space $C$ of a labeled CMDP $\mathcal{M}_L$.

Assumption 1. There exists $\Gamma \in \mathbb{Z}^+$ and $C[1], \cdots, C[\Gamma]$ such that $\mathcal{C} = \prod_{i=1}^{\Gamma} C[i]$. For $c = (c[1], \cdots, c[\Gamma]) \in C$, we call $c[i]$ the $i$th context parameter of $c$. We say $\Gamma$ is the dimension of the context space $C$, referred to as $\dim(C)$.

The assumption of a box-shaped context space $C$ of a labeled CMDP $\mathcal{M}_L$ allows us to establish a mapping from the transitions in the reward machine to context parameters.

Problem statement. Given a labeled CMDP $\mathcal{M}_L$, a reward machine $\mathcal{R}$ that encodes the non-Markovian reward function of $\mathcal{M}_L$, and a target context distribution $\varphi$, we want to obtain a policy that maximizes the expected discounted return in contexts $c$ drawn from $\varphi$, namely,

$$\max_{\omega} \mathbb{E}_{\varphi(c), \nu(c), \pi_\omega(a|s, c)}\sum_{t=0}^{T-1} \gamma^t R^L_c(h_t),$$

where $h_t = s_0a_0 \cdots s_ta_ts_{t+1}$ is the history at time $t$. Note that as the reward machine $\mathcal{R}$ encodes the reward function $R^L_c$ for any context $c$, we have $R^L_c(h_t) = R(\ell_1, \cdots, \ell_{t+1})$ with labels $\ell_\tau = L_c(s_{\tau-1}, a_{\tau-1}, s_{\tau})$ for $\tau \in [t+1]$.

5 METHOD

We first present an intermediate self-paced RL algorithm by adopting the approach by [carte et al. 2018a], which runs an RL algorithm using reward machines. Then, we discuss how contexts affect the transitions in a reward machine, and define a reward-machine-context mapping. Finally, integrating the proposed mapping into the intermediate algorithm, we develop a reward-machine-guided self-paced RL algorithm.

5.1 INTERMEDIATE SELF-PACED RL

We construct a product contextual MDP that combines a labeled contextual MDP $\mathcal{M}_L$ and its reward machine $\mathcal{R}$.

Definition 5. Given a labeled contextual Markov decision process $\mathcal{M}_L$ and a reward machine $\mathcal{R}$, we define a product contextual MDP as the tuple $\mathcal{M}^L_R = (\bar{S}, A, C, \bar{M}^L)$.
that has an extended state space \( \bar{S} = S \times Q \), an action space \( A \), a context space \( C \), and a mapping \( M^L \) from the context space to product MDP parameters. A member of this product contextual MDP is a tuple \( M^L(c) = (\bar{S}, A, \bar{p}_c, \bar{R}_c, \bar{\phi}_c, \gamma, P, L_c) \) with a probabilistic transition function \( \bar{p}_c : \bar{S} \times A \times \bar{S} \rightarrow [0,1], \) a reward function \( \bar{R}_c : \bar{S} \times A \times \bar{S} \rightarrow \mathbb{R}, \) and an initial state distribution \( \bar{\phi}_c : \bar{S} \times \{q_t\} \rightarrow [0,1]. \) We define them as

\[
\bar{p}_c((s, q), a, (s', q')) = \begin{cases} 
\bar{p}_c(s, a, s') & \text{if } q' = \delta_q(L_c(s, a, s')); \\
0 & \text{otherwise}, \end{cases}
\]

(4)

\[
\bar{R}_c((s, q), a, (s', q')) = \delta_r(q, L_c(s, a, s')),
\]

(5)

\[
\bar{\phi}_c(s, q_t) = \bar{\phi}_c(s),
\]

(6)

where states \( s, s' \in S \) and \( q, q' \in Q \) come from labeled contextual MDP \( M^L \) and reward machine \( R \), respectively.

A trajectory of length \( T \) on the product MDP \( M^L(c) \) is \( \bar{\tau}_i = (s_0, a_0, \bar{r}_1, s_1), \ldots, (s_{T-1}, a_{T-1}, \bar{r}_T, s_T) \), where \( \bar{r}_t = \bar{R}_c(s_{t-1}, a_{t-1}, s_t), t \in \{1, 2, \ldots, T\} \). The intermediate self-paced RL algorithm replaces the contextual MDP trajectories with the product contextual MDP trajectories. Therefore, an RL agent can capture the temporal task structure by learning a policy via trajectories rolled out in a product contextual MDP. The intermediate self-paced RL algorithm optimizes the following objective to generate context distribution

\[
\max_{\nu_c} \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{T_i-1} \gamma^t \frac{\varphi(c_t || \nu_k)}{\varphi(c_t || \nu_{k-1})} r_{i,t+1} - \alpha_k D_{KL}(\nu(c || \nu_k) \left| \left| \nu(c || \varphi(c)) \right\rangle \right. \\
\text{s.t. } D_{KL}(\nu(c || \nu_k) \left| \left| \nu(c || \varphi(c)) \right\rangle \right. \leq \epsilon,
\]

(7)

where \( \alpha_k \) is the regularization coefficient at the context distribution update \( k \). Appendix B provides the pseudocode for this algorithm.

### 5.2 FROM REWARD MACHINES TO CONTEXTS

In the two-door environment (see Figure 2), we observe that the first context parameter, i.e., the position of the first door, determines which product contextual MDP transitions enable the agent to pass the first door, yielding label \{d1\}. If we change the value of the first context parameter, then we have a different set of product contextual MDP transitions that yield label \{d1\}. However, this modification has no impact on the transitions that enable the agent to pass the second door, i.e., to obtain label \{d2\}. Taking this observation into account, we show how context parameters influence the transitions in a reward machine, then we define reward machine-context mapping \( F : Q \times Q \rightarrow 2^D \), which outputs the smallest set of context parameters that determines if a transition in the reward machine happens.

**Definition 6.** Given a product contextual MDP \( M^L_z \), we define a set \( G \subseteq D = \{1, 2, \cdots, \dim(C)\} \), as the set of identifier context parameters on a transition \((q, a, a')\) if

\[
\forall c, c' \in C, c[i] = c'[i], \forall i \in G \implies \\
\delta_q(L_c(s, a, s')) = \delta_{q'}(L_{c'}(s, a, s')),
\]

(8)

where \((q, a, s') \in Q \times S \times A \times S. \) That is, \( G \) is the set of indices of the context parameters that identify the next state of the reward machine given a state \( q \) of the reward machine and a transition \((s, a, s')\) in the labeled MDP.

Notice that \( D \) is a set of identifier context parameters for all \((q, a, s') \in Q \times S \times A \times S. \)

**Lemma 1.** If \( G_1 \) is a set of identifier context parameters on \((q, a, s, s')\), and \( G_1 \subseteq G_2 \subseteq D \), then \( G_2 \) is also a set of identifier context parameters on \((q, a, s, s')\).

**Proof.** Suppose \( c[i] = c'[i], \forall i \in G_2 \), then \( c[i] = c'[i] \) by definition. \( \square \)

We note that if the empty set \( \emptyset \) is a set of identifier context parameters on \((q, a, s, s')\), then the corresponding transition \( \delta_q(L_c(s, a, s')) \) in the reward machine does not depend on the choice of context \( c \in C \).

**Theorem 2.** Under Assumption\(^\dagger\) \( G_1 \) and \( G_2 \) are sets of identifier context parameters on \( (q, a, s, s') \) if and only if \( G_1 \cap G_2 \) is a set of identifier context parameters on \( (q, a, s, s') \).

**Proof Sketch.** The backward statement comes from Lemma\(^\dagger\). For the forward statement, let \( c''_G = [c[i]]_{i \in G} \). Then, for any \( c, c' \in C \) that satisfy \( c''_G = c''_{G_1} \cap c''_{G_2} \), by Assumption\(^\dagger\), there exists \( c'' \) for which \( c''_{G_1} = c''_G \) and \( c''_{G_2} = c''_G \). Therefore, \( \delta_q(L_c(s, a, s')) = \delta_q(L_{c''}(s, a, s')) = \delta_q(L_{c''_G}(s, a, s')) \).

**Corollary 1.** Under Assumption\(^\dagger\), the set \( \Gamma \) containing all sets of identifier context parameters on \((q, a, s, s')\) is closed under arbitrary unions and finite intersections.

**Proof.** Lemma\(^\dagger\) and Theorem\(^\dagger\) guarantee that \( \Gamma \) is closed under unions and finite intersections, respectively. \( \square \)

Corollary\(^\dagger\) guarantees that there is a set of identifier context parameters that is contained by every set of identifier context parameters. In Definition\(^\dagger\), we define a mapping that provides such a set for any transition \((q, a, s, s')\).

**Definition 7.** Given a product contextual MDP \( M^L_z \), we define a mapping \( H_{\min} : Q \times S \times A \times S \rightarrow 2^D \) such that we call \( H_{\min}(q, a, s, s') \) “the smallest set of identifier context parameters on \((q, a, s, s')\)” if \( H_{\min}(q, a, s, s') = \)

---

\(^\dagger\)See Appendix A for the complete proof.
\[ \bigcap_{q_i \in \mathcal{G}_i} \mathcal{G}_i, \text{ where } \Gamma \text{ is the set containing all possible sets of identifier context parameters on } (q, s, a, s'). \]

For practical applications, the design of \( H_{\text{min}} \) is not trivial, as one needs to separately analyze every transition in a labeled contextual MDP \( \mathcal{M}_c \). On the contrary, it is trivial to work with the transitions in a reward machine \( \mathcal{R} \), as the number of transitions in \( \mathcal{R} \) is smaller than the number of transitions in \( \mathcal{M}_c \) in general. Therefore, we define a set of identifier context parameters for every transition in \( \mathcal{R} \).

**Definition 8.** Given a product contextual MDP \( \mathcal{M}_c \) and the mapping \( H_{\text{min}} \), we define a reward machine-context mapping \( F : Q \times Q \rightarrow 2^D \) that outputs "a set of identifier context parameters for the transition \((q, q')\)" as

\[
F(q, q') = \bigcup_{B(q, q')} H_{\text{min}}(q, s, a, s'),
\]

where \( B(q, q') = \{ (q, s, a, s') \in Q \times S \times A \times S : \bar{q}(q, L_c(s, a, s')) = q' \text{ for some } c \in C \} \).

**Theorem 3.** \( F(q, q') \) is the smallest set that is a set of identifier context parameters for all \((q, s, a, s') \in B(q, q')\).

**Proof.** By Corollary 3, \( F(q, q') \) is a set of identifier context parameters for all \((q, s, a, s') \in B(q, q')\). Also, a set \( \mathcal{U} \) that is guaranteed to be a set of identifier context parameters for all \((q, s, a, s') \in B(q, q')\) must contain \( F(q, q') \) by construction. Hence, \(|\mathcal{U}| \geq |F(q, q')|\). \( \square \)

An expert designs mapping \( F \) by asking questions about the task structure. For instance, for the transition \((q_1, q_5)\) in Figure 3, the expert should ask: Is there a transition \((s, a, s')\) in the labeled contextual MDP \( \mathcal{M}_c \) such that it causes the agent to hit the wall for some context \( c \) but lets the agent pass through the door, i.e., \((q_1, q_2)\), for a different context \( c' \)? The idea is to find the context parameters \( i \in D \) for which a change of value, e.g. \( c[i] \neq c'[i] \), prevents a transition \((q, q')\) in the reward machine from happening. For \((q_1, q_5)\), the mapping outputs the first context parameter, \( F(q_1, q_5) = \{1\} \), as the identifier, since it determines the position of the first door. In short, when the agent is in the second room and moves into the first door/wall with an upward action, then the position of the first door determines whether it moves into the door or the wall. Here, the position of the second door does not identify which transition will happen.

### 5.3 REWARD-MACHINE-GUIDED, SELF-PACED REINFORCEMENT LEARNING

Klink et al. [2020b]'s self-paced RL algorithm uses an importance weight in \( \mathcal{G}_i \) as the ratio between probabilities of a context with respect to the current and previous contexts distributions. In other words, the algorithm assumes that every context parameter has an effect on the reward that an environment interaction yields. On the other hand, by Theorem 3, a reward machine-context mapping \( F \) outputs the smallest set of identifier context parameters for a transition \((q, q')\) in the reward machine \( \mathcal{R} \). Therefore, we can remove the naive assumption of Klink et al. [2020b] and use the context parameters that the mapping provides to compute the importance weight of a reward received in a transition \((q, q')\). We achieve this by utilizing the marginal context distributions for the context parameters in the set \( F(q, q') \) as \( \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\gamma_t} \frac{\gamma_{t_i}(c_i|\nu_{k_i})}{\gamma_{t_i}(c_i|\nu_{k-1})} \). Here, we introduce \( \gamma_{t_i}(c_i|\nu_{k}) \) and \( \gamma_{t_i}(c_i|\nu_{k-1}) \), that are the current and previous marginal context distributions, where the marginal variables are the identifier context parameters in \( F_t \), respectively. We note that for the case \( F_t = \emptyset \), we assign \( \frac{\gamma_{t_i}(c_i|\nu_{k})}{\gamma_{t_i}(c_i|\nu_{k-1})} = 1 \). Consequently, the reward-machine-guided, self-paced RL algorithm optimizes the following objective for context distribution updates, namely,

\[
\max_{\nu_k} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i-1} \gamma_{t_i} \frac{\gamma_{t_i}(c_i|\nu_{k})}{\gamma_{t_i}(c_i|\nu_{k-1})} \bar{r}_{t_i+1} - \alpha_k \sum_{t=0}^{T_i-1} D_{KL}(\rho(c|\nu_k) \parallel \rho(c|\nu_{k-1})) \leq \epsilon,
\]

where \( \alpha_k \) is the regularization parameter at the context distribution update \( k \). Similar to the intermediate self-paced RL, the reward-machine-guided, self-paced RL algorithm runs on a product contextual MDP \( \mathcal{M}_c \), as well. We outline the complete algorithm in Algorithm 1. Lines 3-5 update the policy \( \pi \) using trajectories in the sampled contexts via an RL algorithm \( \Psi \). Line 6 generates context distributions.

### 6 EMPIRICAL RESULTS

We evaluate the proposed RM-guided SPRL and Intermediate SPRL with three state-of-the-art automated curriculum generation methods: SPDL [Klink et al., 2020b], GoalGAN [Florensa et al., 2018], and ALP-GMM [Portelas et al., 2020]. We also include two baseline approaches: Default, which draws samples from the target context distribution without generating a curriculum, and Default*, which extends Default by running the RL algorithm on a product contextual MDP; hence, we observe the effect of capturing temporal abstractions. Appendix B includes more details.

**Two-door environment.** The two-door environment is a variation of the point-mass environment [Klink et al., 2020a, 2020b, 2021, 2022] with a temporal structure. Similar to commonly studied domains such as the office world, craft world, and water world [Carta et al., 2018a, Camacho et al., 2019, Carta et al., 2022], the two-door environment has discrete state and action spaces as a 40-by-40 grid

3See https://github.com/cemahir-kopru/ rm-guided-sprl to access the code repository of this work.
Although Default* obtains a higher return early on, it lags behind RM-guided SPRL and Intermediate SPRL since it does not generate a curriculum, but samples from the target context distribution directly. As we set $K_\alpha$ to 70 for all self-paced algorithms, the agent draws easy contexts from the initial context distribution $\rho(\cdot|\nu_0)$. RM-guided SPRL surpasses Default* quickly, but the agent seems to get stuck in the final phase, finding the goal. Intermediate SPRL does not experience this as its curricula converge later (see red lines in Figure 4(a)). Other approaches cannot learn a policy that accomplishes the task because they do not capture the temporal structure described by the reward machine.

**Customized Swimmer-v3 environment.** We customize the Swimmer-v3 environment [Brockman et al. 2016] by adding two flags, namely checkpoints, to the left, $f_1$, and the right, $f_2$, of the initial position of the swimmer. In comparison to the two-door environment, customized Swimmer-v3 has continuous state (8-dimensional) and action (2-dimensional) spaces. However, the underlying task is the simplest in our experimental setup, with a reward machine of 3 states (see Figure 5). The swimmer has to visit flag 2 first, and then flag 1, obtaining a reward of 100 and 1000, respectively. Inspired by Icarte et al. [2022], we use a control penalty, noted as CP, for rewards received by the agent.
following the self-loop transitions in the reward machine, to discourage the agent from applying large forces to the joints. The context space is 2-dimensional and determines the positions of the flags: $C_2 = [-0.6, 0] \times [1, 1.6]$. The target context distribution is $\mathcal{N}((0.6, 1.6, f_3) \sim 1.6 \cdot 10^{-7})$.

Figure 6 shows that only RM-guided-SPRL and Default* achieve 100% success ratio in median. RM-guided SPRL converges faster, and is more reliable as the quartiles converge before the training ends, as well. Default*’s performance evidence that this task does not require a curriculum as much as the two-door environment. Intermediate SPRL’s failure supports this argument, as it cannot achieve a success ratio of more than 20%, in the median. The other algorithms, again, fail to accomplish the task. [Table] indicates that the curricula variance of RM-guided SPRL is not significantly different than Intermediate SPRL. Nevertheless, RM-guided SPRL is reliable as it is 100% successful (median) while avoiding the redundant exploration of the context space.

**Customized HalfCheetah-v3 environment.** We also customize the HalfCheetah-v3 environment [Brockman et al., 2016] by adding three flags, $f_1$, $f_2$, and $f_3$, ordered in ascending distance to the right of the cheetah’s initial position. The reward machine in Figure 8 describes the following task: The cheetah has to visit flags 1 and 2, then go back to flag 1, and finally pass flag 3. The underlying task requires the cheetah to change direction 3 times. The original HalfCheetah-v3 [Brockman et al., 2016] and its variation in [carte et al., 2022] are single-task environments and reward the cheetah for running forward, only. In comparison, customized HalfCheetah-v3 has a running backward subtask (the transition ($q_2, q_3$) in the reward machine in Figure 8), which is challenging for the agent. Similar to customized Swimmer-v3, customized HalfCheetah-v3 has continuous state (17-dimensional) and action (6-dimensional) spaces. The 3-dimensional context space determines the flag positions: $C_3 = [0.5, 4] \times [2, 7] \times [3.5, 10]$. The target context distribution is $\mathcal{N}((4, 7, 10), I_3 \cdot 1.6 \cdot 10^{-7})$.

Figure 7 shows that RM-guided SPRL is the only algorithm that can learn a policy that accomplishes the target contexts in every independent training run. Intermediate SPRL fails in one run, where the generated curricula cannot converge to the target context distribution during the training (see Appendix B). SPDL suffers from a similar issue because they...
both obtain a negative expected discounted return, which
sets $\alpha_k$ in (10) to zero, hence the generated curricula do
not approach, even diverge from, the target context distribu-
tion. Default*, Default, GoalGAN, and ALP-GMM are
unsuccessful in this domain. In Table 1, we exclude SPDL,
as none of its curricula converge to the target before the
training ends. Similar to Case-2, RM-guided SPRL genera-
tes curricula whose variance is significantly smaller with
$p < 0.001$.

7 CONCLUSIONS

We propose two self-paced RL algorithms that exploit the
high-level structural knowledge about long-horizon plan-
ning tasks via reward machines. First, we present an in-
termediate self-paced RL algorithm that uses reward ma-
chines to update the policy and value functions of an RL
agent. Then, we establish a mapping, called reward-machine-
context mapping, that, given a transition in the reward ma-
chine, outputs the smallest set of identifier context param-
eters that determines whether the transition occurs or not.
Lastly, we develop a reward-machine-guided, self-paced
RL algorithm that builds on the intermediate algorithm and
navigates the automated curriculum generation via reward-
machine-context mapping. We evaluate the proposed algo-
risms in three domains. We empirically show that existing
approaches fail to accomplish the given long-horizon plan-
ning tasks, whereas the proposed algorithms can capture the
temporal structure of such tasks. Compared to the interme-
diate algorithm, the reward-machine-guided, self-paced RL
algorithm is more reliable, as it achieves successful com-
pletion of the task in every use case, and it also reduces curricula variance by up to four orders of magnitude.

Limitations. The limitations come from the self-paced RL
algorithm used in the proposed approaches, the assumption
of a priori available reward machine, and task knowledge
to construct a reward-machine-context mapping: 1) Interme-
diate SPRL and RM-guided SPRL employ a self-paced RL
algorithm, SPDL [Klink et al. 2020b], which uses a paramet-
ric family of context distributions to generate a curriculum.
Similar to [Klink et al. 2020b, 2021], we study Gaussian
context distributions. Hence, SPDL does not address set-
ings with arbitrary target context distributions. 2) We focus
on long-horizon planning tasks with a priori available re-
ward machines. The proposed approaches require a reward
machine to construct a product contextual MDP, which cap-
tures the temporal task structure. 3) Task knowledge about
the connection between the reward machine and the context
space enables the design of a reward-machine-context mapping.
Unless such knowledge is available, RM-guided SPRL and
Intermediate SPRL become equivalent, as the latter do
not utilize the mapping.

Future Work. Taking into account the limitations of the
proposed approaches, we will study how to infer a reward
machine and a reward-machine-context mapping of a do-
main online to remove the need for a priori available task
knowledge. In addition, we will extend RM-guided SPRL
to address arbitrary context distributions, which [Klink et al.
2022] studies without integrating high-level structural task
knowledge.

Acknowledgements

This work is supported by the Office of Naval Research
(ONR) under grant number N00014-22-1-2254, and the
National Science Foundation (NSF) under grant number
1646522.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas
Schneider, Rachel Fong, Peter Welinder, Bob Mc-
Grew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech
Zaremba. Hindsight experience replay. In NeurIPS,
2017.

Fahiem Bacchus, Craig Boutilier, and Adam Grove. Re-
warding behaviors. In National Conference on Artificial
Intelligence, pages 1160–1167, 1996.

Adrien Baranes and Pierre-Yves Oudeyer. Intrinsically moti-
vated goal exploration for active motor learning in robots:


