

# Task Phasing: Automated Curriculum Learning from Demonstrations

Vaibhav Bajaj,<sup>1</sup> Guni Sharon,<sup>1</sup> Peter Stone<sup>2</sup>

<sup>1</sup>Texas A&M University

<sup>2</sup>The University of Texas at Austin and Sony AI  
vaibhavbajaj@tamu.edu, guni@tamu.edu, pstone@cs.utexas.edu

## Abstract

Applying reinforcement learning (RL) to sparse reward domains is notoriously challenging due to insufficient guiding signals. Common RL techniques for addressing such domains include (1) learning from demonstrations and (2) curriculum learning. While these two approaches have been studied in detail, they have rarely been considered together. This paper aims to do so by introducing a principled task phasing approach that uses demonstrations to automatically generate a curriculum sequence. Using inverse RL from (suboptimal) demonstrations we define a simple initial task. Our task phasing approach then provides a framework to gradually increase the complexity of the task all the way to the target task, while retuning the RL agent in each phasing iteration. Two approaches for phasing are considered: (1) gradually increasing the proportion of time steps an RL agent is in control, and (2) phasing out a guiding informative reward function. We present conditions that guarantee the convergence of these approaches to an optimal policy. Experimental results on 3 sparse reward domains demonstrate that our task phasing approaches outperform state-of-the-art approaches with respect to asymptotic performance.

## 1 Introduction

In domains with sparse reward signals, a reinforcement learning (RL) agent (Portelas et al. 2020b; Narvekar et al. 2020) receives little to no signal regarding its performance. This phenomenon results in a limited ability to train and learn an optimized policy. As a result, a stream of publications (Nair et al. 2018; Burda et al. 2019; Salimans and Chen 2018; Reddy, Dragan, and Levine 2019; Ecoffet et al. 2019; Zhu et al. 2022) presented various solutions towards effective RL in sparse reward settings. Two of the most common approaches considered in such cases are curriculum learning (CL) (Bengio et al. 2009; Soviany et al. 2021; Wei et al. 2020; Narvekar et al. 2020) and learning from demonstrations (Salimans and Chen 2018; Zhu et al. 2022; Nair et al. 2018). In this paper, we investigate the impact of combining these two general approaches toward a CL continuum which is defined through demonstrations. We suggest applying inverse RL (IRL) to the provided demonstrations in order to obtain a dense reward function and/or a demonstration

policy. The IRL outcomes are then used to define an initial simple task for a curriculum along with a continuous curriculum continuum. This continuum is defined by a convex combination between the initial and target tasks. By training an RL agent on the resulting CL continuum (with progressively increasing complexity), we show—both theoretically and empirically—that an RL agent can be effectively trained to solve tasks that are challenging otherwise. We provide theoretical guarantees that the proposed CL will return an optimal policy under the assumption that the optimal policy as a function of the task continuum is continuous.

Two domain-independent approaches are presented for defining the task continuum, namely *Temporal-Phasing* and *Reward-Phasing*. *Temporal-Phasing* is designed to provide gradually increased control to the RL agent in lieu of a demonstrator agent obtained using IRL. This approach is shown to be especially effective in domains with no/few catastrophic actions, i.e., they provide options for recovering from actions that hinder performance. *Reward-Phasing*, on the other hand, results in a task continuum where each task is some convex combination of an informative dense reward (provided by the IRL agent) and the target (sparse) reward. This approach is shown to be especially effective in domains where a meaningful guiding reward function can be extracted from the provided demonstrations. The theory provided in this paper proves that, under reasonable assumptions, the curriculum defined by *Reward-Phasing* produces a monotonically non-decreasing policy return in expectation—a desirable outcome in many real-world environments. Such theory is a novel addition to existing CL theory, that otherwise focuses on the effects of different curriculum strategies on the convergence rate of the policy (Weinshall, Cohen, and Amir 2018; Weinshall and Amir 2020; Yengera et al. 2021).

Experimental results are provided for 3 continuous sparse reward domains. The results suggest that our proposed approaches are successful in converging to an optimized policy, whereas baseline RL algorithms fail to do so. Moreover, our proposed approaches also outperform prior approaches that apply CL, learning from demonstrations, or both.

In summary, the contributions of this paper are:

1. Define two domain-independent *Task Phasing* approaches that utilize sub-optimal demonstrations.
2. Present state-of-the-art asymptotic performance for

sparse reward RL domains.

3. Present convergence guarantees for *Task Phasing*, under reasonable assumptions.
4. Prove that the curriculum defined by *Reward-Phasing* produces a monotonically non-decreasing policy return in expectation.

## 2 Preliminaries

In reinforcement learning (RL) an agent is assumed to learn through interactions with an underlying *Markov decision process* (MDP) defined by:  $\mathcal{S}$  – the state space,  $\mathcal{A}$  – the action space,  $\mathcal{P}(s_t, a, s_{t+1})$  – the transition function of the form  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ ,  $R(s, a)$  – the reward function of the form  $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ , and  $\gamma$  – the discount factor. The agent is assumed to follow an internal policy,  $\pi$ , which maps states to actions, i.e.,  $\pi : \mathcal{S} \mapsto \mathcal{A}$ .<sup>1</sup> The agent’s chosen action ( $a_t$ ) at the current state ( $s_t$ ) affects the environment such that a new state emerges ( $s_{t+1}$ ) as well as some reward ( $r_t$ ) representing the immediate utility gained from performing action  $a_t$  at state  $s_t$ , given by  $R(s_t, a_t)$ . We use  $\tau$  to denote a finite horizon trajectory of the form  $\{s_0, a_0, r_0, s_1, \dots, a_{t-1}, r_{t-1}, s_t\}$ .

In this paper, the MDP comprises of two parts, (a) the *environment* and (b) the *task*. The environment defines the state space ( $\mathcal{S}$ ), action space ( $\mathcal{A}$ ), transition function ( $\mathcal{P}$ ), and discount factor  $\gamma$ . The task defines the time steps when the RL agent is given control (at such time steps  $a_t$  is determined by the RL agent). The task also defines the reward function  $R$ . Using this terminology allows sharing an environment across MDPs, each corresponding to a different task.

The expected sum of discounted rewards for a given task,  $\mathcal{K}$ , is denoted by  $J_\pi^\mathcal{K} = \mathbb{E}_{\tau \sim \pi} \sum_t \gamma^t R^\mathcal{K}(s_t, a_t)$  where  $R^\mathcal{K}$  is the task reward function. The observed task rewards are used to tune a policy such that  $J_\pi^\mathcal{K}$  is maximized. The policy  $\operatorname{argmax}_\pi [J_\pi^\mathcal{K}]$  is the optimal policy and is denoted by  $\pi_{\mathcal{K}}^*$ .<sup>2</sup>

In *sparse reward domains* a relatively high proportion of the reward signals ( $r_t$ ) are similar, making it challenging to obtain a meaningful gradient in  $J_\pi^\mathcal{K}$  with respect to  $\pi$ . Such domains are notoriously challenging to solve (i.e., identify an optimal policy for) (Nair et al. 2018).

### 2.1 Related work

Prior RL approaches for solving sparse reward domains can be divided into three broad classes. (1) boosted exploration, (2) demonstration guidance, and (3) curriculum learning.

**Boosted exploration approaches** Boosted exploration approaches (Ecoffet et al. 2019; Burda et al. 2019; Zhao et al. 2020; Nair et al. 2018; Durugkar et al. 2021) are mostly domain-independent approaches to finding the optimal policy for a sparse reward task, by improving the manner in which exploration is conducted in the target environment. These methods use intrinsic motivation (Barto 2013; Oudeyer and Kaplan 2009, 2013) where the agent presents

<sup>1</sup>Policies can also be defined as stochastic (soft policy), i.e., mapping states to a distribution over actions.

<sup>2</sup>For some tasks  $\operatorname{argmax}_\pi [J_\pi^\mathcal{K}]$  is not unique. In such cases,  $\pi_{\mathcal{K}}^*$  may refer to any optimal policy.

itself with exploration rewards that are different from those of the given task-specific rewards. Despite their effectiveness, the time to convergence for such approaches can be significant, as they require significant exploration.

**Demonstration-guided approaches** This class of algorithms (Ho and Ermon 2016a; Torabi, Warnell, and Stone 2018; Fu, Luo, and Levine 2017; Reddy, Dragan, and Levine 2019) attempt to learn policies that minimize the mismatch between the RL state-action visitation distribution and a demonstrator’s state-action visitation distribution. The performance of the policies learned by such techniques is often limited by the demonstrator’s performance, as they aim to only mimic the demonstrator and do not attempt to explore for better policies. On the other hand, Self-Adaptive Imitation Learning (SAIL) (Zhu et al. 2022) proposed an off-policy imitation learning approach that can surpass the demonstrator by encouraging exploration along with distribution matching and replacing sub-optimal demonstrations with superior self-generated trajectories. Consequently, we consider this approach for comparison in our experiments.

**Curriculum learning** This class of algorithms (Bengio et al. 2009; Soviany et al. 2021; Narvekar et al. 2020) aims to break down complex tasks into simpler tasks. CL approaches often require a human domain expert to divide the task into simpler tasks and then design a curriculum that decides the sequence in which those tasks are learnt (Ionescu et al. 2016; Lotfian and Busso 2019; Pentina, Sharmanska, and Lampert 2015; Jiménez-Sánchez et al. 2019). Recent work on CL (Portelas et al. 2020b) aims to automate the curriculum design, but still requires some domain knowledge provided by a human expert, such as defining sub-tasks (Portelas et al. 2020a; Matiisen et al. 2019), defining sub-goal conditions (Zhao et al. 2020; Nair et al. 2018), or scaling domain design features (Dennis et al. 2020).

**Note:** *Transfer learning* in RL (Taylor and Stone 2009) is a subprocess of CL. As such, it is related to but not directly comparable to CL. For further details refer Narvekar et al. (2016).

## 3 Task phasing

Consider a sparse reward RL task defined by  $\mathcal{K}$  and a given environment. Assume that  $\mathcal{K}$  cannot be solved efficiently using common RL approaches (Haarnoja et al. 2018; Schulman et al. 2017) due to reward signal sparsity. *Task Phasing* addresses this inefficiency by introducing a set of simplified tasks with progressively increasing complexity (similar to CL). In contrast to most past CL algorithms, it does not rely on expert knowledge and/or domain-specific assumptions for defining the simplified tasks but defines them through demonstrations in a principled domain-independent way.

Assume some initial simplified task denoted  $\mathcal{K}_s$ , that can be efficiently solved by common RL algorithms, and a target (complex) task,  $\mathcal{K}_f$ , that cannot be solved by common RL algorithms. Next, consider a convex combination function  $\operatorname{Con}(\beta, \mathcal{K}_s, \mathcal{K}_f) = \beta\mathcal{K}_s + (1 - \beta)\mathcal{K}_f$  which provides a task continuum,  $\mathcal{K}_\beta$ , with  $\beta \in [0, 1]$ , between two given tasks  $\mathcal{K}_s, \mathcal{K}_f$ . We can now define the general task phasing curriculum procedure shown in Algorithm 1.

---

**Algorithm 1:** Task phasing curriculum learning.

---

**Input:** initial (simplified) task,  $\mathcal{K}_s$ ; target (complex) task,  $\mathcal{K}_f$ ; step size,  $\alpha$ ; initial policy,  $\pi$

**Output:** optimized policy for  $\mathcal{K}_f$ ,  $\pi_f^*$

```
1:  $\pi^* \leftarrow \text{train}(\pi, \mathcal{K}_s)$ 
2:  $\mathcal{K}_\beta \leftarrow \mathcal{K}_s$ 
3:  $\beta \leftarrow 0$ 
4: while  $\mathcal{K}_\beta \neq \mathcal{K}_f$  do
5:    $\beta \leftarrow \beta + \alpha$ 
6:    $\mathcal{K}_\beta \leftarrow \text{Con}(\beta, \mathcal{K}_s, \mathcal{K}_f)$ 
7:    $\pi^* \leftarrow \text{re-train}(\pi^*, \mathcal{K}_\beta)$ 
8: end while
9: return  $\pi^*$ 
```

---

The ‘train’ (Line 1) and ‘re-train’ (Line 1) functions can be implemented with any off-the-shelf RL algorithm (assuming sufficient exploration, see Sec 3.3 for details). Note that this general approach only introduces a single hyperparameter over the underlying RL solver, the  $\alpha$  step size. A large value of  $\alpha$  is desirable as it quickly leads to the optimization of the RL policy on the target task,  $\mathcal{K}_f$ . But, this may lead to training instability due to a large shift in the objective between consecutive tasks obtained from the task continuum. Determining the dynamic upper bound on the value of  $\alpha$ , that maintains training stability, at any instance of the *Task Phasing* process is an interesting problem that we leave to future work. We found (empirically) that using a larger value of  $\alpha$  during the initial portion of phasing and a smaller value as  $\beta \rightarrow 1$  results in stable training.

Algorithm 1 raises two questions that need to be addressed.

1. How can we obtain the initial simplified task ( $\mathcal{K}_s$ ) for a general (domain-independent) MDP?
2. How can we define the task continuum, i.e.,  $\text{Con}(\beta, \mathcal{K}_s, \mathcal{K}_f)$  for any  $\beta \in [0, 1]$ ?

We address these questions by assuming a set of trajectories,  $\mathcal{D}$ , obtained from a (sub-optimal) demonstrating policy.

### 3.1 Temporal Task Phasing

The *Temporal-Phasing* approach can be viewed as gradually shifting the control from a demonstrating policy to a learned RL policy. This approach assumes that a demonstrator policy,  $\pi^d$ , can be retrieved from  $\mathcal{D}$ , e.g., by using *imitation learning* (Ho and Ermon 2016a; Fu, Luo, and Levine 2017).

*Temporal-Phasing* follows some internal logic for determining in which time steps the RL-agent is given control, instead of the demonstrator, during online exploration. A similar approach was previously proposed (Dey et al. 2021) for smooth policy transitions. At any state, if the RL-agent is in control, the chosen action follows the RL internal policy. If, by contrast, the demonstrator is given control, then  $\pi^d$  is followed. Transitions that follow the demonstrator’s policy might also be used to train the RL-agent if off-policy learning (Levine and Koltun 2013) is enabled.

**Initial simplified task ( $\mathcal{K}_s$ ).**  $\mathcal{K}_s$  is defined by setting the probability of providing control to the RL-agent at any state

to be 0.

**task continuum ( $\text{Con}(\beta, \mathcal{K}_s, \mathcal{K}_f)$ ).** The environment continuum is defined by setting the probability of providing control to the RL-agent at any state to be  $\beta$ . Relevant protocols include:

#### V1: random step

$$\forall t, \pi_t = \begin{cases} \pi^{RL}, & U[0, 1] < \beta \\ \pi^d, & \text{else} \end{cases}$$

where  $U[0, 1]$  is a random value drawn from a uniform distribution in the range  $[0, 1]$ .

#### V2: random $m$ steps

$$\forall t : (t \bmod m) = 0, \pi_{t:t+m-1} = \begin{cases} \pi^{RL}, & U[0, 1] < \beta \\ \pi^d, & \text{else} \end{cases}$$

#### V3: fixed steps

$$\forall t, \pi_t = \begin{cases} \pi^d, & (t \bmod \beta T) = 0 \\ \pi^{RL}, & \text{else} \end{cases}$$

where  $T$  is the total number of steps for the episode.

We implemented and experimented with all three variants and found ‘V1: random step’ to be superior in most cases. Refer to Fig. A.2 in the Appendix for a comparison of the performance of the *Temporal-Phasing* variants.

**Limitations:** *Temporal-Phasing* is expected to perform poorly in domains where actions have unrecoverable outcomes. For example, consider a robotic gripper arm task where the goal is to carry an object from one location to another. The demonstrator’s policy never opens the gripper’s palm, steadily holding the object. An exploratory RL agent, by contrast, will try various actions (including opening the gripper’s palm) that result in unrecoverable situations and reduced probability of reaching the goal state. This probability diminishes exponentially as the exploratory RL agent is provided control in more time steps (factoring in the probabilities of avoiding unrecoverable actions per step). This phenomenon is especially harmful in sparse reward domains as a guiding signal from the goal state is rarely observed.

### 3.2 Reward Task phasing

The *Reward-Phasing* approach can be viewed as gradually shifting the Reward function from a dense, informative reward signal to the true (sparse) reward. The dense reward is initially used to guide the RL agent such that it learns to imitate the demonstrator. Next, the dense reward is gradually phased out leaving the true reward as the only policy guiding signal. Doing so allows the RL agent to learn optimized policies that may diverge from the demonstrator. This approach assumes that an initial dense reward function,  $R^d$ , can be retrieved from  $\mathcal{D}$ , e.g., by using *inverse RL* (Fu, Luo, and Levine 2017; Abbeel and Ng 2004). It further assumes that  $R^d$  and the target reward function are of similar scale. This, however, is not a limiting assumption as  $R^d$  can be scaled arbitrarily with no impact on the IRL efficiency.

**Initial simplified task ( $\mathcal{K}_s$ ).** Let  $R^d$  be the learnt dense reward function and let  $R^f$  be the target (sparse) reward function.  $\mathcal{K}_s$  is defined by setting the reward function  $R^s = R^d + R^f$ .

**Task continuum ( $Con(\beta, \mathcal{K}_s, \mathcal{K}_f)$ ).** The task continuum,  $Con(\beta, \mathcal{K}_s, \mathcal{K}_f)$ , is defined by a phased reward function that is a combination of  $R^d$  and  $R^f$ . Relevant protocols include:

**V1: constant phasing**

$$R^\beta = (1 - \beta)R^d + R^f$$

**V2: random phasing**

$$R^\beta = \begin{cases} R^d + R^f, & U[0, 1] > \beta \\ R^f, & \text{else} \end{cases}$$

We implemented and experimented with both variants and found ‘V1: constant phasing’ to be superior in most cases. Refer to Fig. A.4 and Fig. A.5 in the Appendix for a comparison of the performance of the *Reward-Phasing* variants.

**Limitations:** *Reward-Phasing* is expected to perform poorly in domains where the initial dense reward function directs the RL agent towards a locally optimal policy (with respect to the target reward). For example in a capture-the-flag game (e.g., our PyFlag domain as described in Appendix A.1), an initial reward that is biased towards guarding the player’s flag will make it challenging to learn a policy that steals the enemy flag. This is less of an issue for *Temporal-Phasing* as it trains a “demonstrator” surrogate function that can be queried for states that lay outside of the demonstration’s state distribution, allowing the RL agent to explore further from the provided demonstrations.

### 3.3 Convergence condition

**Definition 1 ( $RL_\epsilon$ ).** Define  $RL_\epsilon$  to be an RL algorithm that explores and returns the optimal policy within some bounded  $\epsilon$  KL-divergence from a given stochastic policy,  $\pi$ . That is,  $RL_\epsilon$  will return

$$\pi^{b*} = \arg \max_{\pi' | \mathbb{E}_{s \sim \pi} [KL(\pi(s), \pi'(s))] \leq \epsilon} J_\pi$$

where  $\pi^{b*}$  is the optimal policy within the exploration bound.

For simplicity of presentation,  $KL(\pi, \pi')$  is used to represent  $\mathbb{E}_{s \sim \pi} [KL(\pi(s), \pi'(s))]$  hereafter.

Examples of  $RL_\epsilon$  include Trust Region Policy Optimization (Schulman et al. 2015) and Proximal Policy Optimization (Schulman et al. 2017).

Consider two tasks  $\mathcal{K}_1$  and  $\mathcal{K}_2$  each with an affiliated optimal policy  $\pi_1^*$  and  $\pi_2^*$ . It is easy to see that  $RL_\epsilon$  will return  $\pi_2^* \leftarrow \text{re-train}(\pi_1^*, \mathcal{K}_2)$  if  $KL(\pi_1^*, \pi_2^*) \leq \epsilon$ . This is because the optimal policy  $\pi_2^*$  is within the exploration range from the initial policy  $\pi_1^*$ .

Consider applying Algorithm 1 for a given  $\mathcal{K}_s$ ,  $\mathcal{K}_f$ , and some RL algorithm,  $RL_\epsilon$ . Assume that  $\pi_s^*$  is within the initial KL-divergence bound of  $RL_\epsilon$ , yet  $\pi_f^*$  is not. Further assume that,  $KL(\pi_s^*, \pi_f^*) > \epsilon$ . That is,  $RL_\epsilon$  might fail to identify  $\pi_f^* \leftarrow \text{re-train}(\pi_s^*, \mathcal{K}_f)$ .

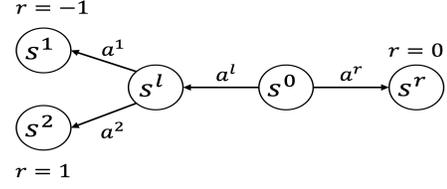


Figure 1: An example MDP where both *Temporal-Phasing* and *Reward-Phasing* using sub-optimal demonstrations result in a non-continuous policy space. The rewards shown in the MDP represent the target reward,  $R^f$ . **Temporal phasing:** assume  $\pi^d([s^0, s^l]) = [a^r, a^l]$ , for  $\beta < 0.5$ ,  $\pi_\beta^*(s^0) = a^r$  yet for  $\beta > 0.5$ ,  $\pi_\beta^*(s^0) = a^l$ . **Reward phasing:** assume  $R^d([(s^0, a^r), (s^0, a^l), (s^l, a^1), (s^l, a^2)]) = [2, 0, 0, 0]$  and  $\gamma = 1$ . Using constant phasing,  $R^\beta = (1 - \beta)R^d + R^f$ , for  $\beta < 0.5$ ,  $\pi_\beta^*(s^0) = a^r$  yet for  $\beta > 0.5$ ,  $\pi_\beta^*(s^0) = a^l$ .

**Lemma 1 (Convergence).** *if the optimal policy,  $\pi_\beta^*$ , as a function of  $\beta$  (the optimal policy for  $\mathcal{K}_\beta$ ) is continuous for  $\beta \in [0, 1]$  and a given  $\mathcal{K}_\beta = Con(\beta, \mathcal{K}_s, \mathcal{K}_f)$  function, then Algorithm 1 with a small enough  $\alpha$ , using  $RL_\epsilon$  as the underlying solver, will converge on  $\pi_f^*$  within  $1/\alpha$  iterations.*

*Proof.* If the optimal policy as a function of  $\beta$  is continuous then there must exist a fine enough  $\beta = \{\beta_0 = 0, \beta_1, \dots, \beta_n = 1\}$  decomposition such that  $\forall i \in \{0, \dots, n - 1\}$ ,  $KL(\pi_{\beta_i}^*, \pi_{\beta_{i+1}}^*) \leq \epsilon$ . Following Definition 1,  $RL_\epsilon$  will return  $\pi_{\beta_i}^* \leftarrow \text{re-train}(\pi_{\beta_{i-1}}^*, \mathcal{K}_i)$  for every  $i \in \{1, \dots, n\}$  where  $\beta_n = 1$ , i.e.,  $\mathcal{K}_n = \mathcal{K}_f$ . That is,  $RL_\epsilon$  will find all optimal policies along the resulting curriculum, until  $\pi_f^*$ .  $\square$

A continuous  $\pi_\beta^*$  function for both the *Temporal-Phasing* (Sec 3.1) and *Reward-Phasing* (Sec 3.2) approaches does not hold for the case of a sub-optimal demonstrator (from which  $\mathcal{D}$  is sampled). Figure 1 presents a counter examples for both spaces (Temporal and Reward phasing). As a result, in the worst case, an arbitrary small step in the task continuum can lead to a re-train procedure that is as hard as training a policy from scratch. Fortunately, this issue can be mitigated by considering a MaxEnt soft policy objective (Haarnoja et al. 2018) which smoothens the policy continuum,  $\pi_\beta^*$  resulting in a continuous shift between the policies in the re-train procedure. Reconsider the example from Figure 1, with the addition of an entropy maximizing term. That is,  $\pi_\beta^* = \sum_t r_t + \mathcal{H}(\pi(\cdot | s_t))$ . The reader is encouraged to validate that, in this case, the  $\pi_\beta^*$  function is indeed continuous. Hence, for the reported experiments we choose an  $RL_\epsilon$  algorithm that includes an entropy coefficient parameter (see Section 4.1).

### 3.4 Theoretical results for *Reward-Phasing*

Next we show that Algorithm 1 results in monotonically non-decreasing policy performance (assuming  $\pi_{\beta+\alpha}^* \leftarrow \text{re-train}(\pi_\beta^*, \mathcal{K}_{\beta+\alpha})$  is found in each iteration). This result is important and useful, since assuming the initial performance is equivalent to that of the demonstrator, it guarantees

that the learned policy never underperforms the demonstrator (in expectation). Note that this result relates to V1 (constant *Reward-Phasing*). Nonetheless, the same result can be extended to apply for V2 following the fact that the affiliated reward functions are equal in expectation. That is,  $\mathbb{E}[R_{V2}^\beta] = (1 - \beta)(R^d + R^f) + \beta R^f = (1 - \beta)R^d + R^f = R_{V1}^\beta$ .

Define  $\mathcal{R}_\pi^d := \mathbb{E}_{\tau \sim \pi} \sum_t \gamma^t R^d(s_t, a_t)$ .

Define  $\mathcal{R}_\pi^f := \mathbb{E}_{\tau \sim \pi} \sum_t \gamma^t R^f(s_t, a_t)$ .

Consequently, we can rewrite

$$\pi_\beta^* := \arg \max_{\pi} (1 - \beta)\mathcal{R}_\pi^d + \mathcal{R}_\pi^f$$

**Theorem 1** (Monotonic improvement).  $\mathcal{R}_{\pi_\beta^*}^f$  is monotonically non-decreasing with  $\beta$ .

*Proof.* by contradiction, assume some  $\pi = \pi_{\beta'}^*$ ,  $\pi' = \pi_{\beta'}^*$ , with  $\beta < \beta'$ , for which  $\mathcal{R}_\pi^f > \mathcal{R}_{\pi'}^f$ .

**Case 1:**  $\mathcal{R}_\pi^d \geq \mathcal{R}_{\pi'}^d$

This would imply that  $(1 - \beta')\mathcal{R}_\pi^d + \mathcal{R}_\pi^f < (1 - \beta')\mathcal{R}_{\pi'}^d + \mathcal{R}_{\pi'}^f$  contradicting the assumption that  $\pi' = \pi_{\beta'}^*$ .

**Case 2:**  $\mathcal{R}_\pi^d < \mathcal{R}_{\pi'}^d$

Let  $\alpha = \beta' - \beta$ . Then

$$(1 - \beta')\mathcal{R}_\pi^d + \mathcal{R}_\pi^f = (1 - \beta)\mathcal{R}_\pi^d + \mathcal{R}_\pi^f - \alpha\mathcal{R}_\pi^d \quad (1)$$

$$\leq (1 - \beta)\mathcal{R}_\pi^d + \mathcal{R}_\pi^f - \alpha\mathcal{R}_{\pi'}^d \quad (2)$$

$$< (1 - \beta)\mathcal{R}_\pi^d + \mathcal{R}_\pi^f - \alpha\mathcal{R}_\pi^d = (1 - \beta')\mathcal{R}_\pi^d + \mathcal{R}_\pi^f \quad (3)$$

contradicting the assumption that  $\pi' = \pi_{\beta'}^*$ .

(Eq 2) because  $\pi = \pi_\beta^*$  (Eq 3) by the Case 2 assumption  $\square$

## 4 Experiments and Results

The experiments are designed to study the performance of our two *Task Phasing* variants when paired with a state-of-the-art RL solver. Specifically, they are designed to answer the following questions.

1. Can our *Task Phasing* variants learn an optimized policy in sparse-reward domains where the paired RL algorithm (without task phasing) cannot?
2. Can the proposed task phasing variants learn a policy that outperforms the demonstrator, where the demonstrator’s policy is used for collecting the demonstrations in  $\mathcal{D}$ ?
3. Are the limitation reported for each of our two *Task Phasing* variants (Sections 3.1 and 3.2) observed empirically?
4. How does *Task Phasing* compare to state-of-the-art algorithms designed to run in sparse reward domains and/or leverage demonstrations?

Our results provide a positive answer to Questions 1–3 and show clear advantages over previous state-of-the-art algorithms with respect to Question 4. In order to support full reproducibility of the reported results, our codebase along with detailed running instructions are provided online<sup>3</sup>. The experiments are carried out in three continuous control, sparse reward environments, namely:

- **PyFlags (Erceth 2020) (GNU General Public License v3.0)** - This task requires a tank to fire at an opponent tank and capture its flag while defending its own flag.
- **FetchPickAndPlace-v1(P&P) (Plappert et al. 2018) (MIT license)** - This task requires a robotic arm to grab a randomly spawning block on a tabletop and carry it to a random goal location within the arms reach.
- **FetchSlide-v1(FS) (Plappert et al. 2018) (MIT license)** - This task requires a robotic arm to push a randomly spawning block on a tabletop such that it slides to a stop at a random goal location outside the robotic arms reach.

The FetchPickAndPlace and FetchSlide domains are used as sparse reward benchmark domains in the HER with demonstrations paper (Nair et al. 2018). All 3 domains are of special interest as they correspond to the limitations reported for our *Task Phasing* variants. Specifically, the PyFlags domain (capture the flag) has many local optimums in the policy space, which is expected to have a stronger negative impact on *Reward-Phasing* compared to *Temporal-Phasing*. On the other hand, the two Fetch domains have unrecoverable actions, which are expected to have a stronger negative impact on *Temporal-Phasing* compared to *Reward-Phasing*.

A snapshot from each of the reported domains is provided in Figure A.1. The technical description for the domains is provided in Appendix A.1. All experiments are repeated with 3 different random seeds and the mean of their results is reported along with a  $1-\sigma$  shaded region.

### 4.1 Settings

We use a common  $RL_\epsilon$  algorithm, denoted Proximal Policy Optimization (PPO) (Schulman et al. 2017), as our RL algorithm (for computing ‘train’, Line 1, and ‘re-train’, Line 1, in Algorithm 1). All algorithms used in the experiments apply the ADAM optimizer for training. The hyperparameters values for our approach as well as for the baseline algorithms are provided in Appendix A.2.

**Temporal phasing** *Temporal-Phasing* requires an online demonstrator. The demonstrator can be learnt from demonstrations using techniques such as IRL (Fu, Luo, and Levine 2017) or GAIL (Ho and Ermon 2016a). In order to focus the study on the phasing approach, we skip the IL phase and directly use a suboptimal rules-based demonstrator (same one used to collect the demonstrations). We report results for the best performing *Temporal-Phasing* variant ‘V1: random step’. We provide a comparison of the various variants of *Temporal-Phasing*, as well as ablation studies conducted on them, in the Appendix A.3. We found that using a dynamic  $\alpha$  step size performed better compared to using a static one. In our dynamic setting,  $\beta$  is incremented by  $\beta \leftarrow \beta + \alpha$  only if the average performance of the current policy ( $\pi_\beta$ ) is greater than a set threshold. The threshold values for each domain are provided in Appendix A.2. Also, for *Temporal-Phasing* we used importance sampling to train the RL-policy over transitions that follow from the demonstrator control (off-policy training). We used a PPO importance sampling approach that was introduced by (Levine and

<sup>3</sup><https://github.com/ParanoidAndroid96/Task-Phasing.git>

Koltun 2013). However, instead of using Differential Dynamic Programming to generate “guiding samples” (Levine and Koltun 2013), we use the demonstrator’s actions.

**Reward-Phasing.** *Reward-Phasing* requires a dense reward function to provide initial guidance to the RL policy. In our approach, we use the Adversarial IRL (Fu, Luo, and Levine 2017) approach to learn the dense reward function for the task. Results are reported for the best performing *Reward-Phasing* variant (V1: constant phasing). Similar to *Temporal-Phasing*, we provide a comparison of the various variants of *Reward-Phasing* in Appendix A.3. In order to allow reasonable running times, we employ a fixed interval approach, where  $\beta \leftarrow \beta + \alpha$  is applied every 200 training episodes. That is, Line 1 in Alg 1 trains for a fixed 200 episodes (not necessarily until convergence to  $\pi_\beta^*$ ).

The experiments conducted indicate that the performance of both *Temporal-Phasing* and *Reward-Phasing* is sensitive to the  $\alpha$  parameter (step size), learning rate, and entropy coefficient. However, annealing these parameters throughout the learning process leads to stable (insensitive) results for both approaches. Further details on the hyperparameters used can be found in Appendix A.2.

**Baselines** We compare the proposed approaches against the following baseline algorithms that are considered state-of-the-art for sparse reward domains:

**Hindsight Experience Replay (HER) with demonstrations (Nair et al. 2018).** This approach automatically generates a learning curriculum. It does so by setting intermediate goals in the environment that the policy can reach and providing rewards based on how close the agent is to a goal state. This approach requires some domain knowledge as the goal state for the environment must be defined. We find that the agent learns to retrieve the flag when the goal state for the PyFlag task is set such that the agent is positioned at its own base with the opponent’s flag in its possession (distance to red flag = 0) and its own flag is not in the opponent’s possession. When computing rewards, a weight of 1.0 is provided for possession of the opponent’s flag and preventing the opponent from taking the agent’s flag, in order to encourage offensive and defensive strategies. A lower weight, 0.1, is provided for the agent being positioned at its base so that it will prioritize attempting to steal the flag instead of staying close to its base. The goal state and the reward for the Fetch domains is the same as in (Nair et al. 2018), with the exception that the reward is scaled up from  $\{-1, 0\}$  to  $\{0, 1\}$ .

**Self-adaptive Imitation Learning (SAIL) (Zhu et al. 2022).** Instead of relying purely on exploration or on exploiting demonstrations, this approach aims to effectively strike a balance between exploiting sup-optimal demonstrations and efficiently exploring the environment to surpass the performance of the demonstrator. It maintains two replay buffers, for caching teacher demonstrations and self-generated transitions, respectively. During iterative training, SAIL dynamically adds high-quality self-generated trajectories into the teacher demonstration buffer.

**Random Network Distillation (RND) (Burda et al. 2019).** This approach encourages the exploration of new states in the environment but does not rely on demonstrations. The exploration boost is provided by combining an intrinsic reward with an extrinsic (true) reward. The intrinsic reward is the error of a neural network predicting features of the observations given by a fixed randomly initialized neural network.

**Other relevant baselines** algorithms include: HER+DDPG (Andrychowicz et al. 2017), BC+HER, GAIL (Ho and Ermon 2016b), DAC (Kostrikov et al. 2018), DDPGfD (Vecerik et al. 2017), POfD (Kang, Jie, and Feng 2018). However, these baselines are omitted from our reported results as they were shown (Nair et al. 2018; Zhu et al. 2022; Burda et al. 2019) to be inferior to the aforementioned 3 state-of-the-art baselines.

## 4.2 Results

We start by comparing our task phasing variants against the baseline algorithms. Figure 2 presents learning curves for both our best *Temporal-Phasing* (V1: random step) and *Reward-Phasing* (V1: constant phasing) variants.

**Temporal Phasing** In the PyFlags domain, see Figure 2(a), *Temporal-Phasing* outperforms all other algorithms with respect to the final performance. *Temporal-Phasing* is also the only approach that is able to outperform the demonstrator.<sup>4</sup> The peak in performance is achieved when the RL policy has full control ( $\beta = 1$ ) learning a policy that outperforms the demonstrator (game score  $> 0$ ). These results provide a positive answer to the experiments’ Questions 1, 2, and 4. The results in the Fetch domains, shown in Figure 2(b,c), indicate that *Temporal-Phasing* performed poorly in these domain, with the demonstrator policy only being phased out 50% on average (not able to achieve the phasing threshold beyond 50%). This highlights the limitations of the *Temporal-Phasing* approach in learning to perform sensitive tasks that suffer from unrecoverable actions. This is apparent in the P&P domain where the robotic arm can take one wrong action, such as opening the gripper, leading it to drop the block. A similar phenomenon can be observed in the FS domain where pushing the block in certain directions leads to the block becoming unreachable. In the PyFlags domain, by contrast, the demonstrator’s policy is more likely to recover from a limited number of bad actions, making the *Temporal-Phasing* approach more appropriate. These results provide a positive answer to the experiments’ Question 3.

**Reward Phasing** Figure 2 paints a picture where *Reward-Phasing* achieves state-of-the-art performance in both Fetch domains. It can be observed that during the phasing process *Reward-Phasing* results in mostly monotonic improvement (in expectancy), supporting the claims made in Theorem 1.

In the Fetch domains, the policy learned using *Reward-Phasing* not only learns to perform the task but also outperforms the demonstrator policy. These results provide further positive support to experiments’ Questions 1, 2, and 4. In the

<sup>4</sup>In the PyFlags domain, matching the demonstrator’s performance is achieved when the average game score is zero (as it is a zero-sum game).

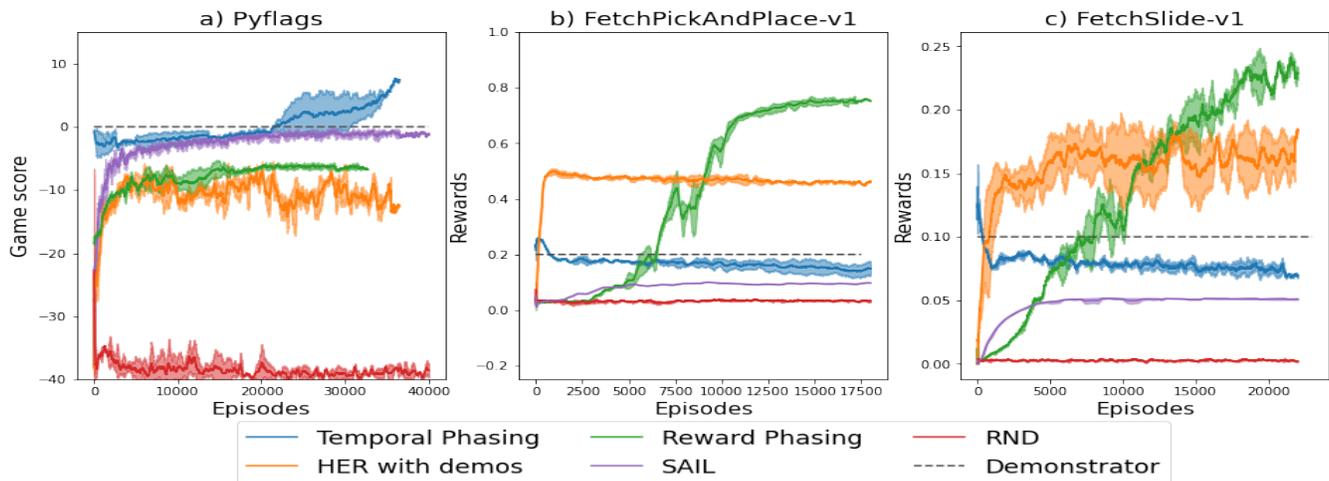


Figure 2: Learning curves of different algorithms tested in the PyFlag, FetchPickAndPlace, and FetchSlide domains. The y-axis represents the average episode score/reward over 300 runs. In the PyFlags domain, the game score is the difference between the number of times the blue and red flags are stolen. In PyFlags 1 episode = 20480 steps; P&P, FS 1 episode = 1024 steps.

PyFlags domain, although the policy learnt using *Reward-Phasing* does not outperform the demonstrator policy, we observe that the learned policy is still able to steal the enemy’s flag 5 times per episode on average. This indicates that *Reward-Phasing* was successful in learning a policy that retained useful behaviors in sparse reward settings. However, the RL policy seemed to get stuck in a local minima where it is unable to outperform the opponent. This provides further positive support to the experiments’ Question 3.

**HER with demonstrations** Despite being able to learn a reasonable sub-optimal policy, the HER algorithm did not outperform both our best task phasing approaches in the reported domains. The results highlight the drawbacks of the HER algorithm in adversarial settings such as the PyFlags domain. The HER algorithm provides a reward to the agent for achieving sub-goals based on how close they are to the true goal state, but it does not consider how good those sub-goals are with respect to the true reward. For instance, the agent may receive a large reward when it reaches a state close to the opponent’s flag, but it is not penalized if this state leads the agent directly into the opponent’s line of fire.

**Random Network Distillation** The RND algorithm was not able to perform meaningful learning in any of the reported domains. This indicates that although exploration-based approaches are known to provide good results, they require a substantial training period compared to algorithms that utilize demonstrations or some form of domain knowledge. Similar trends for RND have been previously observed and reported (Yang et al. 2021).

**Self-adaptive Imitation Learning** SAIL was unable to outperform at least one of our task-phasing approaches in all of the reported domains. While in the PyFlags domain, it was able to outperform *Reward-Phasing*, it still underperformed *Temporal-Phasing*. SAIL exhibited similar limitations to *Temporal-Phasing* in the Fetch domains where

it failed to even match the demonstrator’s performance. We speculate that similar to *Temporal-Phasing*, SAIL is unable to effectively deal with unrecoverable actions which prevents it from experiencing sparse (hard to reach) rewards.

## 5 Limitations and Future Work

A limitation of task phasing is the need for human input to choose between its two (and potentially more) variants, *Temporal-Phasing* or *Reward-Phasing*. Future work aims to improve *Reward-Phasing* by fusing it with the *Dataset Aggregation* (Ross, Gordon, and Bagnell 2011) approach for no-regret online learning. We expect such an approach to outperform *Temporal-Phasing* in all of the domains, making *Reward-Phasing* a default variant choice. In doing so, however, the availability of an online interactive demonstrator must be assumed. Such an assumption can be motivated by RL domains that target optimizing existing, suboptimal controllers. Examples include Traffic Signal Controllers (Ault and Sharon 2021; Sharon 2021; Ault, Hanna, and Sharon 2020) and Autonomous Driving (Kiran et al. 2021).

## 6 Summary

This paper presents two general, domain-independent approaches for designing a curriculum continuum from demonstrations: *Temporal-Phasing* and *Reward-Phasing*. The curriculum continuum allows decomposing a complex RL task into a set of tasks with progressively increasing complexity. We show that under the assumption of a continuous (task, optimal policy) space, a task phasing approach with a sufficiently small step size, is guaranteed to learn the optimal policy for any task. We show that the *Reward-Phasing* curriculum must result in policies that are monotonically non-decreasing with respect to the expected return in the target task. Experimental results in sparse reward domains indicate that *Temporal-Phasing* and/or *Reward-Phasing* can significantly surpass state-of-the-art algorithms

in terms of asymptotic performance. The results indicate that *Temporal-Phasing* is more applicable for tasks that do not require high precision or are prone to catastrophic actions, and that *Reward-Phasing* produces sub-optimal results when IRL fails to accurately mimic the demonstrator.

### Acknowledgements

1. The reported work has taken place in the *PiStar* AI and Optimization Lab at Texas A&M. PiStar research is supported in part by NSF (IIS-2238979).
2. A portion of this work was funded by *Ultra Electronics Advanced Tactical Systems, Inc.* and *Ultra Labs*.
3. A portion of this work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, FAIN-2019844), ONR (N00014-18-2243), ARO (W911NF-19-2-0333), DARPA, GM, Bosch, and UT Austin’s Good Systems grand challenge.
4. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.
5. A portion of this work was funded by the Mays Innovation Research Center at Texas A&M

### References

- Abbeel, P.; and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 1.
- Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; and Zaremba, W. 2017. Hindsight experience replay. *Advances in neural information processing systems*, 30.
- Ault, J.; Hanna, J. P.; and Sharon, G. 2020. Learning an Interpretable Traffic Signal Control Policy. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS ’20*, 88–96. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.
- Ault, J.; and Sharon, G. 2021. Reinforcement Learning Benchmarks for Traffic Signal Control. In *Proceedings of the 35th Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks*.
- Barto, A. G. 2013. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, 17–47. Springer.
- Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. volume 60, 6.
- Burda, Y.; Edwards, H.; Storkey, A.; and Klimov, O. 2019. Exploration by random network distillation. In *Seventh International Conference on Learning Representations*, 1–17.
- Dennis, M.; Jaques, N.; Vinitzky, E.; Bayen, A. M.; Russell, S. J.; Critch, A.; and Levine, S. 2020. Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design. *ArXiv*, abs/2012.02096.
- Dey, S.; Pendurkar, S.; Sharon, G.; and Hanna, J. P. 2021. A Joint Imitation-Reinforcement Learning Framework for Reduced Baseline Regret. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3485–3491.
- Durugkar, I.; Tec, M.; Niekum, S.; and Stone, P. 2021. Adversarial Intrinsic Motivation for Reinforcement Learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NeurIPS 2021)*.
- Ecoffet, A.; Huizinga, J.; Lehman, J.; Stanley, K. O.; and Clune, J. 2019. Go-Explore: a New Approach for Hard-Exploration Problems. *ArXiv*, abs/1901.10995.
- Erceth. 2020. PyFlags.
- Fu, J.; Luo, K.; and Levine, S. 2017. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*.
- 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.
- Ho, J.; and Ermon, S. 2016a. Generative Adversarial Imitation Learning. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.
- Ho, J.; and Ermon, S. 2016b. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29.
- Ionescu, R. T.; Alexe, B.; Leordeanu, M.; Popescu, M.; Papadopoulos, D.; and Ferrari, V. 2016. How Hard Can It Be? Estimating the Difficulty of Visual Search in an Image. 2157–2166.
- Jiménez-Sánchez, A.; Mateus, D.; Kirchoff, S.; Kirchoff, C.; Biberthaler, P.; Navab, N.; Ballester, M. Á. G.; and Piella, G. 2019. Medical-based Deep Curriculum Learning for Improved Fracture Classification. In *MICCAI*.
- Kang, B.; Jie, Z.; and Feng, J. 2018. Policy optimization with demonstrations. In *International conference on machine learning*, 2469–2478. PMLR.
- Kiran, B. R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A. A.; Yogamani, S.; and Pérez, P. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*.
- Kostrikov, I.; Agrawal, K. K.; Dwibedi, D.; Levine, S.; and Tompson, J. 2018. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*.
- Levine, S.; and Koltun, V. 2013. Guided Policy Search. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 1–9. Atlanta, Georgia, USA: PMLR.
- Lotfian, R.; and Busso, C. 2019. Curriculum Learning for Speech Emotion Recognition From Crowdsourced Labels. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(4): 815–826.

- Matiisen, T.; Oliver, A.; Cohen, T.; and Schulman, J. 2019. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9): 3732–3740.
- Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6292–6299. IEEE.
- Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M. E.; and Stone, P. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Journal of Machine Learning Research*, 21(181): 1–50.
- Narvekar, S.; Sinapov, J.; Leonetti, M.; and Stone, P. 2016. Source Task Creation for Curriculum Learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, AAMAS '16*, 566–574. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450342391.
- Oudeyer, P.-Y.; and Kaplan, F. 2009. What is intrinsic motivation? A typology of computational approaches. *Frontiers in neurobotics*, 1: 6.
- Oudeyer, P.-Y.; and Kaplan, F. 2013. How can we define intrinsic motivation ?
- Pentina, A.; Sharmanska, V.; and Lampert, C. 2015. Curriculum learning of multiple tasks. 5492–5500.
- Plappert, M.; Andrychowicz, M.; Ray, A.; McGrew, B.; Baker, B.; Powell, G.; Schneider, J.; Tobin, J.; Chociej, M.; Welinder, P.; Kumar, V.; and Zaremba, W. 2018. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. *ArXiv*, abs/1802.09464.
- Portelas, R.; Colas, C.; Hofmann, K.; and Oudeyer, P.-Y. 2020a. Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments. In Kaelbling, L. P.; Kragic, D.; and Sugiura, K., eds., *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, 835–853. PMLR.
- Portelas, R.; Colas, C.; Weng, L.; Hofmann, K.; and Oudeyer, P.-Y. 2020b. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*.
- Reddy, S.; Dragan, A. D.; and Levine, S. 2019. SQIL: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*.
- Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635. JMLR Workshop and Conference Proceedings.
- Salimans, T.; and Chen, R. 2018. Learning Montezuma’s Revenge from a Single Demonstration. *arXiv preprint arXiv:1812.03381*.
- Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust Region Policy Optimization. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 1889–1897. Lille, France: PMLR.
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms.
- Sharon, G. 2021. Alleviating Road Traffic Congestion with Artificial Intelligence. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 4965–4969. International Joint Conferences on Artificial Intelligence Organization. Early Career.
- Soviany, P.; Ionescu, R. T.; Rota, P.; and Sebe, N. 2021. Curriculum Learning: A Survey. *ArXiv*, abs/2101.10382.
- Taylor, M. E.; and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7).
- Torabi, F.; Warnell, G.; and Stone, P. 2018. Behavioral cloning from observation. 4950–4957.
- Vecerik, M.; Hester, T.; Scholz, J.; Wang, F.; Pietquin, O.; Piot, B.; Heess, N.; Rothörl, T.; Lampe, T.; and Riedmiller, M. 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*.
- Wei, J.; Suriawinata, A.; Ren, B.; Liu, X.; Lisovsky, M.; Vaickus, L.; Brown, C.; Baker, M.; Nasir-Moin, M.; Tomita, N.; Torresani, L.; Wei, J.; and Hassanpour, S. 2020. Learn like a Pathologist: Curriculum Learning by Annotator Agreement for Histopathology Image Classification.
- Weinshall, D.; and Amir, D. 2020. Theory of curriculum learning, with convex loss functions. *Journal of Machine Learning Research*, 21(222): 1–19.
- Weinshall, D.; Cohen, G.; and Amir, D. 2018. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International Conference on Machine Learning*, 5238–5246. PMLR.
- Yang, T.; Tang, H.; Bai, C.; Liu, J.; Hao, J.; Meng, Z.; and Liu, P. 2021. Exploration in deep reinforcement learning: a comprehensive survey. *arXiv preprint arXiv:2109.06668*.
- Yengera, G.; Devidze, R.; Kamalaruban, P.; and Singla, A. 2021. Curriculum Design for Teaching via Demonstrations: Theory and Applications. *Advances in Neural Information Processing Systems*, 34: 10496–10509.
- Zhao, E.; Deng, S.; Zang, Y.; Kang, Y.; Li, K.; and Xing, J. 2020. Potential Driven Reinforcement Learning for Hard Exploration Tasks. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2096–2102. International Joint Conferences on Artificial Intelligence Organization. Main track.
- Zhu, Z.; Lin, K.; Dai, B.; and Zhou, J. 2022. Self-Adaptive Imitation Learning: Learning Tasks with Delayed Rewards from Sub-optimal Demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8): 9269–9277.