Unsupervised Training Sequence Design: Efficient and Generalizable Agent Training

Wenjun Li, Pradeep Varakantham
Singapore Management University
wjli.2020@phdcs.smu.edu.sg, pradeepv@smu.edu.sg

Abstract

To train generalizable Reinforcement Learning (RL) agents, researchers recently proposed the Unsupervised Environment Design (UED) framework, in which a teacher agent creates a very large number of training environments and a student agent trains on the experiences in these environments to be robust against unseen testing scenarios. For example, to train a student to master the "stepping over stumps" task, the teacher will create numerous training environments with varying stump heights and shapes. In this paper, we argue that UED neglects training efficiency and its need for a very large number of environments (henceforth referred to as infinite horizon training) makes it less suitable for training robots and non-expert humans. In real-world applications where either creating new training scenarios is expensive or training efficiency is of critical importance, we want to maximize both the learning efficiency and learning outcome of the student. To achieve efficient finite horizon training, we propose a novel Markov Decision Process (MDP) formulation for the teacher agent, referred to as *Unsupervised Training Sequence Design* (UTSD). Specifically, we encode salient information from the student policy (e.g., behaviors and learning progress) into the teacher's state space, enabling the teacher to closely track the student's learning progress and consequently discover the optimal training sequences with finite lengths. Additionally, we explore the teacher's efficient adaptation to unseen students at test time by employing the context-based meta-learning approach, which leverages the teacher's past experiences with various students. Finally, we empirically demonstrate our teacher's capability to design efficient and effective training sequences for students with varying capabilities.

Introduction

In order to train generalizable Reinforcement Learning (RL) agents that are robust to various challenges and unseen scenarios, researchers have proposed Distributional RL (Bellemare, Dabney, and Munos 2017; Brunke et al. 2022), model-based RL (Kaiser et al. 2019; Moerland et al. 2023) and adversarial RL (Pinto et al. 2017; He et al. 2022). Recently, Unsupervised Environment Design (Dennis et al. 2020; Jiang, Grefenstette, and Rocktäschel 2021; Jiang et al. 2021; Parker-Holder et al. 2022; Li, Varakantham, and Li 2023; Li, Li, and Varakantham 2023; Tio and Varakantham 2023)) formulates a training framework between a teacher agent and a student agent, where the teacher creates a very large number of training environments (e.g., mazes with different positions of obstacles and car-racing games with various track designs) to improve the student's generalization ability such that it is robust to "out-of-distribution" (OOD) scenarios. UED algorithms have been empirically shown to help RL agents achieve state-of-the-art generalization performance.

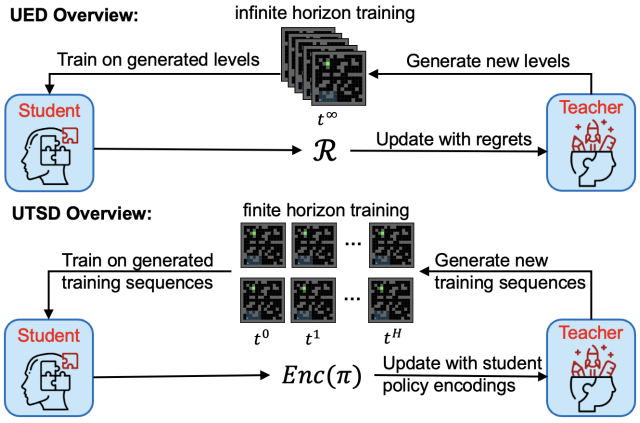

Figure 1: A high-level overview of UED and USTD.

The existing UED algorithms seek to generate an infinite number of environments and open-endedly train the student based on the *regret* notion (we will introduce regret in detail in Section 2). For example, to help the student generalize to a variety of mazes, the state-of-the-art algorithm (Parker-Holder et al. 2022) creates hundreds of thousands of training mazes, but such a massive amount of training is usually low-efficient and the student cannot learn anything in most of the mazes. Consequently, such infinite training is less applicable to real-world tasks where training efficiency is critical. On top of achieving well-generalizing agents, we also want to maximize the agent training efficiency with as few environments or tasks as possible in real-world tasks. For example, to train a robot to climb stairs, we cannot afford to create tens of thousands of real stairs with various heights and slopes in the laboratory. Instead, it is desirable for the robot to generalize well to a variety of stairs with just a few training

environments. Besides, to teach non-expert humans cooking skills, we want to make sure the cooking tasks in the training curriculum are optimally arranged regarding both training efficiency and training outcome. Both example training tasks require efficient finite horizon training, which cannot be addressed by the UED framework.

To that end, we make four key contributions. First, we propose a new training framework, UTSD, to enable the teacher to discover efficient training sequences. Second, we employ the Quality Diversity approach to select diverse validation environments with respect to the student policy and subsequently encode the student policy into the state space of the teacher. Third, we explore the teacher's rapid adaptation to unseen students by employing the context-based meta-RL approach to encode students with various properties into latent variables. Finally, we empirically demonstrate that our proposed teacher method can not only discover training sequences that maximize students' learning efficiency and final generalization performance but also rapidly adapt to unseen students with just a few interactions.

Background

In this section, we briefly review the background of unsupervised environment design and quality diversity.

Unsupervised Environment Design

The Unsupervised Environment Design (UED) framework assumes a teacher agent and a student agent, where the teacher creates an infinite number of environments to train the student such that it can generalize well to a variety of environments. UED problems are formally defined on an Underspecified Partially Observable Markov Decision Process (UPOMDP) using the following tuple:

$$\langle S, A, \Theta, I, O, T, R, \gamma \rangle$$

where S, A and O are the set of states, actions and observations respectively. T and R are the transition and reward functions respectively. γ is the discount factor. The most important element in the tuple is Θ, which is the representation of environments. A particular representation $\theta \in \Theta$ (can be an encoding of the environment, or sequence of values) defines an environment and can affect the transition and observation function, i.e. $T : S \times A \times \Theta \to S$ and $I : S \times \Theta \to O$. The goal of the teacher agent policy Λ is to generate a distribution over the next set of environment parameter values (i.e., $\Delta(\Theta)$) to train the student such that the student is robust to any given $\theta \in \Theta$, which can be written as:

$$\Lambda : \Pi \to \Delta(\Theta) \quad \text{s.t.}$$

$$\max_{\pi} V^{\theta}(\pi) = \max_{\pi} \mathbb{E}_{\pi} V^{\theta}(\tau) = \max_{\pi} \mathbb{E}_{\pi} \Big[\sum_{t=0}^{H} r_t^{\theta} \cdot \gamma^t \Big]$$

where Π is the set of possible policies of the teacher, r_t^{θ} is the reward obtained by student policy π in an environment with environment representation θ at time step t.

In all approaches for solving UED, the student always optimizes a policy that maximizes its value on the given θ. The different approaches for solving UED vary on the method adopted by the teacher. We now elaborate on the Λ employed by existing UED algorithms. Two of the earliest approaches to UED are Domain Randomization (DR, (Jakobi 1997; Tobin et al. 2017)), which uniformly randomizes the environment configurations regardless of the student's policy, and Minimax (Morimoto and Doya 2005; Pinto, Davidson, and Gupta 2017) that adversarially generates challenging environments to minimize the rewards of the student's policy. These fundamental approaches are significantly outperformed by the Protagonist Antagonist Induced Regret Environment Design (PAIRED, (Dennis et al. 2020)) algorithm which focuses on the principled generation of environments based on the *regret* notion. The regret is defined approximately as the difference between the maximum and the mean return of the student's policy:

$$regret^{\theta}(\pi) \approx \max_{\tau \sim \pi} V^{\theta}(\tau) - \mathbb{E}_{\tau \sim \pi} V^{\theta}(\tau)$$

Later on, PLR$^{\perp}$ (Jiang et al. 2021) combines the random environment generator with environment prioritization based on an efficient approximation of regret, referred to as *positive value loss* (a customized form of Generalized Advantage Estimation (GAE)).

$$gae^{\theta}(\pi) = \frac{1}{H} \sum_{t=0}^{H} \max \left(\sum_{k=t}^{H} (\gamma\lambda)^{k-t} \delta_k, 0 \right) \tag{1}$$

where γ and λ are the MDP and GAE discount factors respectively, H is the time horizon and δ_k is the TD-error at time step k.

The state-of-the-art algorithm, ACCEL (Parker-Holder et al. 2022), performs random edits on the generated high-regret environments through human-defined step size so that it can efficiently explore the environment space and achieve a smoothly evolving curriculum.

The leading UED algorithms all rely on regret to create an infinite number of new training environments for the student. However, such infinite horizon training is not suitable for training robots and humans, and we aim to address efficient finite horizon training in this paper.

Quality Diversity

Quality Diversity (QD, (Mouret and Clune 2015; Pugh, Soros, and Stanley 2016)) aims to generate a set of diverse solutions to the given problem, where each solution ϕ is evaluated based on both its quality and its diversity. A QD problem defines an objective function $f : \mathbb{R}^n \to \mathbb{R}$ that evaluates the solution ϕ's quality, and a measure function $m : \mathbb{R}^n \to \mathbb{R}^k$ that measures the solution diversity in the descriptor space D. The objective function is typically the expected cumulative rewards and the measure function is used to differentiate between solutions and computes a descriptive vector of features in the Cartesian space, e.g., how long the agent takes to learn a task and the agent's unit fuel consumption. Practically, the continuous descriptor space D is discretized into N cells, and each solution is mapped to a cell $d \in D$ based on its measure $m(\phi)$. Note that each cell contains at most one solution and empty cells have an objective value of 0. The solutions that occupy cells form

an *archive* of solutions. A QD algorithm aims to find a set of solutions $\{\phi_1, \phi_2, ..., \phi_N\}$ that maximize the QD-score, which is defined as:

$$\sum_{i=1}^{N} f(\phi_i) \tag{2}$$

The performance of a QD algorithm is also evaluated by the coverage of the descriptor space: $\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\phi_i}$, where $\mathbf{1}_{\phi_i}$ means cell i is occupied by a solution ϕ_i.

We can put QD into the context of UED and assume a student policy π acts in a UPOMDP defined on a parameter space Θ. When applying QD to UED, QD evaluates an environment with an objective value $f(\theta, \pi) : \mathbb{R}^n \to \mathbb{R}$ and a measure function $m(\theta, \pi) : \mathbb{R}^n \to \mathbb{R}^k$. In this setting, the QD algorithm aims to find environments that maximize f but are diverse with respect to the diversity measure m. In the literature, researchers (Fontaine and Nikolaidis 2020; Fontaine et al. 2021a,b; Bhatt et al. 2022) employ QD to generate diverse training and testing environments with a fixed π. In this paper, we adopt the QD approach to select validation environments that elicit diverse policy behaviors and subsequently encode comprehensive information about the student policy with the student's validation performance.

UTSD

In UED, the teacher is learning an infinite horizon MDP, whose state space has not been formally defined. To discover efficient finite horizon training, we introduce a new MDP formulation for the teacher, which is referred to as the teacher MDP and can be formally written as:

$$M = \langle S, A, T, R, H \rangle$$

where S, A, T and R are the set of teacher's state space, action space, transition function and reward function respectively, and H is the horizon limit. Two significant differences between this teacher MDP and the loosely defined infinite MDP in UED are the state space and the horizon. First, the teacher's state space S encodes salient information about the student's policy (e.g., overall abilities and behaviors), allowing the teacher to closely monitor the student's learning progress. Second, teacher MDP has a horizon constraint that motivates the teacher to discover highly efficient and effective training sequences. We refer to this finite horizon training framework as *Unsupervised Training Sequence Design* (UTSD), where the student learns in a low-level environment defined on a UPOMDP and the teacher learns on the high-level teacher MDP.

In this paper, we denote the teacher policy and student policy by Π and π respectively. The teacher's objective is to maximize its expected rewards:

$$\begin{aligned} \arg\max_{\Pi} \mathbb{E}_{\Pi}\Big[\sum_{t=0}^{H} r_t \cdot \gamma^t\Big] \\ \text{s.t. } r_t &= V^{\Theta}(\pi_t) \\ \pi_t &= \text{PPO}(\theta_t, \pi_{t-1}) \\ \theta_t &= \Pi(\cdot|\pi_{t-1}) \end{aligned} \tag{3}$$

where θ_t is a particular training environment at time step t, PPO can be replaced with any deep RL algorithms, $V^{\Theta}(\cdot)$ evaluates the student's generalization performance across any $\theta \in \Theta$. In this paper, we use the *validation performance* to approximate $V^{\Theta}(\cdot)$, and we will elaborate on this method in the next section. Through this bi-level training, the teacher finally finds a finite horizon training sequence that maximizes the student's generalization performance.

Approach

The most challenging part of UTSD is how to include the student policy into the state space of the teacher, i.e., how to encode the student policy from deep neural network parameters into meaningful compact representations. To ensure scalability, we propose to encode the student policy network with its validation performance in diverse environments selected with the Quality Diversity method. Additionally, to achieve efficient teacher adaptation to unseen students, we employ the context-based meta-RL method to help the teacher aggregate past experiences with various students. Specifically, our approach makes two key contributions:

1. A scalable agent policy encoding method, which can help the teacher in UTSD closely track the student's overall ability and behaviors and consequently design efficient training sequences with finite length.
2. Train a generalizable teacher that can rapidly adapt to unseen students with various learning patterns and capabilities by employing the context-based meta-RL approach.

Student Policy Encoding

In this section, we elaborate on how to collect a set of diverse environments regarding the student agent policy behaviors with the Quality Diversity method.

Assume a UPOMDP environment defined on a parameter space $\Theta \subseteq \mathbb{R}^m$, where a particular parameter vector $\theta \in \Theta$ determines the environment properties, e.g., gravity and friction coefficient. To construct the set of validation environments, we don't directly select validation environments in the parameter space because of two main reasons. First, the parameter values may not be linear to policy behaviors. Because of such non-linearity, sampling in the parameter space

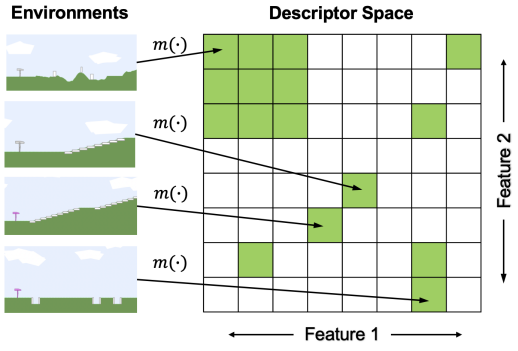

Figure 2: An illustration of the QD method.

does not necessarily create a validation set with high coverage in the policy behavior space. A validation set with low coverage in the behavior space can result in poor estimation of the agent's generalization ability. Second, the size of the validation set is exponential to the dimension of the parameter space and thus it's hard to select a validation set with reasonable size when m is large. By considering the diversity regarding policy behaviors in validation environments, we could reduce the size of the validation set by eliminating environments eliciting similar policy behaviors.

Instead of directly sampling in the environment parameter space, we strive to identify environments that elicit diverse policy behaviors and unique skills. To accomplish this, we employ QD in the environment generation process to create diverse environments regarding policy behaviors. We provide the procedures for constructing the validation set in Algorithm 2 in the appendix. We further provide the ablation study of how QD affects the teacher's performance and reduce the size of the validation set in the appendix.

We concatenate the student's performance in each validation environment and form a compact encoding of the student policy regarding its overall ability. On top of using the validation performance as an encoding method for the student policy, we further design the teacher's reward function based on the student's validation performance. It's straightforward to assign the teacher higher rewards when it helps enhance the student's overall abilities and achieve better validation performance. The encoding and reward function is formally written in Equation (4) and Equation (5) respectively,

$$Enc_i(\pi_t) = \mathbb{E}_{\tau\sim\pi_t}[V^{\theta_i}(\tau)] \tag{4}$$

$$r_t = \sum_{\theta_i\sim\tilde{\theta}} \mathbb{E}_{\tau\sim\pi_t}[V^{\theta_i}(\tau)] \tag{5}$$

where $Enc_i(\pi_t)$ is the i-th element in the encoding vector, $V^{\theta_i}(\pi_t)$ evaluates the student policy π_t on the validation environment configured by θ_i. θ_i is drawn from $\tilde{\theta}$, which is the set of validation environment parameters and $\tilde{\theta} \in \Theta$. Note that the teacher MDP is terminated once the student achieves optimal performance in all validation environments.

Context-based Student Embedding

To achieve practical application potential, where the teacher should efficiently adapt to students with varying learning patterns and capabilities, we further explore the teacher's transferability by employing the meta-RL approach.

Meta-RL aims to train an agent that can quickly adapt to any task $\mathcal{T}$ drawn from a distribution of tasks $\rho(\mathcal{T})$, where each task is modeled as an MDP:

$$\mathcal{T} = \{p(s_0), p(s_{t+1}|s_t, a_t), r(s_t, a_t)\}$$

where $p(s_0)$ is the initial state distribution, and $p(s_{t+1}|s_t, a_t)$ and $r(s_t, a_t)$ are the transition function and reward function respectively. Meta-RL assumes all tasks in $\rho(\mathcal{T})$ share the same state and action space but may differ in transition and reward functions. The goal of meta-RL is to maximize the expected reward of the agent policy π over the task distribution:

$$\mathbb{E}_{\mathcal{T}\in\rho(\mathcal{T})}\Big[\mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{H} r(s_t, a_t)\cdot\gamma^t]\Big] \tag{6}$$

where $r(s_t, a_t)$ is the reward to the agent at time step t and γ is the MDP discount factor.

There are two threads in meta-RL: (1) *gradient-based* algorithms learn a policy initialization that can attain single-task level performance on new tasks after one or few policy gradient steps, e.g., (Finn, Abbeel, and Levine 2017; Rothfuss et al. 2018; Sung et al. 2017; Houthooft et al. 2018). (2) *context-based* algorithms learn from past experience by leveraging "context", e.g., (Duan et al. 2016; Wang et al. 2016; Fakoor et al. 2019). In general, context-based methods are more sample-efficient regarding adaptation and can generalize better to new tasks (Rakelly et al. 2019). Therefore, given our objective of enabling the teacher to efficiently generalize to new students, we adopt the context-based meta-RL method to encode various students' learning patterns on the training tasks into latent variables, enabling reasoning over student uncertainty and rapid adaptation to new students.

Context-based meta-RL decomposes the meta-RL solution into two parts: a context encoder that infers the task context and a conditional policy based on the context. In the literature, *context* c refers to the past collections of transitions. Assume $c_n^{\mathcal{T}} = (s_n, a_n, r_n, s'_n)$ be one transition in the task $\mathcal{T}$ such that $c_{1:N}^{\mathcal{T}}$ comprises the experience collected so far. The context encoder q_ϕ is an inference network that computes a latent variable z given the task context $c_{1:N}^{\mathcal{T}}$. The conditional agent policy acts based on z, i.e., $\pi(a|s, z)$. Context-based meta-RL algorithms differ in context representation methods, context sampling approaches, etc.

We now formulate the student training as a meta-RL problem. Assume the teacher is training various students to generalize well on a UPOMDP defined on Θ, and each student policy π corresponds to a task:

$$\mathcal{T}_\pi^\Theta = \{p(\pi_0), p(\pi_{t+1}|\pi_t, a_t), r(\pi_t, a_t)\}$$

where π_t is the student policy at time step t and can be encoded into a compact representation, for example, using the policy encoding method introduced in this paper. $p(\pi_{t+1}|\pi_t, a_t)$ is the transition function conditioned by the student's learning patterns and capabilities, and $r(\pi_t, a_t)$ is the reward function to the teacher. Note that a unique student policy drawn from $\rho(\pi)$ corresponds to a teacher MDP defined in Section 3.

Meta-teacher Training

In this section, we build the algorithm to train a teacher agent with generalizability, enabling it to aggregate past experiences from interactions with various students and rapidly adapt to new students. After aligning the student training task with meta-RL formulations, we can directly apply meta-RL algorithms to the teacher. To ensure sample-efficient training, we employ the PEARL (Rakelly et al. 2019) algorithm, which utilizes the context replay buffer and context sampler to achieve sample-efficient off-policy meta-learning. Please refer to the appendix for more details about

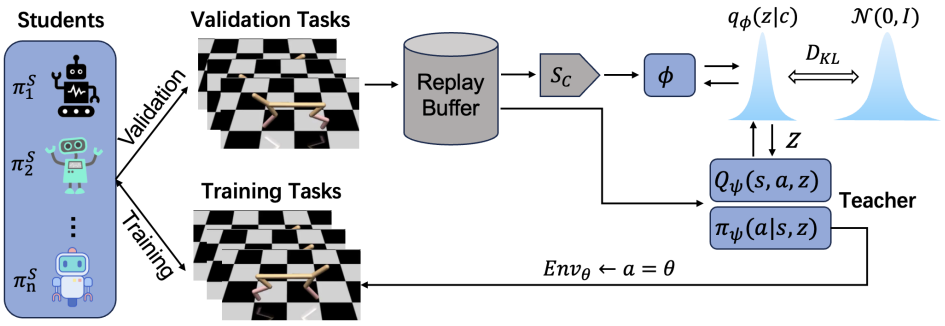

Figure 3: An Overview of the meta-teacher algorithm: at each step t, the teacher performs the following actions: (1) samples an action a_t and creates a new training environment with θ_t (note that $a_t = \theta_t$); (2) trains the student on the environment Env_{θ_t} and encodes the student policy with its validation performance; (3) computes the teacher reward. When the teacher MDP terminates, we update the replay buffer with the collected transitions and train the meta-teacher with the PEARL algorithm.

the PEARL algorithm and our implementations. We refer to our proposed algorithm as the meta-teacher and provide an overview of it in Figure 3.

Our objective is to train a well-generalizing teacher that can rapidly adapt to any new students with various learning properties, including initial ability level, maximal capability, learning speed, etc. The teacher is trained on tasks associated with different students, where each task is a UTSD problem and is modeled as a teacher MDP. The students learn on the training sequences provided by the teacher and can have different learning patterns (i.e., by adopting different RL algorithms, e.g., DQN (Mnih et al. 2013), PPO (Schulman et al. 2017), and SAC (Haarnoja et al. 2018)), and varying initial abilities (i.e., by different amount of pre-training). After a certain amount of learning, the teacher measures the learning progress and overall abilities of the students by evaluating them in validation environments. The transitions $(\pi_t, a_t, r_t, \pi_{t+1})$ are accumulated in a replay buffer and the teacher is trained on these transitions with the PEARL algorithm. The detailed training procedures of the meta-teacher are summarized in Algorithm 1.

Algorithm 1: Train meta-teacher

Input: a random teacher policy with actor Π_ψ and critic Q_ψ, a distribution of students $\rho(\pi)$, set of validation environment parameters $\tilde{\theta}$, horizon H, context sampler S_c

Initialize replay buffers $\mathcal{B}$
while *not converged* **do**
  Sample an initial student policy $\pi_0 \sim \rho(\pi)$
  Initialize context $c = \{\}$
  Encode student policy, $s_0 = Enc(\pi_0)$
  for $t = 0, 1, 2, ..., H$ **do**
    Sample $z \sim q_\phi(z|c)$
    Teacher samples an action, $a_t \sim \Pi_\psi(\cdot|s, z)$
    Teacher creates a new training environment with $\theta_t = a_t$
    Train student on θ_t and update its policy
    Encode student policy, $s_t = Enc(\pi_t)$
    Compute teacher reward, $r_t = \sum_{\tilde{\theta}} \mathbb{E}[V^{\tilde{\theta}}(\pi_t)]$
  end
  Update $\mathcal{B}$ with $c = \left\{(s_t^\pi, a_t, s_{t+1}^\pi, r_t)\right\}_{t=0:H}$
  for *step in training steps* **do**
    Sample RL batch $b \sim \mathcal{B}$ and context $c \sim S_c(\mathcal{B})$
    Update Π_ψ, Q_ψ and q_ϕ with b and c
  end
end

Experiments

In this section, we empirically validate the effectiveness of the UTSD framework and demonstrate the transferability of the meta-teacher by comparing it to a set of leading baselines in UED: Domain Randomization (DR), PAIRED, PLR$^\perp$, and ACCEL. We conduct experiments on three popular yet distinct benchmarks in UED: Bit-Flipping, Lunar-Lander, and Minigrid. Note that these experiments serve as proof-of-concept empirical studies. In future work, we will transition to human-related training. For extended experiment results on teacher training, an ablation study on how the QD-based validation set affects algorithm performance, implementation details about teacher and student structures, and hyperparameter settings, please refer to the appendix.

Specifically, we aim to demonstrate two key results. First, teachers trained in UED struggle to discover efficient training sequences, whereas our proposed meta-teacher, trained with UTSD, can maximize both training efficiency and training outcomes. Second, the proposed meta-teacher can quickly adapt to new students and provide efficient training sequences correspondingly.

Bit-Flipping Domain

The Bit-Flipping environment, introduced by (Andrychowicz et al. 2017), is widely used in RL for its efficiency. In this task, the agent starts with two uniformly randomized binary vectors with the same length – the initial vector and the target vector. The agent needs to flip the bits in the ini-

tial vector one by one until it matches the target vector. Assume the vector length is n, then the agent has a state space $S = \{0, 1\}^n$, and an action space $A = \{0, 1, ..., n-1\}$. The agent selects an action i to flip the i-th bit in the initial vector. To parameterize the Bit-Flipping environment and distinguish between different tasks, we introduce a variable called *number of free bits* (denoted by m, where $m \leq n$). In our experiments, the maximum training sequence length is set to 12 and the training amount on each task is fixed at 5k steps. We use m, where $1 \leq m \leq 12$, to represent a Bit-Flipping task with m free bits. A larger m corresponds to a more challenging task, requiring the agent to take more actions on the initial vector to achieve the goal. In Bit-Flipping, the sequencing of tasks is crucial. Specifically, the student agent cannot directly master the most difficult task (i.e., $m = 12$). In contrast, the student can learn to solve the most difficult task given a carefully curated training curriculum that progresses from easy tasks to hard tasks, such as $TS = \{1, 2, 3, ..., 11, 12\}$, which is designed by humans. Note that we use *TS* to represent a training sequence.

First, we demonstrate that teachers trained with UED algorithms struggle to find efficient training sequences for student agents, whereas our proposed meta-teacher excels in maximizing training efficiency. Averaged across all students in the experiments, our teacher policy converges to discover efficient training sequences with a length of 9.08 ± 0.13 (average $\pm$ standard deviation, calculated based on five independent runs). In contrast, none of the UED teachers can reduce the training sequence length. A typical training sequence found by meta-teacher is $TS = \{2, 3, 3, 6, 6, 6, 8, 10, 12\}$ with a length of 9. In comparison, the human-designed training sequence has a length of 12, which takes three more training environments to achieve the same effect as that of the meta-teacher. Figure 4 illustrates the student's generalization performance after learning in 12 environments with a max learning amount of 60k steps.

Second, we show that UED teachers cannot adapt to new students rapidly. We transfer the trained teachers to new students with varying learning properties and capabilities, including different RL algorithms and network structures/sizes, and illustrate the teacher's adaptation performance in Figure 5. Note that, we enable the learning and gradient updates for UED teachers during the transfer, especially for those (i.e., PLR$^\perp$ and ACCEL) heavily reliant on replay buffers. These teachers are not immediately transferable because their replay buffers are empty upon transfer to new students. Besides, the PAIRED teacher fails to adapt to new students efficiently due to overfitting during training and the iterative generation of hard tasks. In contrast, the proposed meta-teacher can swiftly and effectively adapt to new students with diverse learning properties and capabilities, providing efficient and effective training sequences.

Lunar-Lander Domain

We also conduct experiments in a continuous domain, Lunar-Lander (Brockman et al. 2016), as depicted in Figure 6 (a). In Lunar-Lander, the agent aims to land on the flat platform by controlling its main engine power (MEP) and side engine power (SEP). These engines provide vertical and rotational force, respectively. The reward is based on the quality of the landing, considering factors such as fuel usage and impact velocity. Safe landings yield high positive rewards, while crashes result in high negative rewards. The parameter space in Lunar-Lander consists of two continuous variables: MEP and SEP, defining the engine power ranges. In Lunar-Lander, we set the maximum training sequence length to 12. The teacher can sequentially create 12 training environments with various MEP and SEP values. The training amount on each training environment is fixed at 25k steps. In our experiment, the teacher creates training environments with MEP and SEP values in [10, 18] and [1, 6], respectively. The testing environment parameters are uniformly sampled from [8, 22] and [0, 8] for MEP and SEP, respectively. This allows us to assess the student's generalization performance in out-of-distribution (OOD) environments.

In Lunar-Lander, the meta-teacher demonstrates the ability to reduce the training sequence length from 12 to an optimized value of 10.38 ± 0.11, whereas UED methods fail to discover training sequences with fewer than 12 steps until convergence. The averaged student generalization performance, based on learning in 12 environments with a maximum of 300k steps, is presented in Figure 7.

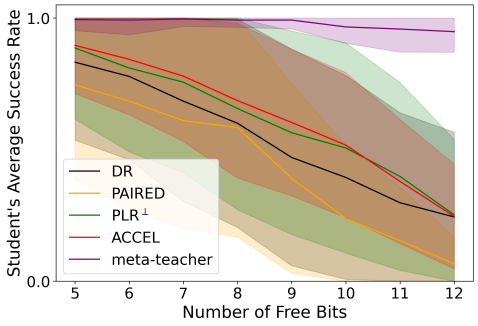

Figure 4: Normalized student's generalization performance in Bit-Fipping after learning on 12 environments with a maximum learning amount of 60k steps.

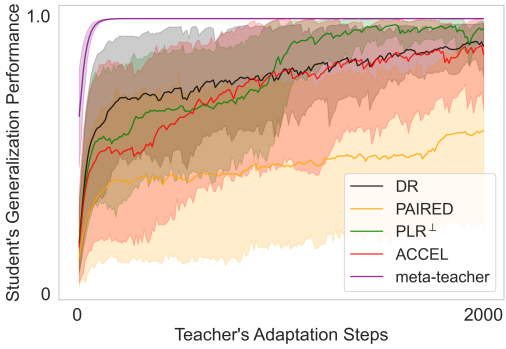

Figure 5: Normalized student's generalization performance with respect to teacher's adaptation steps over five independent runs (mean and standard error) in Bit-Flipping domain, where one step corresponds to one training environment.

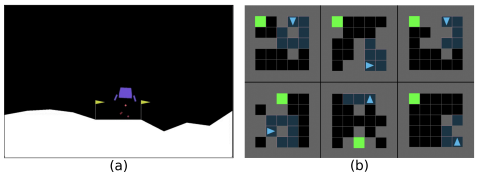

Figure 6: (a) Lunar-Lander domain. (b) Examples of training environments in Minigrid domain.

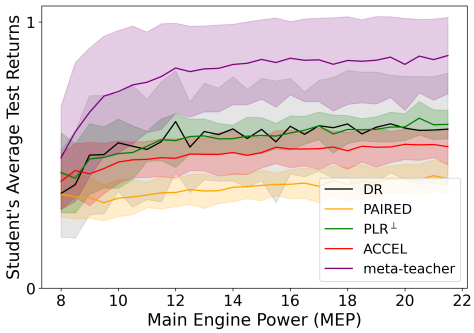

Figure 7: Normalized student's generalization performance in Lunar-Lander after learning on 12 environments with a maximum learning amount of 300k steps.

After convergence, we evaluate the teacher's transfer performance to new students and plot the curves in Figure 8. It's noteworthy that, in Lunar-Lander, as the student's learning amount increases, its generalization performance decreases. This is attributed to the student gradually overfitting the training (MEP, SEP) values, and the knowledge gained in these environments becomes less applicable to settings with different parameter values. Despite this phenomenon, the proposed meta-teacher consistently helps students achieve better generalization performance. An additional advantage of using the validation performance encoding is that it enables us to halt the training once the student attains optimal performance, preventing overfitting – a challenge not effectively addressed by standard UED algorithms.

Minigrid Domain

Unlike the above domains, Minigrid is a non-parameterized environment where a particular environment (i.e. a maze) cannot be represented by a compact parameter vector due to its high complexity. To simplify the teacher's action space and focus on validating the effectiveness of our proposed method, teachers sample mazes from a diverse set generated by the recently proposed DSAGE algorithm (Bhatt et al. 2022). Refer to the appendix for more details about the DSAGE algorithm. It's crucial to note that the Minigrid domain is very hard to learn. Even the state-of-the-art algorithm, ACCEL, creates hundreds of thousands of mazes to train a well-generalizing agent. To alleviate the massive training demand, we choose to conduct experiments in 5x5 mazes, where the student has a partial observability of 3x3.

To ensure a reliable and straightforward comparison, we

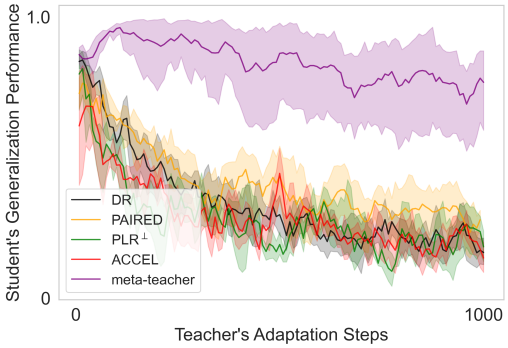

Figure 8: Normalized student's generalization performance with respect to teacher's adaptation steps over five independent runs (mean and standard error) in Lunar-Lander domain, where one step corresponds to one environment.

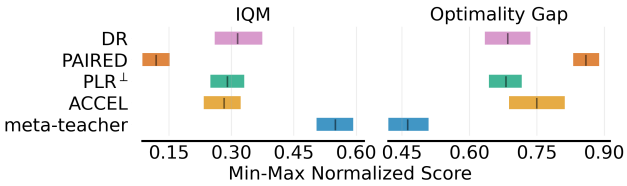

Figure 9: Aggregate generalization performance of students in Minigrid Domain across five independent runs. Higher IQM scores and lower optimality gaps are better.

adopt the standardized RL evaluation method (Agarwal et al. 2021) recently introduced. This method shows the aggregate inter-quartile mean (IQM) and optimality gap. After training the teachers, we transfer them to new students and have them generate a training sequence of fifty mazes. The students' generalization performances are then measured and the standardized results are presented in Figure 9. Notably, the meta-teacher consistently produces the best training outcomes. Due to the space limitations, additional experimental results in the Minigrid domain are provided in the appendix.

Conclusion

In this paper, we addressed finite horizon training by proposing a novel MDP formulation for the teacher agent, referred to as Unsupervised Training Sequence Design (UTSD). We employed the Quality Diversity approach to select diverse validation environments and subsequently encode salient information about the student policy into the state space of the teacher agent. This allows the teacher to track the learning progress and overall ability of the student during training. To make the teacher more applicable to real-world tasks, we further enhance the teacher's transferability to new students by utilizing the context-based meta-RL method. In our experiments, we validated that the proposed meta-teacher can not only create efficient and effective finite horizon training sequences but is also capable of rapidly transferring to new students with varying learning properties.

Acknowledgments

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-017).

References

Agarwal, R.; Schwarzer, M.; Castro, P. S.; Courville, A. C.; and Bellemare, M. 2021. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34: 29304–29320.

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.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. In *International conference on machine learning*, 449–458. PMLR.

Bhatt, V.; Tjanaka, B.; Fontaine, M.; and Nikolaidis, S. 2022. Deep surrogate assisted generation of environments. *Advances in Neural Information Processing Systems*, 35: 37762–37777.

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

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

Dennis, M.; Jaques, N.; Vinitsky, E.; Bayen, A.; Russell, S.; Critch, A.; and Levine, S. 2020. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33: 13049–13061.

Duan, Y.; Schulman, J.; Chen, X.; Bartlett, P. L.; Sutskever, I.; and Abbeel, P. 2016. RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. arXiv:1611.02779.

Fakoor, R.; Chaudhari, P.; Soatto, S.; and Smola, A. J. 2019. Meta-q-learning. *arXiv preprint arXiv:1910.00125*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

Fontaine, M.; and Nikolaidis, S. 2020. A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy. *arXiv preprint arXiv:2012.04283*.

Fontaine, M. C.; Hsu, Y.-C.; Zhang, Y.; Tjanaka, B.; and Nikolaidis, S. 2021a. On the importance of environments in human-robot coordination. *arXiv preprint arXiv:2106.10853*.

Fontaine, M. C.; Liu, R.; Khalifa, A.; Modi, J.; Togelius, J.; Hoover, A. K.; and Nikolaidis, S. 2021b. Illuminating Mario Scenes in the Latent Space of a Generative Adversarial Network. arXiv:2007.05674.

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.

He, X.; Yang, H.; Hu, Z.; and Lv, C. 2022. Robust lane change decision making for autonomous vehicles: An observation adversarial reinforcement learning approach. *IEEE Transactions on Intelligent Vehicles*, 8(1): 184–193.

Houthooft, R.; Chen, Y.; Isola, P.; Stadie, B.; Wolski, F.; Jonathan Ho, O.; and Abbeel, P. 2018. Evolved policy gradients. *Advances in Neural Information Processing Systems*, 31.

Jakobi, N. 1997. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive behavior*, 6(2): 325–368.

Jiang, M.; Dennis, M.; Parker-Holder, J.; Foerster, J.; Grefenstette, E.; and Rocktäschel, T. 2021. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34: 1884–1897.

Jiang, M.; Grefenstette, E.; and Rocktäschel, T. 2021. Prioritized level replay. In *International Conference on Machine Learning*, 4940–4950. PMLR.

Kaiser, L.; Babaeizadeh, M.; Milos, P.; Osinski, B.; Campbell, R. H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. 2019. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*.

Li, D.; Li, W.; and Varakantham, P. 2023. Diversity Induced Environment Design via Self-Play. *arXiv preprint arXiv:2302.02119*.

Li, W.; Varakantham, P.; and Li, D. 2023. Effective Diversity in Unsupervised Environment Design. *arXiv preprint arXiv:2301.08025*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Moerland, T. M.; Broekens, J.; Plaat, A.; Jonker, C. M.; et al. 2023. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1): 1–118.

Morimoto, J.; and Doya, K. 2005. Robust reinforcement learning. *Neural computation*, 17(2): 335–359.

Mouret, J.-B.; and Clune, J. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

Parker-Holder, J.; Jiang, M.; Dennis, M.; Samvelyan, M.; Foerster, J.; Grefenstette, E.; and Rocktäschel, T. 2022. Evolving Curricula with Regret-Based Environment Design. *arXiv preprint arXiv:2203.01302*.

Pinto, L.; Davidson, J.; and Gupta, A. 2017. Supervision via competition: Robot adversaries for learning tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1601–1608. IEEE.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, 2817–2826. PMLR.

Pugh, J. K.; Soros, L. B.; and Stanley, K. O. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3: 40.

Rakelly, K.; Zhou, A.; Finn, C.; Levine, S.; and Quillen, D. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, 5331–5340. PMLR.

Rothfuss, J.; Lee, D.; Clavera, I.; Asfour, T.; and Abbeel, P. 2018. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*.

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

Sung, F.; Zhang, L.; Xiang, T.; Hospedales, T.; and Yang, Y. 2017. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv preprint arXiv:1706.09529*.

Tio, S.; and Varakantham, P. 2023. Transferable Curricula through Difficulty Conditioned Generators. *arXiv preprint arXiv:2306.13028*.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 23–30. IEEE.

Wang, J. X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J. Z.; Munos, R.; Blundell, C.; Kumaran, D.; and Botvinick, M. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.