# Marginal Benefit Driven RL Teacher for Unsupervised Environment Design

**Dexun Li, Wenjun Li, Pradeep Varakantham**

Singapore Management University
{dexunli.2019, wjli.2020, pradeepv}@smu.edu.sg

## Abstract

Training generally capable agents in complex environments is a challenging task that involves identifying the "right" environments at the training stage. Recent research has highlighted the potential of the Unsupervised Environment Design framework, which generates environment instances/levels adaptively at the frontier of the agent's capabilities using regret measures. While regret approaches have shown promise in generating feasible environments, they can produce difficult environments that are challenging for an RL agent to learn from. This is because regret represents the best-case (upper bound) learning potential and not the actual learning potential of an environment. To address this, we propose an alternative mechanism that employs marginal benefit, focusing on the improvement (in terms of generalized performance) the agent policy gets for a given environment. The advantage of this new mechanism is that it is agent-focused (and not environment focused) and generates the "right" environments depending on the agent's policy. Additionally, to improve the generalizability of the agent, we introduce a representative state diversity metric that aims to generate varied experiences for the agent. Finally, we provide detailed experimental results and ablation analysis to showcase the effectiveness of our methods. We obtain SOTA results among RL-based environment generation methods.

## 1 Introduction

The advancements in Reinforcement Learning (RL) have led to significant successes in various applications, such as game playing (Mnih et al. 2015; Silver et al. 2016), robot control (Levine et al. 2016; Akkaya et al. 2019), and many others. However, training RL agents with general capabilities remains a major challenge due to the millions of experiences required to train an RL agent in each environment, which is both time-consuming and expensive.

One promising approach to address this problem is to "shallowly" train an agent on a sequence of tasks or environments (Dennis et al. 2020; Parker-Holder et al. 2022; Li, Varakantham, and Li 2023). In this method, instead of extensively training on each environment with millions of experiences, the RL agent is exposed to a limited number of experiences in each individual environment. This adaptive curriculum of environments, tailored to the agent's policy, has been demonstrated to produce more robust agents in fewer training steps (Portelas et al. 2020a; Jiang, Grefenstette, and Rocktäschel 2021). This methodology is referred to as Unsupervised Environment Design (UED).

Protagonist Antagonist Induced Regret Environment Design (PAIRED) (Dennis et al. 2020) introduced a self-supervised RL paradigm in which the RL teacher employs regret, obtained from the student agent's performance to generate new environments. This approach leverages regret as a measure of learning potential to create environments that are at the edge of the capabilities of the student. However, computing the regret value, which is approximated by the difference between the expected payoffs of the student (protagonist) and the expert (antagonist), is computationally expensive owing to costly interactions between the three agents and the environment. Furthermore, PAIRED suffers from catastrophic forgetting of past environments due to learning on new environments. To address these concerns, subsequent works such as PLR (Jiang, Grefenstette, and Rocktäschel 2021) and ACCEL (Parker-Holder et al. 2022) introduced multiple changes: (1) They eliminated RL-based generation and instead employed a random generation of environments; (2) They used an approximate version of regret, namely Generalized Advantage Estimate (GAE); (3) They employed the replay of past environments to prevent catastrophic forgetting; and (4) They Edited the generated environments to ensure thorough training on a challenging sequence of environments.

While there has been significant progress in UED on account of above-mentioned algorithms, there exist some significant issues. The random environment generation-based training has been shown to provide good zero-shot generalization performance (in ACCEL and PLR), however, there exists a significant variance in performance across different runs. As indicated earlier, regret represents the upper bound on the learning potential of an environment and not the real learning potential. Furthermore, it is unclear if that potential can really be achieved. As shown in Figure 2, there is a positive correlation between what the agent can learn from the environment and the learning potential of the environment, but the significant variance suggests that high potential does not necessarily lead to substantial performance gains.

Therefore, we propose to employ RL-based training but with an alternative measure for environment generation called marginal benefit. Marginal benefit quantifies the actual im-

provement (on validation environments) obtained by the student agent due to "training" on an environment.

**Contributions:** Towards operationalizing the idea of utilizing marginal benefit-based measure, we make the following key contributions:

- We define the marginal benefit measure and use it in a teacher agent (RL-based or random) to generate environments. More importantly, we propose an RL-based teacher that is more scalable compared to the original RL generator UED algorithm, PAIRED, yet retains its beneficial features.
- To facilitate general capability, we define "representative state diversity" in environments and propose a mechanism to intentionally train in "diverse" environments with high marginal benefit.
- We demonstrate the effectiveness of our proposed methods *MBeED* and *MBeDED* through extensive experiments on a wide range of benchmark problems from the literature.

# 2 Background

In this section, we provide a brief background on UED and discuss relevant methods for UED. We offer a more detailed discussion of related work in Appendix A.

## 2.1 Unsupervised Environment Design, UED

In UED, we train a student agent to perform well across a set of in-distribution and out-of-distribution environments. To accomplish this, UED utilizes a teacher agent that provides a sequence of environment parameter values and generates the corresponding environment to train the student to generalize well to unseen environments/levels. We use environment and level interchangeably in this work. The UED problem is formally described as an Underspecified Partially Observable Markov Decision Process (UPOMDP):

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

$S$, $A$ and $\Omega$ are the set of states, actions, and observations, respectively. $R : S \to \mathbb{R}$ is the reward function, and $\gamma$ is the discount factor. The crucial element is $\Theta$, which denotes the set of all possible environment configurations. A particular parameter configuration, $\theta \in \Theta$ (can be a vector or sequence of values) defines a level and can impact the reward model, transition dynamics, and the observation function, i.e. $R : S \times \Theta \to \mathbb{R}$, $T : S \times A \times \Theta \to S$ and $O : S \times \Theta \to \Omega$. The UPOMDP is underspecified because training with all values of $\theta$ ($\in \Theta$) is infeasible, as $\Theta$ can be infinitely large. The goal of the student policy $\pi$ in a UPOMDP is to maximize its discounted expected rewards for any given $\theta \in \Theta$. In the model-free setting, where transition and observation functions are not known *a priori*, this objective is formulated as: $\max_{\pi} V^{\theta}(\pi) = \max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{H} r_t^{\theta} \cdot \gamma^t\right]$. where $r_t^{\theta}$ is the reward obtained by the student policy $\pi$ in a level with environment parameter $\theta$ at time step $t$. Consequently, the student needs to be trained on a series of $\theta$ values that maximize its generalization capability across all possible levels from $\Theta$. To achieve this objective, the teacher agent is employed. The goal of the teacher is to generate a distribution over the next set of environment parameter values to train the student, i.e., $\Lambda : \Pi \to \Delta(\Theta)$ to achieve good generalization performance, where $\Pi$ is the set of possible policies of the teacher.

## 2.2 Existing Methods

Dennis et al. (2020) first formalized the UED and introduced the Protagonist Antagonist Induced Regret Environment Design (PAIRED) algorithm, which is a three-agent game: the protagonist $\pi^P$ (student), the antagonist $\pi^A$ (expert) and the environment generator $\mathcal{G}$ (teacher). The environment generator $\mathcal{G}$ learns to control the distribution of environmental parameters $\theta$ by maximizing regret, which is approximated by the difference between the cumulative reward obtained by the protagonist and the antagonist under the same environment with parameters $\theta$:

$$\text{REGRET}^{\theta}(\pi^P, \pi^A) = V^{\theta}(\pi^A) - V^{\theta}(\pi^P) \tag{1}$$

Both the protagonist and antagonist are trained to maximize their own cumulative reward in the current environment $\theta$. Note that the environment generator (teacher) is discouraged from generating levels that can not be solved because they will have a maximum regret of 0. This teacher-student-expert framework co-evolves the policies, creating an adaptive curriculum learning approach where the teacher constantly creates an emergent class of levels that get progressively more difficult along the borderline of the student's ability, allowing agents to learn a good policy that enables zero-shot transfer. However, this framework struggles because of teacher efficiency, as it uses a RL generator and requires expensive interactions with the environment to collect millions of samples to train Protagonist and Antagonist agents separately.

As an alternative regret-based UED approach, Jiang, Grefenstette, and Rocktäschel (2021) proposed Prioritized Level Replay (PLR), where a student policy is challenged by two co-evolving teachers, Generator and Curator. In PLR, Generator randomly creates new environments, while the Curator prioritizes the replay probability for each environment based on the estimated learning potential. By adapting the sampling of the previously encountered levels to train, PLR is an active learning strategy that improves sample efficiency and generalization. In addition, PLR uses Generalized Advantage Estimation (GAE) (Schulman et al. 2015) to approximate the regret compared to the more expensive regret definition used in PAIRED. Specifically, the regret $F_{gae}(\theta)$ of the environment $\theta$ is defined as:

$$F_{gae}(\theta) = \frac{1}{T} \sum_{t=0}^{T} \max\{\sum_{k=t}^{T} (\gamma\lambda)^{k-t} \delta_k, 0\} \tag{2}$$

where $\lambda$ and $\gamma$ are the exponential weight discount and MDP discount factor respectively. $\delta$ is the TD-error at timestep $t$. The agent trained by PLR shows good generalization ability in terms of empirical results. However, PLR is still limited as it is unable to exploit any previously discovered level structure and can only curate randomly sampled levels. Moreover, the random search will be heavily affected by the high-dimensional design space, making it highly unlikely to sample levels at the frontier of the agent's current capabilities.

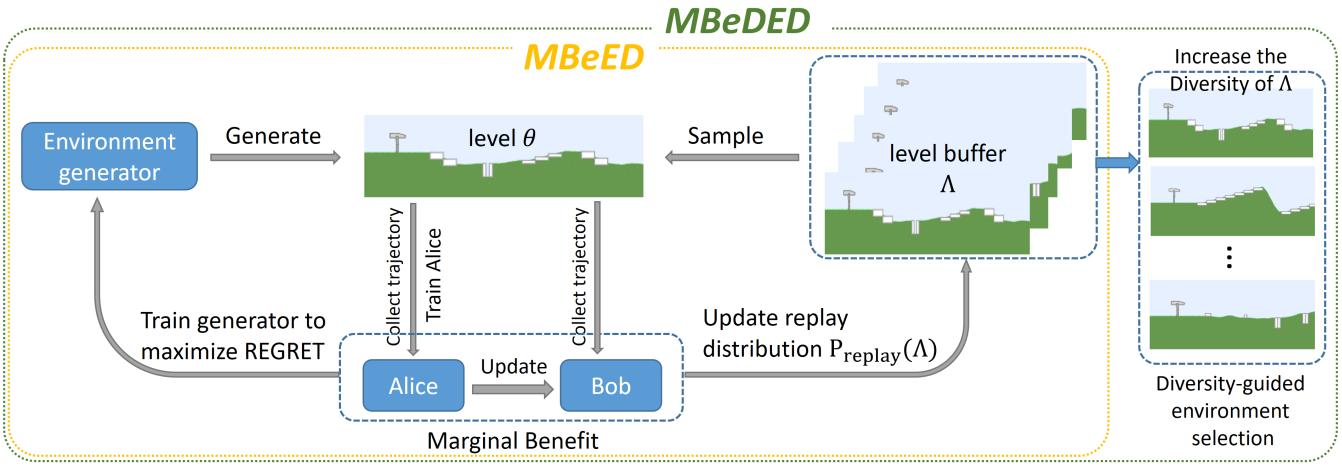


Figure 1: Overall framework of our approaches, *MBeED* and *MBeDED*.

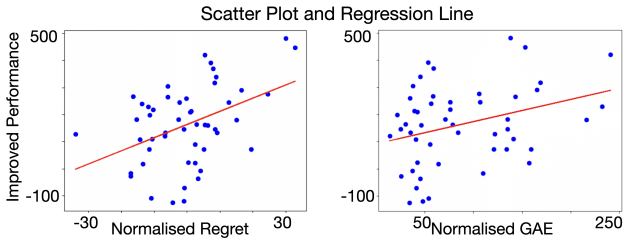


Figure 2: Correlation between performance changes after training for 10k timesteps in various environments of LunarLander for two metrics: 1. Regret defined as the performance gap between expert and student agents (Left, Equation 1), 2. Regret approximated using GAE values (Right, Equation 2).

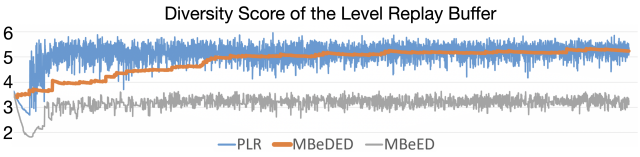


Figure 3: Comparison of normalized diversity scores for level replay buffer. Higher scores indicate higher diversity. *MBeDED* extends *MBeED* by intentionally training agents in "diverse" environments with high marginal benefits to facilitate general capability.

# 3 Approach

In this section, we describe the technical details of our proposed UED methods, Marginal Benefit driven Environment Design (*MBeED*) and its extension, Marginal Benefit and Diversity driven Environment Design (*MBeDED*). Our algorithms, *MBeED* and *MBeDED*, rely on the teacher-student framework, which consists of an environment generator (teacher) and two student agents that help compute the marginal benefit of a generated environment. Algorithm 1 provides the pseudocode of these algorithms, and Figure 1 illustrates the overall framework. Typically:

- *MBeED* incorporates the Marginal benefit-driven environment generator;
- *MBeDED* further extends *MBeED* to include representative state diversity-guided environment selection.

## 3.1 Marginal Benefit Based RL Environment Generator: *MBeED*

*MBeED* incentivizes the teacher to generate marginal benefit maximizing environments. To compute marginal benefit, two versions of the student policy, Alice and Bob, are maintained. While Alice represents the student policy that is current (after training on the generated environment), Bob represents the **base** policy (slightly outdated policy without training on the generated environment). We define the marginal Benefit ($\mu$) of a generated environment ($\theta$) as the difference in the expected values obtained by Alice and Bob's policies in that environment:

$$\mu^{\theta}(\pi^{A}, \pi^{B}) = V^{\theta}(\pi^{A}) - V^{\theta}(\pi^{B}) \tag{3}$$

For the marginal benefit to be positive, the teacher needs to generate environments that improve the student policy, providing an incentive to continuously generate environments that boost student performance. Alice and Bob each collect their trajectories, $\tau^{A}$ and $\tau^{B}$, in the current environment $\theta$ (Line 7 and 15 in Algorithm 1). The marginal benefit of training in the environment $\theta$ is then computed as the difference between the cumulative rewards they received (Line 9). Note that since Bob is "outdated" Alice, we can use the same set of trajectories $\tau$ to evaluate Bob and train Alice, and collect trajectories from updated Alice to compute the marginal benefit value, $\mu^{\theta}(\pi^{A}, \pi^{B})$.

There are alternative methods to compute the marginal benefit, such as using the difference in the scoring function based on the average magnitude of the GAE, we provide discussions in Appendix B.1. In this work, we opted to compute the marginal benefit as the difference between Bob and Alice's expected cumulative returns because of its simplicity and promising results in creating challenging yet solvable environments. Similar to PAIRED, if the teacher generates

Algorithm 1: *MBeED* and *MBeDED*

1: **Input:** Level buffer $\Lambda$, replay probability $p$.
2: **Initialize**: policy Alice $\pi^A$, Bob $\pi^B$, level generator (teacher) $\mathcal{G}$ and replay distribution $P_{replay}$;
3: **while** Not converged **do**
4: Sample a replay decision, $\epsilon \sim U[0,1]$
5: **if** $\epsilon \geq p$ **then**
6: Generate $\theta$ from $\mathcal{G}$, and create POMDP $\mathcal{M}_\theta$
7: Collect Alice's and Bob's trajectories $\tau^A$ and $\tau^B$ in $\mathcal{M}_\theta$, and compute $V^\theta(\pi^B) = \sum_{t=0}^{T} \gamma^t r_t^\theta$
8: Train $\pi^A$ to maximize $V^\theta(\pi^A)$
9: compute $\mu(\pi^A, \pi^B)$ using Eq. 3
10: Train $\mathcal{G}$ with RL Update and reward $\mu(\pi^A, \pi^B)$
11: Determine $S_\theta$ {For *MBeDED*}
12: If $F_{div}(,)$ of $\theta$ is higher than the lowest one in the $\Lambda$, replace that one by $\theta$ to $\Lambda$ and update $P_{replay}$ according to Eq. 21
13: **else**
14: Sample $\theta$ from $\Lambda$ according to $P_{replay}$, and create POMDP $\mathcal{M}_\theta$
15: Collect Alice's and Bob's trajectories $\tau^A$ and $\tau^B$ in $\mathcal{M}_\theta$, and compute $V^\theta(\pi^B) = \sum_{t=0}^{T} \gamma^t r_t$
16: Train $\pi^A$ to maximize $V^\theta(\pi^A)$
17: compute $\mu(\pi^A, \pi^B)$ using Eq. 3
18: Update $S_\theta$ {For *MBeDED*}
19: Update $P_{replay}$ according to Eq. 21
20: **end if**
21: Update $\pi^B$ by letting $\pi^B = \pi^A$
22: **end while**

unsolvable environments, Alice and Bob would perform the same (marginal benefit is zero) and the teacher would get a reward of zero, but if the teacher generates environments the student agent will get a high improvement after training on the environments, the teacher achieves a positive reward. By focusing on improvement in student policy for a generated environment, we adopt a student-centric perspective, which allows for generating environments at a suitable pace and provides a more accurate representation of the student's actual improvement during training.

In order to generate a challenging environment at a suitable pace, the teacher $\mathcal{G}$ is trained to maximize the marginal benefit $\mu$(Line 10). Alice is trained to maximize its cumulative reward (Line 8 and Line 16). A crucial aspect of this approach is the direct policy copying from Alice to Bob (Line 21), which avoids the need for costly interactions with the environment to train an optimal antagonist's policy at the current level, as required in PAIRED (Dennis et al. 2020). The self-regulating feedback loop between Alice and Bob enables the environment generator to establish an adaptive curriculum, where new levels that could significantly improve Alice and Bob's behavior are constantly generated. Additionally, we provide a discussion in Appendix B.1, demonstrating that our base model $\pi^B$ can be approximated to learn a minimax policy similar to PAIRED. Furthermore, the definition of our marginal benefit $\mu^\theta(\pi^A, \pi^B)$ in Eq. 3 offers greater scalability compared to the regret definition in Eq. 1 for PAIRED, as it bypasses the need for resource-intensive training of an expert agent in each environment. Such agent-focused measure not only generates the "right" environments depending on the agent's policy but also simplify the training process.

### 3.2 Diversity Guided Environment Selection: *MBeDED*

The objective of this section is to incorporate representative state diversity into the UED framework to enhance exploration and improve the generalizability of the student agent. Existing algorithms (Jiang et al. 2021; Jiang, Grefenstette, and Rocktäschel 2021; Parker-Holder et al. 2022) that utilize random generation also consider a setup whereby a level buffer $\Lambda$ is introduced to store the top $K$ visited levels with the highest learning potential. The key intuition behind introducing diversity is that while the marginal benefit-induced RL generation is a good criterion for generating an environment, it may not contribute to generalization if the level replay buffer $\Lambda$ contains many similar environments (in terms of key states that are visited). In Figure 3, we show the normalized diversity score calculated by our representative state diversity measurement. Random-based generation methods such as PLR encounter fewer similarities due to their inherent randomness. However, RL-based generation is more likely to consistently produce *similar* or *redundant* environments over time, as the *MBeED* records the lowest diversity. Thus, determining the level buffer by the learning potential alone may result in preserving similar or repeated levels, resulting in low exploration of the environment space, and the agent will not learn much from training on these similar environments.

Therefore, we propose to selectively add diverse environments to the level buffer, so as to ensure that the student agent gets exposure to a wider variety of environments (as shown in Figure 3, *MBeDED* can achieve the same diversity level compared to random generation approaches). In this section, we first provide a measure for representative state diversity based on the representative observed state vector when the agent executes its policy in an environment. We then, describe how we ensure the level replay buffer remains diverse and finally we explain our overall algorithm.

**Representative Observed State Vector** We first define the representative observed state vector $S_\theta$ of environment $\theta$ given a student policy. We also provide other forms of $S_\theta$ in Appendix B.2.

**Definition 1** ***(Representative observed state vector)*** *For an environment $\theta$, given multiple trajectories collected by the current student policy, we have the set of all visited states in environment $\theta$, $\mathbb{S}_\theta = \{s_1, s_2, ...s_n\}$. We define the representative observed state vector, $S_\theta$ as*

$$S_\theta = \{\tilde{s}_1, \tilde{s}_2, ....\tilde{s}_m\},\ \textit{where}\ m \ll n$$

*and $S_\theta \subset \mathbb{S}_\theta$ represents the set of representative states visited in the trajectories.*

$S_\theta$ consists of two types of states:

- **important** states: we rank the observed states according to their TD-error, $\delta = r_t + \gamma V(s_{t+1}) - V(s_t)$. States with high TD errors are considered important as they have a significant impact on training. We select the top $m_1$ states with the highest TD error and include them in $S_\theta$.

- **representative** states: we also aim to ensure that $S_\theta$ effectively represents the environment $\theta$. This means that for every observed state in $\mathbb{S}$, there should be a similar state in $S_\theta$. We add the remaining $m_2 = m - m_1$ states to ensure that $S_\theta$ provides a comprehensive representation of the level $\theta$.

Formally, the representative score of $S_\theta$ is defined as:

$$F_{rep}(S_\theta) = \sum_{s_i \in \mathbb{S}} \max_{s_j \in S_\theta} \{k(s_i, s_j)\} \tag{4}$$

$k(.,.)$ represents the similarity kernel between states. One choice for the similarity kernel is the cosine similarity, defined as: $k(s_1, s_2) = \frac{s_1 s_2^\top}{\|s_1\|\|s_2\|}$, where $\|s\|$ is the norm of observed state $s$.

While finding the important states is simple, determining the $S_\theta$ that maximizes $F_{rep}(.)$ is NP-hard, and it is computationally expensive when $\mathbb{S}$ is typically very large. Motivated by (Fang et al. 2019), we propose a heuristic way. We first randomly sample a set $S' \subset \mathbb{S}$ of size $n'$, where $m < n' < n$. Then we initialize $S_\theta$ as a set including those top $m_1$ important states, and use a greedy algorithm to pick the top $m_2$ observed states from $S'$ to get the final $S_\theta$. Specifically, at each step, we will add the observed state $s$ that maximizes the marginal gain to $S_\theta$, where the marginal gain $F_{rep}(\{s\}|S_\theta)$ is defined as the difference in $F_{rep}(\cdot)$ when adding the observed state $\{s\}$ to $S_\theta$:

$$F_{rep}(\{s\}|S_\theta) = F_{rep}(\{s\} \cup S_\theta) - F_{rep}(S_\theta) \tag{5}$$

Because $F_{rep}(S_\theta)$ is a submodular function, the greedy algorithm can provide a solution, $S^*_\theta$ with an approximation factor of $1 - \frac{1}{e}$ (Nemhauser, Wolsey, and Fisher 1978).

**Diverse Level Replay Buffer** We now describe how to utilize the representative observed state vector $S_\theta$ to maintain a diverse level replay buffer $\Lambda$. A diverse level buffer $\Lambda$ is more informative and will contribute to the effective exploration and improve the generalizability of the agent. The formal definition to measure the state-aware diversity among the level buffer $\Lambda$ for a given student policy is as follows.

**Definition 2** *(**Representative state diversity score among** $\Lambda$) Consider the level replay buffer $\Lambda = \{\theta_1, \ldots, \theta_K\}$, each level $\theta_i \in \Lambda$ has its corresponding representative observed state vector, $S_{\theta_i}$. We can compute the state-aware diversity score $F_{div}(\Lambda, \Lambda)$ of level replay buffer $\Lambda$ as follows:*

$$\begin{aligned} &F_{div}(\Lambda, \Lambda) = \sum_{\theta_i \in \Lambda} F_{div}(\theta_i, \Lambda \setminus \{\theta_i\}) \text{ and} \\ &F_{div}(\theta_i, \Lambda \setminus \{\theta_i\}) = -\sum_{s_i \in S_{\theta_i}} \max_{s_j \in \{S_{\theta_j}\}_{i \neq j}} \{k(s_i, s_j)\} \end{aligned} \tag{6}$$

*$F_{div}(\theta_i, \Lambda \setminus \{\theta_i\})$ measures the representative state diversity score between the level $\theta_i$ and a set of levels $\Lambda \setminus \{\theta_i\}$.*

Unlike computing the representative score in Eq. 4, we introduce a negative sign before the kernel function to assess the representative state diversity between two states. A high similarity between two representative observed state vectors indicates a low diversity score between those two levels, and vice versa. When generating a new environment $\theta_{new}$, if that new level is added to the level replay buffer $\Lambda$, we want to increase the diversity among $\Lambda$, which means the representative state diversity score among $\Lambda$, i.e., $F_{div}(\Lambda, \Lambda)$, should increase. We now provide a heuristic way to determine whether a newly generated level $\theta_{new}$ should be added to the level replay buffer $\Lambda$:

[1] Consider the level buffer $\Lambda = \{\theta_1, \ldots, \theta_K\}$ and the newly generated level $\theta_{new}$, each of which has its corresponding observed state representative vector $S_\theta$. For any level $\theta \in \{\theta_{new}\} \cup \Lambda$, we can compute its state-aware diversity score $F_{div}(,)$ with other levels $\theta' \in \{\theta_{new}\} \cup \Lambda \setminus \{\theta\}$ as follows:

$$F_{div}(\theta, \{\theta_{new}\} \cup \Lambda \setminus \{\theta\}) = -\sum_{o_i \in S_\theta} \max_{o_j \in S'_\theta} \{k(o_i, o_j)\}, \tag{7}$$

[2] If $F_{div}(\theta_{new}, \Lambda) > \min_{\theta_i \in \Lambda} \{F_{div}(\theta_i, \{\theta_{new}\} \cup \Lambda \setminus \{\theta_i\})\}$, add $\theta_{new}$ to $\Lambda$ and remove $\theta_i$ that has the lowest $F_{div}(\theta_i, \{\theta_{new}\} \cup \Lambda \setminus \{\theta_i\})$ value, as a higher diversity score of $\theta_{new}$ indicates a lower likelihood of finding a similar observed state within the levels in $\Lambda$.

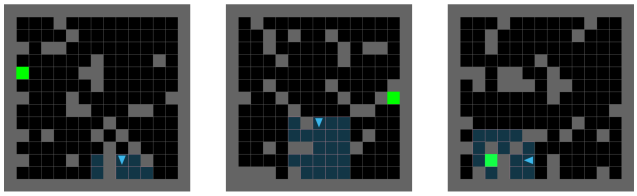

Figure 4: Example levels generated by *MBeDED* (placing up to 50 blocks) during training.

### 3.3 Overall Algorithm

At the beginning of each iteration, *MBeDED* either generates new levels (with probability $p$, line 5 and 6) or samples a mini-batch of levels in the level buffer to train the student (line 13 and 14 of Algorithm 1). When sampling levels from buffer $\Lambda$, we assign each level $\theta_i$ a probability based on the combination of its diversity score (indicating the generalization through training on diverse environments) and most importantly its marginal benefit (representing the learning potential from an environment for the agent).

First, we provide the probability computation corresponding to marginal benefit. Given the marginal benefit $\mu^{\theta_i}$ as the proxy for its learning potential for each level, we rank them accordingly and use a prioritization function $h$ to decide how differences in learning potential are translated into differences in prioritization. As a result, we obtain a learning potential prioritized distribution $P_{mb}(\Lambda)$ over the level buffer, and the probability for $\theta_i$ is

$$P_{mb}(\theta_i|\Lambda, \mu^{\theta_i}) = \frac{h\left(\text{rank}(\mu^{\theta_i})\right)^{1/\beta}}{\sum_j h\left(\text{rank}(\mu^{\theta_i})\right)^{1/\beta}} \tag{8}$$

where $\beta$ is the temperature parameter that tunes how much $h(\text{rank}(\mu^{\theta_i}))$ determines the resulted probability, and $h(\text{rank}(\mu^{\theta_i}))$ is the rank of level $\theta_i$ sorted in descending order

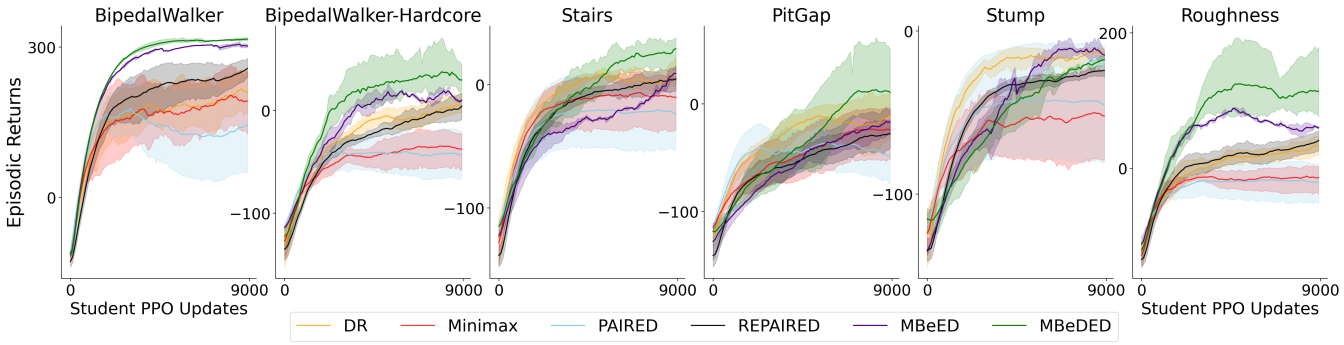


Figure 5: Transfer performance on test environments during training (mean and standard error).

among the level buffer. Similar to (Jiang, Grefenstette, and Rocktäschel 2021), we employ $h\left(\text{rank}(\mu^{\theta_j})\right) = \frac{1}{\text{rank}(\mu^{\theta_i})}$. Note that different surrogates for learning potential can be used, such as the regret (Equation 1) and GAE value (Equation 2). Our experiments in the Lunar Lander environment indicate that marginal benefit is a better measure of learning potential for improving the agent's generalization capabilities, and a detailed comparison is provided in Appendix D.2.

Similarly, we can calculate the diversity scores with Equation 14 and then translate the diversity scores into a prioritized distribution $P_{div}(\Lambda)$ over the level buffer. We update the overall replay probability distribution $P_{replay}(\Lambda)$ over $\Lambda$ by combining $P_{mb}(\Lambda)$ (based on marginal benefit) and $P_{div}(\Lambda)$ (based on diversity score) as (Line 19):

$$P_{replay}(\Lambda) = (1-\rho)\cdot P_{mb}(\Lambda) + \rho \cdot P_{div}(\Lambda) \qquad (9)$$

The hyperparameter $\rho \in [0, 1]$ balances the trade-off between marginal benefit and diversity. Details regarding the replay process are explained in Appendix B.4 in the appendix.

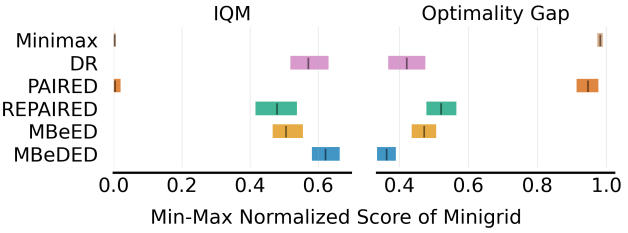


Figure 6: Aggregate zero-shot OOD test performance in Minigrid Domain across 5 independent runs. Specifically, IQM focuses on the middle 50% of combined runs, discarding the bottom and top 25%, thereby providing a robust measure of overall performance. The optimality gap captures the amount by which the algorithm fails to meet a desirable target (e.g., 95% solved rate), beyond which further improvements are considered unimportant. Higher IQM scores and lower optimality gap scores indicate better performance.

## 4 Experimental Results

In this section, we present our experimental results in the domains of *BipedalWalker*, *Minigrid*, and *CarRacing* to demonstrate the superior performance of our approach when a trained agent is transferred to new environments. We compare our approach against existing UED methods: Domain Randomization (DR) (Tobin et al. 2017), Minimax (Wang et al. 2019), PAIRED (Dennis et al. 2020), REPAIRED (Jiang et al. 2021). We have focused on comparing our approach to other RL-controlled environment generation methods while ignoring random-based approaches such as PLR (Jiang, Grefenstette, and Rocktäschel 2021), ACCEL (Parker-Holder et al. 2022) as they have a huge variance in performances and also require human-defined edits and a predetermined starting point for environments. However, we wish to point out that our contributions are complementary and can be utilized along with ACCEL and other random-based environment generation approaches. We show the average and variance of the performance for our method, baselines with five random seeds. Table 1 in the appendix provides a summary of the key characteristics of all approaches.

**Performance on BipedalWalker:** We first evaluate our approach on *BipedalWalker* environment. This environment entails continuous control with dense rewards. Similar to (Wang et al. 2019),we use a modified version of *BipedalWalker-Hardcore* from OpenAI Gym (Brockman et al. 2016). In *BipedalWalker*, there are 8 parameters that indirectly represent the intensity of four kinds of terrain-based obstacles for a two-legged robot: the minimum/maximum roughness of the ground, the minimum/maximum height of stump obstacles, the minimum/maximum width of pit gap obstacles, and the minimum/maximum size of ascending and descending flights of stairs. We provide an illustration of these four kinds of obstacles in Figure 8 in the Appendix.

The *BipedalWalker* environment provides the student with a 24-dimensional proprioceptive state with respect to its lidar sensors, angles, and contacts. The action space is continuous and consists of four values that control the torques of the agent's four motors. The teacher learns to control eight parameters that correspond to the range of four kinds of obstacles and then combines a random seed to generate a specific level. All agents are trained using Proximal Policy Optimization (PPO) (Schulman et al. 2017). For a fair comparison, during training, we presented a vanilla *BipedalWalker*, a challenging *BipedalWalker-Hardcore* environment, and four specific levels in the context of isolated challenges in {Roughness,

Stump height, Pit gap, Stair step} to evaluate our algorithm and baselines.

Figure 5 shows the transfer performance throughout training. *MBeED* and *MBeDED* consistently outperform other baselines across most of test environments, while also achieving faster convergence. Additionally, our approaches demonstrate greater scalability compared to the RL-based generator algorithms, PAIRED and REPAIRED, as detailed in Table 5 in Appendix D.6. These results provide strong evidence in support of the key principles driving *MBeDED*'s design: a student-centric generator that progressively produces environments to facilitate the improvement of the student policy, and the maintenance of diverse environments to enhance effective exploration and improve generalizability. We further summarize the empirical results of all approaches in Figure 10 in Appendix D.1.

**Performance on Minigrid:** We investigate the maze navigation environment, which is based on *Minigrid* (Chevalier-Boisvert, Willems, and Pal 2018). We train the environment generator to learn how to build maze environments by choosing the location of the obstacles, the goals, and the starting location of the agent. Specifically, at the beginning of each iteration, the generator will place the student agent and the goal, and then every time step afterward, the generator outputs a location where the next obstacle will be placed. There will be up to 50 blocks that can be placed. Several examples of generated mazes during training are illustrated in Figure 4.

The maze is partially observable, where the student agent's view is shown as a blue-shaded area in Figure 4. The student agent (blue triangle) must explore the maze to find a goal (green square). In order to deal with the partially observable setting, our agents use PPO with a Recurrent Neural Network structure. We compare our agents' transfer ability trained by different approaches on human-designed levels.

Due to space constraints, the test environments and corresponding performance results are provided in Figure 11 in Appendix D.1. It shows that the original RL-based approach, PAIRED, performs poorly, as the long time horizon to build the environment and the sparse reward are challenging for an RL-based generator, and minimax performs worse than PAIRED. While DR is a strong baseline in this domain, *MBeDED* can achieve a similar or higher mean return. We summarize the empirical results of all approaches in Figure 6. Despite both REPAIRED and DR can learn well on the Minigrid domain, MBeDED outperforms them on both IQM and optimality gap.

**Performance on Car Racing:** In the CarRacing environment, we design each track as a closed loop for the student agent to drive around, with the goal of completing a full lap. To enhance the expressiveness of the original CarRacing environment, we reparametrize the tracks using Bézier curves. Specifically, each track is constructed from a Bézier curve (Mortenson 1999) based on 12 control points randomly sampled within a fixed radius, $B/2$, of the center of the $B \times B$ playfield. The track is comprised of a sequence of $L$ polygons, with the student receiving a reward of $1000/L$ for driving over each previously unvisited polygon. In addition, the student receives a penalty of $-0.1$ at each time step. Following the methodology of (Jiang et al. 2021), we do not penalize

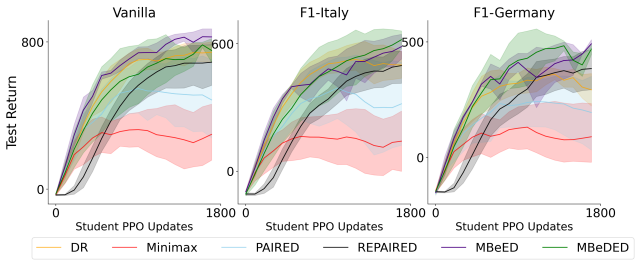


Figure 7: Zero-shot transfer performance on the OOD F1 tracks: Vanilla, Italy and Germany.

the agent for driving out of bounds but terminate the episode if it deviates too far off track. To reduce the complexity of the observation space, we provide the student with a $96 \times 96 \times 3$ pixel observation in RGB channels, clipped to an egocentric bird's-eye view of the vehicle centered horizontally in the top $84 \times 96$ portion of the frame. The remaining $12 \times 96$ portion of the frame displays the dashboard, which visualizes the agent's latest action and return.

We present per-track zero-shot transfer returns of policies trained by each method on some of the human-designed Formula One (F1) tracks throughout training in Figure 7. Note that these tracks are significantly different, out-of-distribution (OOD) from environments generated in training and they can not be generated within 12 control points. Remarkably, both *MBeED* and *MBeDED* are able to achieve better performance than other baselines in mean performance, providing further evidence of the benefits of the induced curriculum and diverse level buffer. We also summarize the empirical results of all approaches in Figure 12 in Appendix D.1. More experimental details can be found in Section C in the appendix.

**Ablation and additional Experiments:** We conduct detailed ablation studies to evaluate the impact of different design choices. Additionally, we use the Lunar Lander, which is a classic rocket trajectory optimization problem, to explore the relationship between transfer performance improvements and three different measures: 1. regret, 2. GAE, 3. marginal benefit. We show that marginal benefit is a more reliable measure to indicate the benefit accrued from training in terms of generalization capabilities in the current environment. All detailed results are included in Section D in the Appendix.

## 5 Conclusion

In this paper, we introduce *MBeED* and *MBeDED* for unsupervised environment design. Our approaches utilize a curriculum to automatically create a distribution of training environments by employing a novel marginal benefit-based measure. Moreover, to enhance effective exploration and improve generalizability, *MBeDED* selectively revisits previously generated environments by prioritizing those with higher estimated learning potential and representative state diversity. Finally, we conducted experiments in various benchmark environments and demonstrated that our RL-based generation algorithms achieve superior zero-shot transfer performances.

## 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

Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; Ribas, R.; et al. 2019. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*.

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

Chevalier-Boisvert, M.; Willems, L.; and Pal, S. 2018. Minimalistic gridworld environment for openai gym.

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.

Du, Y.; Abbeel, P.; and Grover, A. 2022. It Takes Four to Tango: Multiagent Selfplay for Automatic Curriculum Generation. *arXiv preprint arXiv:2202.10608*.

Fang, M.; Zhou, T.; Du, Y.; Han, L.; and Zhang, Z. 2019. Curriculum-guided hindsight experience replay. *Advances in neural information processing systems*, 32.

Florensa, C.; Held, D.; Wulfmeier, M.; Zhang, M.; and Abbeel, P. 2017. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, 482–495. PMLR.

Igl, M.; Farquhar, G.; Luketina, J.; Boehmer, W.; and Whiteson, S. 2020. The impact of non-stationarity on generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*.

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.

Klink, P.; Yang, H.; D'Eramo, C.; Peters, J.; and Pajarinen, J. 2022. Curriculum reinforcement learning via constrained optimal transport. In *International Conference on Machine Learning*, 11341–11358. PMLR.

Kostrikov, I. 2018. PyTorch Implementations of Reinforcement Learning Algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail.

Kostrikov, I.; Yarats, D.; and Fergus, R. 2020. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1): 1334–1373.

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.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Mortenson, M. E. 1999. *Mathematics for computer graphics applications*. Industrial Press Inc.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming*, 14(1): 265–294.

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*.

Portelas, R.; Colas, C.; Hofmann, K.; and Oudeyer, P.-Y. 2020a. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, 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*.

Raileanu, R.; Goldstein, M.; Yarats, D.; Kostrikov, I.; and Fergus, R. 2020. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

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

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Sukhbaatar, S.; Lin, Z.; Kostrikov, I.; Synnaeve, G.; Szlam, A.; and Fergus, R. 2017. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*.

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, K.; Kang, B.; Shao, J.; and Feng, J. 2020. Improving generalization in reinforcement learning with mixture regularization. *Advances in Neural Information Processing Systems*, 33: 7968–7978.

Wang, R.; Lehman, J.; Clune, J.; and Stanley, K. O. 2019. Paired open-ended trailblazer (poet): Endlessly generating

increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*.

Weinshall, D.; and Amir, D. 2020. Theory of curriculum learning, with convex loss functions. *Journal of Machine Learning Research*, 21(222): 1–19.

Whiteson, S. 2009. Generalized domains for empirical evaluations in reinforcement learning.

Wu, X.; Dyer, E.; and Neyshabur, B. 2020. When do curricula work? *arXiv preprint arXiv:2012.03107*.

Zhang, Y.; Abbeel, P.; and Pinto, L. 2020. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33: 7648–7659.