Replay-Guided Adversarial Environment Design

Anonymous Author(s)
Affiliation
Address
email

Abstract

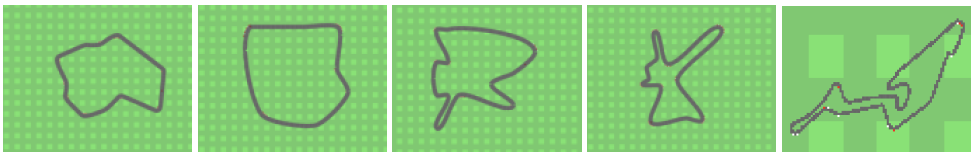
Deep reinforcement learning (RL) agents may successfully generalize to new settings if trained on an appropriately diverse set of environment and task configurations. Unsupervised Environment Design (UED) is a promising self-supervised RL paradigm, wherein the free parameters of an underspecified environment are automatically adapted during training based on an agent's policy, leading to the emergence of diverse training environments. Here, we cast Prioritized Level Replay (PLR), an empirically successful but theoretically unmotivated method which can be used to choose from a set of levels, as a type of UED. We argue that through curating completely random levels, PLR, too, can generate novel and complex levels for effective training. This insight reveals a natural class of UED methods we call Dual Curriculum Design (DCD). Crucially, DCD includes both PLR and a popular UED algorithm, PAIRED, as special cases and inherits similar theoretical guarantees. This connection allows us to develop novel theory for PLR, providing a version with a robustness guarantee. Furthermore, our theory suggests a highly counterintuitive improvement to PLR: by stopping the agent from updating its policy on uncurated levels (training on *less data*), we can improve the convergence to Nash equilibria. Indeed, our experiments confirm that our new method, $\text{PLR}^{\perp}$, obtains better results on a suite of zero-shot out-of-distribution generalization tasks, in addition to demonstrating that PLR improves the performance of PAIRED, from which it inherited its theoretical framework.

1 Introduction

While deep reinforcement learning (RL) approaches have led to many successful applications in challenging domains like Atari [18], Go [31], Chess [32], Dota [3] and StarCraft [36] in recent years, RL agents remain quite brittle, often failing to transfer to environments only slightly different from those encountered during training [39, 7]. To ensure learning of robust and well-generalizing policies, agents must train on sufficiently diverse and informative variations of environments (e.g. see Section 3.1 of [6]). However, it is not always feasible to specify an appropriate training distribution or a generator thereof. Thus, agents may benefit greatly from methods that automatically adapt the training distribution over environment variations throughout training [8, 15]. Throughout this paper we will call a particular environment instance or configuration (e.g. an arrangement of blocks, race tracks or any of the environment's entities) a *level*.

Two recent works [8, 15] have sought to empirically demonstrate this need for a more targeted agent-adaptive mechanism of selecting which levels to train RL agents on in order to ensure efficient learning and generalization to unseen levels, as well as to provide methods implementing such mechanisms. The first method—Protagonist Antagonist Induced Regret Environment Design (PAIRED) [8]—introduces a self-supervised RL paradigm called Unsupervised Environment Design (UED). Here, an environment generator (a *teacher*) is co-evolved together with a *student* policy that trains on levels *actively* proposed by the teacher, leading to a form of adaptive curriculum learning. The aim of this

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.



(a) **DR** (b) **PAIRED** (c) **REPAIRED** (d) **PLR$^{\perp}$** (e) **Human**

Figure 1: Randomly drawn samples of `CarRacing` tracks produced by different methods. (a) Domain Randomization (DR) produces tracks of average complexity, with few sharp turns. (b) PAIRED often overexploits the difference in the students, leading to simple tracks that incidentally favor the antagonist. (c) Tracks generated by REPAIRED combine both challenging and easy sections. (d) Robust PLR selects the most challenging randomly generated tracks, resulting in tracks that more closely resemble human-designed tracks, such as (e) the Nürburgring Grand Prix.

coevolution is for the teacher to gradually learn to generate environments that exemplify properties of those that might be encountered at deployment time, and for the student to simultaneously learn a good policy that enables zero-shot transfer to such environments. PAIRED's specific adversarial approach to UED ensures a useful robustness characterization of the final learned policy in the form of a minimax regret guarantee—assuming that its underlying teacher-student multi-agent system arrives at a Nash equilibrium [NE, 21]. In contrast, the second such method—Prioritized Level Replay (PLR) [15]—embodies an alternative form of dynamic curriculum learning that assumes not control of level generation, but the ability to selectively replay existing levels. PLR tracks levels previously proposed by a blackbox environment generator, and for each level, maintains meta-data from which it can estimate a *score* of the agent's *learning potential* in terms of how useful it would be to gather new experience from a particular level again in the future. The PLR algorithm exploits these scores to determine an adaptive schedule for revisiting or *replaying* levels to maximize learning potential. PLR has been shown to produce scalable and robust results, improving both sample complexity of agent training and the generalization of the learned policy in diverse environments. However, unlike PAIRED, PLR is motivated by heuristic arguments and as such lacks a useful theoretical characterization of its learning behavior.

In this paper, we argue that PLR is, in itself, an effective form of UED: Through curating even randomly generated levels, PLR can generate novel and complex levels for learning robust policies. This insight leads to a natural class of UED methods which we call *Dual Curriculum Design* (DCD). In DCD, a student policy is challenged by a team of two co-evolving teachers. One teacher *actively generates* new, challenging levels, while the other *passively curates* existing levels for replaying, by prioritizing those estimated to be most suitably challenging for the student. We show that PAIRED and PLR are distinct members of the DCD class of algorithms and prove in Section 3 that all DCD algorithms enjoy similar minimax regret guarantees to that of PAIRED.

We make use of this result to provide the first theoretical characterization of PLR, which immediately suggests a simple yet highly counterintuitive adjustment to PLR: By only training on trajectories in replay levels, PLR becomes provably robust. We call this resulting variant PLR$^{\perp}$ (Section 4). From this perspective, we see that, in a diametrically opposite manner to PAIRED, PLR effectively performs level design through prioritized selection rather than active generation.

A second corollary to the provable robustness of DCD algorithms (Section 5) shows that PLR can be extended to make use of the teacher from PAIRED as a level generator while preserving the robustness guarantee of PAIRED, resulting in a method called *Replay-Enhanced PAIRED* (REPAIRED). We hypothesize that PLR can play a complementary role to PAIRED in robustifying student policies.

In Section 6, we investigate the learning dynamics of both PLR$^{\perp}$, REPAIRED, and their close counterparts in the DCD class on a challenging maze domain, as well as a novel continuous control UED setting based on the popular `CarRacing` environment [4]. In both of these highly distinct settings, our new methods provide significant improvements over PLR and PAIRED, even producing agents that can perform out-of-distribution (OOD) generalization to a variety of human designed Formula One tracks.

In summary, we present the following contributions: (i) We establish a common framework, Dual Curriculum Design, that encompasses PLR and PAIRED. This allows us to develop new theory,

which provides the first robustness guarantees for PLR, as well as for REPAIRED, which augments PAIRED with a PLR-based replay mechanism. (ii) Crucially, this new theory suggests a *highly counterintuitive* improvement to PLR: the convergence to Nash Equilibrium should be assisted by training *on fewer data* when using PLR—namely by only taking policy-gradient updates from data that originates from the PLR buffer, and only using samples from the environment distribution to populate the buffer. (iii) Our experiments across a maze domain and a novel car-racing domain show that this indeed improves the performance of PLR. Both this new variant of PLR and REPAIRED reach new state-of-the-art performance in zero-shot OOD generalization in these challenging settings.

2 Background

2.1 Unsupervised Environment Design

Unsupervised Environment Design (UED), as introduced by [8], is the problem of automatically designing a distribution of environments that adapts to the learning agent. UED is defined in terms of an *Underspecified POMDP* (UPOMDP), given by $\mathcal{M} = \langle A, O, \Theta, S^{\mathcal{M}}, \mathcal{T}^{\mathcal{M}}, \mathcal{I}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, \gamma \rangle$, where A is a set of actions, O is a set of observations, S is a set of states, $\mathcal{T} : S \times A \to \mathbf{\Delta}(S)$ is a transition function, $\mathcal{I} : S \to O$ is an observation (or inspection) function, $\mathcal{R} : S \to \mathbb{R}$ is a reward function, and γ is a discount factor. This definition is identical to a POMDP with the addition of Θ to represent the free-parameters of the environment. These parameters can be distinct at every time step and incorporated into the transition function $\mathcal{T}^{\mathcal{M}} : S \times A \times \Theta \to \mathbf{\Delta}(S)$. For example, Θ could represent the possible positions of obstacles in a maze. We refer to the environment resulting from a fixed θ as $\mathcal{M}_\theta$. We define the value of π in $\mathcal{M}_\theta$ to be $V^\theta(\pi) = \mathbb{E}[\sum_{i=0}^{T} r_t \gamma^t]$ where r_t are the rewards achieved by π in $\mathcal{M}_\theta$. Aligning with terminology from [15], we refer to a particular fully-specified environment as a *level*.

2.2 Protagonist Antagonist Induced Regret Environment Design

Protagonist Antagonist Induced Regret Environment Design [PAIRED, 8] presents a UED approach consisting of simultaneously training agents in a three player game: the protagonist π_P and the antagonist π_A are trained in environments $\tilde{\boldsymbol{\theta}}$, generated by the teacher $\mathbf{\Theta}$. The objective of this game is defined by $U(\pi_P, \pi_A, \theta_A) = \mathbb{E}_{\theta \sim \tilde{\boldsymbol{\theta}}}[\textsc{Regret}^\theta(\pi_A, \pi_B)]$, where regret is defined by $\textsc{Regret}^\theta(\pi_A, \pi_B) = V^\theta(\pi_P) - V^\theta(\pi_A)$. The protagonist and antagonist are both trained to maximize their discounted environment returns while the teacher is trained to maximize U. As shorthand, we will sometimes refer to the protagonist and antagonist jointly as the *student agents*. The counterclockwise loop beginning at the student agents in Figure 2 summarizes this approach, with the student(s) being both the protagonist and antagonist.

As both student agents grow more adept at solving different levels, the teacher continues to adapt its level designs to exploit the weaknesses of the protagonist in relation to the antagonist. As this dynamic unfolds, PAIRED produces an emergent curriculum of progressively more complex levels along the boundary of the protagonist's capabilities. PAIRED is a creative method in the sense that the teacher may potentially generate an endless sequence of novel levels. However, as the teacher only adapts through gradient updates, it is inherently slow to adapt to changes in the student policies. Further, over a long training horizon, PAIRED amounts to a continual learning setting, where the student policies become susceptible to forgetting of previously useful behaviors, leading to a slowdown and potential cycling.

2.3 Prioritized Level Replay

Prioritized Level Replay [PLR, 15] is an active-learning strategy shown to improve the sample efficiency and generalization to unseen levels of a policy trained and evaluated on levels from a common UPOMDP, typically implemented as a seeded simulator. PLR maintains a level buffer $\mathbf{\Lambda}$ of the top K visited levels with highest learning potential as estimated by the time-averaged L1 value loss of the learning agent over the last episode on each level. At the start of each training episode, with some predefined replay probability p, PLR uses a bandit to sample the level from $\mathbf{\Lambda}$ to maximize the estimated learning potential; otherwise, with probability $1 - p$, PLR samples a new level from the simulator. In contrast to the generative but slow-adapting PAIRED, PLR cannot by itself create new

levels, but instead it acts as a fast-adapting curation mechanism for selecting the next training level among a set of previously encountered levels based on some prioritization metric. As described in Section 5, REPAIRED exploits this complementary properties to arrive at a novel UED algorithm capable of both generating a diverse, unbounded sequence of levels, as well as quickly switching among them to maximize a training objective while mitigating forgetting.

3 The Robustness of Dual Curriculum Design

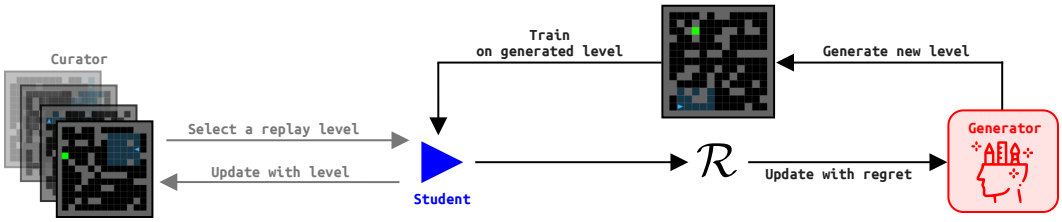


Figure 2: Overview of Dual Curriculum Design. The student agent learns in the presence of two co-adapting, teachers that aim to maximize the student's regret: The generative teacher (generator) designs new levels to challenge the agent, and the curating teacher (curator) prioritizes a set of levels already created, and selectively samples them for replay.

The prior approaches of PAIRED and PLR reveal a natural duality: approaches which gradully learn to generate levels like PAIRED and methods which cannot generate levels, but rather quickly curate existing ones, like PLR. This duality suggests combining slow level generators, with fast level curators. We call this novel class of UED algorithms Dual Curriculum Design (DCD). For instance, PLR can be seen as having a prioritized sampling mechanism for a curator alongside a random generator, while PAIRED can be seen a regret-maximizing generator and no curator. DCD naturally spans the combination of the two, in addition to the degenerate case of Domain Randomization (DR) where we have no replay mechanism, and a random level generator.

To theoretically analyze this space of methods we model DCD as a three player game among a student agent and two teachers called the *dual curriculum game*. However, to formalize this game, we must first formalize the single-teacher context. Suppose the UPOMDP is clear from context. Then, given a utility function for the teacher $U_t(\pi, \theta)$ we can naturally define a game, we will call the *base game* between the student s and the teacher t as $G = \langle S = S_s \times S_t, U = U_s \times U_t \rangle$ where $S_t = \Pi$ is the strategy set of the student, $S_t = \Theta$ is the strategy set of the teacher and $U_s(\pi, \theta) = V^\theta(\pi)$ is the utility function of the student. We will be studying different choices for utility functions for the teacher, focusing mostly on the maximum-regret objective $U_t^R(\pi, \theta)$ and the uniform objective $U_t^U(\pi, \theta)$, each defined as follows (for any constant C):

$$U_t^R(\pi, \theta) = \operatorname*{argmax}_{\pi^* \in \Pi} \{V^\theta(\pi^*) - V^\theta(\pi)\} \quad (1) \qquad U_t^U(\pi, \theta) = C \quad (2)$$

In the dual curriculum game, one teacher plays the game with probability p and the other with probability $(1-p)$, or, more formally, it is the tuple $\overline{G} = \langle \overline{S} = S_s \times S_t \times S_t, U = \overline{U}_s \times \overline{U}_t \times \overline{U}_t \rangle$ where the new utility functions $\overline{U}_s, \overline{U}_t^1, \overline{U}_t^2$ as follows:

$$\overline{U}_t^1(\pi, \theta^1, \theta^2) = pU_t^1(\pi, \theta^1) \quad (3) \qquad \overline{U}_t^2(\pi, \theta^1, \theta^2) = (1-p)U_t^2(\pi, \theta^2) \quad (4)$$

$$\overline{U}_s(\pi, \theta^1, \theta^2) = pU_s(\pi, \theta^1) + (1-p)U_s(\pi, \theta^2) \quad (5)$$

Where U_t^1 and U_t^2 are the utility functions for the first and second teacher respectively. Our main theorem is that equalibria in the dual curriculum game are approximate equalibria of both the base game for either of the original teachers and the base game with an teacher maximizing the joint-reward of $pU_t^1 + (1-p)U_t^2$, where the quality of the approximations depend on the mixing probability p.

Theorem 1. *Let B be the maximum difference between U_t^1 and U_t^2, and let $(\pi, \theta^1, \theta^2)$ be a NE for $\overline{G}$. Then $(\pi, p\theta^1 + (1-p)\theta^2)$ is an approximate NE for the base game with either teacher or for a teacher optimizing their joint objective or more precisely it is a $4Bp$-approximate NE when $U_t = pU_t^1 + (1-p)U_t^2$ or when $U_t = U_t^1$ and a $4B(1-p)$-approximate NE when $U_t = U_t^2$.*

The intuition behind this theorem is that, since the two teachers do not affect each other's behavior, their best response to a fixed π_s is to choose a parameter θ which maximizes U_t^1 and U_t^2 respectively.

Moreover, the teacher's strategies can be viewed as a combined strategies for the joint-objective of the base game, or for either teacher's objective in the base game and is an approximate best-response in each of these contexts when both teachers are playing their individual best responses. Thus when we reach a NE of the dual curricula game the teachers are an approximate best response for both the joint objective and their individual objectives, and so they are in an approximate NE of both base games. The full details of this proof are outlined in Appendix A.

4 Robustifying PLR

In this section, we provide theoretical justification for the empirically observed effectiveness of PLR, and in the process, motivate a counterintuitive adjustment to the algorithm.

Algorithm 1: Robustified PLR (PLR$^{\perp}$)

Randomly initialize policy $\pi(\phi)$ and an empty level buffer, $\mathbf{\Lambda}$ of size K.
while *not converged* **do**
  Sample replay decision $d \sim P_D(d)$
  if $d = 0$ **then**
    Sample level θ from level generator
    Collect π's trajectory τ on θ, with a stop-gradient $\phi_{\perp}$ *i.e. Suppress policy update*
    Compute PLR score, $S = \textbf{score}(\tau, \pi)$
    Update $\mathbf{\Lambda}$ with θ using score S^P
  else
    Use PLR to sample a replay level from the level store, $\theta \sim \mathbf{\Lambda}$
    Collect policy trajectory τ on θ and update π with rewards $\boldsymbol{R}(\tau)$
  end
end

4.1 Achieving Robustness Guarantees with PLR

PLR provides strong empirical gains in generalization, but lacks any theoretical guarantees of robustness. One step towards achieving such a guarantee is to replace its L1 value-loss prioritizaton with a regret prioritization, using the methods we discuss in Section 4.2: While L1 value loss may be good for quickly training the value function, it can bias the long-term training behavior toward high-variance policies. However, even with this change, PLR holds weaker theoretical guarantees because the random generating teacher can bias the student away from minimax regret policies and instead, toward policies that sacrifice robustness in order to excel in unstructured levels. We formalize this intuitive argument in the following corollary of Theorem 1.

Corollary 2. *Let $\overline{G}$ be the dual curriculum game in which the first teacher maximizes regret so $U_t^1 = U_t^R$ and the second teacher plays arbitrarily so $U_t^2 = U_t^U$. Let $V^\theta(\pi)$ be bounded in $[B^+, B^-]$ for all θ, π. Further, suppose that $(\pi, \theta^1, \theta^2)$ is a Nash equalibria of $\overline{G}$. Let $R^* = \max_{\pi_A \in \Pi}\{\min_{\theta, \pi_B \in \Theta, \Pi}\{\textsc{Regret}^\theta(\pi_A, \pi_B)\}\}$ be optimal worst-case regret. Then π is $4(B^+ - B^-)p$ close to having optimal worst-case regret, or formally $\min_{\theta, \pi_B \in \Theta, \Pi}\{\textsc{Regret}^\theta(\pi, \pi_B)\} \geq R^* - 4(B^+ - B^-)p$. Moreover, there exists environments for all values of p within a constant factor of achieving this bound.*

The proof of Corollary 2 follows from a direct application of Theorem 1 to show that a NE of $\overline{G}$ is an approximate NE for the base game of the first teacher, and through constructing a simple example where the average differs from the minimax regret. These arguments are described in full in Appendix A. This corollary provides some justification for why PLR improves robustness of the equilibrium policy, since it biases the resulting policy towards a minimax regret policy. However, it also points a way towards further improving PLR: if the probability of using a teacher generated level directly was set to 0, Corollary 2 shows that in equilibrium the resulting policy converges to a minimax regret policy. Consequently, we arrive at the counterintuitive idea of avoiding gradient updates from trajectories collected from randomly sampled levels, to ensure that at NE, we find a minimax regret policy. From a robustness standpoint, it is thus *optimal to train on less data*. The modified PLR algorithm PLR$^{\perp}$ with this counterintuitive adjustment is summarized in Algorithm 1, in which this small change relative to the original algorithm is highlighted in blue.

4.2 Estimating Regret

In general, levels may differ in maximum achievable returns, making it impossible to know the true regret of a level without access to an oracle. As the L1 value loss typically employed by PLR does not generally correspond to regret, we turn to alternative scoring functions that better approximate regret. Two approaches, both effective in practice, are discussed below.

Positive Value Loss Averaging over all transitions with positive value loss amounts to estimating regret as the difference between maximum achieved return and predicted return on an episodic basis. However, this estimate is highly biased, as the value targets are tied to the agent's current, potentially suboptimal policy. As it only considers positive value losses, this scoring function leads to optimistic sampling of levels with respect to the current policy. When using GAE [29] to estimate bootstrapped value targets, this loss takes the following form, where λ and γ are the GAE and MDP discount factors respectively, and δ_t, the TD-error at timestep t:

$$\frac{1}{T}\sum_{t=0}^{T}\max\left(\sum_{k=t}^{T}(\gamma\lambda)^{t-k}\delta_t^P, 0\right),$$

Maximum Monte Carlo (MaxMC) We can mitigate some of the bias of the positive value loss by replacing the value target with the highest return achieved on the given level so far during training. By using this maximal return, the regret estimates no longer depend on the agent's current policy. This estimator takes the simple form of $(1/T)\sum_{t=0}^{T} R_{\max} - V(s_t)$. In our dense-reward experiments, we resort to computing this loss as the difference between the maximum achieved return and the value prediction at the initial state.

5 Replay-Enhanced PAIRED (REPAIRED)

A straightforward extension of PLR is to replace the random teacher (i.e. the level generator) used by PLR with the PAIRED teacher. This addition then necessarily entails introducing a second student agent, the antagonist, also equipped with its own PLR level buffer. In each episode, with probability p, the student agents train on a newly generated level and with probability $1-p$ train on a level sampled from each student's own PLR buffer, prioritizing levels by highest estimated regret. We will refer to this extension as *Replay-Enhanced PAIRED* (REPAIRED). An overview of REPAIRED is provided by black arrows in Figure 2, with the student(s) being both the protagonist and antagonist, while the full procedure is outlined in Appendix B.

Since REPAIRED's variant of PLR and PAIRED promote regret in equilibrium, it would be reasonable to believe the combination of the two would promote regret in equilibrium. As a straightforward corollary of Theorem 1, which we describe in Appendix 1, we can show that, in a theoretically ideal setting, combing these two algorithms as is done in REPAIRED will still find minimax regret strategies in equilibrium.

Corollary 3. *Let $\overline{G}$ be the dual curriculum game in which the both teachers maximize regret so $U_t^1 = U_t^2 = U_t^R$. Further, suppose that $(\pi, \theta^1, \theta^2)$ is a Nash equalibria of $\overline{G}$. Then $\pi \in$* $\text{argmax}_{\pi_A \in \Pi}\{\min_{\theta, \pi_B \in \Theta, \Pi}\{\textsc{Regret}^{\theta}(\pi_A, \pi_B)\}\}$

This result gives us some amount of assurance that, if our method finds a Nash equilibrium, then the protagonist will be implementing a minimax regret strategy, which has the benefits outlined in [8]: since a minimax regret policy solves all solvable environments, whenever this is possible and sufficiently well-defined, we should expect policies resulting from the equalibria behavior of REPAIRED to be robust and versatile across all environments in the domain.

6 Experiments

Our experiments firstly aim to (1) assess the empirical performance of the theoretically motivated PLR$^{\perp}$, and secondly, seek to better understand the effect of replay on unsupervised environment design, specifically (2) its impact on the zero-shot generalization performance of the induced student policies, and (3) the complexity of the levels designed by the teacher. To do so, we compare PLR and REPAIRED against their replay-free counterparts, DR and PAIRED in the two highly distinct settings of discrete control with sparse rewards and continuous control with dense rewards. We provide environment descriptions alongside model and hyperparameter choices in Appendix D.

6.1 Partially-Observable Navigation

Each navigation level is a partially-observable maze requiring student agents to take discrete actions to reach a goal and receive a sparse reward. Our agents use PPO [30] with an LSTM-based recurrent policy to handle partial observability. Before each episode, the teacher designs the level in this order: beginning with an empty maze, it places one obstructing block per time step up to a predefined block budget, and finally places the agent followed by the goal.

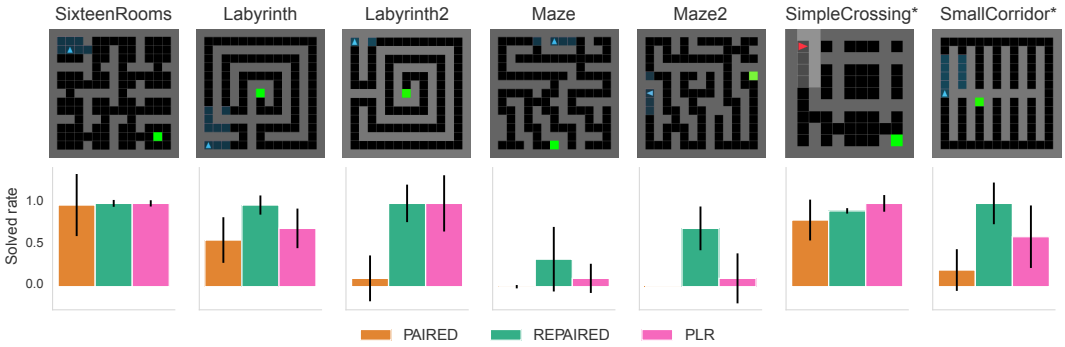


Figure 3: Zero-shot transfer performance for variants of PAIRED and REPAIRED in 7 different environments after 250M training steps. The plots show median test returns over 7 runs and the interquartile range.

Zero-Shot Generalization We train policies with each method for 250M steps and evaluate zero-shot generalization on several challenging OOD environments, in addition to levels from the full distribution of two procedurally-generated environments, `SimpleCrossing` and `SmallCorridor`. We also compare against DR and minimax baselines trained for 3B steps reported in [8]. As reported in Figure 7, we find that PLR$^{\perp}$ with replay rate of 0.5 matches the sample-efficiency of PLR on a gradient-update basis, *despite taking half as many gradient steps*. While this result corroborates the theoretical basis of PLR$^{\perp}$, we opt to use standard PLR in our maze experiments to minimize the number of effective environment steps taken due to the extremely large number of steps needed for full comparison with PAIRED. Our results in Figure 3 and 4 show that PLR$^{\perp}$ and REPAIRED attain significantly better zero-shot generalization than their replay-free counterparts. Notably, well before 250 million steps, both methods significantly outperform PAIRED after 3 billion training steps as reported in [8]. On `Labyrinth` and `Maze`, this amounts to an over $40\times$ speed-up in sample efficiency. Notably, PLR tends to see worse transfer on levels requiring backtracking, such as `Maze`, `Maze2`, and `SmallCorridor`. This suggests that, while randomly sampling the space of levels provides enough variation for PLR to induce a robustifying curriculum, levels that robustify against backtracking may be hard to find through random search alone, and instead benefit from a trained active teacher that more deliberately searches for regret-inducing levels.

Emergent Complexity As the student agents improve, the teachers must generate more challenging levels to maintain regret. We measure the resultant emergent complexity by tracking the number of blocks in each level and the shortest path length to the goal (where unsolvable levels are assigned a length of 0). These results, summarized in Figure 7, show that PAIRED slowly adapts the complexity over training while REPAIRED takes even longer due to its less frequent gradient updates, but maintains higher complexity metrics, even after adjusting proportionately by replay rate. This is likely due to its PLR mechanism inducing more robust student policies. Our extended results in Appendix C confirm that both PAIRED and REPAIRED slowly increase complexity over time, eventually matching that attained in a fraction of the number of gradient steps by PLR$^{\perp}$. This result shows that random search is surprisingly efficient at continually discovering levels of increasing complexity, given an appropriate curation mechanism such as PLR. Figure 7 of Appendix C.1 shows that like methods with a regret-maximizing teacher, PLR finds levels exhibiting complex structure.

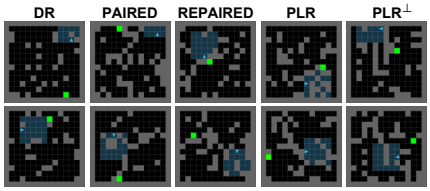


Figure 5: Examples of emergent structures generated by each method.

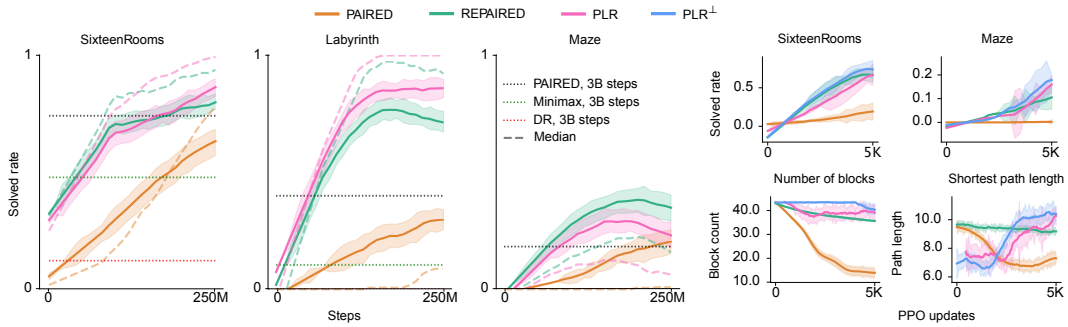


Figure 4: Zero-shot transfer performance over training for PAIRED and REPAIRED variants. The plots show mean and standard error across 10 runs. The dotted lines mark the mean performance of PAIRED, minimax, and DR after 3B training steps, as reported in [8], while dashed lines indicate median returns.

6.2 Pixel-Based Car Racing with Continuous Control

To test the versatility and scalability of our methods, we turn to an extended version of the `CarRacing` environment from OpenAI Gym [4]. This environment entails continuous control with dense rewards, a 3-dimensional action space, and partial, pixel observations, with the goal of driving a full lap around a track. To enable UED of any closed-loop track, we reparameterize CarRacing to generate tracks as Bézier curves [19] with arbitrary control points. The teacher generates levels by choosing a sequence of up to 12 control points, which uniquely defines a Bézier track within specific, predefined curvature constraints. After 5M steps of training, we test the zero-shot transfer performance of policies trained by each method on 20 levels replicating official human-designed Formula One (F1) tracks (see Figure 11 in the Appendix for a visualization of the tracks). Note that these tracks are significantly OOD, as they cannot be defined with just 12 control points. In Figure 6 we show the progression of zero-shot transfer performance for the original `CarRacing` environment, as well as three F1 tracks of varying difficulty, while also including the final performance on the full distribution of F1 tracks. For the final performance, we also evaluated the state-of-the-art `CarRacing` agent from [34] on our new F1 benchmark.

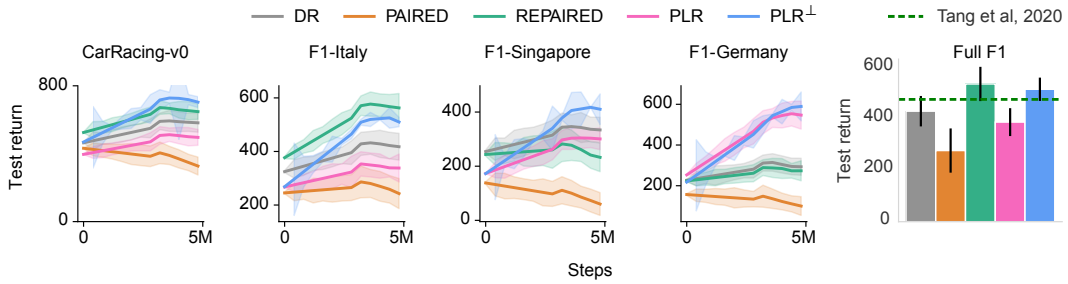


Figure 6: Training and zero-shot transfer performance. Plots show mean and standard error over 10 runs.

Unlike in the sparse, discrete navigation setting, we find DR leads to moderately successful policies for zero-shot transfer in `CarRacing`. Dense rewards simplify the learning problem and random Bezier tracks occasionally contain the challenges seen in F1 tracks, such as hairpin turns and observations showing parallel tracks due to high local curvature. Still, we see that learning from regret-based prioritized replay of tracks significantly outperforms uniformly sampling from randomly generated tracks, producing policies demonstrating better zero-shot transfer to F1 tracks. Remarkably, with a replay rate of 0.5, PLR$^{\perp}$ sees drastic gains over PLR in zero-shot performance, despite directly training on half the data and half as many gradient updates. We thus find strong empirical support for the robustness property of PLR$^{\perp}$. Once again, we see that random search with curation via PLR produces a rich enough selection of levels and an effective curriculum.

We also see that PAIRED struggles to train a robust protagonist in `CarRacing`. Specifically, PAIRED overexploits the relative strengths of the antagonist over the protagonist, finding curricula that steer the protagonist towards policies that ultimately perform poorly even on simple tracks, leading gradual reduction in level complexity. We present training curves revealing this dynamic in Appendix C. As shown in Figure 6, REPAIRED improves upon this significantly, inducing a policy competitive with

that trained using PLR$^{\perp}$ in mean performance on the full F1 benchmark. Furthermore, both of the new replay-based algorithms outperform the state-of-the-art AttentionAgent [34], which was specifically designed for this environment, in the OOD F1 domain. Figure 8 in Appendix C further reveals that PLR$^{\perp}$ and REPAIRED induce `CarRacing` policies that tend to achieve higher minimum returns on average compared to other methods that do not have such a robustness guarantees, providing further evidence for the benefits of the minimax regret property.

7 Related Work

In inducing parallel curricula, DCD follows a rich lineage of curriculum learning methods [2, 28, 25, 20]. Many previous curriculum learning algorithms resemble the curator in DCD, sharing similar underlying selective-sampling mechanisms as PLR$^{\perp}$. Most similar is TSCL [16], which prioritizes levels based on return rather than value loss, and has been shown to overfit to training levels in some settings [15]. In our setting, replayed levels can be viewed as past strategies from a level-generating teacher. This links our replay-based methods to fictitious self-play [FSP, 11], and more closely, Prioritized FSP, which selectively samples opponents based on historic win ratios.

Recent approaches that make use of a generating adversary include Asymmetric Self-Play [33, 23], wherein one agent proposes tasks for another in the form of environment trajectories, and AMIGo [5], wherein the teacher is rewarded for proposing reachable goals. While our methods do not presuppose a goal-based setting, others have made progress here using generative modeling [10, 26], latent skill learning [12], and exploiting model disagreement [40]. These methods are less generally applicable than PLR$^{\perp}$, and unlike our DCD methods, they do not provide well-principled robustness guarantees.

Other recent algorithms can be understood as forms of UED and like DCD, framed in the lens of decision theory. POET [37, 38], a coevolutionary approach [24], uses a population of *minimax* (rather than minimax regret) adversaries to construct terrain for a BipedalWalker agent. In contrast to our methods, POET requires training a large population of both agents and environments and consequently a sizable compute overhead. APT-Gen [9] also procedurally generates tasks, but requires access to target tasks, whereas our methods seek to improve zero-shot transfer.

The DCD setting can be seen as a form of adaptive domain randomization [DR, 17, 13]. DR has seen success in assisting sim2real transfer for robotics [35, 14, 1, 22]. DR itself is subsumed by procedural content generation [27], for which UED and DCD may be seen as providing a formal, decision-theoretic framework allowing development of provably optimal algorithms.

8 Conclusions

We developed a novel connection between PLR and minimax-regret UED approaches like PAIRED. We demonstrated that PAIRED, a slow but generative method for level design can be combined with PLR, a fast but non-generative method that relies instead on selective replay of the most promising, existing levels. In order to theoretically ground this new setting, we introduced Dual Curriculum Design (DCD), where a student policy is challenged by a team of two co-adapting, regret-maximizing teachers—one, a generator that creates new levels and the other, a curator that selectively samples existing levels for replay. This formalism allows us to prove robustness guarantees for PLR at NE, and leads to the counterintuitive result that PLR can be made provably robust by training on *less data*, specifically trajectories on levels sampled for replay. Further, we develop Replay-Enhanced PAIRED (REPAIRED), the natural instantiation of DCD. Empirically, we find REPAIRED significantly improves zero-shot generalization over PAIRED in two distinct environments, while also inducing policies robust to situations where PLR$^{\perp}$ falls short.

Our theoretical contribution may extend to more general continual learning settings in which agents must learn to master a sequence of tasks with predefined (or inferred) episode boundaries, if tasks are assumed to be designed by a regret-optimizing teacher. This suggests DCD-derived methods like PLR$^{\perp}$ may also be applied to continual learning. We believe this to be a promising direction for future work.

References

[1] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

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

[5] A. Campero, R. Raileanu, H. Kuttler, J. B. Tenenbaum, T. Rocktäschel, and E. Grefenstette. Learning with AMIGo: Adversarially motivated intrinsic goals. In *International Conference on Learning Representations*, 2021.

[6] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2048–2056, 2020.

[7] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research. PMLR, 2019.

[8] M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[9] K. Fang, Y. Zhu, S. Savarese, and F.-F. Li. Adaptive procedural task generation for hard-exploration problems. In *International Conference on Learning Representations*, 2021.

[10] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1515–1528. PMLR, 10–15 Jul 2018.

[11] J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 805–813, 2015.

[12] A. Jabri, K. Hsu, A. Gupta, B. Eysenbach, S. Levine, and C. Finn. Unsupervised curricula for visual meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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

[14] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *1st Conference on Robot Learning*, 2017.

[15] M. Jiang, E. Grefenstette, and T. Rocktäschel. Prioritized level replay. *CoRR*, abs/2010.03934, 2020.

[16] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher-student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, PP, 07 2017.

[17] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 1162–1176. PMLR, 2019.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[19] M. E. Mortenson. *Mathematics for Computer Graphics Applications: An Introduction to the Mathematics and Geometry of CAD/Cam, Geometric Modeling, Scientific Visualizati*. Industrial Press, Inc., USA, 2nd edition, 1999.

[20] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. Taylor, and P. Stone. Curriculum learning for reinforcement learning domains: A framework and survey. 03 2020.

[21] J. F. Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

[22] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019.

[23] O. OpenAI, M. Plappert, R. Sampedro, T. Xu, I. Akkaya, V. Kosaraju, P. Welinder, R. D'Sa, A. Petron, H. P. de Oliveira Pinto, A. Paino, H. Noh, L. Weng, Q. Yuan, C. Chu, and W. Zaremba. Asymmetric self-play for automatic goal discovery in robotic manipulation, 2021.

[24] E. Popovici, A. Bucci, R. P. Wiegand, and E. D. De Jong. *Coevolutionary Principles*, pages 987–1033. Springer Berlin Heidelberg, 2012.

[25] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. Automatic curriculum learning for deep rl: A short survey. In C. Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4819–4825. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Survey track.

[26] S. Racaniere, A. Lampinen, A. Santoro, D. Reichert, V. Firoiu, and T. Lillicrap. Automated curriculum generation through setter-solver interactions. In *International Conference on Learning Representations*, 2020.

[27] S. Risi and J. Togelius. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8):428–436, Aug 2020.

[28] J. Schmidhuber. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *Frontiers in psychology*, 4:313, 06 2013.

[29] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

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

[32] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[33] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.

[34] Y. Tang, D. Nguyen, and D. Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2020.

[35] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. 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)*, pages 23–30, 2017.

[36] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[37] R. Wang, J. Lehman, J. Clune, and K. O. Stanley. Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions. *CoRR*, abs/1901.01753, 2019.

[38] R. Wang, J. Lehman, A. Rawal, J. Zhi, Y. Li, J. Clune, and K. Stanley. Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9940–9951, 2020.

[39] A. Zhang, N. Ballas, and J. Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

[40] Y. Zhang, P. Abbeel, and L. Pinto. Automatic curriculum learning through value disagreement. In *Advances in Neural Information Processing Systems*, volume 33, pages 7648–7659, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? Yes
   (b) Did you describe the limitations of your work? Yes
   (c) Did you discuss any potential negative societal impacts of your work? Not applicable
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? Yes
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? Yes
   (b) Did you include complete proofs of all theoretical results? Yes
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? Yes
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? Yes
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? Yes
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? No
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? Yes
   (b) Did you mention the license of the assets? Yes
   (c) Did you include any new assets either in the supplemental material or as a URL? Not applicable.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? Not applicable.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? Not applicable.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? Did not use human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? Did not use human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? Did not use human subjects.