Beyond Discrete Environments: Benchmarking Regret-Based Automatic Curriculum Learning in MuJoCo

Chin-Jui Chang[1], Chen-Xing Li[1,2], Jan Seyler[2] and Shahram Eivazi[1,2]
[1]*Department of Computer Science, Eberhard Karls University of Tübingen, Tübingen, Germany*
[2]*Festo, Esslingen, Germany*
{*chin-jui.chang, chenxing.li, shahram.eivazi*}*@student.uni-tuebingen.de, jan.seyler@festo.com*

Keywords: Reinforcement Learning, Curriculum Learning, Unsupervised Environment Design, Regret-Based Methods, Robustness, Generalization, Automatic Curriculum Learning, Benchmarking.

Abstract: Training robust reinforcement learning (RL) agents capable of performing well in unseen scenarios remains a significant challenge. Curriculum learning has emerged as a promising approach to build transferable skills and enhance overall robustness. This paper investigates regret-based adversarial methods for automatically generating curricula, extending their evaluation beyond simple environments to the more complex MuJoCo suite. We benchmark several state-of-the-art regret-based methods against traditional baselines, revealing that while these methods generally outperform baselines, the performance gains are less substantial than anticipated in these more complex environments. Moreover, our study provides valuable insights into the application of regret-based curriculum learning methods to continuous parameter spaces and highlights the challenges involved. We discuss promising directions for improvement and offer perspectives on how current automatic curriculum learning techniques can be applied to real-world tasks.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Reinforcement learning (RL) has evolved into a key AI component, advancing fields from robotics (OpenAI et al., 2020) to gaming (Silver et al., 2018; Mnih et al., 2015), as documented in recent surveys (Arulkumaran et al., 2017; Li, 2018). However, traditional RL faces two major challenges: sparse rewards, where feedback is infrequent (Andrychowicz et al., 2017), and limited robustness in novel situations (Cobbe et al., 2019b; Kirk et al., 2021). Curriculum learning addresses these limitations by incrementally increasing task complexity (Narvekar et al., 2020; Portelas et al., 2020), helping agents manage both sparse rewards and adaptation to new scenarios.

Curriculum learning in RL has several advantages: it allows agents to master hard tasks that would be impossible to learn if approached directly (Bengio et al., 2009; Kulkarni et al., 2016; Dietterich, 2000; Held et al., 2018); it breaks down complex tasks into a series of incrementally harder sub-tasks so agents can build skills incrementally; and it promotes the development of more adaptable agents that can handle a wide range of tasks and environments. Moreover, curriculum learning can also improve sample efficiency, agents can learn more with less data (Narvekar et al., 2020; Portelas et al., 2020), which is valuable in real world applications where data collection is time consuming or expensive.

Historically, designing curricula in reinforcement learning has been primarily hand-crafted (Graves et al., 2017; Narvekar et al., 2020; Bengio et al., 2009; Taylor and Stone, 2009). While effective in certain scenarios, hand-crafted curricula are time consuming to create and require domain-specific knowledge. Moreover, these curricula lack flexibility and fail to adapt to an individual agent's evolving capability. This becomes especially problematic when dealing with large or unknown task spaces or scenarios that require frequent curriculum updates.

To address these problems, the Unsupervised Environment Design (UED) (Dennis et al., 2020) framework has been proposed which views curriculum generation as an automated process. UED formulates the learning process as a game between the agent and an environment generator. The generator aims to create environments that maximize the agent's regret - the difference between the optimal performance and the agent's current performance in a given environment. Meanwhile, the agent tries to minimize this regret by improving its policy. This ensures the generated environments are the most challenging yet still solvable,

Chang, C.-J., Li, C.-X., Seyler, J. and Eivazi, S.
Beyond Discrete Environments: Benchmarking Regret-Based Automatic Curriculum Learning in MuJoCo.
DOI: 10.5220/0013116400003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 2*, pages 133-143
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

so the agent improves in areas where it has the most room to grow. Unlike simpler return-based adversarial methods (Sukhbaatar et al., 2018; Wang et al., 2019; Florensa et al., 2017) which often create impossible or trivial environments leading to inefficient learning, regret-based methods find the balance that promotes skill development.

While UED family of research has been demonstrated to be promising, most of the research has been limited to a few simple benchmark environments like MiniGrid, BipedalWalker and CarRacing (Dennis et al., 2020; Jiang et al., 2021b; Parker-Holder et al., 2022; Azad et al., 2023), which may not fully capture the complexity and challenges of more realistic scenarios. The limited scope of these test environments raises questions about the scalability of UED methods to more complex and realistic tasks. Many of these benchmark environments have simplified dynamics and constrained action spaces that may not adequately represent the continuous, high-dimensional state and action spaces found in real-world applications. This gap in current research motivates us to explore UED in more complex simulation environments that better reflect the challenges of real-world tasks.

Our research benchmarks various UED algorithms across a wider range of environments, providing a comprehensive landscape of their effectiveness. By extending our testing to more complex scenarios, we aim to identify the strengths and limitations of current approaches within the UED framework. Our primary focus is on comparing these algorithms to determine which ones demonstrate superior performance in unseen environments. We evaluate their ability to generate effective curricula, transfer learned skills to new situations, and adapt to different environment parameterizations, particularly in challenging continuous control tasks. Through this comparative analysis, we seek to offer practical insights for researchers and practitioners interested in applying UED methods to realistic domains, potentially bridging the gap between simplified benchmarks and real-world robotics applications.

2 RELATED WORKS

2.1 RL Benchmarks

Several RL benchmarks have been proposed to evaluate various aspects of RL algorithms. The Arcade Learning Environment (ALE) (Machado et al., 2018) incorporates Atari 2600 games to assess RL algorithms' performance across diverse environments. Bsuite (Osband et al., 2020), a compact benchmark suite, tests algorithms' robustness to noise and evaluates core RL agent capabilities including generalization, exploration, and long-term consequence handling. CARL (Benjamins et al., 2021) extends established RL environments to contextual RL problems, providing a consistent theoretical framework for studying generalization. It allows researchers to create environment variations by modifying goal states or altering transition dynamics, offering a flexible platform for in-depth RL research.

Another category of benchmarks employs Procedural Content Generation (PCG) to create diverse environments with varying complexity levels. The Procgen Benchmark (Cobbe et al., 2019a) is a prominent example, featuring 16 procedurally generated game-like environments designed to evaluate both sample efficiency and generalization in reinforcement learning. Procgen uses PCG to create a vast array of levels with randomized layouts, assets, and game-specific details, forcing agents to learn robust policies that generalize across diverse scenarios. Another notable example is Obstacle Tower (Juliani et al., 2019), which offers a rich 3D environment with complex navigation challenges. While these PCG-based environments provide a range of difficulty levels and emphasize generalization, they lack built-in curricula or structured training paces for agents. On the other hand, our work focuses on curriculum generation algorithms that automatically design optimal training paths and paces.

To our knowledge, TeachMyAgent (Romac et al., 2021) is the only existing benchmark specifically focused on curriculum learning algorithms in reinforcement learning. It unifies various curriculum reinforcement learning (CRL) methods under a teacher-student framework, evaluating diverse teacher algorithms such as ALP-GMM (Portelas et al., 2019), RIAC (Baranes and Oudeyer, 2009), CovarGMM (Moulin-Frier et al., 2014), SPDL (Klink et al., 2020), ADR (Plappert et al., 2019), and GoalGAN (Florensa et al., 2018). While TeachMyAgent provides a structured platform for benchmarking these algorithms, our work differentiates itself by focusing on regret-based curriculum learning methods, which are not included in their benchmark. Additionally, we extend our evaluation to a broader range of MuJoCo environments, offering a comprehensive assessment of algorithm performance across diverse and complex continuous control tasks.

2.2 Alternative Approaches to Curriculum Learning

While our work focuses specifically on regret-based curriculum learning within the UED framework, it's important to contextualize our approach within the broader landscape of curriculum learning in reinforcement learning. Besides regret-based methods, many existing curriculum learning approaches employ different strategies for task generation and sequencing. Narvekar et al. (2020) (Narvekar et al., 2020) provide a comprehensive survey and framework for these diverse approaches. They categorize curriculum learning methods based on various dimensions, including task generation, sequencing methods, and transfer learning techniques. The survey highlights several key approaches such as sample sequencing strategies, co-learning strategies, and methods that modify reward functions or state distributions.

Some recent curriculum learning approaches have also shown promising results. The Paired Open-Ended Trailblazer (POET) algorithm (Wang et al., 2019) maintains a population of environment-agent pairs, continuously performing three key tasks: generating new environments by mutating existing ones, improving agents within their paired environments, and attempting to transfer successful agents between environments. The process creates a co-evolutionary dynamics of agents and environments. However, POET requires maintaining a population of environments and testing all agents in their paired environments, which require more computational resources. Additionally, they require a manually decided threshold to discard environments, a limitation not present in the regret-based methods we benchmark.

The CURROT (Klink et al., 2022) algorithm reformulates curriculum generation as a constrained optimal transport problem. Unlike previous methods that use KL divergence, CURROT uses Wasserstein distance to measure distribution similarity. The algorithm enforces a strict performance constraint across all tasks in the curriculum, avoiding the pitfall of mixing trivial and infeasible tasks. By representing the curriculum as a set of particles and utilizing context buffer (or environment parameter buffer), CURROT balances the trade-off between exploration and exploitation. However, a key limitation of CURROT is its reliance on a pre-defined target task distribution. This contrasts with the open-ended approaches we evaluate in this work, which can generate ever-expanding curricula without the need for predefined target distributions, offering greater flexibility in task exploration and adaptation.

3 BACKGROUND

In this section, we provide an overview of the background knowledge necessary to enhance understanding of the algorithms discussed in Section 4.

3.1 Unsupervised Environment Design

Unsupervised Environment Design (UED) (Dennis et al., 2020) is a paradigm that aims to automatically generate a curriculum of levels or tasks for a student agent, with the goal of achieving systematic generalization across all possible levels. In this framework, levels (hereafter used interchangeably with environment parameters) are typically produced by a generator, or teacher. This generator operates by maximizing a utility function, $U_t(\pi, \theta)$, where π represents the student agent's policy and θ denotes the level parameters. The utility function serves as a measure of the educational value or challenge provided by a given level, guiding the curriculum's progression to optimize the agent's learning trajectory. This approach enables a dynamic and adaptive learning process, continually tailoring the environment to the agent's evolving capabilities.

UED methods employ teachers that maximize regret, defined as the difference between the expected return of the current policy and the optimal policy. The teacher's utility is then defined as:

$$U_t^R(\pi, \theta) = \underset{\pi^* \in \Pi}{\operatorname{argmax}} \{\mathrm{REGRET}_\theta(\pi, \pi^*)\} \quad (1)$$

$$= \underset{\pi^* \in \Pi}{\operatorname{argmax}} \{V_\theta(\pi^*) - V_\theta(\pi)\}. \quad (2)$$

Regret-based objectives are desirable because they promote the generation of the simplest levels that the student cannot currently solve (Dennis et al., 2020). Formally, if the learning process reaches a Nash equilibrium, the resulting student policy π provably converges to a minimax regret policy, defined as:

$$\pi = \underset{\pi \in \Pi}{\operatorname{argmin}} \left\{ \max_{\theta, \pi^* \in \Theta, \Pi} \{\mathrm{REGRET}_\theta(\pi, \pi^*)\} \right\}. \quad (3)$$

However, without access to π^* for each level, UED algorithms must approximate the regret. In practice, regret is estimated as the difference in return attained by the main student agent (i.e., protagonist) and a second agent (i.e., antagonist).

3.2 Domain Randomization

Domain Randomization (DR) (Tobin et al., 2017) is a technique used to bridge the reality gap between simulated and real-world environments, particularly

in the context of robotic learning and computer vision tasks. Unlike Unsupervised Environment Design (UED) methods, DR does not actively generate a curriculum but instead randomly samples a large number of environment configurations from a predefined distribution. The randomization process can be formalized as sampling from a distribution $p(\theta)$ over environment parameters θ, where the goal is to train a model f_ϕ (e.g., a neural network) that minimizes the expected loss $\mathcal{L}$ across this distribution:

$$\phi^* = \arg\min_\phi \mathrm{E}_{\theta\sim p(\theta)}[\mathcal{L}(f_\phi, \theta)]. \quad (4)$$

DR's objective is to let policies generalize to unseen scenarios by making the model encounter a wide array of variations during training. While it shows very promising results for various robotic tasks and computer vision, the approach of DR may prove inefficient in more complex domains because of the low probability of sampling relevant environment configurations in comparison with more targeted curriculum learning approaches used by UED.

3.3 Prioritized Level Replay

Prioritized Level Replay (PLR) (Jiang et al., 2021a) is a method designed to enhance learning efficiency by selectively sampling training levels based on their estimated learning potential. The potential value for future learning is calculated using the average magnitude of the Generalized Advantage Estimate (GAE) over the episode trajectory, defined as:

$$\mathrm{Score}(l) = \frac{1}{T}\sum_{t=1}^{T} |\delta_t| \quad (5)$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD-error at time step t, and T is the length of the episode on level l. The intuition is that higher magnitude TD-errors indicate a greater discrepancy between expected and actual returns, suggesting more potential for learning.

Once scores are assigned, PLR employs a rank-based prioritization scheme for sampling levels. The probability of sampling a level is inversely proportional to its rank, given by:

$$P(l) \propto \frac{1}{\mathrm{rank}(l)^\alpha} \quad (6)$$

where α is a hyperparameter controlling the degree of prioritization. To prevent scores from becoming stale, PLR incorporates a staleness factor that increases the sampling probability for levels that haven't been played recently. This is achieved by modifying the sampling probability:

$$P(l) \propto \frac{1}{\mathrm{rank}(l)^\alpha} + \beta \cdot \mathrm{staleness}(l) \quad (7)$$

where β is a hyperparameter balancing the importance of staleness, and staleness(l) is a measure of how long it has been since level l was last played. This combination of prioritization and staleness awareness allows PLR to adaptively focus on high-potential levels while still maintaining exploration of the level space.

3.4 MuJoCo Environments

MuJoCo (Multi-Joint dynamics with Contact) is a high-performance physics engine that has become a cornerstone in robotics and reinforcement learning research for its precise simulation capabilities. The environments provided in Gymnasium (Towers et al., 2024) offer a standardized implementation of MuJoCo-based tasks, featuring accurate physical dynamics and continuous control challenges. These environments excel in simulating complex dynamic systems with realistic contact dynamics and joint interactions, making them particularly valuable for developing and evaluating advanced control algorithms in continuous action spaces. The simulation framework incorporates sophisticated physics modeling, including friction, contact forces, and multi-joint dynamics, providing researchers with reliable benchmarks for testing reinforcement learning approaches.

4 METHOD

4.1 Automatic Curriculum Learning Methods Included

Herein, we outline the algorithms included in our benchmark study. We have picked algorithms representing a different variety of approaches to UED: from adversarial frameworks, including PAIRED and REPAIRED, to evolutionary methods including ACCEL and structured manifold sampling with CLUTR, and foundational techniques including DR, PLR, and Robust PLR. In this paper, we include the current algorithms that benchmark a wide spectrum of UED strategies, providing insight into their relative strengths and weaknesses and helping retain the most efficient strategies in training robust, generalizable policies.

4.1.1 REPAIRED

Replay-Enhanced PAIRED (REPAIRED) (Jiang et al., 2021b) extends the PAIRED approach by incorporating a replay mechanism that focuses on levels causing the highest regret. Levels generated by

PAIRED is only used once and then discarded. REPAIRED addresses this inefficiency and instability by reusing and concentrating on past environment levels with the highest regret. Adhering to the principles of robust PLR (introduced in Section 4.1.5), REPAIRED avoids training agents on newly generated levels. It strategically utilizes a curated selection of previously encountered levels.

4.1.2 ACCEL

Adversarially Compounding Complexity by Editing Levels (ACCEL) (Parker-Holder et al., 2022) introduces an iterative approach to curriculum generation. It focuses on editing existing high-regret environments, progressively increasing their complexity through evolutionary methods. The key insight of ACCEL is that if a particular environment level challenges the agent's current capabilities, then strategically edited versions of this level should continue to push the boundaries of the agent's competence. This approach leads to the efficient discovery of increasingly challenging scenarios. Importantly, ACCEL aligns with the principles of robust PLR and REPAIRED by not training on newly generated levels, maintaining a clear separation between environment generation and agent training.

4.1.3 CLUTR

Curriculum Learning Using Task Representations (CLUTR) (Azad et al., 2023) addresses a key challenge in the PAIRED framework: the simultaneous learning of a task space and curriculum, which can lead to training instability. CLUTR innovates by pretraining a variational autoencoder (VAE) (Kingma and Welling, 2013) to represent the environment's parameter space. This approach enables concurrent generation of all environment parameters, fostering more structured and coherent task creation. By mapping similar tasks to proximal points in latent space, CLUTR facilitates smoother curriculum progression. The VAE's latent representation provides a stable, lower-dimensional space for the teacher to explore, simplifying task generation and mitigating the instability issues associated with simultaneous learning.

4.1.4 DR and PLR

Domain Randomization (DR) (Tobin et al., 2017) and Prioritized Level Replay (PLR) (Jiang et al., 2021a) serve as baselines in our study, chosen for their simplicity and proven effectiveness. DR enhances agent robustness by training across a diverse range of randomly generated scenarios. PLR refines this approach by prioritizing levels based on their estimated TD (temporal difference) error, focusing the agent's learning on the most informative experiences. These methods provide valuable benchmarks against which more advanced curriculum learning algorithms can be compared.

4.1.5 Robust PLR (PLR$\perp$)

Robust PLR introduced in the REPAIRED paper (Jiang et al., 2021b) is an enhanced iteration of the original Prioritized Level Replay (PLR) that offers guaranteed theoretical robustness. It diverges from the original PLR in two crucial aspects: firstly, it employs regret prioritization for level sampling, replacing the L1 value-loss metric used in the original PLR. Secondly, it constrains the agent's training exclusively to sampled levels, avoiding training on newly generated environments. These strategic modifications yield significant improvements in training stability and theoretical guarantees. At equilibrium, this approach ensures the convergence of the resulting policy to a minimax regret policy. This refinement addresses key limitations of the original PLR, offering a more principled approach to level sampling.

4.1.6 PAIRED

Protagonist Antagonist Induced Regret Environment Design (PAIRED) (Dennis et al., 2020) is a pioneering algorithm in Unsupervised Environment Design (UED), introducing a novel regret-based adversarial framework for automatic environment generation. PAIRED employs a tripartite system comprising a teacher, a student agent, and an antagonist agent. The teacher is trained using regret, defined as the difference between the rewards of the student and antagonist agents. This approach motivates the teacher to create levels that are challenging yet solvable for the student agent. If a level is unsolvable, both the student and antagonist would receive low rewards, providing a natural balance. PAIRED's innovative structure enables the generation of a curriculum that continuously adapts to the student agent's improving capabilities, fostering efficient learning and generalization.

4.2 Extended MuJoCo Environments

We extend environments based on the MuJoCo physics engine (Todorov et al., 2012) to create underspecified partially observable Markov decision processes (UPOMDPs) by introducing adjustable environment parameters. The specific level parameters subject to modification are detailed in Section 5. Our environments differ significantly from those previously used to benchmark UED algorithms in two key

Table 1: Comparison of Environment Generation Algorithms.

Algorithm	Env Generator	Uses PLR
DR	Fixed distribution	No
PLR	Fixed distribution	L1 value-loss sampler
PLR⊥	Fixed distribution	Regret-based sampler
ACCEL	Fixed distribution + Evolution Editing	Regret-based sampler
PAIRED	PAIRED	No
REPAIRED	PAIRED	Regret-based sampler
CLUTR	PAIRED + Pretrained VAE	No

aspects: firstly, the environments in this work feature a continuous parameter space, allowing for fine-grained variations in environment dynamics. Secondly, alterations in these parameters—such as gravity or motor gear ratios—directly influence the state transition function, fundamentally changing the environment's dynamic and the optimal policy, rather than merely increasing state complexity.

These MuJoCo-based UPOMDPs more closely approximate the challenges inherent in sim-to-real transfer learning scenarios. In such cases, the discrepancies between simulated training environments and real-world conditions can significantly impact the performance of learned policies. By testing UED algorithms on these new UPOMDP environments, we aim to evaluate their capability in constructing policies that are robust to variations in environment dynamics, potentially facilitating more effective sim-to-real transfer. This approach provides a testbed for assessing the adaptability and generalization capabilities of regret-based curriculum algorithms in dynamic, physics-based environments.

5 EXPERIMENT

To evaluate the effectiveness of UED methods in complex, continuous control tasks, we extended our experiments to include six MuJoCo-based environments from Gymnasium (Towers et al., 2024): `HalfCheetah-v5`, `Ant-v5`, `Swimmer-v5`, `Hopper-v5`, `Walker2d-v5`, and `Humanoid-v5`. These environments offer diverse locomotion challenges with continuous action and state spaces, closely resembling real-world robotic control tasks.

For all environments except `Swimmer-v5`, we made the motor gear ratio and gravity adjustable. In `Swimmer-v5`, we replaced gravity with viscosity, reflecting its fluid medium simulation, since gravity plays a less significant role in this environment. The ranges for these parameters were set to within ±30% of their original values during training, allowing for significant task difficulty variation while maintaining physical plausibility.

In our study, we employ Proximal Policy Optimization (PPO) (Schulman et al., 2017) to optimize the environment generator (teacher agent), following the approach of previous UED methods. For the student agents, we utilize Soft Actor-Critic (SAC) (Haarnoja et al., 2018) method, given its widespread adoption and superior performance in continuous action spaces compared to other algorithms.

To ensure statistical robustness and provide a more accurate assessment of performance, we conduct training across five random seeds for each environment and algorithm combination. The results presented in our analysis reflect the average scores obtained from these multiple runs. The specific training parameters are detailed in Table 2.

Table 2: Hyperparameters and Configuration.

Parameter	Value
Number of Steps	5,760,000
Number of Processes	16
Model	2 hidden layers with 256 units each
Level Buffer Size	10,000
Replay Buffer Size	1e6
Learning Rate	3e-4
Update Interval	16000 step
Update Times	16000
Batch Size	256
Observation Normalization	True
Reward Normalization	True

5.1 Convergence of Each Algorithm

We initially demonstrate the performance of each algorithm on the unmodified original environment in Figure 1, with scores evaluated during training at 16,000-step intervals and averaged over 10 rollouts. The results indicate that agents across all algorithms successfully learn to achieve satisfactory scores, with ACCEL demonstrating the highest average final score, though not significantly. Regarding consistency and robustness, DR, PLR, PAIRED, and CLUTR exhibit wider confidence intervals throughout training, possibly due to continuous training on newly generated levels (even PLR, despite its level replay buffer). In contrast, PLR⊥, which addresses this issue with a more comprehensive level replay strategy, displays a significantly narrower confidence in-

terval, suggesting improved stability in training and more consistent performance across different runs.

5.2 Transfer of Knowledge to Challenging Scenario

To assess the performance of policies trained with curriculum learning in unseen environments, we designed increasingly challenging scenarios by gradually modifying key parameters. For most environments, we decreased the motor gear ratio to reduce joint movement precision and torque, while increasing gravity to impede body movement and increase the likelihood of falling. These modifications effectively increased the difficulty for all environments except `Swimmer-v5`. For `Swimmer-v5` specifically, we increased viscosity instead of gravity to hinder its movement, as gravity changes did not significantly affect its performance. We created three levels of environmental difficulty, detailed in Table 3, with Level 1 representing the easiest variation (parameters slightly beyond the training range) and Level 3 being the most challenging. This approach evaluates the robustness and generalization capabilities of policies when faced with progressively more demanding and unfamiliar conditions. Tables 4, 5, and 6 present the results for each difficulty level, with each value representing an average from 50 rollouts generated by 5 models (10 rollouts per model) trained with different random seeds. For ease of analysis, we normalized the scores so that the highest average score for each unmodified environment is set to 1.0.

Table 3: Parameter modifications for different difficulty levels. The percentages indicate changes applied to the original values.

Level	Gravity/Viscosity Change	Motor Gear Ratio Change
1	+0%	−22%
2	+11%	−45%
3	+33%	−45%

Table 4: Comparison of different methods across various level 1 environments.

	`Cheet`	`Ant`	`Swim`	`Hop`	`Walk`	`Hum`
DR	0.91	0.92	0.93	0.92	0.93	0.92
PLR	0.92	0.91	0.92	**0.95**	0.92	0.91
PLR⊥	**0.95**	0.90	**0.95**	0.91	0.91	0.89
ACCEL	0.93	**0.95**	0.92	0.93	**0.95**	**0.94**
PAIRED	0.89	0.88	0.90	0.89	0.90	0.88
REPAIRED	0.90	0.89	0.91	0.90	0.91	0.90
CLUTR	0.87	0.86	0.88	0.87	0.88	0.86

Table 5: Comparison of different methods across various level 2 environments.

	`Cheet`	`Ant`	`Swim`	`Hop`	`Walk`	`Hum`
DR	0.59	0.79	0.82	0.68	0.68	0.57
PLR	0.61	0.80	0.83	0.70	0.70	0.58
PLR⊥	0.63	0.82	**0.86**	0.71	0.72	0.59
ACCEL	**0.65**	**0.83**	0.85	0.72	**0.74**	**0.60**
PAIRED	0.60	0.78	0.81	0.67	0.67	0.56
REPAIRED	0.62	0.81	0.84	**0.73**	0.71	0.58
CLUTR	0.55	0.75	0.78	0.64	0.62	0.55

Table 6: Comparison of different methods across highly challenging level 3 environments.

	`Cheet`	`Ant`	`Swim`	`Hop`	`Walk`	`Hum`
DR	0.28	0.32	0.35	0.30	0.29	0.22
PLR	0.31	0.36	0.38	0.33	0.32	0.24
PLR⊥	0.35	0.40	**0.45**	0.37	0.36	0.27
ACCEL	**0.38**	**0.43**	0.42	0.39	**0.40**	**0.30**
PAIRED	0.25	0.29	0.32	0.27	0.26	0.19
REPAIRED	0.33	0.38	0.40	**0.41**	0.34	0.25
CLUTR	0.20	0.23	0.25	0.21	0.20	0.14

For difficulty level 1, all methods maintain good performance, most of them are able to maintain a performance of above 90% of the original scores. Only CLUTR drops below 0.90 for all environments. We think the reason is that the original scores achieved by these methods are lower, not due to unable to handle distribution shift.

For difficulty level 2 and 3, ACCEL consistently demonstrates superior performance in most scenarios, particularly excelling in the `HalfCheetah-v5`, `Ant-v5`, `Walker2d-v5`, and `Humanoid-v5` environments. This superior performance can be attributed to ACCEL's unique approach of only editing part of the parameters at the capacity frontier of the current agent. By doing so, ACCEL can more efficiently find other environment parameters that challenge the agent at its current capacity, leading to better environment generation efficiency. This targeted approach allows ACCEL to generate diverse and challenging tasks that promote robust learning more effectively than other methods. Interestingly, PLR⊥ shows the best performance in the `Swimmer-v5` environment for both difficulty levels, indicating its effectiveness in this specific task. REPAIRED, on the other hand, consistently outperforms other methods in the `Hopper-v5` environment, highlighting its strength in some particular domain.

A notable observation is the significant drop in performance scores between level 2 and level 3 environments across all methods and tasks, underscoring

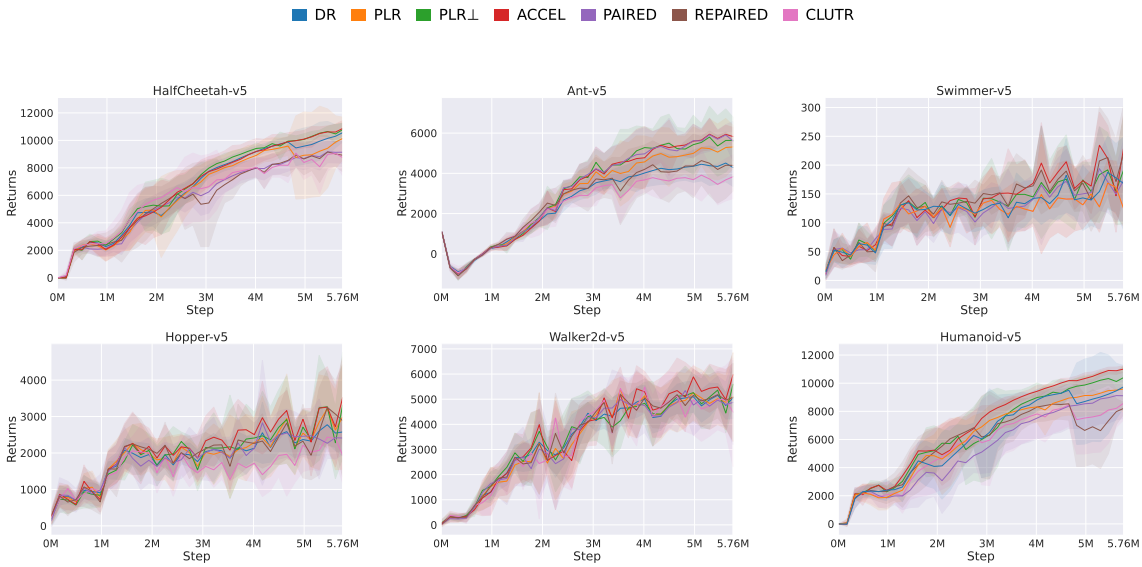

Figure 1: Learning Curves.

the substantially increased difficulty and the challenge of maintaining high performance as task complexity grows. Despite this overall decrease, the relative performance rankings of different methods remain largely consistent, with ACCEL, PLR⊥, and REPAIRED maintaining their positions as top performers in their respective strong suits. ACCEL's ability to maintain its lead even in more challenging environments further supports the effectiveness of its parameter editing strategy in finding the optimal balance between challenge and learnability. It's worth noting that the CLUTR method consistently shows lower performance compared to other approaches in both difficulty levels, suggesting that while promising, it may require further refinement (e.g., in VAE training) to compete with more established UED methods in these specific environments. Overall, the significant performance drop in level 3 environments indicates that there is still substantial room for improvement in the robustness and generalization capabilities of current UED approaches.

5.3 Level Parameter Trends

We illustrate the evolution of level parameters throughout training in Fig. 2. To enhance the clarity of these learning curves and mitigate data noise, we applied the Savitzky-Golay filter with a window size of 29 and a polynomial order of 3. Our observations reveal no significant differences among the trends of the various algorithms, including ACCEL, despite its superior performance relative to other algorithms. Consistent with findings from previous studies (Parker-Holder et al., 2022), the mean difficulty for PAIRED and PLR algorithms does not exhibit an apparent trend. Interestingly, our evaluation shows that this lack of significant change extends to the ACCEL algorithm as well in these MuJoCo environments.

Two potential explanations emerge for this observation. First, in MuJoCo environments, the capacity frontier may not consistently decrease or increase, resulting in the absence of obvious trends in environment parameters. Alternatively, these algorithms may still have room for improvement in generating a meaningful curriculum. This suggests that while these methods demonstrate effectiveness in performance, their approach to curriculum generation might benefit from further refinement to produce more discernible parameter trends over the course of training.

6 DISCUSSION & CONCLUSION

Recent years have seen the proposal of various automatic curriculum learning methods, yet their effective application across a broad range of environments remains insufficiently tested. Our benchmarking of these methods on MuJoCo environments, featuring continuous action spaces more akin to real-world robotic settings, reveals important insights into their performance and limitations.

Our results demonstrate that while these automatic curriculum learning methods generally outperform the baselines (i.e., DR, PLR), the performance gains are less significant than initially anticipated. This finding underscores the challenges of applying these methods to more complex, continuous environ-

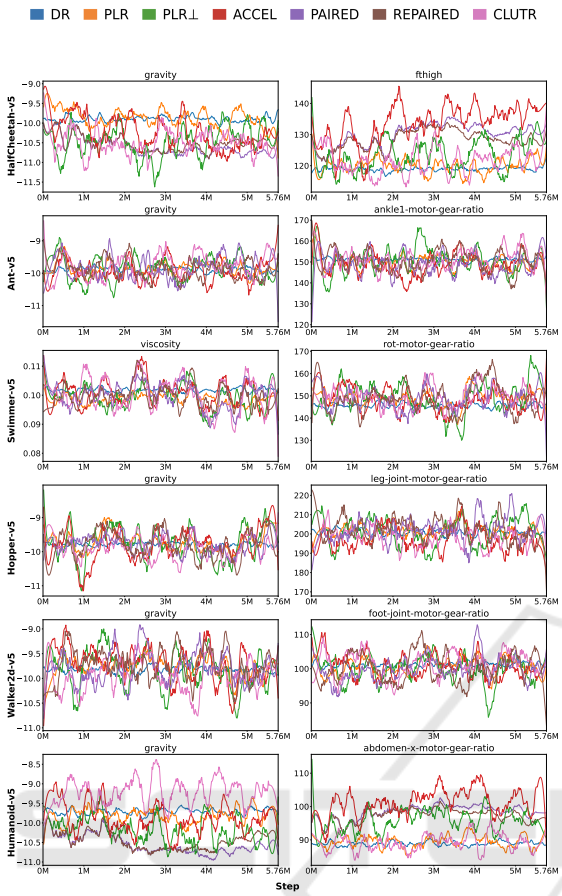

Figure 2: Level Parameters Throughout Training.

ments that closely resemble real-world scenarios.

Among the tested methods, ACCEL emerges as a promising direction for improving automatic curriculum learning. Its efficiency in finding new challenging environment settings stands out, suggesting that its evolutionary-based approach to incrementally increasing environment complexity is particularly effective. This targeted approach allows ACCEL to generate diverse and challenging tasks that promote robust learning more effectively than other methods.

PLR⊥ also shows promise, incorporating two key modifications: not training on newly generated parameters and using a regret-based sampler. These changes have yielded encouraging results, indicating their potential for integration into future methods.

Our study reveals that these methods are less effective for continuous parameter spaces compared to discrete ones. The vast diversity of parameter combinations in continuous spaces poses a significant challenge for efficient exploration, highlighting the need for more sophisticated exploration strategies tailored to continuous domains.

A notable issue with these curriculum learning algorithms is the substantial increase in environment interactions required for agents to achieve robustness. For the MuJoCo environments tested, this overhead was 5.76 times the original value. While generalizing to unseen scenarios is inherently more challenging, this high overhead presents a significant barrier to practical application. Developing methods that can achieve generalization with lower sample complexity could be a fruitful direction for future research.

The limitations exposed by our study underscore the importance of expanding the range of environments used to test these algorithms. In particular, there is a need to include more challenging tasks, especially those that are initially unsolvable, to evaluate whether these methods can effectively break down complex problems into learnable components.

In conclusion, while regret-based automatic curriculum learning methods show promise in continuous control tasks, there remains significant room for improvement. Future work could focus on enhancing the efficiency of these methods in continuous parameter spaces, reducing the sample complexity required for robust learning, and demonstrating their effectiveness in solving previously intractable problems. As we continue to push the boundaries of reinforcement learning towards more realistic and complex scenarios, the development of more sophisticated and efficient curriculum learning methods will play a crucial role in advancing the field.

SCIENCE AND TECHNOLOGY PUBLICATIONS

ACKNOWLEDGEMENTS

The authors acknowledge the use of the AI assistant Claude 3.5 Sonnet, developed by Anthropic, PBC, to enhance the readability and clarity of this manuscript. The content and scientific contributions remain the original work of the authors.

REFERENCES

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38.

Azad, A. S., Gur, I., Emhoff, J., Alexis, N., Faust, A., Abbeel, P., and Stoica, I. (2023). Clutr: Curriculum

learning via unsupervised task representation learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *PMLR*, Honolulu, Hawaii, USA. PMLR.

Baranes, A. and Oudeyer, P.-Y. (2009). R-iac: robust intrinsically motivated exploration and active learning. *IEEE Transactions on Autonomous Mental Development*, 1(3):155–169.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Benjamins, C., Eimer, T., Schubert, F., Biedenkapp, A., Rosenhahn, B., Hutter, F., and Lindauer, M. (2021). Carl: A benchmark for contextual and adaptive reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. (2019a). Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2019b). Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR.

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. In *Advances in Neural Information Processing Systems*, volume 33.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303.

Florensa, C., Held, D., Geng, X., and Abbeel, P. (2018). Automatic goal generation for reinforcement learning agents. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1515–1528.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. (2017). Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pages 482–495.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. (2017). Automated curriculum learning for neural networks. In *International conference on machine learning*, pages 1311–1320. PMLR.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning*, pages 1861–1870.

Held, D., Geng, X., Florensa, C., and Abbeel, P. (2018). Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pages 1515–1528. PMLR.

Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J., Grefenstette, E., and Rocktäschel, T. (2021a). Prioritized level replay.

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

Juliani, A., Khalifa, A., Berges, V.-P., Harper, J., Teng, E., Henry, H., Crespi, A., Togelius, J., and Lange, D. (2019). Obstacle tower: A generalization challenge in vision, control, and planning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 2684–2691. IJCAI.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. (2021). A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*.

Klink, P., D'Eramo, C., Peters, J., and Pajarinen, J. (2020). Self-paced deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc.

Klink, P., Yang, H., D'Eramo, C., Pajarinen, J., and Peters, J. (2022). Curriculum reinforcement learning via constrained optimal transport. In *Proceedings of the 39th International Conference on Machine Learning*, pages 11325–11344. PMLR.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29.

Li, Y. (2018). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562.

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.

Moulin-Frier, C., Nguyen, S. M., and Oudeyer, P.-Y. (2014). Self-organization of early vocal development in infants and machines: The role of intrinsic motivation. *Frontiers in Psychology*, 5:1065.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50.

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvári, C., Singh, S., Van Roy, B., Sutton, R. S., Silver, D., and van Hasselt, H. (2020). Behaviour suite for reinforcement learning. In *8th International Con-*

ference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia.

Parker-Holder, J., Jiang, M., Dennis, M., Samvelyan, M., Foerster, J., Grefenstette, E., and Rocktäschel, T. (2022). Evolving curricula with regret-based environment design.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. (2019). Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*.

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y. (2019). Alp-gmm: Active learning of prioritized skills. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):3796–3803.

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y. (2020). Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853. PMLR.

Romac, C., Portelas, R., Hofmann, K., and Oudeyer, P.-Y. (2021). Teachmyagent: a benchmark for automatic curriculum learning in deep rl. *arXiv preprint arXiv:2103.09815*.

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

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

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

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7):1633–1685.

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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., et al. (2024). Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*.

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*.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS