Exploration by Exploitation: Curriculum Learning for Reinforcement Learning Agents through Competence-Based Curriculum Policy Search

Tabitha Edith Lee[*]
Mila & Université de Montréal

Nan Rosemary Ke[†]
Mila – Quebec AI Institute

Sarvesh Patil[†]
Carnegie Mellon University

Annya Dahmani[‡]
Univ. of California, Berkeley

Eunice Yiu[‡]
Univ. of California, Berkeley

Esra'a Saleh
Mila & Université de Montréal

Alison Gopnik
Univ. of California, Berkeley

Oliver Kroemer[§]
Carnegie Mellon University

Glen Berseth[§]
Mila & Université de Montréal

Abstract

We present CURATE, an algorithm for automatic curriculum learning for reinforcement learning agents to solve a difficult target task distribution with sparse rewards. Initially, due to fundamental exploration challenges without informed priors or specialized algorithms, agents may be unable to consistently receive rewards, leading to inefficient learning. Through "exploration by exploitation," CURATE dynamically scales the task difficulty to match the agent's current competence. By exploiting its current capabilities that were learned in easier tasks, the agent improves its exploration in more difficult tasks. While training the agent, CURATE conducts policy search in the curriculum space to learn a task distribution for the agent corresponding to the easiest unsolved tasks. As the agent's mastery grows, the learned curriculum adapts correspondingly in an approximately easiest-to-hardest fashion, efficiently culminating in an agent that can solve the target tasks. Our experiments demonstrate that the curricula learned by CURATE achieve greater sample efficiency for solving the target tasks than state-of-the-art algorithms and most baselines. Although an incremental, easiest-to-hardest curriculum was more performant for one-dimensional curricula, CURATE shows promising performance for two-dimensional curricula where the optimal task sequencing is not obvious.

1 Introduction

The advent of reinforcement learning (RL) [1, 2] with deep neural networks [3, 4] has ushered in a promising era of impressive milestones in sequential decision making for deep RL [5–12]. Yet, without models, inductive biases, expert trajectories, or dense rewards, model-free deep RL algorithms are markedly sample inefficient due to fundamental challenges with exploration. Initially, the RL agent's actions are essentially random, requiring many interactions with the environment before the agent can learn useful behaviors that accrue rewards. However, agent learning can be structured through *curriculum learning* [13], which specifies how training data should be sequenced in order to achieve two broad aims [14]: to guide training (i.e., increase learning sample efficiency) and to denoise training (i.e., improve learning robustness and generalization through focus on high-

[*]Work also conducted while at Carnegie Mellon. Correspondence: `tabitha-edith.lee@mila.quebec`
[†]Equal contribution. [‡]Equal contribution. [§]Equal supervision.

First Exploration in AI Today Workshop at ICML (EXAIT at ICML 2025).

confidence training data regimes). Indeed, the effectiveness of introducing concepts in an orderly, structured fashion also has support from cognitive neuroscience [15] and effective pedagogy such as problem-based learning and assisted discovery learning [16–19], where knowledge arises from both learner self-discovery of innovations and timely instructor interventions.

For reinforcement learning, the advantages of improving sample efficiency, generalization, and exploration through a curriculum are generally recognized [20–22]. Indeed, achieving the goal of *automatic curriculum learning* — automatically learning the optimal curricula for *any* domain — would have far-reaching implications for the field of reinforcement learning, leading to the *de facto* standard for training RL agents and the significant impact it would entail. However, an automated way of selecting the curriculum remains an open problem, as previous literature suggests that, in the words of Bengio et al. [13], "some curriculum strategies work better than others." Insight from evolutionary algorithms for open-ended learning [23, 24] suggest that innovations can arise in a nonlinear, spontaneous fashion. Conversely, for reformulating single-task RL as multi-task RL with a one-dimensional curriculum, it has been argued that solving tasks in an easy-to-hard fashion is optimal [25], but it is unclear how this insight extends to multiple dimensions. Therefore, it is not generally obvious in what sequence the tasks should be visited for a curriculum. In light of these questions, curricula are often constructed manually by human designers in an *ad hoc* fashion, leading to hand curricula that are tailor-made for specific domains but do not generalize to others.

To answer these questions, we introduce CURATE (**Cur**riculum **A**gent for **T**argeted **E**xploration, App. A), an automatic curriculum learning algorithm for training a model-free, on-policy reinforcement learning agent to solve a difficult target task distribution with sparse rewards. Our approach overcomes fundamental exploration challenges by conducting *exploration by exploitation*, as coined by Leibo et al. [26].[1] Specifically, CURATE adapts the difficulty of the training tasks to the agent's current capabilities, or *competence*, through curriculum policy search. Initially, the agent has not learned useful behaviors, so relatively easier tasks ensure that random exploration is (relatively more) viable. Then, as the agent's competence grows, more difficult training tasks are selected to match the current capabilities of the agent. In other words, the agent improves its ability to explore in more challenging tasks by exploiting its current capabilities that were gained from previous easier tasks. In this way, CURATE trains an RL agent through an approximately easiest-to-hardest progression, quickly training the agent to complete the target task distribution at the end of the curriculum.

2 Related work

Curriculum Learning for Reinforcement Learning As formalized by Bengio et al. [13], curriculum learning concerns how to meaningfully organize data for training machine learning models, including those used for reinforcement learning. In this section, we identify a few relevant works in curriculum learning for RL; please refer to Narvekar et al. [20], Portelas et al. [21], and Parker-Holder et al. [22] for comprehensive surveys. Graves et al. [27] introduce a general curriculum learning method based on a nonstationary multi-armed bandit algorithm. Wang et al. [23, 24] show that curricula can emerge from co-evolving environments and agents. Portelas et al. [28] introduce ALP-GMM, a Gaussian mixture model in the parameter space of the environment. Algorithms from the Unsupervised Environment Design [29] and Dual Curriculum Design [30] frameworks yield implicit curricula that emerge from unsupervised learning. For the case of reinforcement learning, Li et al. [25] proposed that, under certain assumptions, solving tasks from easiest to hardest is optimal. Our algorithm, CURATE, is most similar to Portelas et al. [28] and Li et al. [25]. CURATE also maintains a task distribution similar to ALP-GMM [28], but the curricula learned by CURATE are driven by seeking out the easiest set of unsolved tasks, leading to an approximately easiest-to-hardest curriculum that is similar to Li et al. [25] to maximize learning at the frontier of agent capability.

Unsupervised Environment Design and Dual Curriculum Design First introduced by Dennis et al. [29], the Unsupervised Environment Design (UED) paradigm provides a framework wherein parameters of an underspecified environment are varied by a teacher to produce distributions over environments for a student learner. This paradigm can support various teaching modes, such as domain randomization, minimax regret, or a "environment-generating adversary" [29] in the PAIRED

[1] Leibo et al. describes "exploration by exploitation" as a form of exploration in agents that continuously adapt to exploit their abilities in non-stationary environments [26]. In our work, our environments are not non-stationary, but the training distribution is made non-stationary by the learned curriculum policy.

algorithm. Jiang et al. [30] unifies the UED framework with prior work in replaying experiences with Prioritized Level Replay (PLR) [31] to form the Dual Curriculum Design (DCD) framework, wherein the student learns from either an environment-generating teacher (as in UED) or a teacher that selects past experiences to replay (as in PLR). In so doing, Jiang et al. introduced REPAIRED (replay-augmented PAIRED) and an extension of PLR, Robust PLR (also stylized as PLR$^{\perp}$), in which gradient updates only occur on replayed levels. Later, Parker-Holder et al. [32] introduced ACCEL, an evolutionary-based algorithm that randomly mutates levels starting from environments of minimal complexity. Other UED algorithms include MAESTRO [33], the work of Mediratta et al. [34] to stabilize PAIRED, and ReMiDi [35]. Our algorithm, CURATE, can be placed within the DCD framework by functioning as a teacher that offers levels that are at the leading edge of the student's competence as determined by feedback from the student through sample-based evaluations.

3 Preliminaries

3.1 Underspecified POMDPs

The agent learns within an Underspecified Partially Observable Markov Decision Processs (UPOMDP) framework [29]. The UPOMDP defines a distribution of Partially Observable Markov Decision Process (POMDP) tasks [36, 37] as determined by the selection of environment parameters. The UPOMDP is defined as follows:

$$\mathcal{M} = \langle \mathcal{A}, O, \Theta, \mathcal{S}^{\mathcal{M}}, \mathcal{T}^{\mathcal{M}}, \mathcal{I}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, \gamma \rangle \tag{1}$$

where $a \in \mathcal{A}$ is a set of actions, $o \in O$ is a set of observations, $\theta \in \Theta$ is a set of environment parameters, and γ is a discount factor for future rewards. The remainder of the UPOMDP tuple is defined with respect to the chosen environment parameters θ and are thus superscripted by $\mathcal{M}$. Therefore, for the POMDP $\mathcal{M}_\theta$ specified by θ, $s \in \mathcal{S}^{\mathcal{M}} : \mathcal{S} \times \Theta$ is a set of states from state space $\mathcal{S}$ that are not observable to the agent, $\mathcal{T}^{\mathcal{M}} : \mathcal{S} \times \mathcal{A} \times \Theta \rightarrow \mathcal{S}$ defines the transition function, $\mathcal{I}^{\mathcal{M}} : \mathcal{S} \times \Theta \rightarrow O$ is the observation (i.e., introspection) function, and $R \in \mathcal{R}^{\mathcal{M}} : \mathcal{S} \times \Theta \rightarrow \mathbb{R}$ is the reward function. A task is considered solved if its reward exceeds a solved threshold R_S. Through reinforcement learning, the agent learns a policy $\pi(a|o)$ from maximizing the objective $J(\pi)$, the expected sum of discounted rewards over trajectories τ with maximum timesteps T:

$$J(\pi) = \mathop{\mathbb{E}}_{\tau \sim \pi}[\sum_{i=0}^{T} \gamma^i R_i] \tag{2}$$

Assumptions In principle, the UPOMDP framework allows a temporally-varying trajectory of environment parameters, but in practice, we are concerned with environment parameters that only specify the construction of the initial scene via the underspecified state space $\mathcal{S}^{\mathcal{M}}$. In this view, our use of UPOMDPs is similar to Contextual MDPs [38–40]. We also assume that the environment parameter space Θ is disentangled, i.e., each dimension controls a single factor of variation.

3.2 Curriculum learning within UPOMDPs

Our problem addresses automatic curriculum learning within a UPOMDP for solving a particular target task distribution that is initially challenging or impossible for the agent to complete. Under the assumption that the environment parameters Θ are disentangled, curriculum learning can be conducted within the *axes of generalization* of the UPOMDP's *curriculum space*, i.e., the space of environment parameters Θ. In this work, tasks are ordered by difficulty within the axes of generalization, so that increasing θ generally yields more difficult tasks. Although these assumptions on the structure of the curriculum space are strong, they permit systematic evaluation of different curriculum learning algorithms in this work. The easiest tasks occur at $\min(\Theta)$, and the hardest tasks occur at $\max(\Theta)$. Specifically, the target task distribution is defined by environment parameters $\theta_t = \max(\Theta)$, and the POMDP that specifies this target task is therefore $\mathcal{M}_{\theta_t}$. The time-varying sequence of tasks that arises from a curriculum learning algorithm is called a curriculum $\mathcal{C}$.

4 Methodology

The goal of CURATE is to automatically learn a curriculum $\mathcal{C}$ for training a control policy π to complete a difficult target task distribution $\mathcal{M}_{\theta_t}$. To do this, CURATE conducts policy search using

sample-based evaluations to determine a curriculum policy that shapes the distribution of tasks used for training the agent. Intuitively, the curriculum policy learns a local distribution of unsolved tasks with high rewards that can be sampled for training. As the agent becomes more proficient and begins solving these tasks, this distribution shifts towards more difficult unsolved tasks, leading to an approximately easiest-to-hardest curriculum that has been shown to be optimal under certain conditions [25]. The curriculum policy $\pi_c(\theta; \mu_\theta, \Sigma_\theta)$ is represented by a Gaussian distribution over environment parameters θ with mean μ_θ and covariance Σ_θ. The training procedure is summarized in Sec. 4.1. Section 4.2 describes the curriculum update step, UPDATECURRICULUM.

4.1 Training RL policies with curriculum learning

This section summarizes the procedure for training the RL agent as described in Alg. 1 (App. B). This training procedure is designed to close the RL training loop around the target task distribution $\mathcal{M}_{\theta_t}$, such that RL training ends with only the minimum number of frames needed to solve $\mathcal{M}_{\theta_t}$. First, the control policy π is initialized randomly. Then, the curriculum policy π_c is initialized by the Gaussian distribution that approximates a uniform distribution over the curriculum space. Thereafter, the curriculum policy is updated prior to training with the (initially random) control policy π via UPDATECURRICULUM (Sec. 4.2). For each iteration in the training loop, tasks are sampled from the curriculum policy π_c by first sampling environment parameters θ_i, which are in turn transformed into tasks. Then, a trajectory dataset $\mathcal{D}$ is generated with mean training reward $R_\mathcal{D}$ from rollouts of π in the sampled tasks. Next, π is updated by the reinforcement learning algorithm by performing gradient updates of policy parameters using the dataset $\mathcal{D}$. Although in principle any on-policy reinforcement learning algorithm could be used, we use Proximal Policy Optimization (PPO) [41]. Following the policy update, the curriculum policy π_c is updated via UPDATECURRICULUM if the training reward $R_\mathcal{D}$ meets or exceeds the task solved threshold R_S. This trigger indicates that the agent has mastered proficiency in its current training distribution and is ready for more challenging tasks. Curriculum learning can also be triggered if a maximum number of timesteps since the last curriculum update has been reached. This prevents training stagnation if the current tasks are too difficult for the agent. Finally, the agent is evaluated on the target task distribution $\mathcal{M}_{\theta_t}$ to obtain a task evaluation reward R_t. We typically conduct stochastic, rather than deterministic, policy evaluation. If the agent solves the target task ($R_S \leq R_t$), then training concludes successfully. Otherwise, training continues while the number of maximum training frames has not been reached.

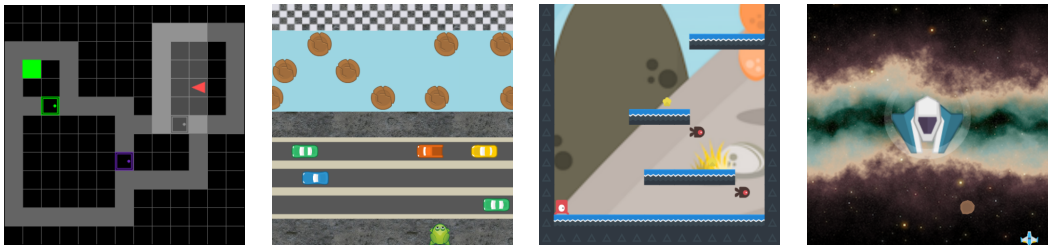
4.2 Updating the curriculum using curriculum policy search

This section summarizes UPDATECURRICULUM, the curriculum update procedure that is fully described in Alg. 2 (App. B). UPDATECURRICULUM is a nonlinear optimization within environmental parameter space to learn π_c by optimizing the curriculum objective $J_c(\pi_c)$:

$$J_c(\pi_c) = \mathop{\mathbb{E}}_{\theta_j \sim \pi_c, \mathcal{M}_{\theta_j} \sim \theta_j, R_j \sim \mathcal{M}_{\theta_j}(\tau \sim \pi)} [\nu_j], \quad \nu_j = \frac{R_j}{R_S} \mathbb{1}_{R_j < R_S} - \lambda_\theta ||\theta_j||_2 \tag{3}$$

where $\mathcal{M}_{\theta_j}$ is a task distribution sampled from π_c via parameters θ_j, R_j is the reward obtained by evaluating π on $\mathcal{M}_{\theta_j}$, ν_j is the curriculum reward, and λ_θ is a regularization hyperparameter.

This method conducts evaluations to assess the current proficiency of the agent in a sample-efficient manner without exhaustive search of the curriculum space. First, the initial parameter distribution for the curriculum policy π_c is provided as $(\mu_\theta, \Sigma_\theta)$. Then, for each of N_r rounds, the agent draws N_s parameter samples from the curriculum policy parameter distribution $(\mu_\theta, \Sigma_\theta)$. For each parameter sample θ_j, the corresponding task $\mathcal{M}_{\theta_j}$ is generated, and the agent is evaluated on this task to yield reward R_j. However, this reward is not used directly for the curriculum learning reward ν_j. Instead, it is assessed whether it meets or exceeds the threshold R_S, i.e., the task is solved. If so, the agent receives zero curriculum reward for this task, as the agent has mastered this task. Otherwise, the curriculum reward is first assessed as R_j/R_S. This reward signal induces the agent towards the easiest (i.e., highest return) tasks that have not yet been solved. Thereafter, the curriculum learning reward is regularized by $\lambda_\theta ||\theta_j||_2$ to become ν_j. This regularization addresses cases where samples consistently return zero reward (e.g., at the start of training, all tasks may be too difficult), leading to a small bias towards easier levels. Then, the parameter samples θ_j and curriculum reward ν_j are appended to buffers. These buffers are used by Relative Entropy Policy Search (REPS) [42] to yield an updated Gaussian distribution that maximizes the curriculum reward, subject to an information



(a) MiniGrid MultiRoom (b) Procgen Leaper (c) Procgen Climber (d) Procgen BossFight

Figure 1: The experimental domains investigated in our work. MiniGrid MultiRoom (a) a 1-dimensional curriculum space with field-of-view state and direction observations. Leaper (b), Climber (c), and BossFight (d) have 2-dimensional curriculum spaces with image observations.

Table 1: Description of each curriculum axis for each domain.

Domain	MultiRoom	Leaper	Climber	BossFight
Θ_1	Num. rooms (1-4)	Num. road lanes (0-3)	Num. platforms (1-10)	Round health (1-9)
Θ_2	n/a	Num. water lanes (0-3)	Enemy prob. % (0-20)	Num. rounds (1-5)

loss bound based on Kullback-Leibler divergence [43]. Lastly, the continuous curriculum parameters $(\mu_\theta, \Sigma_\theta)$ are discretized to yield the updated curriculum policy π_c. This process repeats iteratively N_r times before returning π_c at the conclusion of the curriculum update.

5 Experimental results

In this section, we evaluate CURATE against a variety of curriculum baselines, where performance is quantified by sample efficiency: how much training data is required to train an RL agent to complete the most difficult tasks within each experimental domain. In our experiments, we seek to answer two research questions. First, when compared to a variety of curriculum strategies, such as implicit and explicit curricula, how does CURATE compare when considering a one-dimensional curriculum space with a limited observation space (Q1)? Second, how does CURATE perform against curriculum baselines in two-dimensional curriculum spaces with high-dimensional observations (Q2)?

Experimental domains Figure 1 provides an overview of the experimental domains. All domains use discrete control and discrete environment parameters, varying by the dimensionality of the curriculum space and the observations. Table 1 describes the domain curriculum spaces. More information about the experimental domains can be found in Sec. D.

MiniGrid MultiRoom [44] is a one-dimensional curriculum space that is explored for question Q1. The specific domain is a reimplementation of MultiRoom-Random-N4 [31], except with the typical MiniGrid observation space of field-of-view state and agent direction.

The Procgen Curriculum Suite (Leaper, Climber, BossFight) contains two-dimensional curriculum spaces with image observations for question Q2. The three games in the suite are adaptations of the same games first introduced by Cobbe et al. [45] to structure the levels within each game into a curriculum space and allow for changing the initial state based on the environment parameters. For this work, we use the easy distribution mode for all three games.

Train and test procedure All methods use PPO [41] with the Adam optimizer [46] for training the control policy π. The optimizer runs continuously and is not reset during training. After every control policy update, the agent is evaluated on the target task distribution $\mathcal{M}_{\theta_t}$. If the return achieved in $\mathcal{M}_{\theta_t}$ is at least R_S, training ends. Otherwise, training continues up to a maximum allowable frames.

Baselines We assess CURATE against a variety of baselines, which can be broadly categorized into either *explicit* or *implicit* curriculum learning algorithms. Explicit curricula structure the sequencing of training tasks externally, often with knowledge of the environment parameters. In contrast, implicit curricula emerge through self-discovery by interacting with the tasks based on learning objectives that do not necessarily access the environment parameters or other task-based schedules.

Table 2: Statistics for sample efficiency for MiniGrid MultiRoom in terms of frames required to either solve $\mathcal{M}_{\theta_t}$ or the maximum allowable frames (50 million). C. Type stands for curriculum type. 10 trials are evaluated for each approach. Mean Frames are shown with $\pm$ one standard deviation. Median Frames are shown with $\pm$ one interquartile range (IQR). Trials that do not solve the task still count towards summary statistics and are assessed the maximum allowable frames.

Approach	C. Type	Success Rate	Mean Frames ($\times 10^6$)	Median Frames ($\times 10^6$)
CURATE (ours)	Explicit	100%	6.114 ± 1.551	5.737 ± 2.006
PLR$^\perp$	Implicit	100%	18.280 ± 1.994	18.156 ± 3.190
ACCEL	Implicit	90%	36.905 ± 7.609	34.265 ± 12.490
Dom. Rand.	Random	100%	9.418 ± 1.806	9.136 ± 2.447
Incr. Curr.	Explicit	100%	4.750 ± 0.608	4.663 ± 0.726
Target	None	0%	50.000 ± 0.000	50.000 ± 0.000

Our implicit curriculum baselines are Robust PLR (PLR$^\perp$) [31] and ACCEL [30]. PLR$^\perp$ focuses on student replay of levels, extending PLR [31] by only updating the agent on replayed levels. ACCEL [30] randomly mutates levels replayed by the student and starts from the easiest set of tasks.

Our curriculum baselines without learning are Domain Randomization (DR), Incremental Curriculum (IC), and Target (NC). DR represents a random curriculum. IC is an explicit baseline that represents a hand-designed, easiest-to-hardest curriculum that incrementally increases the task difficulty. Please refer to App. C for how IC is algorithmically generated. For less complex domains, IC can also be considered a pseudo-oracle and an approximation of ground truth. Lastly, NC represents only training on the target tasks without a curriculum.

5.1 Q1: One-dimensional curricula: MiniGrid MultiRoom

MiniGrid MultiRoom requires the agent to master grid-based navigation within the MiniGrid domain [44]. In this domain, tasks consist of mazes composed of random numbers of rooms linked together, from 1 room to 4 rooms. Each task is specified by the environmental parameter $\theta \in \{1, 2, 3, 4\}$ that specifies the number of rooms. The agent must navigate from the starting room to the goal, which is always contained in the last room. Therefore, the target task distribution $\mathcal{M}_{\theta_t}$ requires the agent to solve a distribution of mazes with 4 rooms. MultiRoom is a sparse reward domain; the agent receives a time-discounted reward only upon solving a task. The task solved threshold R_S is 0.7.

Results Results for MiniGrid MultiRoom are shown in Tab. 2, which provides the summary statistics for each approach. Please see App. E for more results, including a visualization of summary statistics (Fig. 5) and representative curricula for each approach (Fig. 6).

In general, CURATE outperforms all approaches except for IC, which can be viewed as a ground truth curriculum in this domain. The performance gap between CURATE and the DCD algorithms, PLR$^\perp$ and ACCEL, is relatively large. Although these algorithms yield a greater increase in training agent return, the increase in target task return is gradual. The implicit currricula that are learned appear to stagnate and reach equilibrium after the training return can no longer be maximized. DR provides a stronger performance than the DCD algorithms, as random curriculum exploration is viable given the relatively bounded nature of Θ in this domain. Lastly, NC represents the performance without using a curriculum. Overall, performance is markedly poor: no trials were successful. The target task distribution is too difficult to solve directly due to the exploration problem that uninformed agents face. CURATE addresses this exploration problem by dynamically changing to simpler tasks, leading to success early that can be bootstrapped into solving harder tasks.

5.2 Q2: Two-dimensional curricula: Procgen Curriculum Suite

The Procgen Curriculum Suite presents more challenging domains for curriculum learning. Curriculum spaces are two-dimensional, and the agent receives an image-based observation of the game. The target task distribution $\mathcal{M}_{\theta_t}$ for each Procgen game are as follows: for Leaper, tasks with 3 road lanes and 3 water lanes; for Climber, tasks with 10 platforms and 20% enemy spawn probability; and for BossFight, tasks with 9 health per round and 5 rounds. The task solved threshold R_S is either 8 (Leaper) or 10 (Climber, BossFight). Leaper is a sparse reward domain, where the agent receives 10

Table 3: Results for the Procgen Curriculum Suite in terms of frames required to either solve $\mathcal{M}_{\theta_t}$ or the maximum allowable frames. C. Type stands for curriculum type. 1 trial is evaluated for each approach. Frames are listed in ($\times 10^6$). For Success, ✓ means $\mathcal{M}_{\theta_t}$ was solved, and X otherwise.

		Leaper		Climber		BossFight	
Approach	C. Type	Frames	Success	Frames	Success	Frames	Success
CURATE (ours)	Explicit	9.372	✓	22.544	✓	40.108	✓
PLR$^{\perp}$	Implicit	99.975	X	99.942	X	99.877	X
ACCEL	Implicit	99.975	X	99.877	X	99.615	X
Dom. Rand.	Random	17.629	✓	32.113	✓	64.225	✓
Incr. Curr.	Explicit	12.354	✓	24.707	✓	53.740	✓
Target	None	99.975	X	23.855	✓	86.770	✓

reward only when solving a task. In contrast, Climber and BossFight are less sparse, offering small rewards throughout the task in addition to a larger reward when the task is solved.

Results Results for the Procgen Curriculum Suite are presented in Tab. 3. For visualizations, please see App. F. In these more complex domains, CURATE continues to outperform DR, PLR$^{\perp}$, and ACCEL. Notably, CURATE now shows promising performance against IC, although the margin is relatively narrow for Leaper and Climber. We hypothesize that multidimensional curriculum spaces offer opportunities for CURATE to forge the path of least learning through the curriculum based on the agent's competence. This approach contrasts with IC, which approximates the shortest path curriculum but may not be optimal in general. Although NC is unsuccessful for Leaper, this approach is viable for Climber and BossFight due to these games having less sparse rewards. Nevertheless, we see that CURATE usually offers greater sample efficiency than not using a curriculum, although the difference is small with Climber. Although our Procgen experiments are only for one trial, we believe that our results present a compelling proof-of-concept that warrants further study.

6 Conclusion

We present CURATE, an automatic curriculum learning approach for training a model-free, on-policy reinforcement learning agent to complete a difficult target task distribution with sparse rewards. CURATE navigates a curriculum through policy search in the curriculum space to establish the best task distribution that matches the agent's current competence. In so doing, CURATE's "exploration by exploitation" approach addresses fundamental exploration challenges through curriculum learning. Moreover, CURATE is effective in discontinuous curriculum spaces without requiring optimal initializations or starting in the easiest tasks. Initial results demonstrate that CURATE outperforms recent state-of-the-art DCD algorithms and most curriculum learning baselines. Although an incremental curriculum was more sample efficient in a one-dimensional curriculum for grid-based navigation, CURATE offers promising performance in two-dimensional curriculum spaces with selected Procgen games. We hypothesize that multidimensional curriculum spaces may showcase CURATE's ability to learn curricula where the best task sequencing is not obvious to specify *a priori*.

Limitations Although CURATE offers promising performance for automatically learning curricula, it is important to note CURATE's assumptions. CURATE requires that the curriculum space is defined, accessible, and structured in difficulty order along certain axes of task variation. These assumptions permit the curriculum policy search that powers CURATE. CURATE also assumes that task evaluations are not limited (e.g., if the target task distribution can only be attempted once). Currently, CURATE's runtime scales with task evaluations, but more efficient ways of exploring high-dimensional curriculum spaces can be investigated to limit the number of task evaluations.

Future work We will explore avenues to address CURATE's assumptions and assess CURATE against more algorithms, such as ALP-GMM [28], PAIRED [29], and Random Network Distillation [47]. We will also evaluate CURATE on continuous control domains, such as robotic control. We are also interested in how cognitive psychology insights into how humans approach curriculum learning [48–50] can inform extensions for CURATE, as well as how lessons learned from CURATE can inform new algorithms for Unsupervised Environment Design.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge the following support for our research: IVADO, including the Postdoctoral Funding Program; the Natural Sciences and Engineering Research Council of Canada (NSERC); the Canadian Institute for Advanced Research (CIFAR); and the National Institute of Standards and Technology of the United States (NIST, award no. 70NANB23H178). We also gratefully acknowledge compute resources provided by Mila and NVIDIA. We thank the Mila IDT team, including Fabrice Normandin and Olexa Bilaniuk, for their help and support. We also thank Minqi Jiang and Michael Dennis for their helpful insight into our work.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

[2] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521:436–444, 2015.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*, 2013.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-Level Control through Deep Reinforcement Learning. *Nature*, 518:529–533, 2015.

[7] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play. *Science*, 362(6419):1140–1144, 2018.

[8] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *Nature*, 575:350–354, 2019.

[9] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning Dexterous In-Hand Manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2019.

[10] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113*, 2019.

[11] Yuxi Li. Deep Reinforcement Learning. *arXiv preprint arXiv:1810.06339*, 2018.

[12] Shengbo Eben Li. Deep Reinforcement Learning. *Reinforcement Learning for Sequential Decision and Optimal Control*, pages 365–402, 2023.

[13] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum Learning. *International Conference on Machine Learning (ICML)*, 2009.

[14] Xin Wang, Yudong Chen, and Wenwu Zhu. A Survey on Curriculum Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9), 2022.

[15] Jeffrey L Elman. Learning and Development in Neural Networks: The Importance of Starting Small. *Cognition*, 48(1):71–99, 1993.

[16] Henk G. Schmidt, Sofie M. M. Loyens, Tamara Van Gog, and Fred Paas. Problem-Based Learning is Compatible with Human Cognitive Architecture: Commentary on Kirschner, Sweller, and Clark (2006). *Educational Psychologist*, 42(2):91–97, 2007.

[17] Sofie M. M. Loyens, Joshua Magda, and Remy M. J. P. Rikers. Self-Directed Learning in Problem-Based Learning and its Relationships with Self-Regulated Learning. *Educational Psychology Review*, 20:411–427, 2008.

[18] Louis Alfieri, Patricia J. Brooks, Naomi J. Aldrich, and Harriet R. Tenenbaum. Does Discovery-Based Instruction Enhance Learning? *Journal of Educational Psychology*, 103(1):1–18, 2011.

[19] Faisal Khan, Xiaojin Zhu, and Bilge Mutlu. How Do Humans Teach: On Curriculum Learning and Teaching Dimension. *Neural Information Processing Systems (NIPS)*, 2011.

[20] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[21] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic Curriculum Learning For Deep RL: A Short Survey. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[22] Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, Frank Hutter, and Marius Lindauer. Automated Reinforcement Learning (AutoRL): A Survey and Open Problems. *Journal of Artificial Intelligence Research*, 74:517–568, 2022.

[23] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. *arXiv preprint arXiv:1901.01753*, 2019.

[24] Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced POET: Open-Ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions. *International Conference on Machine Learning (ICML)*, 2020.

[25] Qiyang Li, Yuexiang Zhai, Yi Ma, and Sergey Levine. Understanding the Complexity Gains of Single-Task RL with a Curriculum. *International Conference on Machine Learning (ICML)*, 2023.

[26] Joel Z. Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the Emergence of Innovation from Social Interaction: A Manifesto for Multi-Agent Intelligence Research. *arXiv preprint arXiv:1903.00742*, 2019.

[27] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated Curriculum Learning for Neural Networks. *International Conference on Machine Learning (ICML)*, 2017.

[28] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher Algorithms for Curriculum Learning of Deep RL in Continuously Parameterized Environments. *Conference on Robot Learning (CoRL)*, 2020.

[29] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent Complexity and Zero-Shot Transfer via Unsupervised Environment Design. *Neural Information Processing Systems (NeurIPS)*, 2020.

[30] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-Guided Adversarial Environment Design. *Neural Information Processing Systems (NeurIPS)*, 2021.

[31] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized Level Replay. *International Conference on Machine Learning (ICML)*, 2021.

[32] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving Curricula with Regret-Based Environment Design. *International Conference on Machine Learning (ICML)*, 2022.

[33] Mikayel Samvelyan, Akbir Khan, Michael Dennis, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, Roberta Raileanu, and Tim Rocktäschel. MAESTRO: Open-Ended Environment Design for Multi-Agent Reinforcement Learning. *International Conference on Learning Representations (ICLR)*, 2023.

[34] Ishita Mediratta, Minqi Jiang, Jack Parker-Holder, Michael Dennis, Eugene Vinitsky, and Tim Rocktäschel. Stabilizing Unsupervised Environment Design with a Learned Adversary. *Conference on Lifelong Learning Agents (CoLLAs)*, 2023.

[35] Michael Beukman, Samuel Coward, Michael Matthews, Mattie Fellows, Minqi Jiang, Michael Dennis, and Jakob Foerster. Refining Minimax Regret for Unsupervised Environment Design. *International Conference on Machine Learning (ICML)*, 2024.

[36] K. J. Åström. Optimal Control of Markov Processes with Incomplete State Information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

[37] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[38] Yasin Abbasi-Yadkori and Gergely Neu. Online Learning in MDPs with Side Information. *arXiv preprint arXiv:1406.6812*, 2014.

[39] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual Markov Decision Processes. *arXiv preprint arXiv:1502.02259*, 2015.

[40] Aditya Modi, Nan Jiang, Satinder Singh, and Ambuj Tewari. Markov Decision Processes with Continuous Side Information. *Algorithmic Learning Theory*, 2018.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[42] Jan Peters, Katharina Mülling, and Yasemin Altün. Relative Entropy Policy Search. *AAAI Conference on Artificial Intelligence*, 2010.

[43] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[44] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks. *Neural Information Processing Systems (NeurIPS)*, 2023.

[45] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. *International Conference on Machine Learning (ICML)*, 2020.

[46] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[47] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. *International Conference on Learning Representations (ICLR)*, 2019.

[48] Annya Dahmani*, Eunice Yiu*, Nan Rosemary Ke, Tabitha Edith Lee, Oliver Kroemer, and Alison Gopnik. Toward Understanding Automated Causal Curriculum Learning in Humans and Reinforcement Learning Agents. *The 6th International Workshop on Intrinsically Motivated Open-ended Learning (IMOL)*, 2023. *Equal contribution.

[49] Annya Dahmani*, Eunice Yiu*, Nan Rosemary Ke, Tabitha Edith Lee, Oliver Kroemer, and Alison Gopnik. Toward Understanding Automated Causal Curriculum Learning in Humans and Reinforcement Learning Agents. *Interactive Causal Learning Conference (ICLC)*, 2023. *Equal contribution.

[50] Annya Dahmani*, Eunice Yiu*, Tabitha Edith Lee, Nan Rosemary Ke, Oliver Kroemer, and Alison Gopnik. From Child's Play to AI: Insights into Automated Causal Curriculum Learning. *Intrinsically Motivated Open-ended Learning Workshop, Thirty-seventh Conference on Neural Information Processing Systems (IMOL@NeurIPS)*, 2023. *Equal contribution.

A Intuition for CURATE

Figure 2 illustrates the intuition behind CURATE.

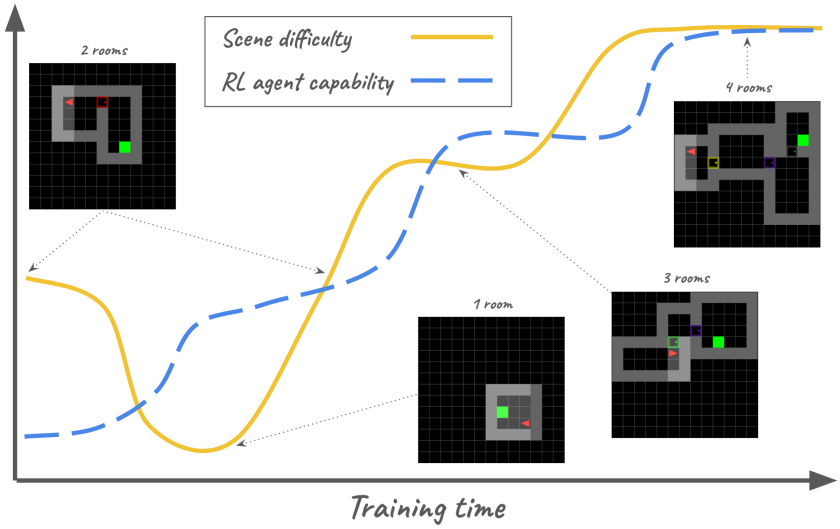


Figure 2: The CURATE algorithm automatically learns a curriculum for training a RL agent to complete a target task distribution that is initially too difficult for the agent. CURATE sequences the RL agent's training data by altering the difficulty of the training task distribution. The RL agent's current capability, or competence, is a measure of its performance in relatively more difficult tasks. In this visualization, the tasks offered by CURATE are initially too difficult, leading to a simplification of tasks. Once the RL agent begins solving these simple tasks, CURATE dynamically adjusts the training data accordingly to offer harder tasks. Finally, the agent solves the target task distribution at the end, indicating that training can conclude. Scenes are from the MiniGrid MultiRoom domain (Sec. 5.1).

B CURATE algorithms

Algorithm 1 (CURATE) describes the training procedure used to train RL agents using CURATE in this work. Algorithm 2 (UPDATECURRICULUM) describes the policy search procedure that CURATE uses to learn the curriculum policy π_c during training.

C Incremental curriculum algorithm

The incremental curriculum baseline in Sec. 5 approximates a handcrafted curriculum that a domain expert may design. Specifically, the incremental curriculum sequentially visits tasks in increasing order of difficulty, approximating a straight line that forms the shortest path curriculum through the curriculum space. (Note that the incremental curriculum does not directly traverse the shortest path curriculum, as the incremental curriculum only increments one dimension of the environment parameters at a time to prevent training instability.) Figure 3 shows the incremental curricula that were used for Leaper, Climber, and BossFight. We generate the incremental curriculum algorithmically using Alg. 3 (GENERATEINCREMENTALCURRICULUM), given the environment parameters of the easiest task θ_i and the environment parameters of the target task θ_t.

Algorithm 3 yields the lowest error approximation of a straight line through the curriculum subject to sequential, cyclic increments of each environment parameter, starting from the easiest task parameters θ_i until the target task parameters θ_t are reached. Intuitively, each dimension of the environment parameters is incremented by a multiple of the respective dimension of $\Delta\theta$ (which for our experiments, is always one). This multiple is the minimum multiple that yields a curriculum point θ' that has dimension d greater than the same dimension of curriculum point θ'_{SP}, which is the projection onto the shortest path curriculum in the direction of the increment. An incremental curriculum is generated

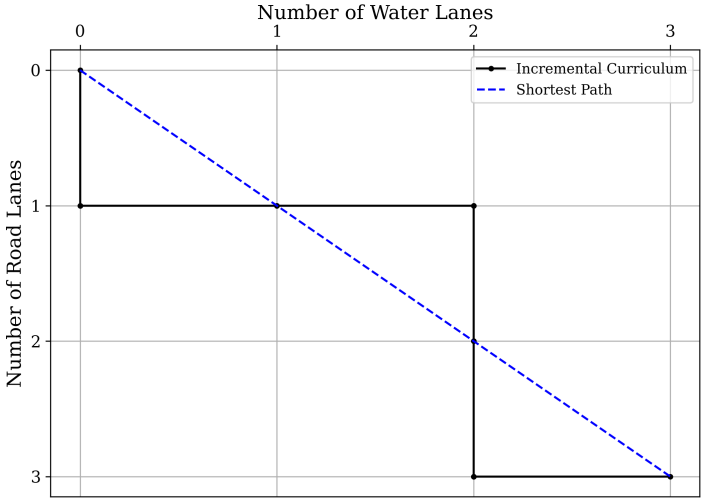
Algorithm 1: CURATE: CURRICULUM AGENT FOR TARGETED EXPLORATION

Input: target task $\mathcal{M}_{\theta_t}$, task solved threshold R_S, maximum number of training frames f_{max}, number of parallelized workers N_v, curriculum advancement on solve $\Delta\mu_\theta$, curriculum covariance for update Σ_{θ_u}, maximum frames between curriculum updates Δf_{sync}

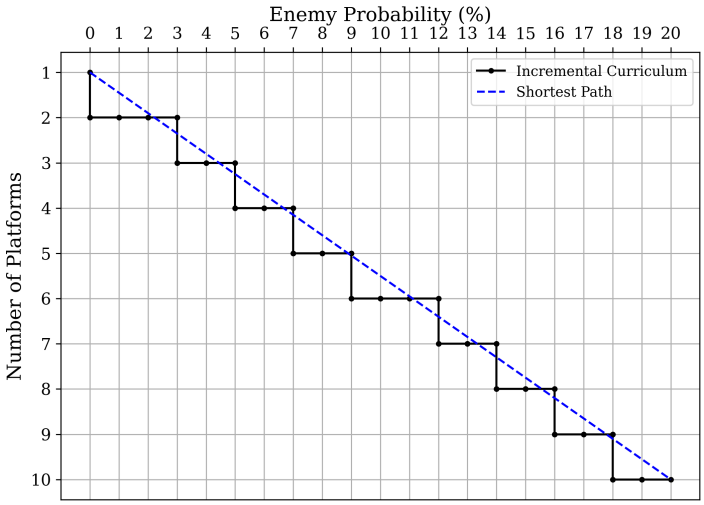
Initialize: training indicator $train \leftarrow$ `True`, target task solved indicator $converged \leftarrow$ `False`, number of training frames $f \leftarrow 0$, control policy $\pi \leftarrow$ INITIALIZERANDOMPOLICY(), curriculum policy and parameters $\pi_c, \mu_\theta, \Sigma_\theta \leftarrow$ INITIALIZERANDOMCURRICULUMPOLICY(), previous curriculum update frame $f_{prev} \leftarrow 0$

```
// Initial curriculum policy update
π_c, μ_θ, Σ_θ ← UpdateCurriculum(π_c, μ_θ, Σ_θ, π)
while train do
    // Sample tasks from the curriculum
    M_θc ← ∅
    for i = 1 to N_v do
        θ_i ~ π_c
        M_θi ← TaskGenerator(θ_i)
        M_θc +← M_θi
    end
    // Collect experience
    D, R_D ← RolloutAgentOnParallelTasks(π, M_θc)
    // Update policy
    π ← UpdateAgent(π, D)
    f = f + NumFrames(D)
    // Update curriculum policy
    if R_S ≤ R_D then
        π_c, μ_θ, Σ_θ ← UpdateCurriculum(π_c, μ_θ + Δμ_θ, Σ_θu, π)
        f_prev ← f
    else if Δf_sync ≤ (f − f_prev) then
        π_c, μ_θ, Σ_θ ← UpdateCurriculum(π_c, μ_θ, Σ_θu, π)
        f_prev ← f
    // Evaluate agent on target task
    R_t ← EvaluateAgent(π, M_θt)
    // Determine whether to continue training
    if R_S ≤ R_t then
        train ← False
        converged ← True
    if f_max ≤ f then
        train ← False
end
```

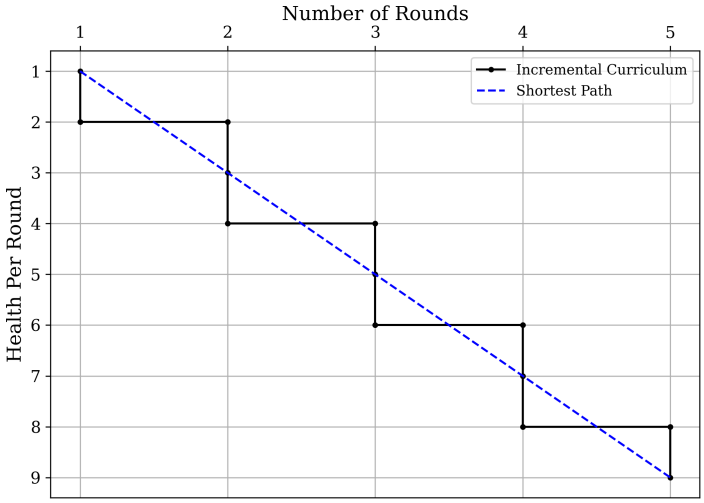
Result: control policy π, target task solved indicator $converged$, number of frames f



(a)



(b)



(c)

Figure 3: Incremental curriculum for (a) Leaper, (b) Climber, and (c) BossFight.

Algorithm 2: UPDATECURRICULUM: Curriculum update for CURATE

Input: curriculum policy π_c, initial curriculum policy mean μ_{θ_0}, initial curriculum policy covariance Σ_{θ_0}, control policy π, task solved threshold R_S, parameter regularization λ_θ, number of rounds N_r, samples per round N_s, relative entropy bound ϵ, minimum temperature η

Initialize: $\mu_\theta \leftarrow \mu_{\theta_0}$, $\Sigma_\theta \leftarrow \Sigma_{\theta_0}$

for $i = 1$ **to** N_r **do**
  `// Reset buffers`
  $\boldsymbol{\theta_{eval}} \leftarrow \emptyset$
  $\boldsymbol{\nu_{eval}} \leftarrow \emptyset$
  for $j = 1$ **to** N_s **do**
    `// Sample task`
    $\theta_j \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$
    $\mathcal{M}_{\theta_j} \leftarrow$ TASKGENERATOR(θ_j)
    `// Evaluate agent on sampled task`
    $R_j \leftarrow$ EVALUATEAGENT$(\pi, \mathcal{M}_{\theta_j})$
    `// Determine reward for curriculum learning`
    if $R_j < R_S$ **then**
      $\nu_{j,init} \leftarrow R_j / R_S$
    else
      $\nu_{j,init} \leftarrow 0$
    $\nu_j \leftarrow \nu_{j,init} - \lambda_\theta ||\theta_j||_2$
    `// Append to buffers`
    $\boldsymbol{\theta_{eval}} \overset{+}{\leftarrow} \theta_j$
    $\boldsymbol{\nu_{eval}} \overset{+}{\leftarrow} \nu_j$
  end
  `// Run REPS and update curriculum policy`
  $\mu_\theta, \Sigma_\theta \leftarrow$ REPSUPDATE$(\boldsymbol{\theta_{eval}}, \boldsymbol{\nu_{eval}}, \epsilon, \eta)$
  $\pi_c \leftarrow$ DISCRETIZEGAUSSIAN$(\mu_\theta, \Sigma_\theta)$
end

Result: updated curriculum policy π_c, updated curriculum policy mean μ_θ, updated curriculum policy covariance Σ_θ

for each possible starting increment of the first dimension, then the incremental curriculum with the least error is returned.

D Experimental details

Implementation Our work is implemented within the Dual Curriculum Design (DCD) codebase [30].[1] We use the official implementations of PLR$^\perp$ and ACCEL as provided in this codebase.

Task solved threshold The task solved threshold R_S indicates when a task has been solved based on its reward. An agent that receives a reward of at least R_S on a task is said to have solved a task. This threshold is used to determine when training is no longer needed in a few ways in this work:

1. When the evaluation reward obtained on the target task distribution meets or exceeds R_S, the RL training procedure concludes.
2. CURATE uses R_S to calculate the rewards ν used for the curriculum policy, which favors learning the set of easiest tasks not yet solved.
3. R_S is used by the incremental curriculum baseline to indicate when it is time to advance to the next set of tasks in the curriculum.

[1] `https://github.com/facebookresearch/dcd`

Algorithm 3: GENERATEINCREMENTALCURRICULUM: Generate incremental curriculum

Input: easiest environment parameters θ_i, target environment parameters θ_t

Initialize: environment parameter dimensionality $N_d \leftarrow \dim(\theta_i)$, environment parameter increment $\Delta_\theta \leftarrow \vec{1}$, shortest path curriculum vector $\vec{v}_{SP} \leftarrow \theta_t - \theta_i$, shortest path curriculum vector magnitude $\hat{v}_{SP} \leftarrow \vec{v}_{SP}/||\vec{v}_{SP}||_2$, number of curriculum steps per dimension $\boldsymbol{N_\Delta} \leftarrow \{\text{ceil}(\vec{v}_{SP}[d]\,/\Delta_\theta[d]) \mid d \in [1, N_d]\}$, total number of curriculum steps $N_{\Sigma_\Delta} \leftarrow \sum_{d=1}^{N_d} \boldsymbol{N_\Delta}[d]$, candidate incremental curricula $\boldsymbol{\mathcal{C}_i} \leftarrow \emptyset$, candidate incremental curricula errors $\boldsymbol{e_i} \leftarrow \emptyset$

```
// Loop over all possible initial increments
for N'_{Δ,init} = 1 to N_Δ[1] do
    θ' ← θ_i
    θ'_SP ← θ_i
    N'_Δ ← 0
    C'_i ← {θ_i}
    e'_i ← 0
    while N'_Δ < N_{Σ_Δ} do
        for d = 1 to N_d do
            while ((θ'[d] ≤ θ'_SP[d]) or (N'_Δ < N'_{Δ,init})) and (θ'[d] < θ_t[d]) do
                // Increment environment parameter along specified dimension
                θ'[d] ← min(θ'[d] + Δ_θ[d], θ_t[d])
                // Update steps taken and add to incremental curriculum
                N'_Δ ← N'_Δ + 1
                C'_i ←+ θ'
                // Calculate error with respect to shortest path
                v_θ' ← θ' − θ_i
                h_e ← v_θ' · v_SP / ||v_SP||_2^2
                v_proj ← h_e · v_SP
                v_⊥ ← v_θ' − v_proj
                e'_i ← e'_i + ||v_⊥||_2
            end
            // Update point on the shortest path
            h_SP ← (θ'[d] − θ_i[d]) / v̂_SP[d]
            θ'_SP ← h_SP · v̂_SP + θ_i
        end
    end
    // Append to buffers
    C_i ←+ C'_i
    e_i ←+ e'_i
end
// Choose the incremental curriculum with the least error
i_h ← argmin(e_i)
C_{i,min} ← C_i[i_h]
```

Result: incremental curriculum with least error approximation to the shortest path $\boldsymbol{\mathcal{C}_{i,min}}$

We assume that R_S is provided as part of the task definition. In practice, we train an RL agent using a random curriculum (i.e., domain randomization) to obtain what the maximum achievable reward in the target task distribution is. Then, we set R_S slightly below that value.

Maximum number of training frames The maximum allowable frames f_{max} provides an upper limit to how long the RL agent is trained. Generally, it is determined as 2-5 times the average frames required for a random curriculum (i.e., domain randomization) to reach a target reward of at least R_S.

D.1 MiniGrid MultiRoom Navigation

MiniGrid MultiRoom requires the RL agent to master grid-based navigation within the MiniGrid domain [44]. In MultiRoom, the agent must navigate through a series of rooms that are sequentially connected with doors separating the rooms. The agent always starts in the first room, and the goal always exists in the last room. The environment is a reimplementation of MultiRoom-Random-N4 [31]. However, we use the typical MiniGrid observation space as described below.

Observation space The agent receives two observations:

1. A field-of-view observation consisting of the states of the world within a 7 x 7 grid within the agent's line of sight. The agent cannot see through walls.
2. The direction the agent is facing, represented as an integer with one of four values that each represent a different direction.

Action space The agent uses discrete actions with action space $|\mathcal{A}| = 7$. The actions include turning left, turning right, going forward, picking up an object, dropping an object, toggling the activation for an object, and doing nothing. As there are no objects for the agent to pick up in this domain, the actions for picking up and dropping an object have no effect. Toggling the activation in front of a door will either open it (if closed) or close it (if opened).

Reward The agent receives a time-discounted reward when solving the level by reaching the goal; zero reward is received otherwise. A task is considered solved if the agent receives at least 0.7 reward.

Curriculum space MultiRoom is a one-dimensional curriculum space, where the curriculum axis $\theta_1 = [1, 4]$ controls the number of rooms in each task.

Target task distribution For MultiRoom, the agent must solve a task distribution consisting of $\theta_t = 4$ rooms.

D.2 Procgen Curriculum Suite

Procgen, as introduced by Cobbe et al. [45], tasks RL agents to master different types of discrete control games. For our experiments, we select three representative games from Procgen: Leaper, Climber, and BossFight.

In our work, each game is adapted such that each level can be changed by specifying causal interventions in the environment parameters to change the initial level state. For example, the intervention $do(\theta_1 = 1, \theta_2 = 3)$ on level seed 0 in Leaper would yield the same level as without interventions, except with 1 road lane and 3 water lanes. Please refer to Fig. 4 for a visualized example. Note that for Leaper, intervention on these parameters may change other aspects of the initial state, such as the initial placement of cars and logs. Therefore, for Leaper, partial entanglement exists between Θ and other variables in the environment. However, for Climber and BossFight, Θ is completely disentangled from the rest of the level generation process.

We implement our extensions of Procgen within the Procgen fork used by Jiang et al. [31][2]

Observation space The agent receives a 64 x 64 RGB image observation of the game.

[2] `https://github.com/minqi/procgen`

Action space The agent uses discrete actions with action space $|\mathcal{A}| = 15$. The actions generally correspond to eight directional actions, six special actions, and one action that does nothing. The actions are game-specific; please refer to Cobbe et al. [45] for a complete description.

Distribution mode We use the easy distribution mode of Procgen to avoid extra computational resources that would be required for the hard distribution mode.

Reward The rewards for Procgen games are game-specific. For Leaper, the agent receives 10 reward when reaching the goal (zero otherwise). Climber and BossFight are less sparse than Leaper. In Climber, the agent receives 10 reward for completing a level by collecting all the coins. Collecting a coin provides 1 reward. For BossFight, the agent receives 10 reward for defeating the boss. The fight is split into separate rounds, and completing a round provides 1 reward.

Curriculum space: Leaper The curriculum axes specify the number of road lanes ($\theta_1 = [0, 3]$) and number of water lanes ($\theta_2 = [0, 3]$).

Curriculum space: Climber The curriculum axes are defined as the number of platforms ($\theta_1 = [1, 10]$) and percentage of an enemy spawning at each platform ($\theta_2 = [0, 20]$).

Curriculum space: BossFight The curriculum axes control the number of health points of the boss per round ($\theta_1 = [1, 9]$) and the total number of rounds ($\theta_2 = [1, 5]$).

Target task distribution Generally, the target task distribution for each game contains the hardest levels that would be obtained in each game under the easy distribution mode (i.e., $\theta_t = \max(\Theta)$). In other words, no level is harder than what would have been possible to experience when randomly sampling levels from Procgen.

D.3 Hyperparameters

Table 4 presents the experimental hyperparameters. For MiniGrid MultiRoom, we generally use the hyperparameters from Jiang et al. [30] for their MiniGrid experiments, except with 16 parallel environments instead of 32 to run experiments with less computational resources. The PPO rollout length was chosen as 192 to fit at least two episodes of duration 80 into the rollout. For Procgen, we generally use the same hyperparameters as the Procgen experiments in Jiang et al. [31] for the easy distribution. However, we use the episode length as defined by each game, and set the PPO rollout length to the nearest power of two. Then, we select minibatches per epoch such that each minibatch has 2048 samples, the same as in Jiang et al. [31].

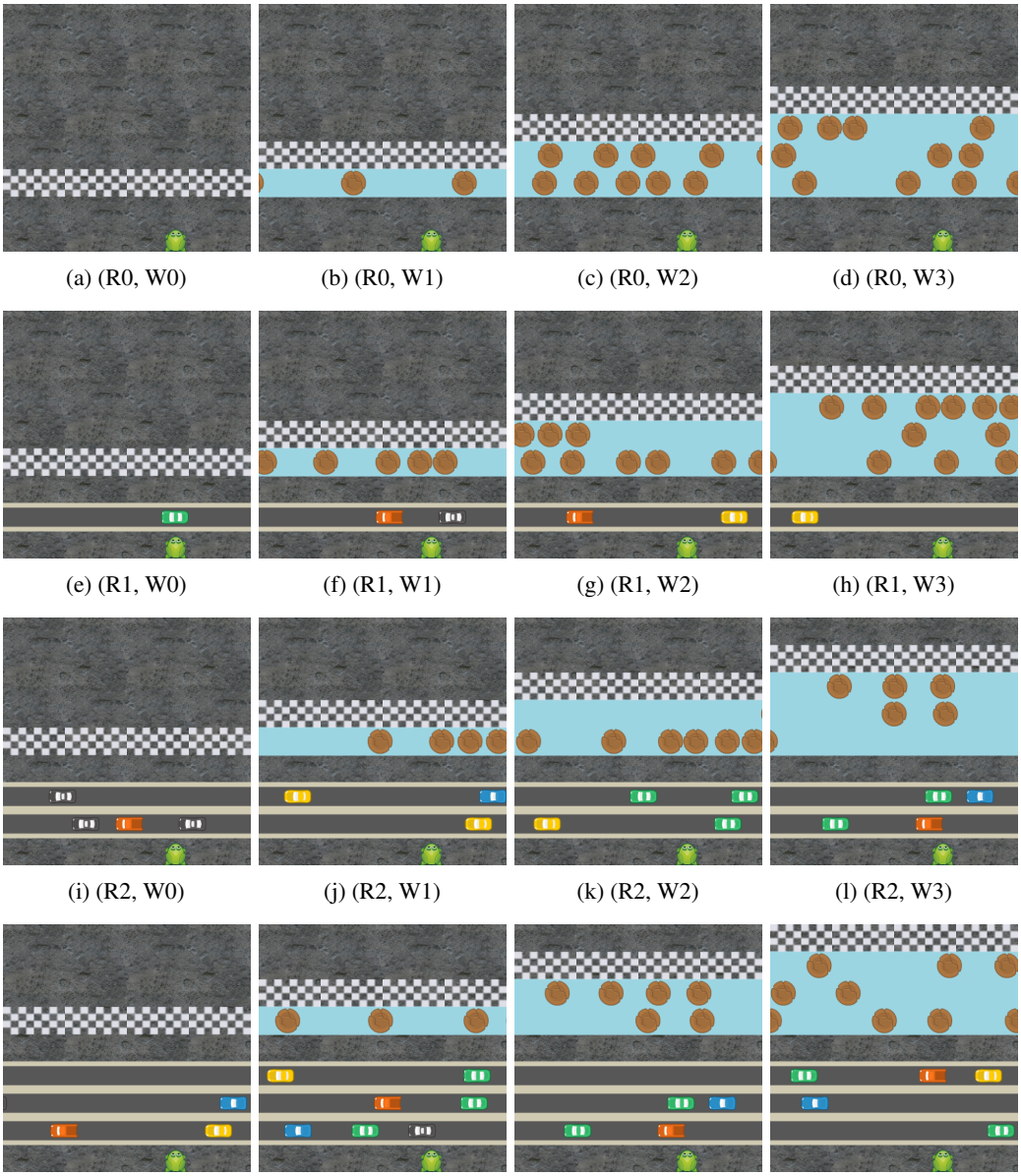
For PLR$^{\perp}$, we generally use the same hyperparameters as Jiang et al. [30] for MultiRoom and Jiang et al. [31] for Procgen. Our ACCEL hyperparameters come from Jiang et al. [30].

E Supplemental results for MiniGrid MultiRoom

Figure 5 visualizes the summary statistics first described in Tab. 2. Figure 6 presents a representative curricula and training/target learning curves for each approach. The representative trial for each approach is the closest trial to the median of all 10 trials used for that approach.

F Supplemental results for Procgen Curriculum Suite

Each approach's learning curves and curricula are visualized for Leaper (Figs. 7–8), Climber (Figs. 9–10), and BossFight (Figs. 11–12).



(a) (R0, W0) (b) (R0, W1) (c) (R0, W2) (d) (R0, W3)

(e) (R1, W0) (f) (R1, W1) (g) (R1, W2) (h) (R1, W3)

(i) (R2, W0) (j) (R2, W1) (k) (R2, W2) (l) (R2, W3)

(m) (R3, W0) (n) (R3, W1) (o) (R3, W2) (p) (R3, W3)

Figure 4: Example of variations in initial scenes for Leaper based on selection of environment parameters. Each figure represents an example task within the task distribution corresponding to the chosen environment parameters. For example, (h) represents a task with $\theta_1 = 1$ road lane and $\theta_2 = 3$ water lanes. All scenes are based on level seed 0.

Table 4: Hyperparameters used for experiments. Note that for CURATE, N_r and λ_θ can take different values depending on whether it is the initial curriculum update or not.

Hyperparameter	MultiRoom	Leaper	Climber	BossFight
Discount factor γ	0.995	0.999	0.999	0.999
λ_{GAE}	0.95	0.95	0.95	0.95
Rollout length	192	512	1024	4096
Epochs	5	3	3	3
Minibatches per epoch	1	16	32	128
Clip range	0.2	0.2	0.2	0.2
Number of parallel environments N_v	16	64	64	64
Return normalization	no	yes	yes	yes
Entropy bonus coefficient	0.0	0.01	0.01	0.01
Value loss coefficient	0.5	0.5	0.5	0.5
Max gradient norm	0.5	0.5	0.5	0.5
Adam learning rate	0.0001	0.0005	0.0005	0.0005
Adam ϵ	0.00001	0.00001	0.00001	0.00001
Recurrent agent	yes	no	no	no
Action space dimensionality $\lvert\mathcal{A}\rvert$	7	15	15	15
Episode length	80	500	1000	4000
Reward threshold R_S	0.7	8.0	10.0	10.0
Min. number of target episodes per eval.	128	64	64	64
Curriculum space dimensionality $\lvert\Theta\rvert$	1	2	2	2
Curriculum space for Θ_1	[1, 4]	[0, 3]	[1, 10]	[1, 9]
Curriculum space for Θ_2	n/a	[0, 3]	[0, 20]	[1, 5]
Max. train frames f_{max} ($\times 10^6$)	50.000	100.000	100.000	100.000
Replay rate	0.5	0.5	0.5	0.5
PLR prioritization	rank	rank	rank	rank
Temperature β	0.3	0.1	0.1	0.1
Staleness coefficient ρ	0.3	0.1	0.1	0.1
Replay buffer size	4000	4000	4000	4000
Scoring function loss	positive value	L1 value	L1 value	L1 value
Edit rate	1.0	1.0	1.0	1.0
Replay rate	0.8	0.8	0.8	0.8
Number of edits	3	3	3	3
Edit method	random	random	random	random
Levels edited	easy	easy	easy	easy
Number rounds N_r	2/2	4/2	4/2	4/2
Samples per round N_s	8	16	16	16
Regularization hyperparameter λ_θ	0.0125/0.0125	0.024/0.0	0.005/0.0	0.01/0.0
REPS relative entropy bound ϵ	0.75	0.75	0.75	0.75
REPS minimum temperature η	0.05	0.05	0.05	0.05
Max. update frames Δf_{sync} ($\times 10^6$)	0.393	4.194	8.389	33.554

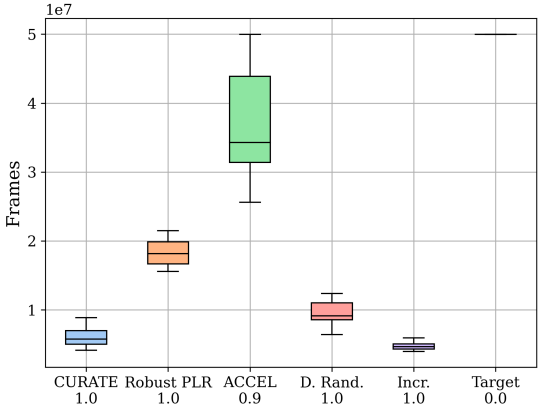
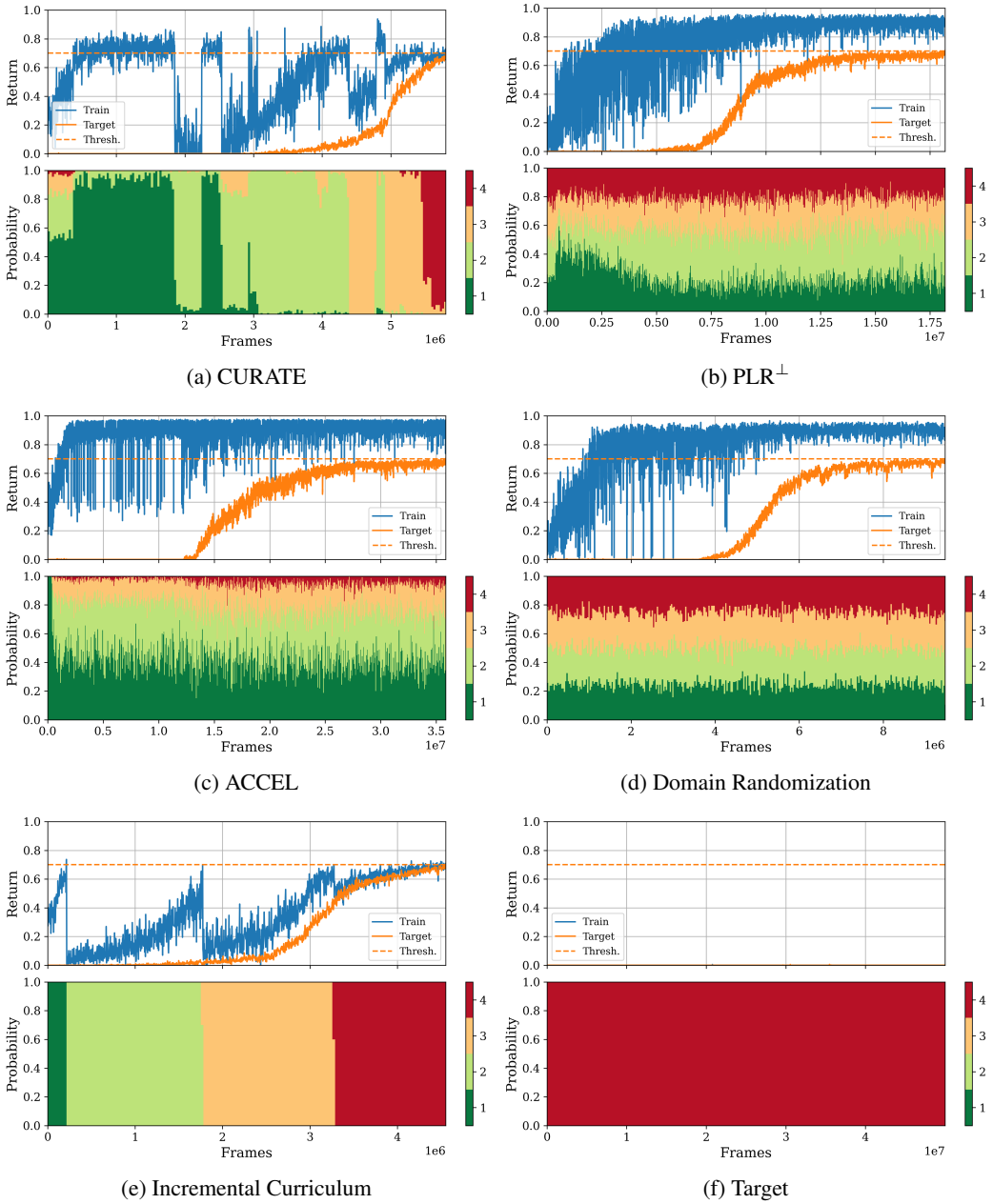


Figure 5: Median statistics for sample efficiency for MiniGrid MultiRoom. The approach success rate is displayed beneath each approach's name. D. Rand stands for Domain Randomization. Incr. stands for Incremental Curriculum. All trials for Target yielded the maximum allowable frames (50 million) with a 0% success rate.



(a) CURATE

(b) PLR$^{\perp}$

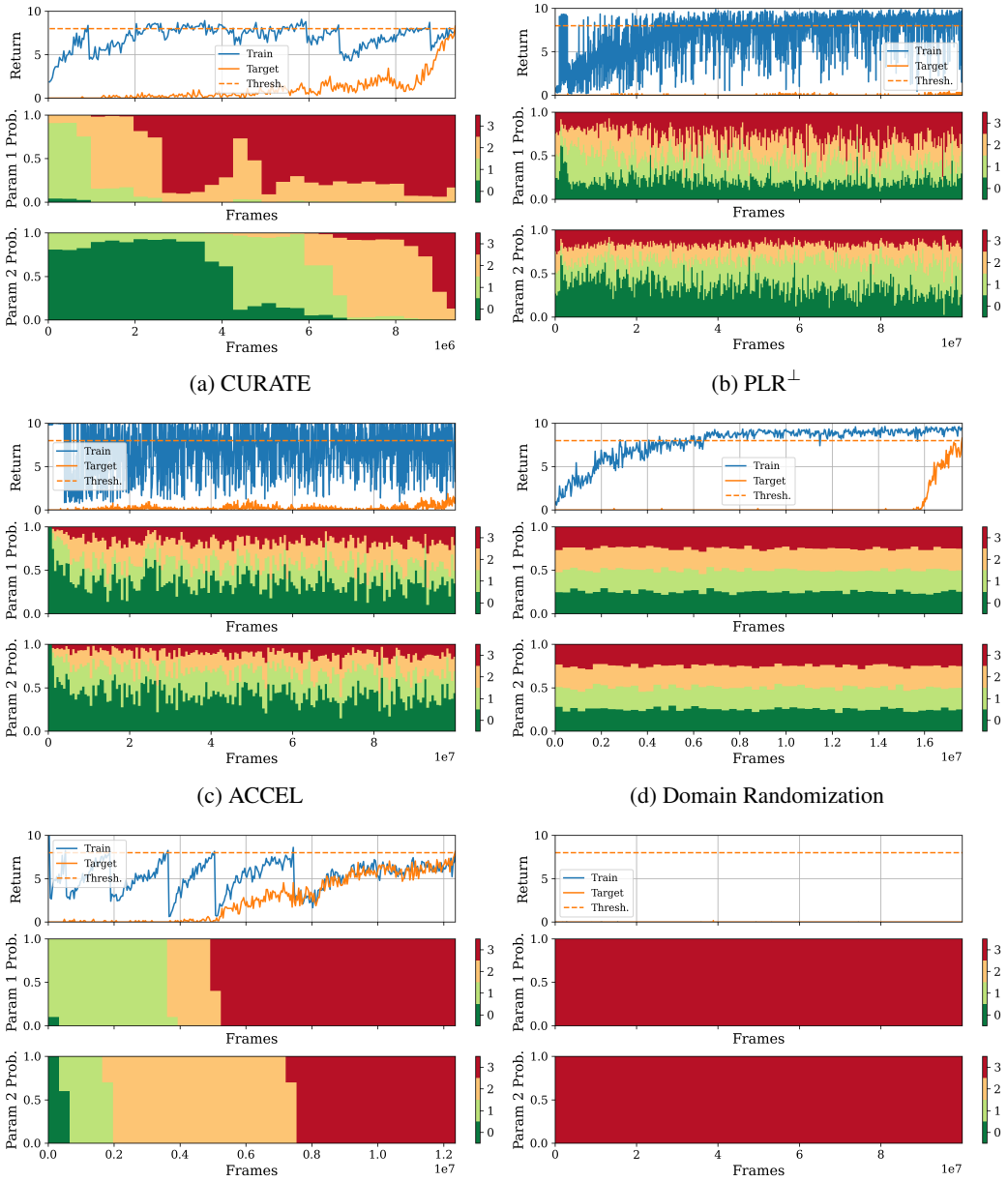
(c) ACCEL

(d) Domain Randomization

(e) Incremental Curriculum

(f) Target

Figure 6: Representative curriculum learning time histories for each approach in MiniGrid MultiRoom. Each time history shows the trial that is closest to the median performance of all 10 trials for each approach. The top figure shows the time history of the return, shown for the training environments and the target task. The bottom figure shows the time history of the curriculum, with time average discretization of 10 updates to better show long-term trends.



(a) CURATE

(b) PLR$^{\perp}$

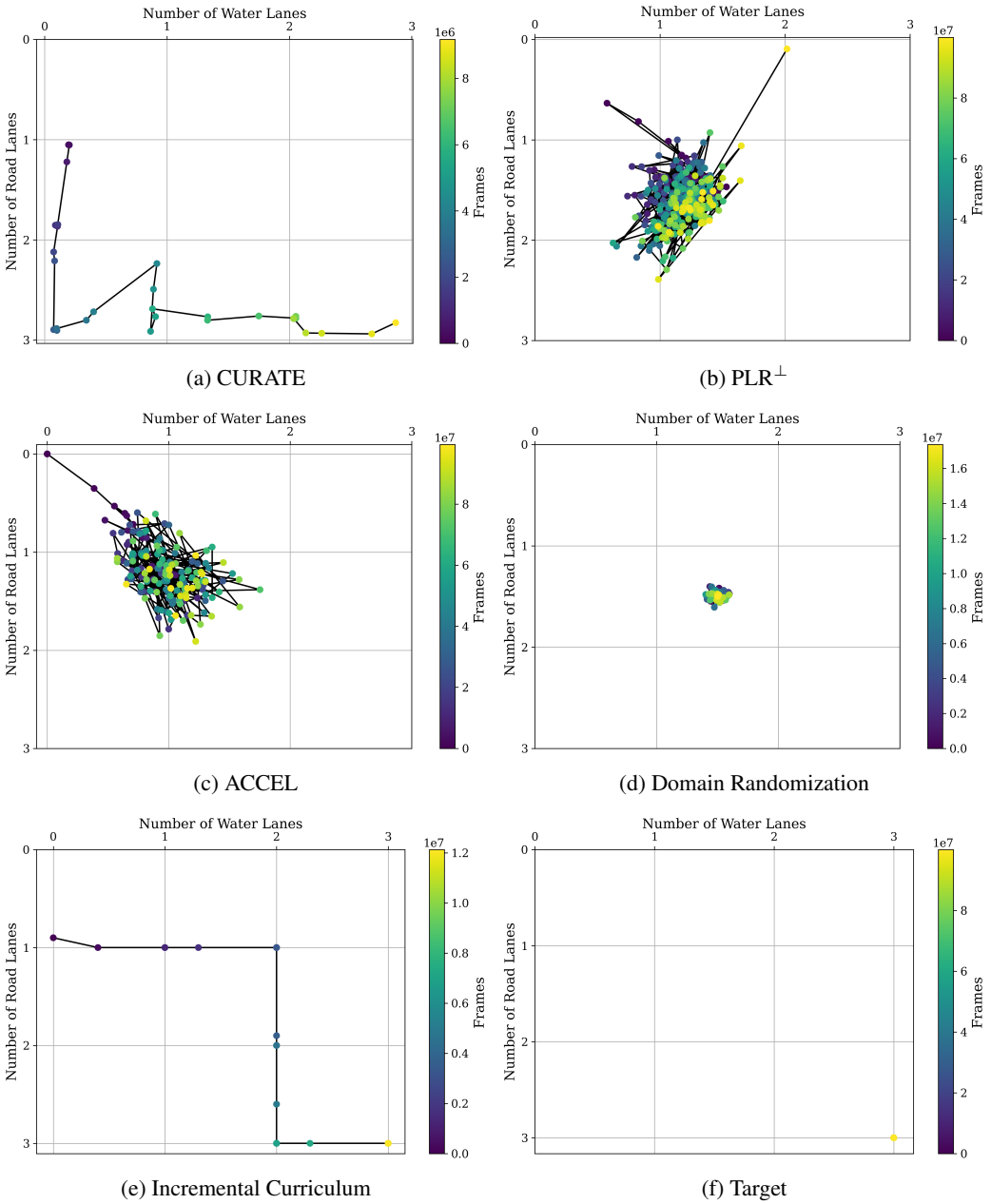
(c) ACCEL

(d) Domain Randomization

(e) Incremental Curriculum

(f) Target

Figure 7: Curriculum learning time histories for each approach in Leaper. The top figure shows the time history of the return, shown for the training environments and the target task. The bottom figures show the time history of the curriculum, with time average discretization of 10 updates to better show long-term trends.



(a) CURATE

(b) PLR$^{\perp}$

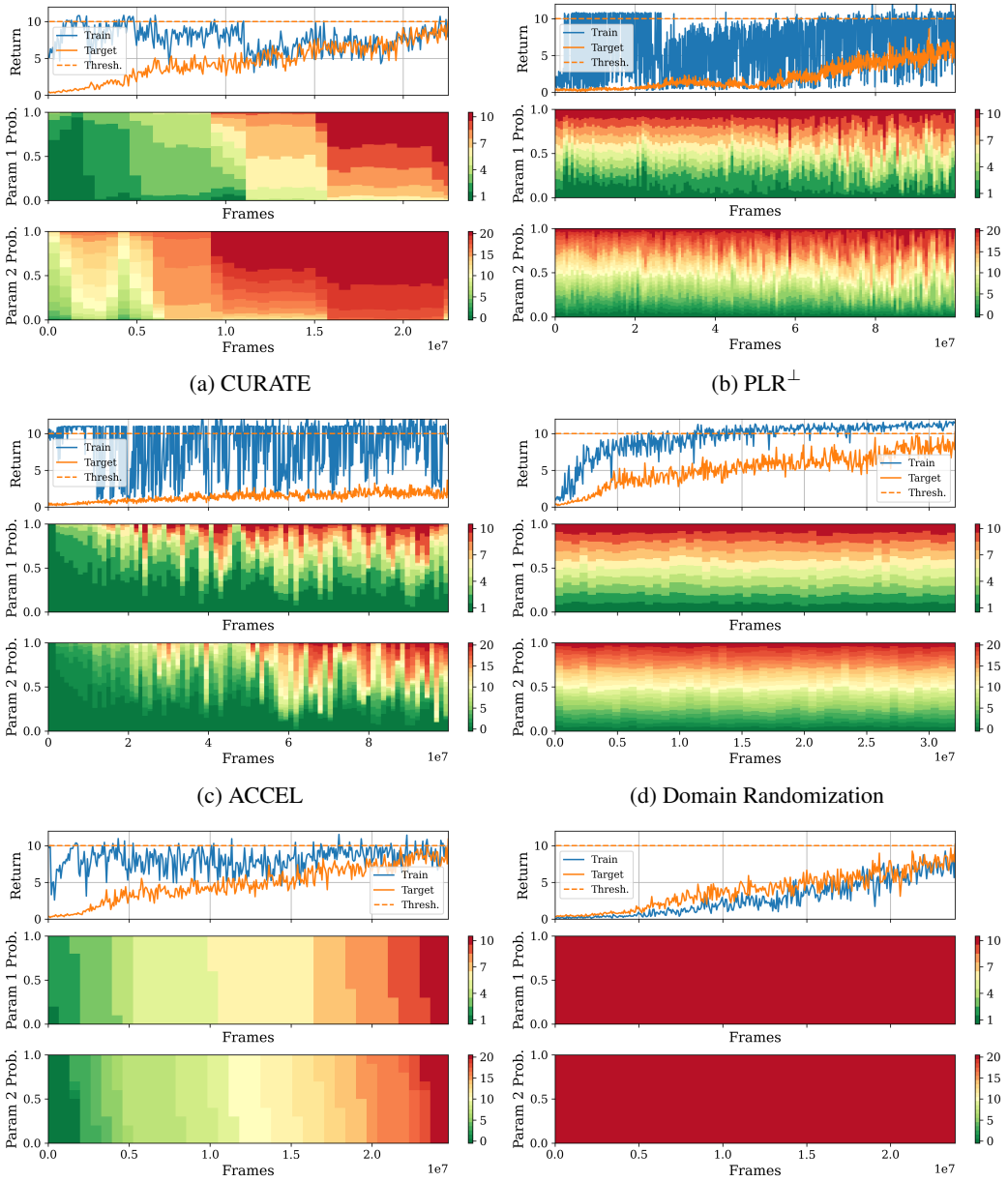
(c) ACCEL

(d) Domain Randomization

(e) Incremental Curriculum

(f) Target

Figure 8: Curriculum for each approach in Leaper as represented by the mean environment parameters of the training tasks with time average discretization of 10 updates to better show long-term trends. Note that the colorbar for each figure has a different maximum value.



(a) CURATE

(b) PLR$^{\perp}$

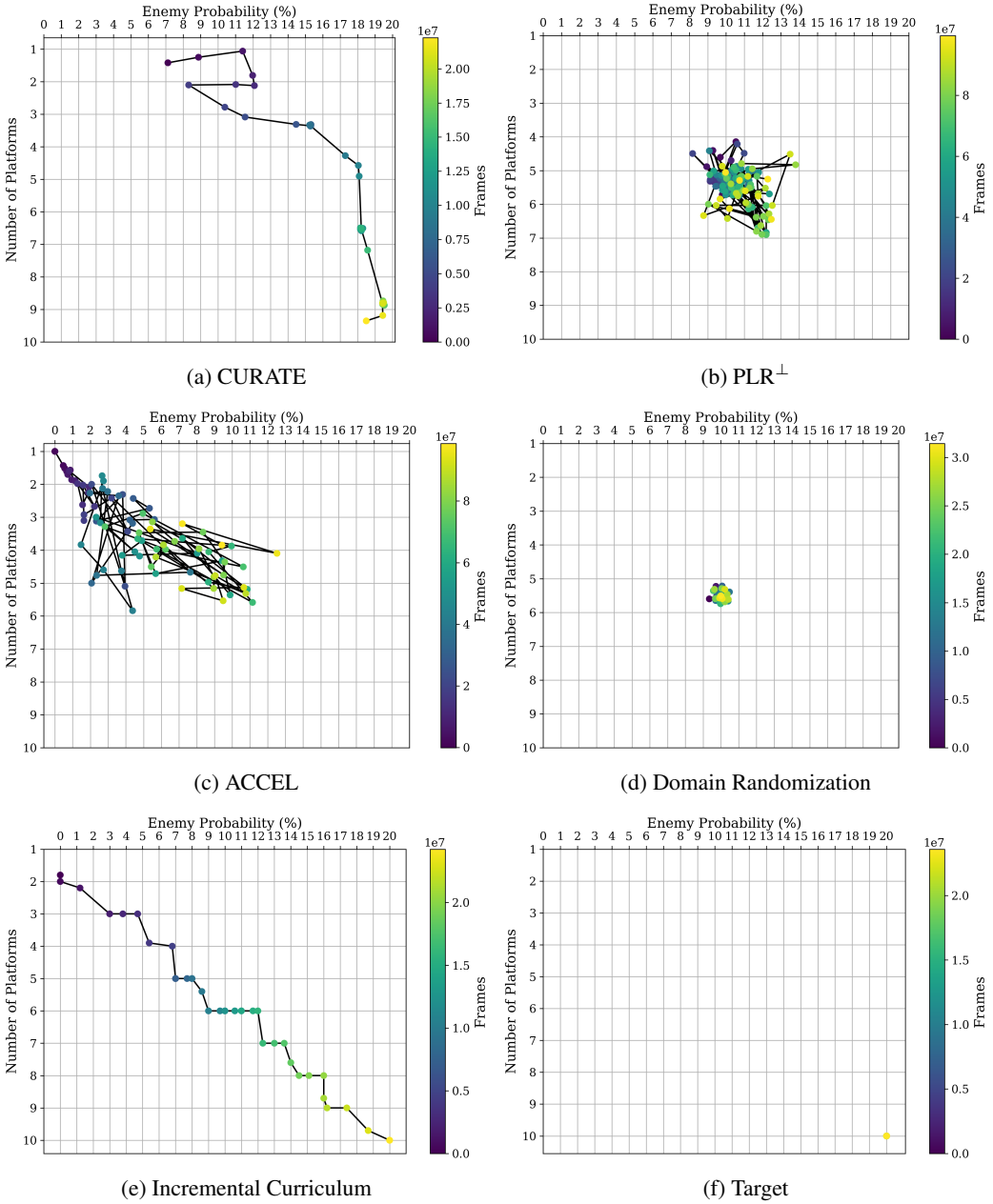
(c) ACCEL

(d) Domain Randomization

(e) Incremental Curriculum

(f) Target

Figure 9: Curriculum learning time histories for each approach in Climber. The top figure shows the time history of the return, shown for the training environments and the target task. The bottom figures show the time history of the curriculum, with time average discretization of 10 updates to better show long-term trends.



(a) CURATE

(b) PLR$^{\perp}$

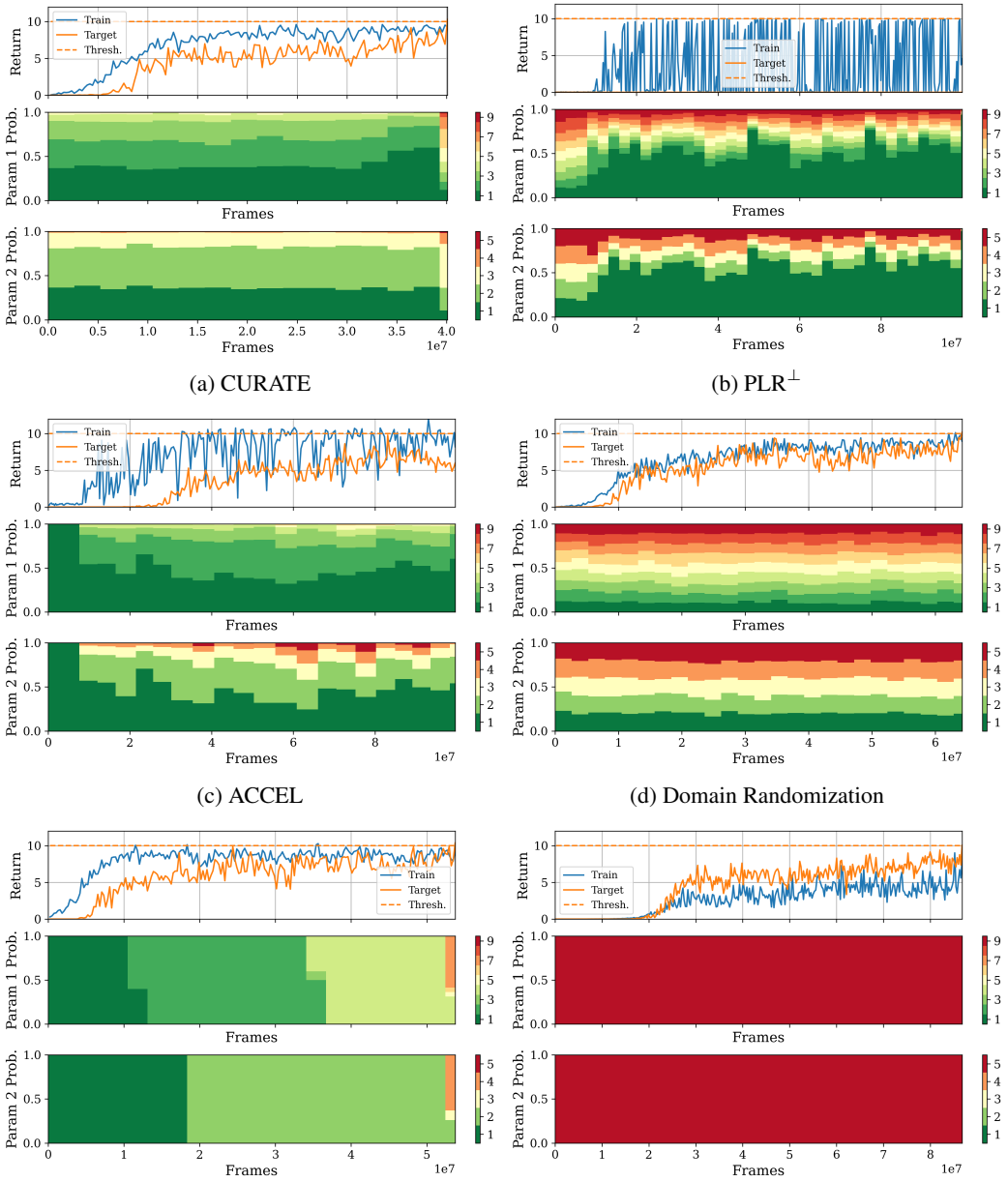
(c) ACCEL

(d) Domain Randomization

(e) Incremental Curriculum

(f) Target

Figure 10: Curriculum for each approach in Climber as represented by the mean environment parameters of the training tasks with time average discretization of 10 updates to better show long-term trends. Note that the colorbar for each figure has a different maximum value.



(a) CURATE

(b) PLR$^{\perp}$

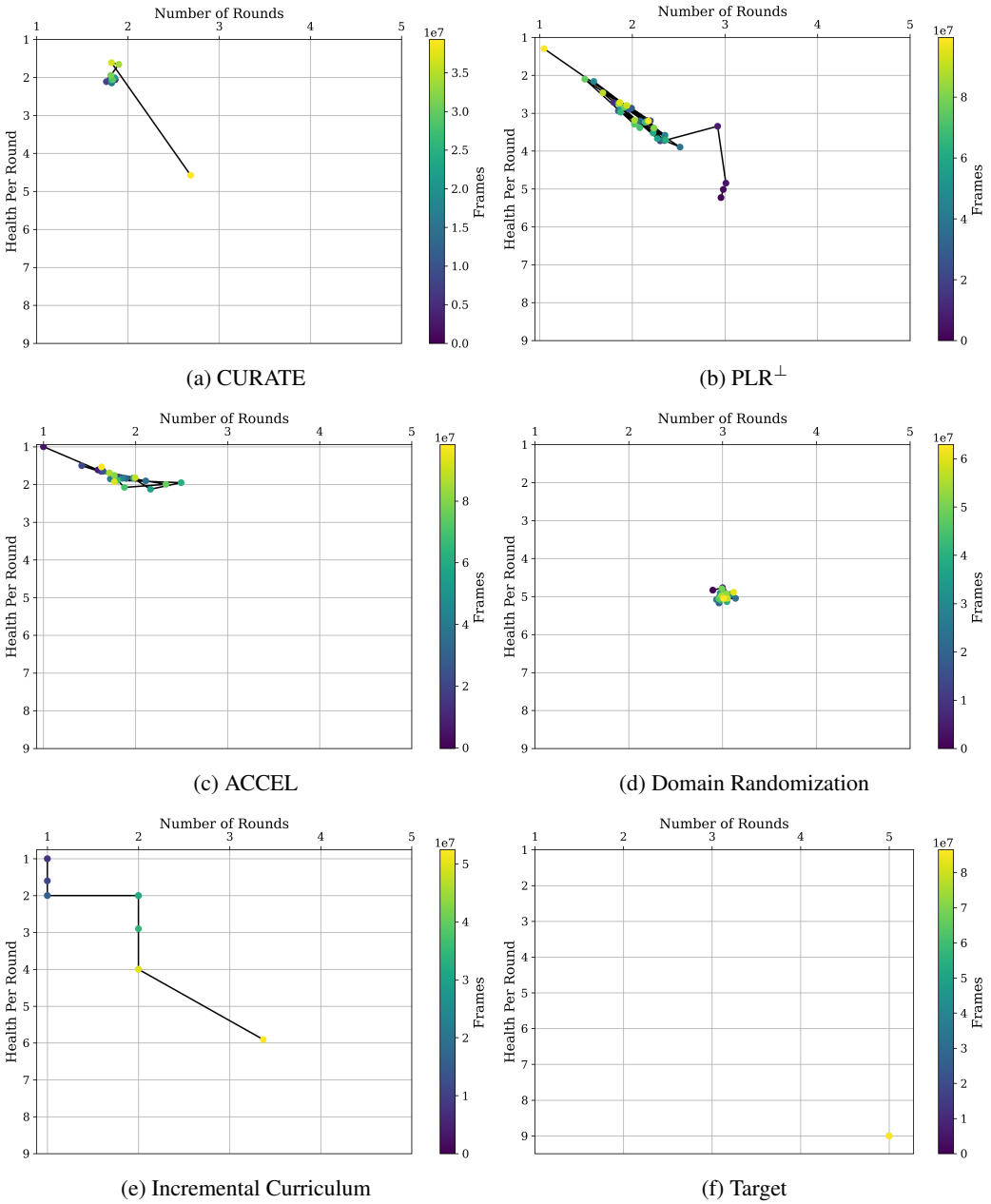
(c) ACCEL

(d) Domain Randomization

(e) Incremental Curriculum

(f) Target

Figure 11: Curriculum learning time histories for each approach in BossFight. The top figure shows the time history of the return, shown for the training environments and the target task. The bottom figures show the time history of the curriculum, with time average discretization of 10 updates to better show long-term trends.



(a) CURATE

(b) PLR$^{\perp}$

(c) ACCEL

(d) Domain Randomization

(e) Incremental Curriculum

(f) Target

Figure 12: Curriculum for each approach in BossFight as represented by the mean environment parameters of the training tasks with time average discretization of 10 updates to better show long-term trends. Note that the colorbar for each figure has a different maximum value.