# An Optimisation Framework for Unsupervised Environment Design

**Anonymous authors**
Paper under double-blind review

**Keywords:** Optimisation, Environment Design, Reinforcement Learning, Robustness

## Summary

For reinforcement learning agents to be deployed in high-risk settings, they must achieve a high level of robustness to unfamiliar scenarios. One method for improving robustness is unsupervised environment design (UED), a suite of methods aiming to maximise an agent's generalisability across configurations of an environment. In this work, we study UED from an optimisation perspective, providing stronger theoretical guarantees for practical settings than prior work. Whereas previous methods relied on guarantees *if* they reach convergence, our framework employs a nonconvex-strongly-concave objective for which we provide a *provably convergent* algorithm in the zero-sum setting. We empirically verify the efficacy of our method, outperforming prior methods in a number of environments with varying difficulties.

## Contribution(s)

1. We provide a reformulation of UED that is strongly concave in the adversary's strategy, allowing for easier convergence.
   **Context:** Dennis et al. (2020)'s initial UED work PAIRED uses a nonconvex-nonconcave objective, which is known to be unstable in training (Wiatrak et al., 2020). Moreover, follow-up works such as (Chung et al., 2024) that improve PAIRED's level generator with generative models maintain this property.
2. We provide convergence guarantees for any score function that is a zero-sum game over the policy's negative return (e.g. regret or negative return).
   **Context:** Prior works in UED (Dennis et al., 2020; Jiang et al., 2021a) assert guarantees if the UED game reaches a saddle point, but fail to converge to one. We propose a method that provably converges.
3. We provide an empirical evaluation of our methods on current UED benchmarks, using relevant optimisation heuristics and a new score function that generalises the work of Rutherford et al. (2024) to general deterministic RL environments.
   **Context:** Learnability (Rutherford et al., 2024) fails to obtain the same guarantees as the zero-sum setting without additional (potentially second-order) optimisation techniques that are beyond the scope of this work (Zeng & Doan, 2024; Hong et al., 2023).

# An Optimisation Framework for Unsupervised Environment Design

**Anonymous authors**
Paper under double-blind review

## Abstract

For reinforcement learning agents to be deployed in high-risk settings, they must achieve a high level of robustness to unfamiliar scenarios. One method for improving robustness is unsupervised environment design (UED), a suite of methods aiming to maximise an agent's generalisability across configurations of an environment. In this work, we study UED from an optimisation perspective, providing stronger theoretical guarantees for practical settings than prior work. Whereas previous methods relied on guarantees *if* they reach convergence, our framework employs a nonconvex-strongly-concave objective for which we provide a *provably convergent* algorithm in the zero-sum setting. We empirically verify the efficacy of our method[1], outperforming prior methods in a number of environments with varying difficulties.

## 1 Introduction

How to train reinforcement learning (RL) agents that are robust to a variety of scenarios is an important and long-studied research question (Morimoto & Doya, 2000). Unsupervised Environment Design (UED) is a contemporary approach to robustness within RL that seeks to learn policies that are versatile to a diverse set of environments. Using a parametrised environment simulator, UED methods progress agents from easy to difficult environment parametrisations, called levels. These methods aim to train agents that perform well in a wide range of unseen levels, and do this by constructing a two-player game between the agent and a level-selecting adversary (Dennis et al., 2020).

Most current methods use a variation of the *minimax regret* approach, in which levels with high regret, meaning an agent performs far from optimal, are selected by an adversary for the agent to learn in. This forces the agent to learn from highly learnable environments (Dennis et al., 2020; Jiang et al., 2021b;a; 2022; Parker-Holder et al., 2023). Using regret as a *score* function is intuitive, since it shows how suboptimal a policy is with regard to a specific level. However, despite some empirical success (Dennis et al., 2020; Jiang et al., 2021b), more recent works have presented a number of issues with the minimax regret formulation (Jiang et al., 2022; Beukman et al., 2024; Rutherford et al., 2024), motivating the creation of new ways of determining the *score* (i.e. learning potential) of a level. The canonical formulation of UED faces several other challenges, including difficulties with convergence and instability when searching for useful levels in a large space. In our work, we provide a reformulation of minimax regret as the expected regret when sampling levels from a categorical distribution. Hence, we provide a gradient-based algorithm that is *provably convergent* to the objective's solution set, in stark contrast to the existing paradigm that only has theoretical guarantees *if* convergence is reached. In particular we compute gradients for both the agent and adversary, and update the adversary with a much larger learning rate than the agent to ensure we find an approximate solution.

[1] Code released upon acceptance.

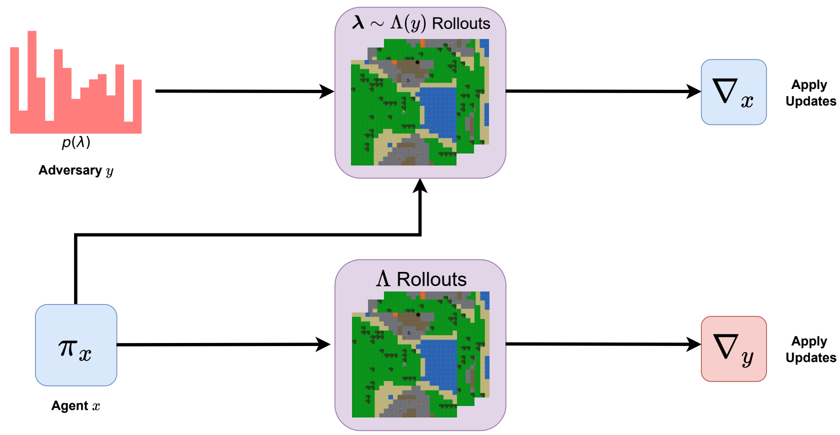



Figure 1: A visual representation of our training loop, which has simultaneous updates for the agent $x$ and the adversary $y$. The agent's update is trained on levels $\boldsymbol{\lambda}$ sampled from $y$, and the adversary's update is computed with scores computed from the policy on all levels from level-buffer $\Lambda$. We use the *Craftax* environment from Matthews et al. (2024) for illustration.

Minimax regret is theoretically attractive because it maintains the zero-sum property between the agent and the adversary, thus allowing for stronger guarantees from single-loss optimisation. However, the levels with highest regret may not be those that lead to efficient learning. Furthermore, regret is intractable in practice, as it requires an optimal policy for each level. Inspired by this, Rutherford et al. (2024) used *learnability* (Tzannetos et al., 2023) as an effective scoring function, but it is limited to deterministic, binary-outcome domains. Motivated by the interpretation of learnability as the variance of agent success, we generalise this score function to arbitrary deterministic settings by using the normalised standard deviation of level-returns. Using this, we develop a practical UED algorithm, `NCC` (see Figure 1), which obtains competitive results in several challenging domains.

In this work, we provide a stepping stone towards more robust RL algorithms by creating a theoretically sound optimisation framework that offers competitive empirical results. We contribute the following:

1. A reformulation of UED that is strongly concave in the adversary's strategy, allowing for easier convergence.
2. Convergence guarantees for any score function that is zero-sum with negative return (e.g., regret).
3. An approximation of our objective in the general-sum case, which induces a better curriculum.
4. A *generalised learnability* score function that is applicable to all deterministic domains.
5. An empirical evaluation demonstrating that our new approach either matches or outperforms the evaluation set performance of current state-of-the-art methods in several domains.

## 2 Limitations and Strengths of Related Works

### 2.1 UED with a Level-Sampling Adversary

The most basic UED method is Domain Randomisation (DR), which trains on randomly sampled levels at every training iteration (Tobin et al., 2017). DR is not a particularly effective method with more difficult testbeds (Matthews et al., 2025), however it was highlighted in Coward et al. (2024) that DR performs better than or equivalently to contemporary methods in simpler environments (i.e., mazes with 25 walls instead of 60).

An important takeaway from DR that has been used in other methods like Prioritised Level Replay (Jiang et al., 2021b;a, PLR) and Sampling For Learnability (Rutherford et al., 2024, SFL) is that the adversary will benefit from performing some variation of an $\epsilon$-greedy policy. Namely, there is value in *replaying* previously-sampled levels, while also training on newly-generated levels to ensure diversity. Moreover, PLR improves DR by allowing for a guided search over the level space via a *dynamic* buffer filled with high-scoring levels. SFL builds on this by using the improved score function *learnability*, and replaces a distribution over a dynamic buffer with a heuristic random search to periodically sample a uniform set of levels.

Such sampling-based methods fail to establish a convergent system due to the use of heuristics in place of gradient-based optimisation. Instead, we establish a game objective that is efficiently optimisable with gradients. Moreover, we provide convergence guarantees for regret, but defer theoretical considerations of learnability to future work, as the general-sum setting induces a departure from our current method. In practice we use a dynamic buffer to search over large level spaces, which can be seen as a modified version of SFL to better suit the policy's optimisation process.

### 2.2 UED with a Level-Generating Adversary

PAIRED (Dennis et al., 2020) places neural policies in zero-sum competition with each other, alongside the use of a level-generating adversary . Moreover, PAIRED's generator has seen success when replaced by probabilistic generative models (Azad et al., 2023; Li & Varakantham, 2024; Garcin et al., 2024). Such nonconvex-nonconcave saddle point problems are generally known to have issues with instability and convergence, especially in the study of generative adversarial networks (Wiatrak et al., 2020). Our method does not use a generative model, but unifies the generative and sampling approaches by *learning* the sampling distribution with gradient optimisation.

## 3 Background

### 3.1 Underspecified POMDP's

The underlying theoretical framework behind UED is the Underspecified Partially Observable Markov Decision Process (UPOMDP). The UPOMDP (Dennis et al., 2020) describes an environment with level space $\mathcal{L}$, such that we train over some subset $\Lambda \subseteq \mathcal{L}$, where each parametrisation $\lambda \in \Lambda$ represents a POMDP. We formally define the UPOMDP by the tuple $(\mathcal{L}, \mathcal{S}, \mathcal{O}, \mathcal{A}, r, P, \rho, \gamma)$. We define $\mathcal{S}$ as the state space, $\mathcal{O}$ as the observation space, where each $o \in \mathcal{O}$ is typically a limited view of the global state of the environment. The action space is $\mathcal{A}$, and the reward function is defined as $r : \mathcal{L} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. We additionally define the transition probability function $P$ which has a varying definition depending on the environment dynamics, but for the discrete-state case we write $P : \mathcal{L} \times \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ is the probability simplex of size equal to the cardinality of set $\mathcal{S}$. We finally define the discount factor $\gamma$ and initial state distribution function $\rho : \mathcal{L} \to \Delta(\mathcal{S})$.

Consider the agent's parameter space $\mathcal{X}$, and policy $\pi : \mathcal{X} \times \mathcal{O} \to \Delta(\mathcal{A})$, where the policy may also consider a hidden state $h$ as input in the partially observable setting. We define the expected return of the agent on a given level to be $J : \mathcal{L} \times \mathcal{X} \to \mathbb{R}$. Moreover, we define the general objective of our agent to be the following, where $\Lambda(y)$ is some distribution over levels parametrised by $y$:

$$\max_{x \in \mathcal{X}} \mathbb{E}_{\lambda \sim \Lambda(y)} \Big[ J(\pi_x, \lambda) \Big], \tag{1}$$

### 3.2 Learning in Games

A game is a scenario where there are agents interacting with each other by taking *actions*, typically under the assumption that each agent is trying to maximise their own utility.

**Solutions of Games** *Learning* a game generally involves solving for an equilibrium point between the players via optimisation techniques. Typically, the hope is that the players will achieve a *Nash*

*Equilibrium* (Nash, 1951, NE), which requires that neither player can unilaterally deviate their strategy to obtain a better utility. In such equilibria, players are *robust* to changes to the opponent's strategy. Hence, the robustness guarantees of prior UED works (Dennis et al., 2020; Jiang et al., 2021a) are derived under an assumption that their systems have converged to a NE.

**First-Order Nash Equilibria** Following Nouiehed et al. (2019), we consider the solution concept of the ($\epsilon$-approximate) first-order Nash Equilibrium. For $\epsilon \geq 0$, unconstrained $x$, and $y$ constrained to $\mathcal{Y}$, a first-order NE $(x^*, y^*)$ of the objective $\min_x \max_{y\in\mathcal{Y}} f(x, y)$, is defined by

$$\begin{aligned} \|\nabla_x f(x^*, y^*)\| &\leq \epsilon \\ \max_{y\in\mathcal{Y}} \langle \nabla_y f(x^*, y), y - y^* \rangle &\leq \epsilon \text{ s.t. } \|y - y^*\| \leq 1\,. \end{aligned} \tag{2}$$

An interpretation of the first-order NE is more clear when one considers a single variable and fixes the other; in particular, neither $x$ or $y$ are able to become more optimal w.r.t. $f$ via first-order gradient optimisation except for by some (small) distance $\epsilon$.

### 3.3 Game Theory and UED

**Zero-sum UED** Prior works frame UED as a zero-sum game between an agent (the policy interacting with the environment) and a level-generating adversary (Dennis et al., 2020; Jiang et al., 2021a). The adversary tends to maximise the agent's *score* on the levels. A common score function is *regret*, defined as $\text{Reg}(\pi_x, \lambda) = J(\pi_*^\lambda, \lambda) - J(\pi_x, \lambda)$, for a level $\lambda \in \Lambda$ and its optimal policy $\pi_*^\lambda$. An agent that maximises its expected return on a given level is equivalently minimising its regret, hence the a regret-maximising adversary is zero-sum with a return-maximising agent.

**Minimax Regret** PAIRED (Dennis et al., 2020), uses a generator parameterised by $y$ as their adversary. While not explicitly stated, the objective being optimised is

$$\min_\pi \max_y \; \mathbb{E}_{\lambda\sim\Lambda(y)} \Big[ \text{Reg}(\pi, \lambda) \Big]. \tag{3}$$

We aim to clarify the analysis of PAIRED, specifically with regards to convergence and policy optimisation. In order to construct a *normal form* game (i.e., a game represented by a payoff matrix for each player), it is assumed that the action space of the agent is the (finite) set of possible deterministic policies. Practically, however, PAIRED trains a stochastic neural network-based policy via PPO (Schulman et al., 2017), and is not deterministic during training. In addition, due to the use of nonlinear neural networks for both the policy and the generator, the objective is in fact nonconvex-nonconcave. Hence, the normal-form construction (which is convex-concave) is not a reasonable representation of the UED problem. Instead, we argue that UED should be viewed as a min-max optimisation problem over the parameters of the agent and adversary.

PAIRED further makes two unrealistic assumptions. Primarily, all of the theoretical results hold only *at* Nash Equilibrium, but there is no guarantee that this NE will be reached. In fact, this is unlikely due to the nonconvex-nonconcave objective, alongside the practical difficulties of min-max optimisation with neural networks. The nonconvex-nonconcave setting is not well-understood in the optimisation literature without additional structural assumptions which are not met (Mertikopoulos et al., 2019; Jin et al., 2020; Cai et al., 2024). Consequently, assuming convergence to NE is an assumption unlikely to be met. Secondly, the minimax theorem does not hold on the account of the nonconvexity/nonconcavity of the optimised variables. We circumvent both issues by constructing a nonconvex-strongly-concave objective for UED and proving that the variables involved converge to a first-order NE without needing to invoke the minimax theorem for analysis.

### 3.4 Choice of Score Function

Beyond issues with the theoretical framework of minimax regret, regret is often not a practically viable choice as a score function. While regret incentivises the adversary to propose levels where the agent has much capacity to improve, these levels may not lead to optimal learning, and in fact may not be conducive to learning at all. Regret also relies on the optimal policy, which is generally not available. Additionally, Beukman et al. (2024) established the *regret stagnation* problem, where due to some stochasticity or partial observability in an environment, the regret is not reducible below some non-minimal value[2]. This problem is in particular an issue when utilising a regret-maximising adversary, because the score function is no longer representative of policy learning potential.

One prevailing alternative to regret is the *learnability* of a level (Rutherford et al., 2024). Considering a policy's trajectory distribution on a level to be $\pi_x(\lambda)$, we denote the policy's return on a trajectory $\tau$ to be $R(\tau, \lambda)$. Learnability can be written:

$$\begin{aligned} &\mathrm{Var}_{\tau\sim\pi_x(\lambda)}\Big[R(\tau,\lambda)\Big] \\ &\text{s.t. } R(\tau,\lambda)\in\{0,1\}. \end{aligned} \tag{4}$$

Learnability has a number of interpretations that are explored in Tzannetos et al. (2023) and Rutherford et al. (2024), but the variance interpretation is intuitive in the sense that levels with low variance of returns are rather too difficult or too easy, and should not be prioritised during training. Thus, we would prefer an adversary that maximizes the variance of the level-returns. Notably, Rutherford et al. (2024) only applied this in the goal-based setting, thus having (deterministic) binary return[3].

## 4 Optimising a Curriculum for UED

We first reformulate UED as a regularised game over expected score. We then describe our general algorithm and finally discuss added heuristics that are typical from UED and model-free RL literature.

### 4.1 Core Optimisation Problem

Consider $s : \mathcal{L} \times \mathcal{X} \to \mathbb{R}$ and vector of scores $s(\pi_x, \Lambda)$ corresponding to the scores of policy $\pi_x$ on each level in $\Lambda$. Additionally, we define $\mathcal{Y} := \Delta(\Lambda)$ as the feasibility set for $y$. Motivated by the unstated formulations of Dennis et al. (2020) and Jiang et al. (2021a), we establish the expected score objective for UED, similar to Qian et al. (2019), which is linear in the adversary's strategy:

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} \mathbb{E}_{\lambda\sim\Lambda(y)}\Big[s(\pi_x,\lambda)\Big] = \min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} y^T s(\pi_x,\Lambda)\,. \tag{5}$$

Extending the *soft UED* framework of Chung et al. (2024), we add an entropy regularization term $\mathcal{H}(y) = -y^T \log y$ to the objective of our adversary:

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} f(x,y) \coloneqq \min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} y^T s(\pi_x,\Lambda) + \alpha\mathcal{H}(y)\,, \tag{6}$$

where $\alpha > 0$ is a temperature coefficient. Our justification is twofold: first, as per Chung et al. (2024) the agent needs to train on several different levels at each iteration, requiring the adversary's distribution to have greater entropy instead of collapsing to a single level whose score is largest. We also introduce an additional justification: entropy regularization ensures that $f$ is strongly concave in $y$, guaranteeing best-iterate convergence (Theorem 5.1). In the case of a general score function, the optimisation problem can instead be written as:

$$\min_{x\in\mathcal{X}} f(x,y) = -y^T J(\pi_x,\Lambda)\,,\ \max_{y\in\mathcal{Y}} g(x,y) = y^T s(\pi_x,\Lambda) + \alpha\mathcal{H}(y)\,. \tag{7}$$

Because UED conventionally uses a nonlinear neural network to parameterise its policy (and value function), we have an objective that is nonconvex in $x$ and concave in $y$. Lin et al. (2020) have

[2]For example, if return $R(\cdot,\lambda) \in [0,1]$, the minimal return value would be 0 but $R(\cdot,\lambda)$ might be irreducible below 0.3.
[3]We abuse notation, as an environment could have a goal, but also (for example) could impose penalties on the return for various agent behaviours.

shown that under certain assumptions ($\zeta$-greediness, and typical continuous activation functions), we can guarantee (best-iterate) convergence using *two-timescale* stochastic gradient descent-ascent, which assumes a separation of learning rates. Moreover, there exists optimisation heuristics to scale this approach to neural networks and high-dimensional parameter spaces (Li et al., 2023). Thus, we propose **N**on**C**onvex-**C**oncave optimisation for UED (NCC) as a theoretically based method for optimisation in the UED setting.

### 4.2 Method for Optimisation

To perform gradient based optimisation for the adversary, we construct score vector $\mathbf{s}$ after each iteration of RL training (TRAIN_RL is agnostic to RL algorithm) to construct $y$'s gradient. For the adversary, we perform projected gradient ascent constrained to the probability simplex, and for $x$ we perform unconstrained gradient ascent. The training loop is summarised in Algorithm 1 (illustrated in Figure 1), where $\mathcal{P}_{\mathcal{X}}(\cdot)$ represents the euclidean projection onto set $\mathcal{X}$. For learning rates $\eta_y \gg \eta_x$, and stochastic gradient estimators $\hat{F}$ and $\hat{G}$ defined in Equations (10) and (11), our update rule can be summarised as the following:

$$x^{t+1} = x^t - \eta_x \cdot \hat{F}, \;\; y^{t+1} = \mathcal{P}_{\mathcal{Y}}\big(y^t + \eta_y \cdot \hat{G}\big). \tag{8}$$

In theory, NCC uses a single stochastic gradient step of $x$ using the gradient estimator in Equation (10), and relies on a static buffer. Furthermore, while we make typical assumptions about the policy architecture (like $\zeta$-greediness) for the sake of theoretical analysis, we do not necessarily use these in practice. However, in implementation, we use tricks that have proven practically useful for UED methods in the past, such as dynamic buffer sampling and mini-batch PPO (Schulman et al., 2017). This provides a trade-off between theoretical convergence guarantees and empirical performance.

### 4.3 Searching the level space

Given that the problem has a large enough level space (i.e. $|\mathcal{L}| \gg |\Lambda|$) especially when there are higher chances of sampling levels that are too easy or too hard, it has been demonstrated that intermittently sampling new levels and exchanging them for low-scoring levels in the level buffer is often necessary for good performance (Jiang et al., 2021b).

In consideration of the tradeoff between a simpler optimisation problem with a static buffer and the non-stationary dynamic buffer problem, we assert that the dynamic buffer is more essential to ensuring good performance. Inherently, if the level space contains a high proportion of irrelevant (low-scoring) levels, the initially-sampled $\Lambda$ would lead to a poor training process if it were kept static. Considering this intuition, alongside the empirical results of Jiang et al. (2021b), we consider the non-stationary case in practice.

To implement such a dynamic buffer, we compute the scores of newly sampled levels at every training iteration, and update the buffer with levels of higher score than the lowest in the buffer *prior* to constructing the adversary's gradient. We give the altered procedure for Algorithm 1 in maroon.

### 4.4 Heuristics and General-Sum UED

**General-Sum UED** In practice, we use the same gradient estimators for $x$ and $y$ regardless of score function, and we find this method with general-sum score functions can lead to performance gains. However, with score functions that are not zero-sum with the policy's negative return, our method is a heuristic that lacks convergence guarantees. However, we note that the baselines that we test our method against (i.e., PLR (Jiang et al., 2021b), DR (Tobin et al., 2017), and SFL (Rutherford et al., 2024)) also lack convergence guarantees. We recognise that one could use the theory of bilevel optimisation (Hong et al., 2023) as an approach to provide a provably convergent method, but that adds the complexities of an unconstrained strongly-concave objective for $y$ with an update rule that often leverages second-order information (Chen et al., 2024). Thus, adding theoretical support for general-sum score functions would induce a significant departure from our current method, and is beyond the scope of our work.

**Algorithm 1** Nonconvex-concave Optimisation for UED (Dynamic Buffer)

**Require:** Initial policy $x^0$, distribution $y^0 = \frac{1}{|\Lambda|}\mathbf{1}$, stepsizes $\eta_x, \eta_y$, initial level set $\Lambda^0$.
**for** $t = 0, 1, \ldots$ **do**
  Sample batch of training levels $\boldsymbol{\lambda} \sim \Lambda^t(y^t)$
  Construct score vector $\mathbf{s} = s(\pi_x, \Lambda)$
  Sample new levels $\Lambda' \sim \mathcal{L}$
  Construct alternate score vector $\mathbf{s}' = s(\pi_x, \Lambda')$
  $\Lambda^{t+1} =$ top $|\Lambda|$ elements from $\Lambda^t \cap \Lambda'$
  Construct merged score vector $\tilde{\mathbf{s}} = s(\pi_x, \Lambda^t)$
  $x^{t+1} = \texttt{TRAIN\_RL}(x^t, \boldsymbol{\lambda}, \eta_x)$
  $y^{t+1} = \mathcal{P}_{\mathcal{Y}}\left(y^t + \eta_y\, \hat{G}(x^t, y^t; \mathbf{s}, \tilde{\mathbf{s}})\right)$ with $\hat{G}$ defined in Equation (11)
**end for**
**return** Best-iterate policy parameters $x^*$

**Generalised Learnability** We extend the learnability score function of Rutherford et al., 2024, to general deterministic domains. As in the binary-outcome case, we aim to prioritise levels of intermediate difficulty for the current policy. We start with the standard deviation of the returns for a given level. However, unlike Equation 4, we cannot entirely rely on a variance metric as we empirically find that in several domains, levels where agents do very poorly have a high return variance. In order to bias scoring against levels that are not of intermediate difficulty, we scale the standard error values with a Gaussian over the mean return of the level buffer $\Lambda$. This reduces the score for levels of significant distance from the mean reward. While this approach could bias scores towards levels with a high range of reward outcomes, empirically we do not find this to be an issue.

Concretely, given a set of $\mathcal{M}$ trajectories on level $\lambda \in \Lambda$, we compute the level-wise empirical mean $\mu_\lambda = \frac{1}{|\mathcal{M}|}\sum_{\tau\in\mathcal{M}} R(\tau, \lambda)$, the overall mean $\mu = \frac{1}{|\Lambda|}\sum_{\lambda\in\Lambda}\mu_\lambda$, the level-wise empirical variance $\sigma_\lambda^2 = (\frac{1}{|\mathcal{M}|}\sum_{\tau\in\mathcal{M}} R(\tau,\lambda)^2) - \mu_\lambda^2$, and the overall variance $\sigma^2 = (\frac{1}{|\Lambda|}\sum_\lambda \mu_\lambda^2) - (\frac{1}{|\Lambda|}\sum_\lambda \mu_\lambda)^2$. We use the Gaussian probability density function $\mathcal{N}(\cdot|\mu, \sigma^2)$ to get the level-wise generalised learnability function:

$$\sigma_\lambda \cdot \mathcal{N}(\mu_\lambda|\mu, \sigma^2). \tag{9}$$

In Appendix B.1 we repeat the score function analysis of Rutherford et al. (2024), demonstrating the score function's effectiveness on Minigrid.

## 5 Provably Convergent UED

In order to guarantee convergence in the zero-sum setting, we use two-timescale stochastic gradient descent-ascent (Lin et al., 2020), as given by Algorithm 1, to find an approximate solution to the optimisation problem defined by Equation (6). We first make the necessary assumptions and definitions, and then state the guarantees of our theoretical algorithm.

### 5.1 Preliminaries and Assumptions

**Notation** We denote $F = \nabla_x f(x, y)$, $G = \nabla_y f(x, y)$, and $H = \nabla f = (F, G)$. Moreover, we let $N = |\Lambda|$ be the number of levels and $\lambda_i$ be the $i$-th level. We denote trajectories by $\tau^t = (o^t, a^t, r^t)$, with $\pi_x(\tau^t) \coloneqq \pi_x(a^t|o^t)$ for short. We write $\Psi^t(\tau, \lambda)$ for the baseline used in policy gradients, corresponding either to the advantage function $A^{\pi_x}(\tau^t, \lambda)$ or the return $R(\tau^t, \lambda)$ from time $t$ onwards. Finally, let $M$ be the batch size used for each gradient update.

In order to prove convergence guarantees, we need to make some basic regularity assumptions on the UED and policy network architecture.

**Assumption 1.** The number of levels $N$ and the longest episode length $T$ are finite, and the reward space is bounded. In particular, we denote by $R_* = \max_{\tau,\lambda} |R(\tau, \lambda)|$ the largest absolute return across trajectories and levels.

**Assumption 2.** $\pi_x$ is an $\zeta$-greedy policy parametrised by a continously differentiable, $L$-Lipschitz neural network. This includes any network composed of fully-connected, convolutional or max-pooling layers, dropout, batch normalization and smooth activation functions (e.g. Sigmoid, Softmax, Tanh, ArcTan, SoftPlus, Softsign) (Virmaux & Scaman, 2018). Moreover, we constrain the adversary probability simplex $\mathcal{Y}$ to be $\xi$-truncated, namely, $\mathcal{Y} = \Delta_\xi(\Lambda) := \{y \in \Delta(\Lambda) \mid y_i \geq \xi \; \forall i\}$.

**Gradient Estimators** For the purpose of analysis, we generalize REINFORCE (Williams, 1992) to the UED setting by defining an unbiased estimator for our agent's gradient $F$ as an expectation over levels $\lambda_i$ sampled from $\Lambda(y)$, with a batch size $M$:

$$\hat{F}(x, y) = -\frac{1}{NM} \sum_{i,j} \sum_{t=0}^{T} \nabla_x \log \pi_x(\tau_{ij}^t, \lambda_i) \Psi^t(\tau_{ij}, \lambda_i)\,, \tag{10}$$

where $\lambda_i \sim \Lambda(y)$ and trajectories $\tau_{ij} \sim \pi_x(\lambda_i)$ are sampled independently. For the adversary, the unbiased estimator gradient is similarly given by

$$\hat{G}(x, y) = \hat{s}(\pi_x, \Lambda) + \alpha \nabla_y \mathcal{H}(y)\,, \tag{11}$$

where $\hat{s}$ is the empirical score vector, given by $\hat{s}(\pi_x, \lambda_i) = -\frac{1}{M} \sum_j R(\tau_{ij}, \lambda_i)$ for $s = -J$ and $\hat{s}(\pi_x, \lambda_i) = \max_\tau R(\tau, \lambda_i) - \frac{1}{M} \sum_j R(\tau_{ij}, \lambda_i)$ for $s = \text{Reg}$.

### 5.2 Convergence Guarantees

**Proposition 1.** Under Assumptions 1 and 2, the estimator $\hat{H} = (\hat{F}, \hat{G})$ defined by Equations (10) and (11), with $M = 1$, has $\sigma^2$-bounded variance with

$$\sigma^2 = 4R_*^2 \left( N + \frac{T^2 L^2}{\zeta^2} \right).$$

Moreover, the corresponding objective function $f(x, y) = \mathbb{E}_\lambda [s(\pi_x, \lambda)] + \alpha \mathcal{H}(y)$ is $\alpha$-strongly concave in $y$ and $\sigma$-Lipschitz. Finally, $\pi_x$ is $K$-smooth for some $K \in \mathbb{R}$ and $f$ is $\ell$-smooth with

$$\ell = \frac{TR_*}{\zeta} \left( TL^2 + K + \frac{L^2}{\zeta} + 2TL \right) + \frac{\alpha}{\xi}.$$

*Proof.* In Appendix A. □

**Theorem 5.1** (Best-Iterate Convergence). *Under Assumptions 1 and 2, let $\sigma, \ell$ be the constants defined in Proposition 1, $\alpha$ the entropy temperature, and $\Delta = \max_y f(x^0, y) - \max_y f(x^*, y)$ the objective distance between initial and optimal policies. For learning rates $\eta_x = \Theta(\alpha^2/\ell^3)$ and $\eta_y = \Theta(1/\ell)$, and a batch size $M = \Theta(\max\{1, \sigma^2 \ell / \alpha \epsilon^2\})$, Algorithm 1 finds an $\epsilon$-stationary policy $\pi_{x^*}$ such that $\|\nabla_x \max_y f(x^*, y)\| < \epsilon$ in*

$$O\left( \frac{\Delta \ell^3}{\alpha^2 \epsilon^2} + \frac{2\ell^3}{\alpha \epsilon^4} \right)$$

*iterations, provided* TRAIN_RL *consists of a single gradient step using the estimator $\hat{F}$.*

*Proof.* Apply Proposition 1 and Lin et al. (2020, Theorem 4.5), with $D \leq \sqrt{2}$ being the diameter of the $\xi$-truncated probability simplex $\mathcal{Y}$ and $\kappa = \ell/\alpha$ the condition number. □

We remark that while we are only concerned with finding a stationary policy $\pi_{x^*}$ in the UED setting, the corresponding optimal distribution $y^* = \operatorname{argmax}_{y \in \mathcal{Y}} f(x^*, y)$ can efficiently be computed via projected gradient ascent due to the strong concavity of $f$ in $y$. The resulting point $(x^*, y^*)$ meets the conditions of Equation (2), and is therefore an $\epsilon$-approximate first-order Nash Equilibrium.

# 6 Experiments

Alongside our theoretical considerations, our method outperforms contemporary works on UED benchmarks after being extended to a practical algorithm. In this section we detail the choice of benchmarks and provide an experimental evaluation of our method's performance.

## 6.1 Experimental Setup

We test our policy on benchmarks from Rutherford et al. (2024), although we decline to use JaxNav as the single-agent setting's results are highly saturated, and the multi-agent setting introduces additional optimisation challenges. Thus, we report results on Minigrid (Chevalier-Boisvert et al., 2023) using the implementation from Coward et al. (2024) and XLand-Minigrid (Nikulin et al., 2023). We refer the reader to Rutherford et al. (2024) for more details on the individual environments. Additionally, we show that our method can obtain competitive performance on a more complex benchmark, *Craftax* (Matthews et al., 2024).

We first observe that our method achieves performance that is competitive to other approaches in a simple environment, and then highlight our method's performance on more difficult benchmarks. All provided plots refer to our method as "NCC" with the attached score function (see Rutherford et al. (2024) for a further discussion of such score functions). We test NCC with both (generalised) learnability and positive value loss (PVL), the latter of which is a regret approximation that we found to lead to more stable training than MaxMC. Our contributions are labeled with **bold** font in the plots. For Minigrid, we use a bar plot for the sake of easier interpretation, whereas for XLand-Minigrid and Craftax we use curve plots to display our final result and rate of learning.

Experiments were written in JAX (Bradbury et al., 2018) and we perform all experiments on a single NVIDIA L40S GPU. Results are averaged across 10 seeds, and standard error from the mean is displayed in the plots. All experiments use PPO (Schulman et al., 2017) as the RL algorithm of choice, although we remark that in theory our method could be used with other algorithms. For NCC we use the TiAda-Adam optimiser (Li et al., 2023) for both our policy and adversary, but for the rest of the benchmarks we used Adam (Kingma & Ba, 2017), as TiAda-Adam is only defined when there are gradients for both $x$ and $y$.

We include all hyperparameters for our experiments in Appendix B. We largely use the hyperparameters from (Rutherford et al., 2024) and (Matthews et al., 2024) when available. We highlight in Appendix C the evolution of Minigrid examples throughout training, and show that our method weighs levels that are more challenging but still solvable in comparison to DR and SfL.

## 6.2 Results

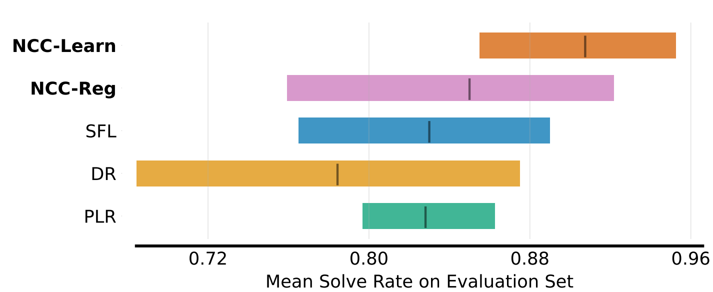
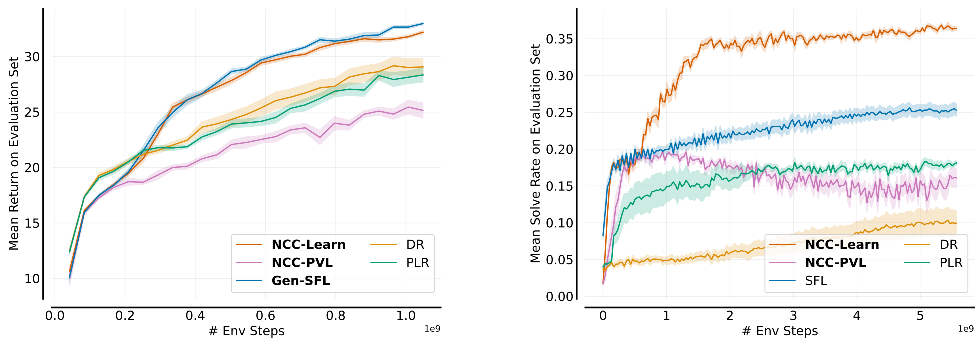



Figure 2: Mean solve rates with 95% confidence intervals on Minigrid, a common UED testbed.

**Minigrid** We observe similar performance to SFL on Minigrid with 60-wall mazes, on the same held-out set of test-levels. Due to the closeness of results, we plot our final mean solve rates, in addition to a 95% confidence interval using the analysis from Agarwal et al. (2021), but using mean instead of interquartile mean to match the other plots. As per the results in Coward et al. (2024) and Rutherford et al. (2024), Minigrid does not leave much room for improvement as a benchmark, and thus there is only a small performance difference across methods. Additionally, because we have access to an optimal return oracle for Minigrid, we test true regret instead of PVL or MaxMC. Despite the stauration, we report results on Minigrid to verify that our method is successful in the benchmarks of prior work.




Craftax XLand-Minigrid

Figure 3: Performance on more difficult benchmarks, with our contributions highlighted in **bold**.

**XLand-Minigrid** Our most significant improvement from prior work is in XLand-Minigrid. We note that out of the given testbeds, this environment has results that are less saturated, and thus leave more room for improvement. We offer a considerably improved solve rate as compared to prior works, although we remark that this is only the case when using learnability as the score function.

**Craftax** We use our new generalisation of learnability to outperform the highest-performing UED baseline from Matthews et al. (2024, PLR-MaxMC), although we find that domain randomisation to perform slightly better than PLR. We remark that we find performance is stronger in Craftax with a *static* buffer, and we highlight this to mention that in environments with a higher density of "good" levels, it may not be necessary to use a dynamic buffer.[4] Moreover, due to such a level space, we found that it more effective to *anneal* our entropy regularisation coefficient $\alpha$ by the rule $\alpha^t = \frac{\alpha}{\sqrt[3]{t+1}}$, thus resulting in a more diverse set of training levels at the start of training. We maintain the training regime of Matthews et al. (2024) by using "inner" and "outer" rollouts, where we update after multiple parallelised sub-sequences within an episode. However, when computing our score vector we roll-out our policy for 500 steps per level. For generalised SFL, our learnability batch set size was 20000, buffer size was 4000, and we changed our buffer every 10 iterations. We attribute similar performance across generalised SFL and NCC with learnability to the algorithms' shared emphasis on levels with high learnability.

## 7 Future Work

Our work obtains best-iterate convergence guarantees, which is commonplace in nonconvex minimax optimisation (Lin et al., 2020; Kalogiannis et al., 2024). However, last-iterate convergence is a more desirable property that has been widely explored in game theory literature (Daskalakis & Panageas, 2020; Lei et al., 2021), and we leave open the question of last-iterate convergence in the UED setting to future work that corresponds with future advancements in the optimisation literature (Lin et al., 2020). Moreover, while it may be possible to analyse the general-sum UED setting under the lens of bilevel optimisation (Hong et al., 2023), we would suggest that future work investigates

[4]In Craftax, the DR distribution of levels is the same as the evaluation distribution.

practical and more scalable ways to produce convergent methods when the zero-sum condition is not met. Finally, considering the emergence of analysis of more sophisticated reinforcement learning algorithms like PPO (Grudzien et al., 2022), we suggest future work that further analyses the more practical variant of our algorithm.

## 8 Conclusion

In this paper, we build on the work of prior UED methods in order to establish the area's first provably convergent method. We incorporate strengths from prior works alongside introducing a new score function that generalises learnability to arbitrary deterministic settings. Moreover, we extend our method to include practicalities such as PPO (Schulman et al., 2017) and specialised optimisers (Li et al., 2023) that achieve competitive performance on UED benchmarks with a wide range of complexity. Ultimately, we believe that our method provides a gateway to the creation of practical robust RL methods with guarantees under reasonable assumptions.

## A Proof of Proposition 1

**Proposition 1.** Under Assumptions 1 and 2, the estimator $\hat{H} = (\hat{F}, \hat{G})$ defined by Equations (10) and (11), with $M = 1$, has $\sigma^2$-bounded variance with

$$\sigma^2 = 4R_*^2 \left( N + \frac{T^2 L^2}{\zeta^2} \right) .$$

Moreover, the corresponding objective function $f(x, y) = \mathbb{E}_\lambda [s(\pi_x, \lambda)] + \alpha \mathcal{H}(y)$ is $\alpha$-strongly concave in $y$ and $\sigma$-Lipschitz. Finally, $\pi_x$ is $K$-smooth for some $K \in \mathbb{R}$ and $f$ is $\ell$-smooth with

$$\ell = \frac{TR_*}{\zeta} \left( TL^2 + K + \frac{L^2}{\zeta} + 2TL \right) + \frac{\alpha}{\xi} .$$

*Proof.* For simplicity, recall Equations (10) and (11) with $M = 1$ from the main text:

$$\hat{F}(x, y) = -\frac{1}{N} \sum_i \sum_{t=0}^{T} \nabla_x \log \pi_x(\tau_i^t, \lambda_i) \Psi^t(\tau_i, \lambda_i) ,$$
$$\hat{G}(x, y) = \hat{s}(\pi_x, \Lambda) + \alpha \nabla_y \mathcal{H}(y) ,$$

where $\lambda_i \sim \Lambda(y)$ and $\tau_i \sim \pi_x(\lambda_i)$ are sampled independently for each level $i$, and $\hat{s}$ is the empirical score vector given by $\hat{s}(\pi_x, \lambda_i) = -R(\tau_i, \lambda_i)$ for $s = -J$ and $\hat{s}(\pi_x, \lambda_i) = \max_\tau R(\tau, \lambda_i) - R(\tau_i, \lambda_i)$ for $s = \text{Reg}$. Finally, denote $z = (x, y)$ for joint parameters.

**(1) Bounded variance.** First note that the variance of the entropy term is zero, hence

$$\begin{aligned} \mathbb{E}\left[\left\|\hat{H}(z) - H(z)\right\|^2\right] &= \mathbb{E}\left[\left\|\hat{F}(z) - F(z)\right\|^2\right] + \mathbb{E}\left[\|\hat{s}(\pi_x, \Lambda) - s(\pi_x, \Lambda)\|^2\right] \\ &\leq \mathbb{E}\left[\left\|\hat{F}(z)\right\|^2\right] + \mathbb{E}\left[\|\hat{s}(\pi_x, \Lambda)\|^2\right] , . \end{aligned}$$

For the second term, we easily obtain, for $M = 1$,

$$\mathbb{E}\left[\|\hat{s}(\pi_x, \Lambda)\|^2\right] \leq \sum_i \mathbb{E}\left[\hat{s}(\pi_x, \lambda_i)^2\right] \leq 4NR_*^2$$

for both $s = -J$ and $s = \text{Reg}$. For the first term, we invoke Lipschitzness and $\zeta$-greediness of the policy $\pi$. For any trajectory $\tau$ and any level $\lambda$, we have $|\sum_t \Psi_t(\tau, \lambda)| \leq 2TR_*$ for both $\Psi^t(\tau, \lambda) = R^t(\tau, \lambda)$ and $\Psi^t(\tau, \lambda) = A^{\pi_x}(\tau^t, \lambda)$, which combined with

$$\left\|\nabla_x \log \pi_x(\tau^t, \lambda)\right\| = \frac{\|\nabla_x \pi_x(\tau^t, \lambda)\|}{\pi_x(\tau^t, \lambda)} \leq \frac{L}{\zeta} ,$$

implies that

$$\mathbb{E}\left[\left\|\hat{F}(z)\right\|^2\right] \le \left\|\sum_t \nabla_x \log \pi_x(\tau^t,\lambda)\Psi^t(\tau,\lambda)\right\|^2 \le \frac{4T^2R_*^2L^2}{\zeta^2}$$

and hence

$$\mathbb{E}\left[\left\|\hat{H}(z)-H(z)\right\|^2\right] \le 4NR_*^2 + \frac{4T^2R_*^2L^2}{\zeta^2} = \sigma^2$$

as required.

**(2) Strong concavity of $f$ in $y$.** Trivial, since $\nabla_y^2 f(x,y) = \mathrm{diag}(-\alpha/y) \preceq -\alpha I$.

**(3) Lipschitzness of $f$.** Follows directly from part **(1)** of the proof by applying Jensen's inequality:

$$\|\nabla f(z)\|^2 = \left\|\mathbb{E}\left[\hat{H}(z)\right]\right\|^2 \le \mathbb{E}\left[\|\hat{H}(z)^2\|\right] \le \sigma^2 \,.$$

**(4) Smoothness of $\pi$.** The composition of Lipschitz functions is Lipschitz, so we need only show that the gradient of any layer in the neural network is Lipschitz. This trivially holds for fully-connected, convolutional, max-pooling and batch-norm since they are piecewise linear, while the second derivative of the chosen activation functions (Sigmoid, Softmax, Tanh, ArcTan, SoftPlus, Softsign) are all bounded. It follows that under Assumption 2, $\left\|\nabla_x^2\pi\right\| \le K$ for some $K \in \mathbb{R}$ which is upper-bounded by the product of Lipschitz constants for each layer's gradient.

**(5) Smoothness of $f$.** The policy gradient Hessian is given by (Shen et al., 2019)

$$\nabla_x^2 J(\pi_x,\lambda) = \mathbb{E}_\tau\left[\sum_t R^t(\tau,\lambda)\Big(\nabla_x \log \pi_x(\tau^t)\nabla_x \log p(\tau \mid \pi_x)^T + \nabla_x^2 \log \pi_x(\tau^t)\Big)\right]$$

where $p(\tau \mid \pi_x) = \rho(s_0)\prod_t P(s_{t+1} \mid s_t, a_t)\pi_x(\tau^t)$ for initial and transition distributions $\rho$ and $P$ (omitting $\lambda$ for convenience). For the first term, writing $P(\tau) = \prod_t P(s_{t+1} \mid s_t, a_t)$, we have

$$\nabla_x p(\tau \mid \pi_x) = \rho(s_0)P(\tau)\sum_t \nabla_x \pi_x(\tau^t)\prod_{s\neq t}\pi_x(\tau^t)$$

which implies

$$\|\nabla_x p(\tau \mid \pi_x)\| \le TL\,,$$

hence the first term is bounded for each $t$:

$$\left\|\nabla_x \log \pi_x(\tau^t)\nabla_x \log p(\tau \mid \pi_x)^T\right\| \le \frac{TL^2}{\zeta}\,.$$

For the second term, we use the smoothness of $\pi$ proven in part **(4)** above to obtain:

$$\left\|\nabla_x^2 \log \pi_x(\tau^t)\right\| = \left\|\frac{\nabla_x^2\pi_x(\tau^t)}{\pi_x(\tau^t)} - \frac{\nabla_x\pi_x(\tau^t)\nabla_x\pi_x(\tau^t)^T}{\pi_x(\tau^t)^2}\right\| \le \frac{K}{\zeta} + \frac{L^2}{\zeta^2}\,.$$

Putting everything together, we obtain

$$\left\|\nabla_x^2 f(z)\right\| = \left\|y^T\nabla_x^2 J(\pi_x,\Lambda)\right\| \le \max_\lambda \left\|\nabla_x^2 J(\pi_x,\lambda)\right\| \le \frac{TR_*}{\zeta}\left(TL^2 + K + \frac{L^2}{\zeta}\right)\,.$$

Now notice that $\left\|\nabla_y^2 f(z)\right\| = \|\mathrm{diag}(-\alpha/y)\| \le \alpha/\xi$ since $y_i \le \xi$ for all $i$. Moreover,

$$\left\|\nabla_{xy}^2 f(z)\right\| = \|\nabla_x J(\pi_x,\Lambda)\| \le \frac{2TR_*L}{\zeta}$$

by the same argument as part **(1)** of the proof, so we conclude

$$\left\|\nabla^2 f(z)\right\| \le \frac{TR_*}{\zeta}\left(TL^2 + K + \frac{L^2}{\zeta} + 2TL\right) + \frac{\alpha}{\xi} = \ell$$

as required. □

## References


Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Abdus Salam Azad, Izzeddin Gur, Jasper Emhoff, Nathaniel Alexis, Aleksandra Faust, Pieter Abbeel, and Ion Stoica. Clutr: curriculum learning via unsupervised task representation learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Michael Beukman, Samuel Coward, Michael Matthews, Mattie Fellows, Minqi Jiang, Michael Dennis, and Jakob Foerster. Refining minimax regret for unsupervised environment design. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Yang Cai, Argyris Oikonomou, and Weiqiang Zheng. Accelerated algorithms for constrained nonconvex-nonconcave min-max optimization and comonotone inclusion. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5312–5347. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/cai24f.html.

Lesi Chen, Jing Xu, and Jingzhao Zhang. On finding small hyper-gradients in bilevel optimization: Hardness results and improved analysis. In Shipra Agrawal and Aaron Roth (eds.), *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pp. 947–980. PMLR, 30 Jun–03 Jul 2024. URL https://proceedings.mlr.press/v247/chen24a.html.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and J Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 73383–73394. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/e8916198466e8ef218a2185a491b49fa-Paper-Datasets_and_Benchmarks.pdf.

Hojun Chung, Junseo Lee, Minsoo Kim, Dohyeong Kim, and Songhwai Oh. Adversarial environment design via regret-guided diffusion models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 63715–63746. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/74953ef4abd9c436344e59d687ad34d3-Paper-Conference.pdf.

Samuel Coward, Michael Beukman, and Jakob Foerster. Jaxued: A simple and useable ued library in jax, 2024. URL https://arxiv.org/abs/2403.13091.

Constantinos Daskalakis and Ioannis Panageas. Last-iterate convergence: Zero-sum games and constrained min-max optimization, 2020. URL https://arxiv.org/abs/1807.04252.

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. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13049–13061. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/985e9a46e10005356bbaf194249f6856-Paper.pdf.

Samuel Garcin, James Doran, Shangmin Guo, Christopher G. Lucas, and Stefano V. Albrecht. Dred: Zero-shot transfer in reinforcement learning via data-regularised environment design. JMLR.org, 2024.

Jakub Grudzien, Christian A Schroeder De Witt, and Jakob Foerster. Mirror learning: A unifying framework of policy optimisation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7825–7844. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/grudzien22a.html.

Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale stochastic algorithm framework for bilevel optimization: Complexity analysis and application to actor-critic. *SIAM Journal on Optimization*, 33(1):147–180, 2023. DOI: 10.1137/20M1387341. URL https://doi.org/10.1137/20M1387341.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 1884–1897. Curran Associates, Inc., 2021a. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/0e915db6326b6fb6a3c56546980a8c93-Paper.pdf.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 18–24 Jul 2021b. URL https://proceedings.mlr.press/v139/jiang21b.html.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Andrei Lupu, Heinrich Küttler, Edward Grefenstette, Tim Rocktäschel, and Jakob Foerster. Grounding aleatoric uncertainty for unsupervised environment design. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 32868–32881. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/d3e2d61af1e9612ddecd099144e50404-Paper-Conference.pdf.

Chi Jin, Praneeth Netrapalli, and Michael Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4880–4889. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/jin20e.html.

Fivos Kalogiannis, Jingming Yan, and Ioannis Panageas. Learning equilibria in adversarial team markov games: A nonconvex-hidden-concave min-max optimization problem. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 92832–92890. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/a8bc668b1559d705221b0e7510c45e48-Paper-Conference.pdf.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Qi Lei, Sai Ganesh Nagarajan, Ioannis Panageas, and xiao wang. Last iterate convergence in no-regret learning: constrained min-max optimization for convex-concave landscapes. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 1441–1449. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/lei21a.html.

Dexun Li and Pradeep Varakantham. Enhancing the hierarchical environment design via generative trajectory modeling, 2024. URL https://arxiv.org/abs/2310.00301.

Xiang Li, Junchi YANG, and Niao He. Tiada: A time-scale adaptive algorithm for nonconvex minimax optimization. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=zClyiZ5V6sL.

Tianyi Lin, Chi Jin, and Michael Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6083–6093. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/lin20a.html.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2024.

Michael Matthews, Michael Beukman, Chris Lu, and Jakob Nicolaus Foerster. Kinetix: Investigating the training of general agents through open-ended physics-based control tasks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zCxGCdzreM.

Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra(-gradient) mile. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg8jjC9KQ.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. In T. Leen, T. Dietterich, and V. Tresp (eds.), *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL https://proceedings.neurips.cc/paper_files/paper/2000/file/e8dfff4676a47048d6f0c4ef899593dd-Paper.pdf.

J.F. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Viacheslav Sinii, Artem Agarkov, and Sergey Kolesnikov. XLand-minigrid: Scalable meta-reinforcement learning environments in JAX. In *Intrinsically-Motivated and Open-Ended Learning Workshop, NeurIPS2023*, 2023. URL https://openreview.net/forum?id=xALDC4aHGz.

Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D Lee, and Meisam Razaviyayn. Solving a class of non-convex min-max games using iterative first order methods. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/25048eb6a33209cb5a815bff0cf6887c-Paper.pdf.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design, 2023. URL https://arxiv.org/abs/2203.01302.

Qi Qian, Shenghuo Zhu, Jiasheng Tang, Rong Jin, Baigui Sun, and Hao Li. Robust optimization over multiple domains. AAAI Press, 2019. ISBN 978-1-57735-809-1. DOI: 10.1609/aaai.v33i01.33014739. URL https://doi.org/10.1609/aaai.v33i01.33014739.

Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 16071–16101. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1d0ed12c3fda52f2c241a0cebcf739a6-Paper-Conference.pdf.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zebang Shen, Alejandro Ribeiro, Hamed Hassani, Hui Qian, and Chao Mi. Hessian aided policy gradient. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5729–5738. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/shen19d.html.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. IEEE Press, 2017. DOI: 10.1109/IROS.2017.8202133. URL https://doi.org/10.1109/IROS.2017.8202133.

Georgios Tzannetos, Bárbara Gomes Ribeiro, Parameswaran Kamalaruban, and Adish Singla. Proximal Curriculum for Reinforcement Learning Agents. *Transactions of Machine Learning Research (TMLR)*, 2023.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/d54e99a6c03704e95e6965532dec148b-Paper.pdf.

Maciej Wiatrak, Stefano V. Albrecht, and Andrew Nystrom. Stabilizing generative adversarial networks: A survey, 2020. URL https://arxiv.org/abs/1910.00927.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. ISSN 0885-6125. DOI: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Sihan Zeng and Thinh T. Doan. Fast two-time-scale stochastic gradient method with applications in reinforcement learning, 2024. URL https://arxiv.org/abs/2405.09660.

# Supplementary Materials

*The following content was not necessarily subject to peer review.*

## B Additional Experimental Details

### B.1 Generalised Learnability Score Function

In Figure 4 we repeat the analysis of UED score functions conducted by Rutherford et al. (2024). To give us a success rate metric, we conduct this analysis in Minigrid using a policy trained for 1100 update steps with SFL (i.e. $1/4$ of a usual training run). We randomly sample 5000 levels and rollout the policy for 2000 timesteps on each. The trend illustrated by the quadratic demonstrates the generalised learnability score function's ability to identify levels of intermediate difficulty.

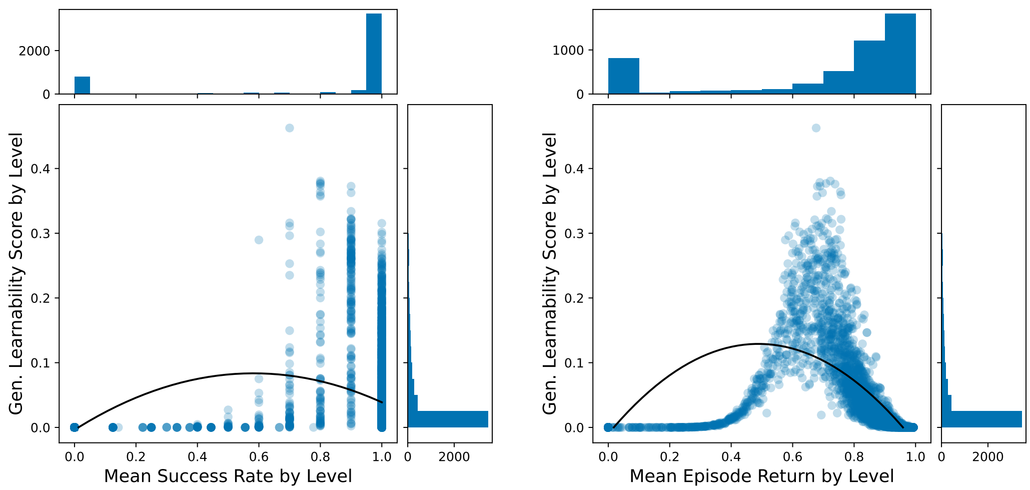



Figure 4: Analysis of Generalised Learnability Score function on Minigrid. The black lines represent a quadratic fit to the scatter data.

### B.2 Compute Time

Table 1 reports the compute time for all experimental evaluations. Each Minigrid seed was run on 1 Nvidia A40 using a server that has 8 Nvidia A40's and two AMD EPYC 7513 32-Core Processor (64 cores in total). Meanwhile, for XLand and Craftax, each individual seed was run on 1 Nvidia L40s using a server that has 8 NVIDIA L40s', two AMD EPYC 9554 processors (128 cores in total).

| Method | Minigrid | XLand | Craftax |
|---|---|---|---|
| NCC Learn | 0:53:45 (0:00:28) | 3:31:57 (0:01:06) | 5:35:06 (0:00:59) |
| NCC Regret | 0:58:41 (0:00:29) | - | - |
| NCC PVL | - | 2:52:31 (0:00:53) | 4:13:16 (0:00:39) |
| SFL | 0:28:19 (0:00:03) | 9:16:31 (0:01:17) | 4:29:17 (0:00:31) |
| PLR | 0:45:16 (0:00:10) | 2:47:53 (0:00:47) | 3:24:46 (0:00:24) |
| DR | 0:43:28 (0:00:16) | 2:42:27 (0:00:43) | 3:28:06 (0:00:08) |

Table 1: Mean and standard deviation of time take for experimental evaluations. Each evaluation consisted of 10 independent seeds.

574 ### B.3 Hyperparameters

| Hyperparameter | Minigrid | XLand | Craftax |
|---|---|---|---|
| $\eta_x$ | 0.001 | 0.0001 | 0.0001 |
| $\eta_y$ | 0.05 | 0.01 | 0.01 |
| $\alpha$ | 0.05 | 0 | 0.05 |
| $\|\Lambda\|$ | 4000 | 4000 | 4000 |
| $\|\Lambda'\|$ | 256 | 8192 | 0 |
| $\|\boldsymbol{\lambda}\|$ | 256 | 8192 | 1024 |
| $\gamma$ | 0.995 | 0.99 | 0.995 |
| GAE $\lambda$ | 0.98 | 0.95 | 0.95 |
| `clip_eps` | 0.2 | 0.2 | 0.2 |
| `critic_coeff` | 0.5 | 0.5 | 0.5 |
| `entropy_coeff` | 0.001 | 0.01 | 0.01 |
| `num_epochs` | 1 | 1 | 4 |
| `max_grad_norm` | 0.5 | 0.5 | 1.0 |
| `num_minibatches` | 1 | 16 | 2 |
| `num_parallel_envs` | 256 | 8192 | 1024 |

Figure 5: NCC Hyperparameters

| Hyperparameter | Minigrid | XLand | Craftax |
|---|---|---|---|
| $\eta_x$ | 0.00025 | 0.0001 | 0.0002 |
| $\|\Lambda\|$ | 4000 | 4000 | 4000 |
| $\gamma$ | 0.995 | 0.99 | 0.99 |
| GAE $\lambda$ | 0.98 | 0.95 | 0.9 |
| `clip_eps` | 0.2 | 0.2 | 0.2 |
| `critic_coeff` | 0.5 | 0.5 | 0.5 |
| `entropy_coeff` | 0 | 0.01 | 0.01 |
| `num_epochs` | 4 | 1 | 5 |
| `max_grad_norm` | 0.5 | 0.5 | 1.0 |
| `num_minibatches` | 4 | 16 | 2 |
| `num_parallel_envs` | 256 | 8192 | 1024 |
| `replay_prob` | 0.5 (0) | 0.95 (0) | 0.5 (0) |
| `staleness_coeff` | 0.3 | 0.3 | 0.3 |
| `temperature` | 1 | 1 | 1 |

Figure 6: PLR (DR) Hyperparameters

| Hyperparameter | Minigrid | XLand | Craftax |
|---|---|---|---|
| $\eta_x$ | 0.00025 | 0.001 | 0.0001 |
| $\lvert\Lambda\rvert$ | 1000 | 8192 | 4000 |
| $\gamma$ | 0.99 | 0.99 | 0.995 |
| GAE $\lambda$ | 0.95 | 0.95 | 0.95 |
| `clip_eps` | 0.04 | 0.2 | 0.2 |
| `critic_coeff` | 0.5 | 0.5 | 0.5 |
| `entropy_coeff` | 0 | 0.01 | 0.01 |
| `num_epochs` | 4 | 1 | 4 |
| `max_grad_norm` | 0.5 | 0.5 | 1.0 |
| `num_minibatches` | 4 | 16 | 2 |
| `num_parallel_envs` | 256 | 8192 | 1024 |
| `batch_size` | 1000 | 40000 | 4000 |
| `num_batches` | 5 | 1 | 5 |

Figure 7: SFL Hyperparameters

# C Difficulty of Levels

To show how our method evolves over time, we compare minigrid levels at halfway and final timesteps in training. Firstly, we plot levels from DR in Figure 8. Levels from DR are not well selected, as there are unsolvable levels, as well as *trivial* levels at the end of training. Secondly, as is explainable by them both selecting for learnability, NCC with learnability (Figure 10) and SFL (Figure 9) both have what appear to be difficult (but not impossible) levels halfway and at the end of training, although we do note that NCC appears to weigh some levels with shorter optimal paths at the end of training in comparison to SFL (particularly the left and middle levels of NCC). This may be to retain diversity in the difficulty of the batch of sampled levels, to prevent overfitting to a certain class of problems. However, our analysis is a hypothesis, as our approach is learned, meaning it is more black-box (i.e. uninterpretable).

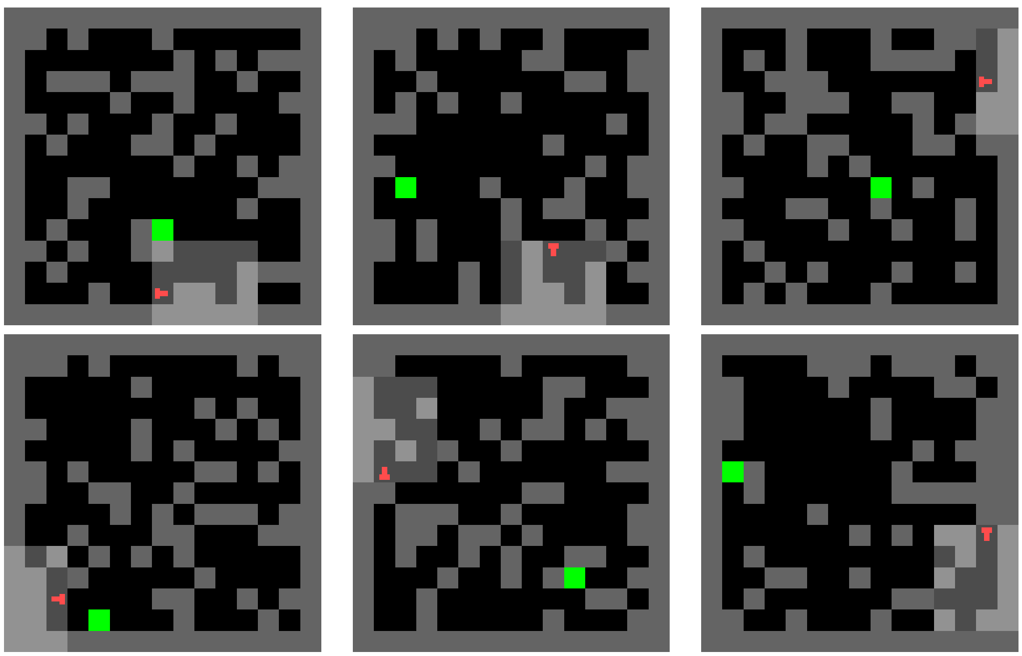


Figure 8: DR: Sampled levels at halfway through training (top row) and the end of training (bottom row)

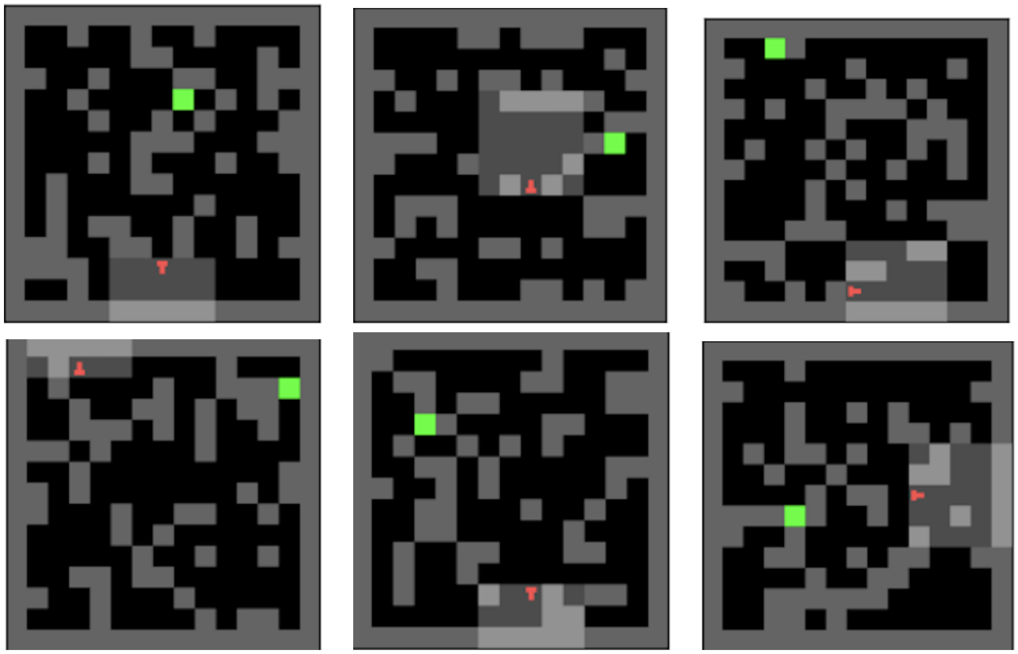


Figure 9: SFL: Highest learnability scoring levels at halfway through training (top row) and the end of training (bottom row)

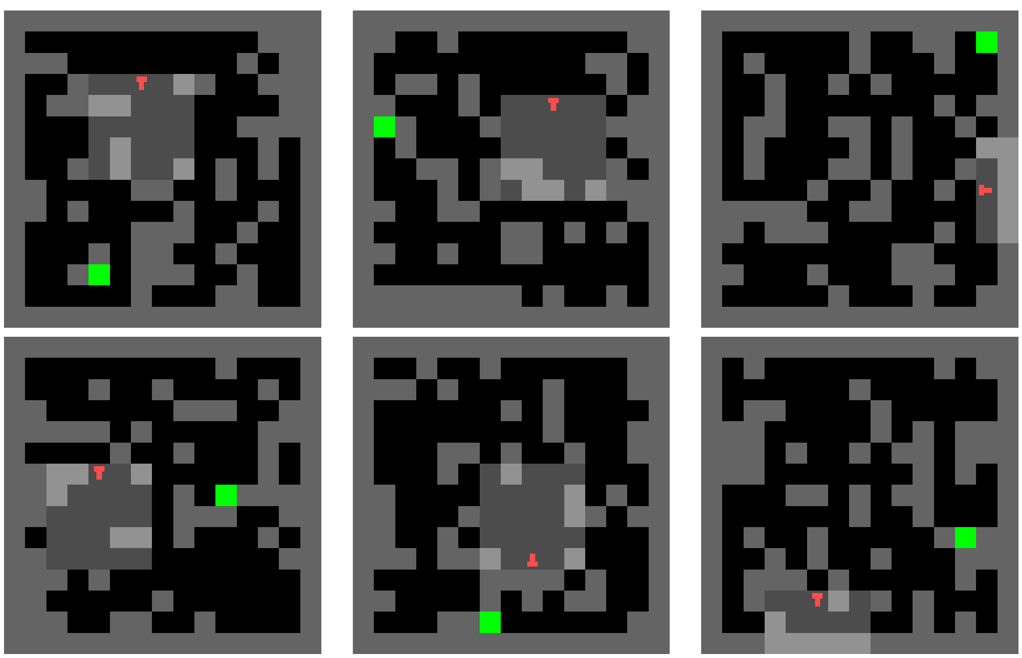


Figure 10: NCC-Learn: Highest weighted levels at halfway through training (top row) and the end of training (bottom row)