An Optimisation Framework for Unsupervised Environment Design

Nathan Monette, Alistair Letcher, Michael Beukman, Matthew T. Jackson, Alexander Rutherford, Alexander D. Goldie, Jakob N. Foerster

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 approach for improving robustness is unsupervised environment design (UED), a suite of methods that aim to maximise an agent's generalisability by training it on a wide variety of environment configurations. 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 on two of three 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 guarantee convergence 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 by introducing a new score function that generalises the work of Rutherford et al. (2024) to general deterministic RL environments.
   Context: Learnability (Rutherford et al., 2024) requires a binary-outcome environment.

An Optimisation Framework for Unsupervised Environment Design

Nathan Monette[1, †, *], **Alistair Letcher**[2], **Michael Beukman**[2], **Matthew T. Jackson**[2], **Alexander Rutherford**[2], **Alexander D. Goldie**[2], **Jakob N. Foerster**[2]
[†]nmonette@uci.edu

[1]**University of California Irvine**
[2]**FLAIR, University of Oxford**

[*]Work undertaken while visiting FLAIR.

Abstract

For reinforcement learning agents to be deployed in high-risk settings, they must achieve a high level of robustness to unfamiliar scenarios. One approach for improving robustness is unsupervised environment design (UED), a suite of methods that aim to maximise an agent's generalisability by training it on a wide variety of environment configurations. 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 on two of three environments with varying difficulties.[1]

1 Introduction

Training reinforcement learning (RL) agents that are robust to a variety of unseen scenarios is an important and long-standing challenge in the field (Morimoto & Doya, 2000; Tobin et al., 2017). One promising approach to addressing this problem is Unsupervised Environment Design (UED), where an adversary automatically proposes a diverse range of training tasks for an agent to learn on, based on its current capabilities (Dennis et al., 2020; Jiang et al., 2021a). In this way, the agent gradually progresses from easy tasks (also called levels) to more difficult ones.

UED is generally posed as a two-player game between a level-selecting adversary and a level-solving agent. 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 (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 on a specific level, i.e., how much it can still improve. 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* of a level. The canonical formulation of UED faces several other challenges, including lack of convergence guarantees 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. Using this reformulation, we provide a gradient-based algorithm that is *provably convergent* to a locally optimal policy, relying on recent results (Lin et al.,

[1]Our code is available at https://github.com/nmonette/NCC-UED.

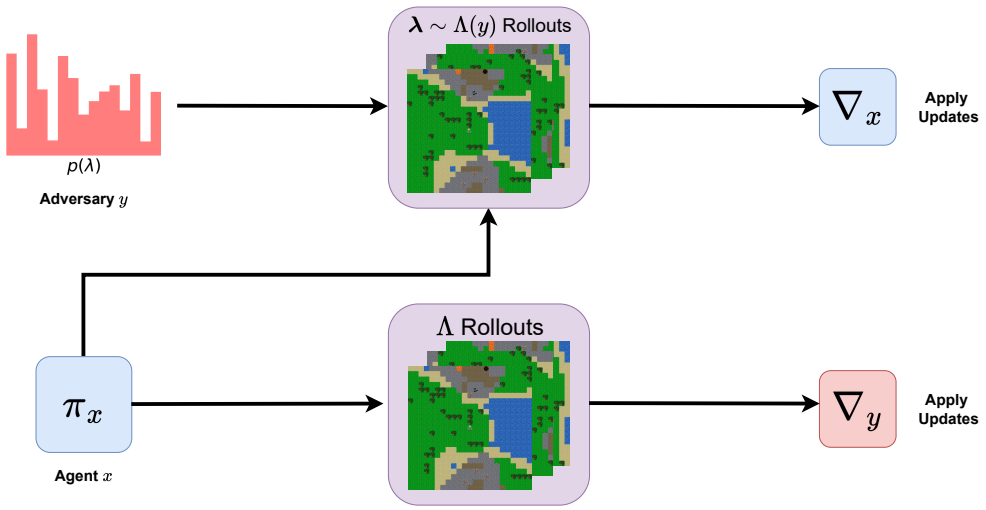


Figure 1: A visual representation of our training loop, which has simultaneous updates for the agent x and the adversary y. The agent's update is trained on levels $\boldsymbol{\lambda}$ sampled from y, and the adversary's update is computed with scores computed from the policy on all levels from level-buffer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

2020) in two-timescale gradient descent that ensure convergence when learning rates for minimiser (agent) and maximiser (adversary) are *separated*. This stands in contrast to the existing paradigm that only has theoretical guarantees *if* convergence is achieved.

Using regret as the score function is theoretically attractive because it maintains the zero-sum property between the agent and the adversary, and therefore allows for stronger guarantees. However, high-regret levels may not lead to efficient learning; moreover, computing regret is generally intractable, as it requires an optimal policy for each level. Inspired by this, Rutherford et al. (2024) use *learnability* (Tzannetos et al., 2023) as an effective scoring function, but their formulation is limited to deterministic, binary-outcome domains. In our work, we generalise this score function to arbitrary deterministic settings, using the interpretation of learnability as the variance of agent success. We finally develop a practical UED algorithm, NCC (see Figure 1), using our reformulation and generalised score function which obtains competitive results in three 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.

2 Background

2.1 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. A UPOMDP is defined by the tuple $(\mathcal{L}, \mathcal{S}, \mathcal{O}, \mathcal{A}, r, P, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{O}$ is the observation space, and 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 $r : \mathcal{L} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The transition probability function P 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 $\mathcal{S}$. Finally, γ denotes the discount factor and $\rho : \mathcal{L} \to \Delta(\mathcal{S})$ denotes the initial state distribution function.

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

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

2.2 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 optimising for an equilibrium point between the players. 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 (ϵ-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 fixes on variable: neither x or y are able to further optimise w.r.t. f via first-order gradient dynamics except for by some (small) distance ϵ.

2.3 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 aims 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 π_*^λ. 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 parametrised by y as their adversary. While not explicitly stated, the objective being optimised is

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

In their analysis, Dennis et al. (2020) 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.

The theoretical results of Dennis et al. (2020) results hold only *at* Nash Equilibrium, but there is no guarantee that this NE will be reached. In fact, the system is unlikely to converge to a NE due to the nonconvex-nonconcave objective, which is not well-understood in the optimisation literature without additional unmet structural assumptions (Mertikopoulos et al., 2019; Jin et al., 2020; Cai et al., 2024). Secondly, the minimax theorem does not hold on the account of the nonconvexity/nonconcavity of the optimised variables (Jin et al., 2020). 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.

2.4 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. There is also the *regret stagnation* problem (Beukman et al., 2024), where due to some stochasticity or partial observability in an environment, the regret is not reducible below some non-minimal value.[2] When irreducible, regret is no longer representative of policy learning potential.

One prevailing alternative to regret is the *learnability* of a level (Rutherford et al., 2024). Assuming a binary-outcome setting, where the return $R(\tau, \lambda)$ is in $\{0, 1\}$ for all trajectories τ and levels λ, learnability is defined as $s(\pi_x, \lambda) = p(1-p)$, where p is the agent's solve rate $p = \mathbb{E}_{\tau \sim \pi_x(\lambda)}\left[R(\tau, \lambda)\right]$. For binary outcomes, this can be rewritten as the variance of returns as follows:

$$s(\pi_x, \lambda) = \text{Var}_{\tau \sim \pi_x(\lambda)}\left[R(\tau, \lambda)\right]. \tag{4}$$

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 either too difficult or too easy, and should not be prioritised during training.

3 Related Work

UED algorithms can generally be categorised into either *sampling*-based and *generative*-based approaches. The former assume access to some function that samples levels; often, this is the uniform distribution over levels $\mathcal{L}$. The latter instead learn a level-generating model (either via RL (Dennis et al., 2020) or self-supervised learning (Garcin et al., 2024)).

The simplest UED method, called Domain Randomisation (DR), directly trains an agent on this uniform distribution (Tobin et al., 2017). In simple environments, DR can perform competitively to other UED algorithms (Coward et al., 2024), but falls behind when the environment becomes more complex (Matthews et al., 2025). Other sampling-based approaches, like Prioritised Level Replay (Jiang et al., 2021b;a, PLR) and Sampling For Learnability (Rutherford et al., 2024, SFL) prioritise training on levels with a high score (e.g., regret, learnability, etc.). However, empirically these methods tend to work best when the agent is *also* trained on some levels from the unfiltered DR distribution, in addition to the high-scoring ones.

While these sampling methods have achieved impressive empirical results, they do not enjoy convergence guarantees due to the use of heuristics in place of gradient-based optimisation. In our work, we establish a two-player 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 significant departure from our current method.

Generative approaches such as PAIRED have also seen success, particularly when replacing the RL level generator with a deep generative model (Azad et al., 2023; Li & Varakantham, 2024; Garcin

[2]For example, if return $R(\cdot, \lambda) \in [0, 1]$, and cannot be increased past 0.7, $\text{Reg}(\cdot, \lambda)$ is irreducible below 0.3.

et al., 2024; Chung et al., 2024). However, the optimisation objective is nonconvex-nonconcave due to the neural adversary, and thus is known to have issues with instability and convergence (Wiatrak et al., 2020). Our method does not use a deep generative model, but unifies the generative and sampling approaches by *learning* the sampling distribution with gradient optimisation.

4 Method

We reformulate UED as a regularised game over expected score, and derive optimisation guarantees from this formulation. We then break the theoretical assumptions for empirical reasons in Section 6.

4.1 Core Optimisation Problem

Consider $s : \mathcal{X} \times \mathcal{L} \to \mathbb{R}$, where $s(\pi_x, \lambda)$ denotes the policy's score on level λ. We abuse notation to rewrite any level-wise function of λ as a function of Λ in **bold** to denote the vector of the function evaluated at all levels $\lambda \in \Lambda$ (e.g. $\boldsymbol{s}(\pi_x, \Lambda)$ is the *score vector*). 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:

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

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

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

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. Second, entropy regularisation ensures that f is strongly concave in y, guaranteeing best-iterate convergence (Theorem 5.1). For general score functions, the optimisation problem is given by:

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

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 strongly concave in y. Lin et al. (2020) have shown that under certain assumptions, we can guarantee (best-iterate) convergence using *two-timescale* stochastic gradient descent-ascent, which assumes a separation of learning rates. Using these results, 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 the score vector $\mathbf{s}$ before each iteration of RL training 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 descent. 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 (9) and (10), 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 (9), and relies on a static buffer. While we make typical assumptions about the policy architecture and policy gradient estimator for the sake of theoretical analysis, we do not necessarily use these in practice due to worsened empirical performance (see Appendix B.3 for more details).

Algorithm 1 Nonconvex-concave Optimisation for UED

Require: Initial policy x^0, distribution $y^0 = \frac{1}{|\Lambda|}\mathbf{1}$, stepsizes η_x, η_y, level set Λ.
for $t = 0, 1, \ldots$ **do**
Sample batch of training levels $\boldsymbol{\lambda} \sim \Lambda(y^t)$
Construct score vector $\mathbf{s} = \boldsymbol{s}(\pi_x, \Lambda)$
$x^{t+1} = x^t - \eta_x \cdot \hat{F}(x^t, y^t; \boldsymbol{\lambda})$ with $\hat{F}$ defined in Equation (9)
$y^{t+1} = \mathcal{P}_{\mathcal{Y}}\left(y^t + \eta_y \cdot \hat{G}(x^t, y^t; \mathbf{s})\right)$ with $\hat{G}$ defined in Equation (10)
end for
return Best-iterate policy parameters x^*

5 Convergence Results

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 our main theorem.

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 λ_i be the i-th level. We denote trajectories by $\tau^t = (o^t, a^t, r^t)$, with $\pi_x(\tau^t) := \pi_x(a^t|o^t)$ for short. We write $\Psi^t(\tau, \lambda)$ for the estimator used in policy gradients, typically taken to be the discounted Q-function $Q^t_{\pi_x}(\tau, \lambda) = \gamma^t Q_{\pi_x}(s^t, a^t, \lambda)$, advantage function $A^t_{\pi_x}(\tau, \lambda) = \gamma^t A_{\pi_x}(s^t, a^t, \lambda)$ or return $R^t(\tau, \lambda) = \sum_{h=t}^{t} \gamma^h r^h$ from time t onwards. The discount ensures that the policy gradient is a true gradient, as clarified by Nota & Thomas (2020).

In order to prove convergence guarantees, we need to make some basic regularity assumptions on the UED and policy network architecture. The first is standard for returns to be finite and for regularity conditions to hold, while the second can be guaranteed by clipping, normalising or otherwise bounding network weights, as discussed in Appendix A.

Assumption 1. The number of levels N and the longest episode length T are finite, and the state and reward spaces are bounded. We consequently write $R_* = \max_{\tau,\lambda} |R(\tau, \lambda)| < \infty$ for the largest absolute discounted return across trajectories and levels.

Assumption 2. The agent policy π_x is a ζ-greedy policy parameterized by an L-Lipschitz and K-smooth function with parameters x. The adversary distribution is constrained to the ξ-truncated probability simplex, 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 N levels λ_i sampled from $\Lambda(y)$, with a batch size of M trajectories for each level:

$$\hat{F}(x, y) = -\frac{1}{NM} \sum_{i,j} \sum_{t=0}^{T} \nabla_x \log \pi_x(\tau^t_{ij}, \lambda_i) \Psi^t(\tau_{ij}, \lambda_i)\,, \tag{9}$$

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

$$\hat{G}(x, y) = \hat{\boldsymbol{s}}(\pi_x, \Lambda) + \alpha \nabla_y \mathcal{H}(y)\,, \tag{10}$$

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 in Equations (9) and (10) has σ_M^2-bounded variance, where

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

Moreover, the corresponding objective $f(x, y) = y^T \boldsymbol{s}(\pi_x, \Lambda) + \alpha\mathcal{H}(y)$ defined in Equation (6) is α-strongly concave in y, Lipschitz, and ℓ-smooth, where

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

Proof. In Appendix A. □

Theorem 5.1 (Best-Iterate Convergence)**.** *Under Assumptions 1 and 2, let ℓ and $\sigma := \sigma_1$ be the constants defined in Proposition 1, α 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 ϵ-stationary policy π_{x^*}, such that $\|\nabla_x \max_y f(x^*, y)\| < \epsilon$, in a number of iterations given by*

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

Proof. Apply Proposition 1 and Lin et al. (2020, Theorem 4.5), with $D \leq \sqrt{2}$ being the diameter of the ξ-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 π_{x^*} in the UED setting, the corresponding optimal distribution $y^* = \text{argmax}_{y\in\mathcal{Y}} f(x^*, y)$ 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 ϵ-approximate first-order Nash Equilibrium.

6 Practical Considerations

To detail the practical extension of our algorithm, we first introduce the *dynamic* level buffer, and then explain our proposed generalisation of the learnability score function from Rutherford et al. (2024). Additional discussion of the differences between our practical and theoretical methods are in Appendix B.3, and the full practical algorithm is given in Algorithm 2.

6.1 Searching the level space

Given that the environment has a large enough level space (i.e. $|\mathcal{L}| \gg |\Lambda|$), 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). Inherently, if the level space contains a high proportion of irrelevant (low-scoring) levels, the initially-sampled Λ 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 dynamic 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 the top $|\Lambda|$ scoring levels *prior* to constructing the adversary's gradient.

6.2 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 lacks convergence guarantees. 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.

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 Λ. 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.

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

$$s(\pi_x, \lambda) = \sigma_\lambda \cdot \mathcal{N}(\mu_\lambda|\mu,\sigma^2)\,. \tag{11}$$

In Appendix B.4 we repeat the score function analysis of Rutherford et al. (2024), demonstrating the score function's effectiveness on Minigrid. Generalised learnability thus allows for the use of generalised SFL ("Gen-SFL" in Figure 3), which obtains superior performance on Craftax.

7 Experiments

Alongside our theoretical considerations, our method outperforms contemporary work 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.

7.1 Experimental Setup

We test our method on benchmarks from Rutherford et al. (2024) and 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 omit JaxNav, as the single-agent setting's results are highly saturated and the multi-agent setting introduces additional optimisation challenges. We refer the reader to Rutherford et al. (2024) for more details on the individual environments, noting only that Minigrid is the only environment with a regret oracle. Additionally, we show that our method can obtain competitive performance on a more complex benchmark, *Craftax* (Matthews et al., 2024).

In all provided plots, we show our contributions in **bold** font, and suffix "NCC" with the particular score function used (see Rutherford et al. (2024) for a further discussion of such score functions). "Reg", "Learn", and "PVL" correspond to regret, learnability, and positive value loss respectively. PVL is an approximation of regret when the latter is unknown, as in XLand-Minigrid and Craftax. For Minigrid, learning curves overlap significantly and we use a bar plot for visual clarity, whereas for XLand-Minigrid and Craftax we plot evaluation results over the course of training to additionally compare sample efficiency.

Experiments were written in JAX (Bradbury et al., 2018) and we discuss experimental details in Appendix B.6. 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.

We largely use the hyperparameters from Rutherford et al. (2024) and Matthews et al. (2024), detailed in Appendix B.7. Moreover, we demonstrate the diversity of levels proposed by our method throughout training in Appendix C.

7.2 Results

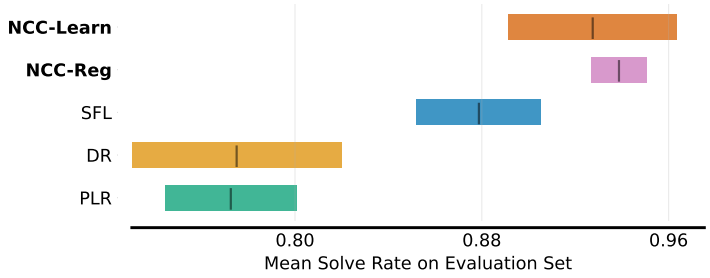
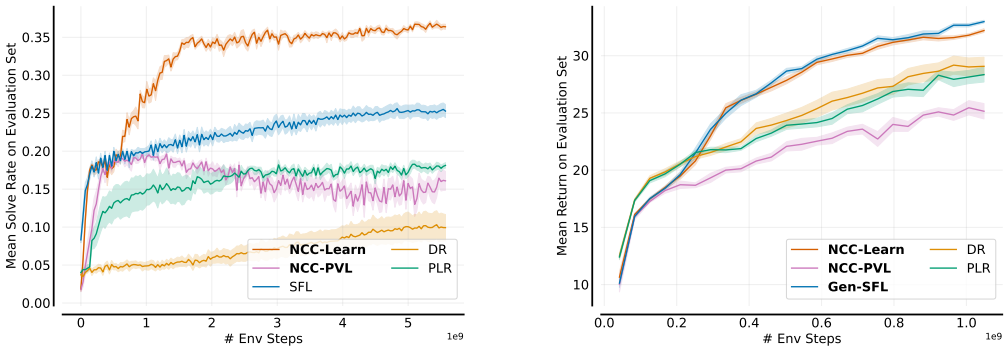


Figure 2: Mean solve rates with standard error bars on Minigrid, a common UED testbed.

Minigrid We observe improved performance on the Minigrid holdout set with 60 walls. For readability, we use a bar plot using recommendations from (Agarwal et al., 2021), but with mean and standard error for consistency. Additionally, we use Minigrid to highlight our method's efficacy given a regret oracle, and thus do not test NCC-PVL.[3] Despite the wisdom from Rutherford et al. (2024) that learnability improves learning, we hypothesise that due to the zero-sum nature of regret it may still lead to more robust policies in the end.



(a) XLand-Minigrid

(b) Craftax

Figure 3: Performance on more difficult benchmarks.

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 leaves more room for improvement. NCC obtains a considerably improved solve rate 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). We remark that we find performance is stronger

[3] We did use PLR-MaxMC instead of PLR-Reg however, because we did not see a improvement in empirical performance.

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] We attribute similar performance across generalised SFL and NCC with learnability to the algorithms' shared emphasis on levels with high learnability. See Appendix B.2 for additional implementation details for Craftax experiments.

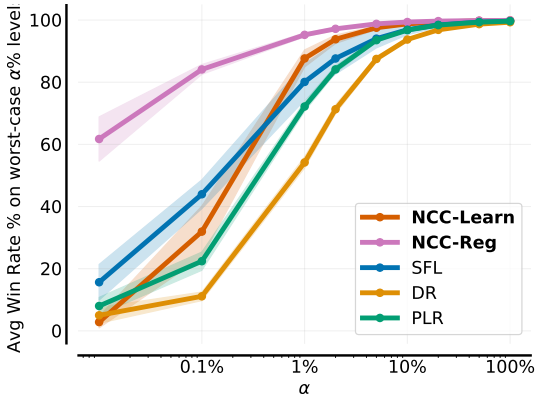


Figure 4: α-CVaR evaluation performance on Minigrid.

Robustness Evaluation We also perform the α-CVaR evaluation protocol from Rutherford et al. (2024), which evaluates policies from 10 seeds per method on the $\alpha\%$ worst-case levels which are still solvable. We find that in XLand-Minigrid and Craftax, the robustness evaluations roughly match the experimental outcomes. However, in Minigrid (where regret is tractable) we find that NCC with regret strongly outperforms any other method, indicating NCC significantly impacts the robustness of the policy. We display our Minigrid result in Figure 4, and defer the other plots to Appendix B.5.

8 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, we leave the question of the more desirable last-iterate convergence property (Daskalakis & Panageas, 2020; Lei et al., 2021) to future work.

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 practical and more scalable ways to produce convergent methods when the zero-sum condition is not met—as in the case when using learnability. Finally, considering the emergence of analysis of more sophisticated reinforcement learning algorithms like PPO (Grudzien et al., 2022), another promising direction for future work is analysing the practical variant of our algorithm.

9 Conclusion

In this paper, we examine the connection between UED and optimisation theory, thereby developing a new framework for optimisation within UED. This framework unlocks the area's first convergence guarantees, and suggests a new method that produces more robust policies. Following prior work, we use our theoretical analysis to develop a practical algorithm that obtains strong results on robustness evaluations. Ultimately, we believe that our work provides a gateway to the creation of practical robust RL methods with guarantees under reasonable assumptions, which will become increasingly important as UED gains prevalence.

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

A Proof of Proposition 1

Before proving the proposition, we briefly discuss Assumption 2. In conjunction with Assumption 1, a simple and sufficient condition for it to hold is for π_x to be parameterized by a neural network with bounded weights, composed of any number of fully-connected, convolutional, max-pooling, recurrent (vanilla / gated / LSTM) layers, dropout, batch normalization and smooth activation functions including Sigmoid, Softmax, Tanh, ArcTan, ELU, SELU, GELU, SoftPlus, Softsign (Virmaux & Scaman, 2018). The condition that weights be bounded may seem restrictive, but (1) typically holds in practice because of weight regularisation, (2) can easily be ensured by clipping weights to an (arbitrarily large) box, which would only alter experimental results if gradients explode, and (3) can also be ensured by normalization across weight matrix rows, which in some cases may improve training stability at a minor cost in performance (Miller & Hardt, 2019).

Proposition 1. Under Assumptions 1 and 2, the estimator $\hat{H} = (\hat{F}, \hat{G})$ defined in Equations (9) and (10) has σ_M^2-bounded variance, where

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

Moreover, the corresponding objective $f(x, y) = y^T \boldsymbol{s}(\pi_x, \Lambda) + \alpha\mathcal{H}(y)$ defined in Equation (6) is α-strongly concave in y, Lipschitz, and ℓ-smooth, where

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

Proof. We first derive the bounds for $M = 1$. Recall Equations (9) and (10) from the main text:

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

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

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

For the second term, we easily obtain

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

for both $s = -J$ and $s = \text{Reg}$. For the first term, we invoke Lipschitzness and ζ-greediness of the policy π. For any trajectory τ and any level λ, we have $|\sum_t \Psi^t(\tau, \lambda)| \leq 2TR_*$ for any choice of estimator $\Psi^t \in \{R^t, Q^t_{\pi_x}, A^t_{\pi_x}\}$, 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

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

and hence

$$\mathbb{E}\left[\left\|\hat{H}(z) - H(z)\right\|^2\right] \leq 4NR_*^2 + \frac{4T^2R_*^2L^2}{\zeta^2} = \sigma_1^2$$

as required. The variance for arbitrary M follows immediately as $\sigma_M^2 = \sigma_1^2/M$.

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

(3) Lipschitzness of f. First note that

$$\|\nabla_y \mathcal{H}\|^2 = \sum_i (1 + \log y_i)^2 \leq N(1 + \log \xi)^2 \,.$$

We combine this with Jensen's inequality and part **(1)** of the proof above to obtain L-Lipschitzness:

$$\begin{aligned}
\|\nabla f\|^2 &= \left\|\mathbb{E}\left[\hat{H}\right]\right\|^2 \leq \mathbb{E}\left[\left\|\hat{H}\right\|^2\right] \\
&= \mathbb{E}\left[\left\|\hat{F}\right\|^2\right] + \mathbb{E}\left[\|\hat{\boldsymbol{s}} + \alpha\nabla_y\mathcal{H}\|^2\right] \\
&= \mathbb{E}\left[\left\|\hat{F}\right\|^2\right] + \mathbb{E}\left[\|\hat{\boldsymbol{s}}\|^2\right] + \alpha^2\|\nabla_y\mathcal{H}\|^2 + 2\alpha\,\mathbb{E}\left[\|\hat{\boldsymbol{s}}\|\right]\|\nabla_y\mathcal{H}\| \\
&\leq \sigma_1^2 + \alpha^2 N(1+\log\xi)^2 + 4\alpha N R_*(1+\log\xi) =: L\,.
\end{aligned}$$

(4) 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 ρ and P (omitting λ 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

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

hence the first term is bounded for each t:

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

For the second term, we use the K-smoothness of π to obtain:

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

Putting everything together, recalling that $|R^t(\tau,\lambda)| \leq R_*$ for all t, we obtain

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

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

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

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

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

as required. □

Acknowledgments

The authors would like to thank Jingming Yan, Ioannis Panageas, and JB Lanier for their helpful feedback on the paper. MJ and AG are funded by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems. MJ is also funded by Amazon Web Services. MB is funded by the Rhodes Trust. AR is partially funded by the EPSRC Programme Grant "From Sensing to Collaboration" (EP/V000748/1).

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.

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.

John Miller and Moritz Hardt. Stable recurrent models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Hygxb2CqKm.

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.

Chris Nota and Philip S Thomas. Is the policy gradient a gradient? In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 939–947, 2020.

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.

Supplementary Materials

The following content was not necessarily subject to peer review.

B Additional Experimental Details

B.1 Practical Algorithm

In practice, we often (although not always) find it helpful to use a dynamic buffer, as per Section 6.1. Moreover, we define the subroutine TRAIN_RL to refer to any form of training from the reinforcement learning literature, however in practice we use mini-batch PPO (Schulman et al., 2017). Moreover, we use TiAda-Adam (Li et al., 2023) as an adaptive optimiser for our optimisation setting, but for all other experiments we use Adam (Kingma & Ba, 2017). We write details for the dynamic buffer in maroon.

Algorithm 2 Practical Nonconvex-concave Optimisation for UED (Dynamic Buffer)

Require: Initial policy x^0, distribution $y^0 = \frac{1}{|\Lambda|}\mathbf{1}$, stepsizes η_x, η_y, initial level set Λ^0.
  for $t = 0, 1, \ldots$ **do**
    Sample batch of training levels $\boldsymbol{\lambda} \sim \Lambda^t(y^t)$
    Construct score vector $\boldsymbol{s}(\pi_x, \Lambda)$
    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 $\tilde{\mathbf{s}} = \boldsymbol{s}(\pi_x, \Lambda^t)$
    $x^{t+1} = \text{TRAIN_RL}(x^t, \boldsymbol{\lambda}, \eta_x)$
    $y^{t+1} = \mathcal{P}_{\mathcal{Y}}\left(y^t + \eta_y \cdot \hat{G}(x^t, y^t; \mathbf{s}, \tilde{\mathbf{s}})\right)$ with $\hat{G}$ defined in Equation (10)
  end for
  return Best-iterate policy parameters x^*

B.2 Additional Craftax Details

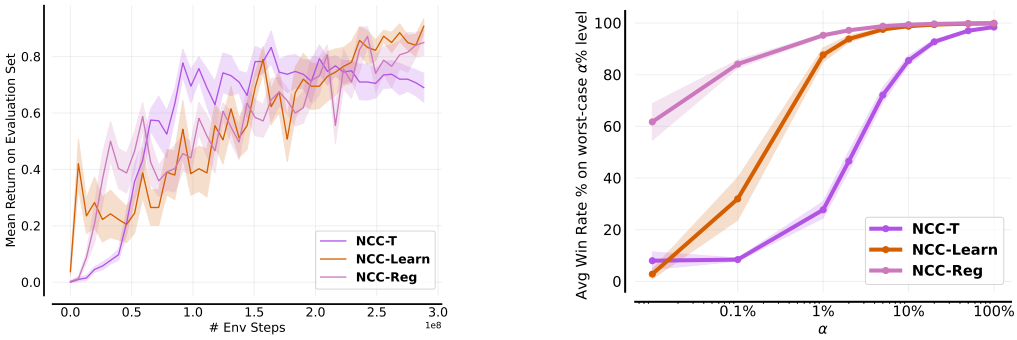
We maintain the training regime of Matthews et al. (2024) by using "inner" and "outer" rollouts, where we update after multiple parallelised sub-sequences within an episode. Moreover, due to such a level space where the levels and test set are so similar, we found that it more effective to *anneal* our entropy regularisation coefficient α 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.

B.3 Comparison Between Theory and Practice

We give the following side by side comparison between our practical and theoretical method:

Table 1: Qualitative comparison between theoretical NCC and practical NCC.

	Theory	Practice
x Gradient Estimator	Equation 9 (REINFORCE)	Any
# minibatches (epochs)	1 (1)	Any (Any)
Dynamic Buffer	Not Allowed	Allowed
Optimiser	SGD	Any
Score Function	Only zero-sum (i.e. Regret and $-J$)	Any
Activation Function	Smooth	Any



(a) Solve Rates

(b) α-CVar Evaluation

Figure 5: Empirical comparison between the different instantiations of NCC.

Moreover, in Figure 5 we experimentally compare the theoretical version of our method ("NCC-T") with the other methods on Minigrid, which is our only testbed with a regret oracle.

B.4 Generalised Learnability Score Function

In Figure 6 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 ($1/4$ 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.

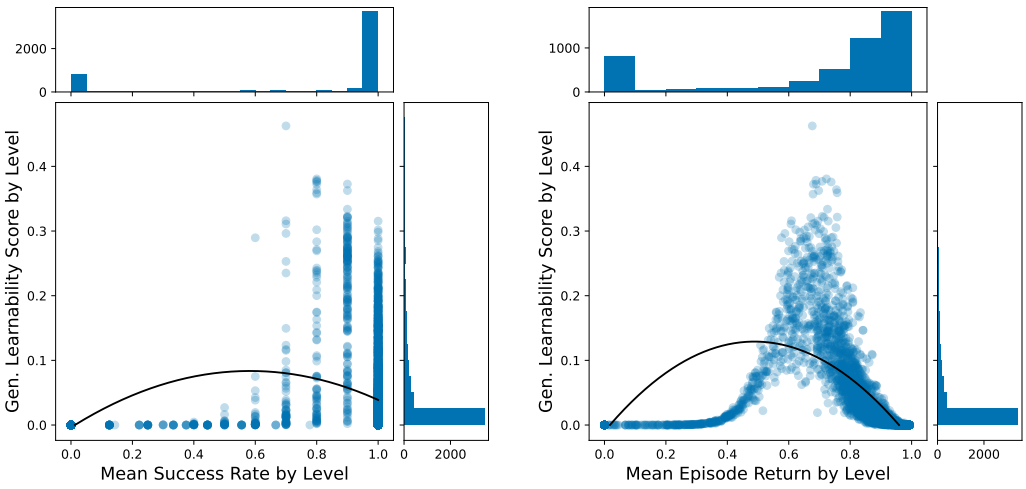
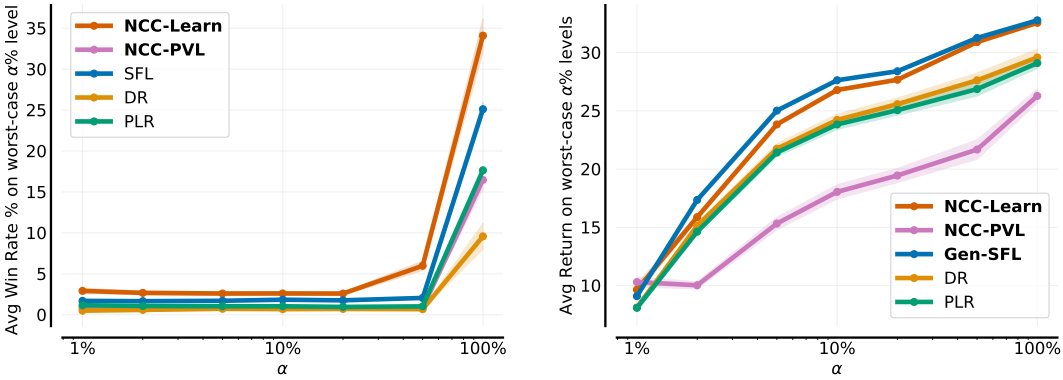


Figure 6: Analysis of Generalised Learnability Score function on Minigrid. The black lines represent a quadratic fit to the scatter data.

B.5 Robustness Scores



(a) XLand-Minigrid

(b) Craftax

Figure 7: Omitted α-CVaR Robustness Evaluation curves.

B.6 Compute Time

Table 2 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). However, NCC experiments for Minigrid were run on the L40s server.

Table 2: Mean and standard deviation of time take for experimental evaluations. Each evaluation consisted of 10 independent seeds.

Method	Minigrid	XLand	Craftax
NCC Learn	1:02:41 (0:00:26)	3:31:57 (0:01:06)	5:35:06 (0:00:59)
NCC Regret	1:01:59 (0:00:11)	-	-
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)

B.7 Hyperparameters

Table 3: NCC (Reg) Hyperparameters

Hyperparameter	Minigrid	XLand	Craftax
η_x	0.001	0.0001	0.0001
η_y	0.1 (0.005)	0.01	0.01
α	0.05 (0.032)	0	0.05
$\|\Lambda\|$	4000	4000	4000
$\|\Lambda'\|$	256	8192	0
$\|\boldsymbol{\lambda}\|$	256	8192	1024
γ	0.995	0.99	0.995
GAE λ	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`	0.25	0.5	1.0
`num_minibatches`	1	16	2
`num_parallel_envs`	256	8192	1024

Table 4: PLR (DR) Hyperparameters

Hyperparameter	Minigrid	XLand	Craftax
η_x	0.00025	0.0001	0.0002
$\|\Lambda\|$	4000	4000	4000
γ	0.995	0.99	0.99
GAE λ	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

Table 5: SFL Hyperparameters

Hyperparameter	Minigrid	XLand	Craftax
η_x	0.00025	0.001	0.0001
$\|\Lambda\|$	1000	8192	4000
γ	0.99	0.99	0.995
GAE λ	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`	4000	40000	4000
`num_batches`	5	1	5

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).

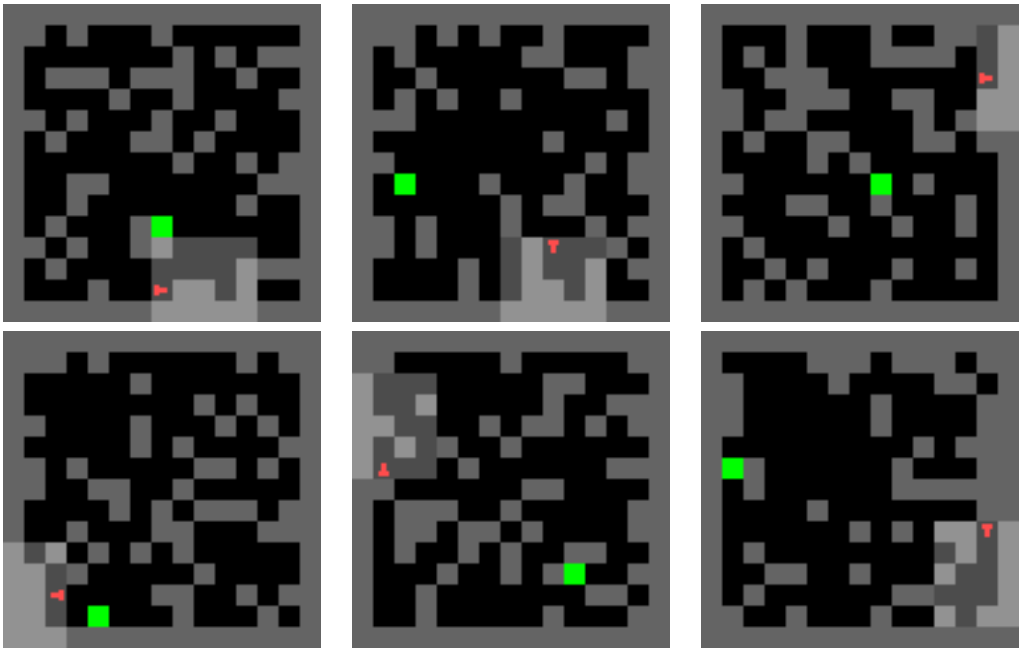


Figure 8: DR: Sampled levels at halfway through training (top row) and the end of training (bottom row)

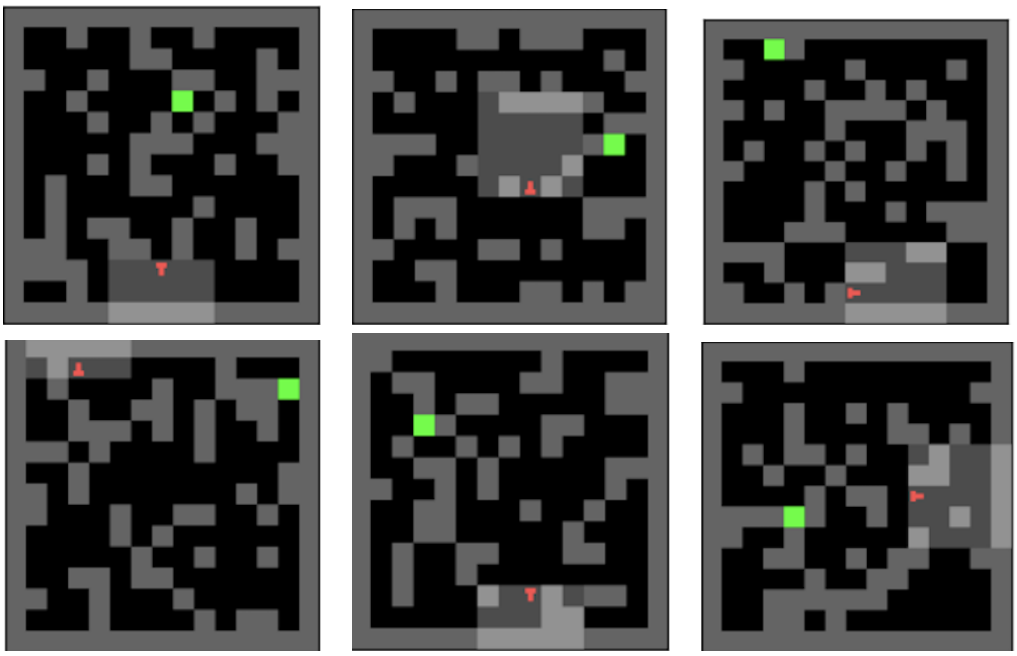


Figure 9: SFL: Highest learnability scoring levels at halfway through training (top row) and the end of training (bottom row)

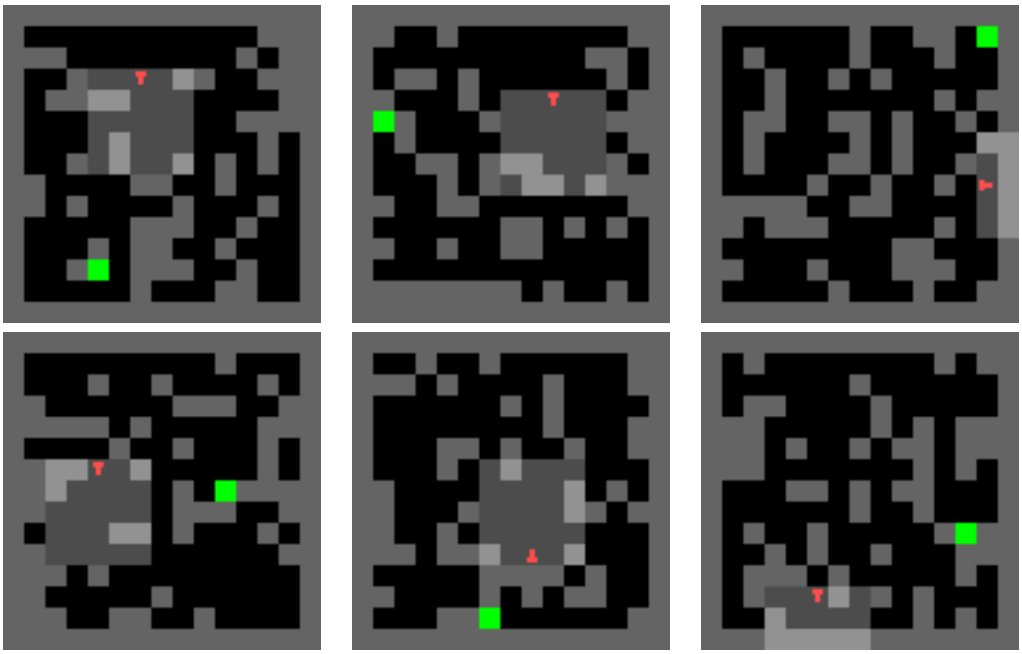


Figure 10: NCC-Learn: Highest weighted levels at halfway through training (top row) and the end of training (bottom row)