CAYLEY MAZE: UNIVERSAL OPEN-ENDED REINFORCEMENT LEARNING ENVIRONMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Parametrizable environments with variable complexity are crucial for advancing fields such as Unsupervised Environment Design (UED), Open-Ended Learning, Curriculum Learning, and Meta Reinforcement Learning. However, the selection of environments in evaluation procedures, along with their complexities, is often either neglected or lacks formal justification. We propose the formal definition of complexity for Markov Decision Processes using Finite Automata and Group Theory machinery. We introduce Cayley Maze, a novel open-ended reinforcement learning environment that naturally generalizes problems like solving Rubik's Cube, sorting, and integer factorization. Cayley Maze is universal: every deterministic MDP is an MDP of a certain instance of Cayley Maze. We demonstrate how Cayley Maze enables control over complexity, simplification, and combination of its instances. Finally, we evaluate UED algorithms on various instances of the Cayley Maze and analyze their capacity to produce agents with robust generalization capabilities.

1 INTRODUCTION

Designing agents capable of generalizing across a diverse array of tasks and environments is both a challenging and exciting problem in modern reinforcement learning. Open-ended learning, unsupervised environment design (UED), and curriculum learning remain attractive approaches to reach this goal (Hughes E., 2024)

There has been a lot of recent progress in the design and implementation of Open-Ended environments, For example Genie (Bruce J., 2024) or Craftax (Matthews M., 2024). While the Genie is a huge achievement by itself, we doubt that such an environment allows for producing challenging, algorithmic, or intelligent problems. On the other hand, some of the Unsupervised Environment Design(UED) algorithms(Beukman M., 2024) , (Parker-Holder J., 2024) are still being evaluated on Minigrid (Boisvert C., 2018) environments. In such an experiment-oriented field like Machine Learning, algorithms cannot be better than their evaluation procedures. We pose the question: what is a good evaluation procedure for Open Ended Learning? The partial answer is that variable complexity is necessarily its component. The ability to produce observations which gradually become more and more complex, is the core assumption of curriculum learning. The diversity of parametrizable environments is an assumption of UED, hence it also relies on some notion of similarity/metric. In this paper we propose the formal definition of the complexity of Reinforcement Learning environments. We discuss, why the current heuristics like the cardinality of state or action space might be a naive estimate for this goal. We do it by translating the certain concepts from the Language of Algebra, Deterministic Finite Automata theory, and Topology, and hopefully prove its usefulness. More than that, we introduce a novel Reinforcement Learning environment called Cayley Maze, which is, in cerain way, universal. We show, that many important problems such as sorting or Rubik's Cube are specific instances of Cayley Maze. Finally, we introduce our implementation of Cayley Maze in JAX, and evaluate UED algorithms in different scenarios.

2 PRELIMINARIES

Here we briefly recap some definitions of algebraic approach to Automata theory. For more context we advise to read J. (2021). *Monoid* is a set M equipped with the operation $(\cdot) : M \times M \to M$ for which the following hold:

- Identity element: there exists $e \in M$ such that for every $m \in M$ $e \cdot m = m \cdot e = m$. Identity element is always unique.
- Associativity: for every elements $f, g, h \in M$, $(f \cdot g) \cdot h = f \cdot (g \cdot h)$

A *group* is monoid, whose every element has an inverse, i.e. for every $g \in G$ there exists $g^{-1} \in G$ such that $g \cdot g^{-1} = g^{-1} \cdot g = e$.
Given a group G, a *G-action* on the set S, is the map $\rho : G \times S \to S$, for which $\forall s \in S$ $\rho(e, s) = s$, and action commutes with multiplication: $\rho(g, \rho(h, s)) = \rho(g \times h, s)$ for all $g, h \in G$ and $s \in S$.
By $[n]$ we denote an n-element set $[n] = \{1, 2, \cdots n\}$. Given set M and S, the set M^S is a set of all functions from S to M. A monoid of functions from n-element set to itself with the operation of composition is denoted by $End([n]) = [n]^{[n]} = \{f : \ f : [n] \to [n]\}$. The following theorem is the core idea of our environment:

Theorem 1 (Cayley's theorem)**.** Every finite monoid N is isomorphic to some submonoid (subset) of $End([n])$ for certain $n \in \mathbb{N}$.

A subset $G \subseteq M$ of monoid M is called a set of *generators* if it generates M, i.e. for every $m \in M$ there is a sequence $g_1, \dots g_n$ such that $m = g_1 \cdot g_2 \cdot \dots g_n$. A free monoid on the set A is a set $M = A^*$, containing all finite sequences of elements of A, with the operation of concatenation, namely for $x = x_m x_{m-1} \dots x_1,\ y = y_n y_{n-1} \dots y_1$, $x \cdot y = x_m x_{m-1} \dots x_1 y_n y_{n-1} \dots y_1$.
A *homomorphism* between monoids M and N is a map between its sets $\phi : M \to N$, such that for every $a, b \in M$ $\phi(a \cdot b) = \phi(a) \cdot \phi(b)$. An *isomorphism* is a bijective homomorphism.
A *congruence* on monoid M is an equivalence relation $\sim$ on a set M, which is compatible with its operation: for every $a, b, c, d \in M$, $a \sim c,\ b \sim d \implies (a \cdot b) \sim (c \cdot d)$. Every congruence induces monoid structure on the set $M/\sim$ of equivalence classes on M and a canonical homomorphism: $\pi : M \to M/\sim,\ \pi(a \cdot b) = [a \cdot b]_\sim = [a]_\sim \cdot [b]_\sim$. $M/\sim$ is called a *quotient* monoid of M. Given a monoid M and its subset L, *syntactic congruence* on M is defined as $a \sim_L b$ if for all $x, y \in M$ $xay \in L \iff xby \in M$. The quotient of this equivalence relation is called a *syntactic monoid*.
We call *Markov Decision Process* a tuple (A, S, s_0, R, T) where A is the set of actions, S - the set of states, s_0 - initial state, $R : A \times S \to \mathbb{R}$ - reward function, and $T : A \to (S \to Pr(S))$ transition function, assigning the transition kernel $T(a) = T_a$ on S to every state. For the mathematical convenience we define a transition kernel T_e of the neutral element e (empty sequence) of the free monoid A to be an identity matrix. Markov Decision Process (A, S, s_0, R, T) is called *sparse*, if there exists a a set of final states $F \subseteq S$, such that

$$\begin{cases} R(a, s) = 1 & T_a(s, f) = 1 \text{ for some } f \in F \\ R(a, s) = 0 & \text{otherwise} \end{cases}$$

For given MDP with action space A, *trajectory* is a sequence of actions $\alpha = a_n \dots a_1$, alternatively, its an element of free monoid on A. Given a trajectory α we call its realization a resulting kernel $T_\alpha = T_n \cdot T_{n-1} \dots T_1$. By $R : A^* \to Pr(\mathbb{R})$ we denote a cumulative reward $R(\alpha)$ after moving along the trajectory α from the initial state s_0.
A *transition monoid* of MDP (A, S, s_0, R, T) is a set of trajectory realizations $M(T) = \{T_\alpha : \alpha \in A^*\}$, equipped with the operation of matrix multiplication.
Finite deterministic automaton is a tuple (Q, A, T, I, F), where Q is a set of states, A - set of actions, $T : A \to (S \to S)$ - transition kernel (by T_a we'll denote a function $T(a, -) : S \to S$), I - set of initial states, F - set of final states. Every DFA induces a directed graph on its states. Given DFA, its *transition monoid* is a set of functions $\{T_\alpha : \alpha \in A^*\}$, equipped with the operation of composition.

3 COMPLEXITY OF MARKOV DECISION PROCESSES

Loosely speaking, we define two MDP's to have same complexity, if the corresponding trajectories yield the same cumulative reward. We begin by proposing the extension of the notion of syntactic

monoid for general structures, such as functions, returning random variables. If one thinks, that the condition $R(a) = R(b)$ as random variables is too strict, then one could compare expectations, or to replace it with some approximation, for example by $d(R(a), R(b)) \leq \varepsilon$ after choosing certain metric on $Pr(\mathbb{R})$ and $\varepsilon \geq 0$.

Definition 1. A reward congruence $\sim_R$ on MDP (A, S, s_0, R, T) is a congruence on its transition monoid $\{T_\alpha : \alpha \in A^*\}$, such that for every $a, b \in M(T)$, $a \sim_R b$ if and only if for all $x, y \in M(T)$ $R(xay) = R(xby)$. Then, an *irreducible monoid* $M(R)$ of MDP is the quotient by the congruence relation $\sim_R$. R is well-defined on $M(R)$.

Proposition 1. A reward congruence $\sim_R$ on $M(T)$ for MDP (A, S, s_0, R, T) is maximal among all congruences preserving reward structure: $\forall a, b \in M(T)\ a \sim b \implies R(a) = R(b)$.

Proof. For a congruence $\sim$ on $M(T)$ and some elements $a, b \in M(T)$, $a \sim b \implies \forall x, y \in M(T)\ xay \sim xby$. Hence $R(xay) = R(xby)$, and $a \sim_R b$. □

There are multiple ways to define the complexity of MDP: for example one could measure the size of state space or action space. While these definitions are reasonable, the definition we propose captures different kind of information.

Definition 2. Two MDP's (A, S, s_0, R, T) , (A', S', s'_0, R', T') are equivalent if there is an isomorphism ϕ between their irreducible monoids $M(R)$, $M(R')$, preserving reward structure, i.e. $\forall a \in M(R), R'(\phi(a)) = R(a)$.

The definition 2 is equivalent to another one:

Definition 3. Two MDP's (A, S, s_0, R, T) , (A', S', s'_0, R', T') are *equivalent* if there is a surjective homomorphism ϕ from A^* A'^* or vice versa, such that reward structure is preserved: $\forall a \in A^*, R'(\phi(a)) = R(a)$. For deterministic MDP's with sparse binary rewards it means, that the trajectory $\alpha \in A^*$ solves the first MDP if and only if $\phi(\alpha)$ solves the second MDP.

Definition 4. Order complexity of MDP (A, S, s_0, R, T) is the minimal possible cardinality of the state space of DFA, whose transition monoid is isomorphic to $M(R)$.

Let's have a look how it works on example:

Example 1. Suppose we want to get on the right side of the grid, which has width 3, and infinite length. In other words, we are given an deterministic MDP (A, S, s_0, R, T), where:

- State space $S = \mathbb{Z}_3 \times \mathbb{Z}$
- Initial state $s_0 = (0, 0)$
- Action space $A = \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$
- Transition kernel $T(i, j)(a, b) = ((a + i) \mod 3,\ b + j)$
- Reward $\begin{cases} R((i, j)(a, b)) = 1 & (i + a) \equiv 2 \mod 3 \\ R((i, j)(a, b)) = 0 & \text{otherwise} \end{cases}$

By the definition 3, we define a reward congruence on $T(M)$:

$$(a, b) \sim_R (c, d) \iff \forall (x, y), (uv),\ R((x, y) + (a, b) + (u, v)) = R((x, y) + (c, d) + (u, v))$$

It is true if and only if $(x+a+u) \mod 3 = (x+c+u) \mod 3$, and so $a = c$, since $a, c \in \{0, 1, 2\}$. Hence the relation $\sim_R$ will have only 3 equivalence classes: $\{\{(i, j) :\ j \in \mathbb{N}\} :\ i \in \mathbb{Z}_3\}$, and the irreducible monoid will have only 3 elements. Hence the irreducible monoid is isomorphic to $\mathbb{Z}_3$, and its order complexity is 3. The main conclusion - MDP with an infinite number of states and bigger action space might be equivalent (reduced) to MDP with 3 states and only 1 action.

We'd like to point out that even though deterministic sparse MDP's are very similar to DFA's, there exists a difference: while in RL the episode stops if agent reaches the final state, for DFA's the word belonging to its language might have an extension not belonging to it. Such difference can be eliminated by modifying the automaton, or by adding the termination action to the agent's action space. This modification looks both useful and reasonable: without it any finite MDP with connected directed graph can be solved by the exhaustive search without any use of environment's output.

4 CAYLEY MAZE

We propose a new Open-Ended Reinforcement Learning Environment: Cayley Maze. The agent's goal is to find the path between initial and final vertices of directed graph, by choosing the edges to move along.

Definition 5. An instance of *Cayley Maze* is defined by the tuple (m, n, T, i, F), where

- $m \in \mathbb{N}$ is the size of the action space , so $A = [m]$
- $n \in \mathbb{N}$ is the size of the state space, so $S = [n]$
- $T : A \to End([n])$ is the correspondence between action and monoid generators: $G = Im(T)$ is called the set of generators, and each generator is denoted by $T_a = T(a)$. For the function T extended to A^*: $T(\alpha) = T_\alpha = T(\alpha_1 \cdot \alpha_2 \dots \alpha_k) = T_{\alpha_1} \circ T_{\alpha_2} \circ \dots T_{\alpha_k}$, the transition monoid of T is the image $M(T) = T(A^*)$.
- $i \in [n]$ is an initial state
- $F \subseteq [n]$ is a set of final states

Then the instance induces a sparse deterministic MDP $([m], [n], i, R, T)$, where R is

$$\begin{cases} R(a, s) = 1 & T_a(s, f) = 1 \text{ for some } f \in F \\ R(a, s) = 0 & \text{otherwise} \end{cases}$$

The opposite appears also to be true:

Proposition 2. Every deterministic sparse finite MDP is an MDP of certain instance of Cayley Maze.

Proof. Since (A, S, s_0, R, T) is deterministic, T can be seen as a function $A \to (S \to S)$. Since MDP is sparse, R is completely defined by the subset of final states $F \subseteq S$. Then, after enumerating A and S, $(|A|, |S|, T, s_0, F)$ is an instance of Cayley Maze with the same MDP. □

From now on all discussed Reinforcement Learning environments and its MDP's are assumed to be deterministic and sparse. While the transition between MDP's and Cayley Maze looks tautological, we see it valuable for several reasons.

4.1 CAYLEY MAZE IS NATURAL

Cayley Maze is a framework which naturally generalizes many important problems. For example, the problem of sorting the array of length n can be seen as the instance of Cayley Maze, where:

- state space is the group of all n-element permutations S_n,
- action space - the set of allowed operations for sorting the array for example it could be the set of all transpositions $\{(i, j) : i, j \leq n\}$. Note, that in this situation every action can be associated with its state,
- the transition monoid is defined by the left multiplication of the state by action
- the initial state is the number array seen as a permutation, and the final state is the identity permutation. In this case the multiplied path from the initial to final state is exactly the right order of the array.

Enumerating the squares of Rubik's Cube allows to translate the problem just like in case of sorting, the only difference - actions will have different kind of permutations.
The examples above are natural examples of Cayley Maze, because unlike for general MDP's, whose transition monoid $M(T)$ acts on the trajectories, in this cases $M(T)$ acts on the state space S by multiplication of action and state. In other words, any local information about paths, which is attained at some state $s_1 \in S$, is valid for any other state $s_2 \in S$. Such remarkable and rare property can be used for evaluating agent's generalization capabilities and architecture's inductive

biases. Nevertheless, that doesn't mean, that such MDP's are easy to solve: MDP's whose transition monoid $M(T)$ is simple in group-theoretic sense, are also irreducible and can have unbounded state space, hence unbounded complexity in the sense of definition 2.

4.2 Cayley Maze has variable computational complexity

Various subfamilies of Cayley Maze have different computational complexity: as it has been shown, the problems of sorting the array, and solving the Rubik's cube both can be represented as instances of Cayley Maze. Sorting problem has polynomial time complexity, while the problem of finding the optimal solution of Rubik's Cube (hence such problem can be expressed by decreasing episode length of usual Cayley Maze representation of Rubik's Cube) is NP-Complete (Demaine E., 2018).

4.3 Most of the Reinforcement Learning environments are not universal

Some of the most popular environments used for evaluation of UED algorithms, like Minigrid mazes (Boisvert C., 2018), make a heavy use of the underlying geometric structure of its state space. We note, that any Open-Ended environment, which can be solely represented by moving on 2-dimensional grid (i.e. for which the directed graph of its MDP can be embedded into the plane respecting grid structure) is not universal in the sense of proposition 2 : for example an MDP with three states $\{A, B, C\}$ and one action $a : A \xrightarrow{a} B \xrightarrow{a} C \xrightarrow{a} A$ cannot be embedded into the plane, since otherwise agent would have to always move in the same direction and return to the initial state. What is less obvious, such environments are not universal even in the sense of definition 2:

Proposition 3. There exists an MDP which is not equivalent in the sense of definition 2 to any MDP, whose directed graph is planar. Consequently, such MDP cannot be represented by moving on 2-dimensional grid.

The proof of this fact is due to Book R. (1976). The witnessing automaton has only 7 states and 6 actions. The further development of this topic and the applications of topology for measuring the complexity of finite automata's can be found in Bonfarte G. (2018)

4.4 Modification and simplification of existing instances

Given an abstract MDP, or the set of game instructions it is often not clear which modification would make it simpler or harder. But it's certainly possible for MDP's: painting all Rubik's cube faces into black makes it much easier to solve. In other words, given the MDP M whose transition monoid acts on the set S, and the coloring of S - a surjective function $h : S \to K$, it's not hard to build *quotient* of M by h, whose construction is similar to it's group-theoretic analogue.
Many constructions on groups, such as products also allow to efficiently combine existing MDP's to produce new ones.

5 Applications and implementation details

We implement Cayley Maze as a parametrizable environment in JAX with Gym interface. It allows to sample any MDP with predefined action space, and the state space, or to slightly modify the existing transition kernel, initial or target state.
Cayley Maze may be used in evaluation procedure of every learning problem, which has the sequential structure, and where generalization, or emergent complexity of observations are crucial. Some of the proposed scenarios are:

1. Since the sampled instances can be very diverse, it may be used for the evaluation of UED algorithms in classical scenario.
2. It is possible to create environment samplers, whose instances have common structure. For example it might be MDP's whose transition monoid are simple groups, or environments which represent $n \times n \times n$ Rubik's Cube.
3. It is possible not only to evaluate on subfamilies of environments, but on its various constructions. For example, we can check, whether the agent, which performed well on some instances would perform well on its product.

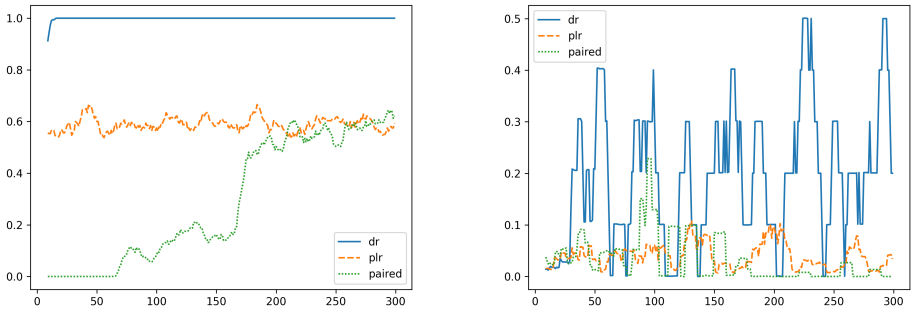

Figure 1: The training and evaluation learning curves of experiment 1

4. Another way to test generalization capabilities is to use the local property of natural Cayley Mazes, as in 4.1. For example, to evaluate on the states which were unreachable during training but have the same properties.
5. Since the process of creating new MDP can be seen as MDP, it can be expressed as an instance of Cayley Maze. Hence the UED scenarios, where the teacher learns to build the environment without student become possible.
6. A model which can't recognize the language, cannot solve it's corresponding MDP. Hence after adding the termination action as it was proposed at the end of section 3, Cayley Maze can be used to evaluate the expressive power of various architectures (Zhou H., 2023).

Another interesting feature of the implementation: while constructing the most general reinforcement learning environment, one might expect the explosion of the teacher's action space. Our implementation allows to set the desired trade-off between the size of action space, representation dimension, and the power of edit per step.

6 Experiments

We've run multiple experiments to evaluate UED algorithms such as PAIRED (Dennis M., 2021) and PLR (Jiang M., 2022) on the Cayley Maze. We basically followed scenarios 3, 1, 2, from the list above. To make the results are comparable we've used the setups proposed in (Dennis M., 2021): all models have LSTM (Hochreiter S., 1997) and were taught for 30000 gradient updates. In all the experiments on both training and evaluation environments the initial and target states have been sampled. All the plots represent the averaged behaviour for 10 seeds, and all the MDP's have the naturality property 4.1. 1 unit of the X axis on the all learning curve figures below corresponds to 100 gradient updates.
In the first experiment models were trained on two separate environments, where one represents the dihedral group of order 6, the other one - the symmetries of 3-cube. Environment used for evaluation represents the wreath product of these two groups. In this experiments underlying MDP's have 10 actions, and 18 states. As it can be seen on the figure 1, the agent, which was trained with the plain domain randomization algorithm, had the best performance on both train and evaluations instances. Even though it's counterintuitive, it can be explained by the fact, that complexity of the instance used in training was lower, than of the instance generated by PAIRED right after initialization. However it's still possible to conclude, that in this case agent, which was trained with domain randomization i general performed better.

In the second experiment model were train on random MDP's with 48 states and 12 actions. They've been evaluated on 6 evaluation instances with various degrees of complexity: dihedral group, group of 3-cube symmetries, symmetric group with transposition actions, wreath product of random groups, and the Rubik's Cube.

As it can be seen on the figure 2, all the UED algorithms performed very poorly on Rubik's Cube. Also, based on the evaluation scores it's still not possible to conclude, that any UED algorithm

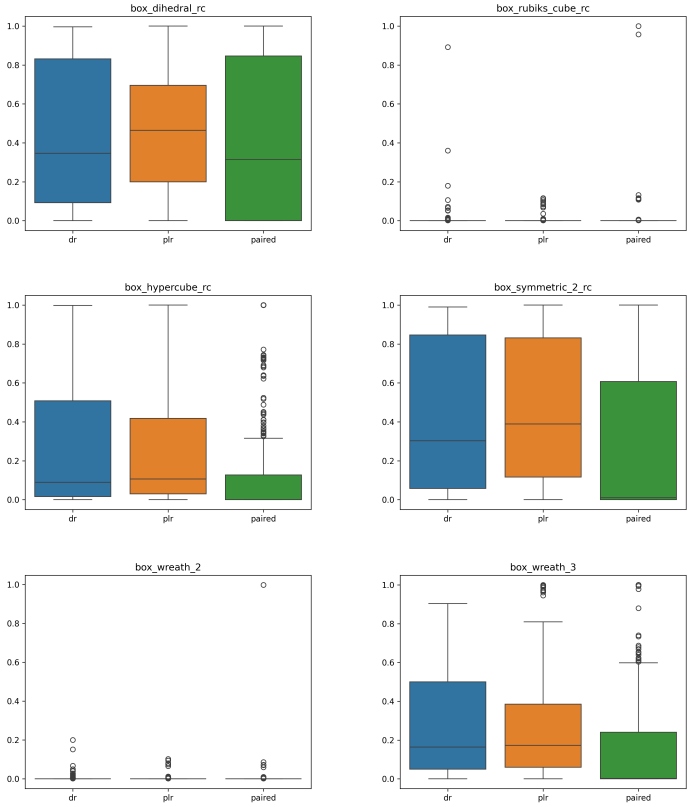

Figure 2: The box plots of evaluation scores of experiment 2

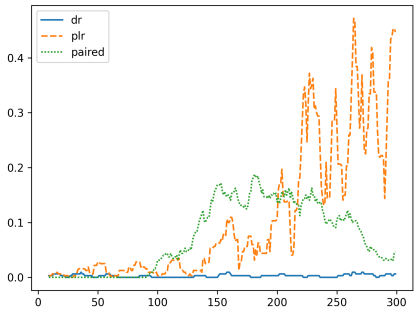

Figure 3: The training learning curve of of experiment 2

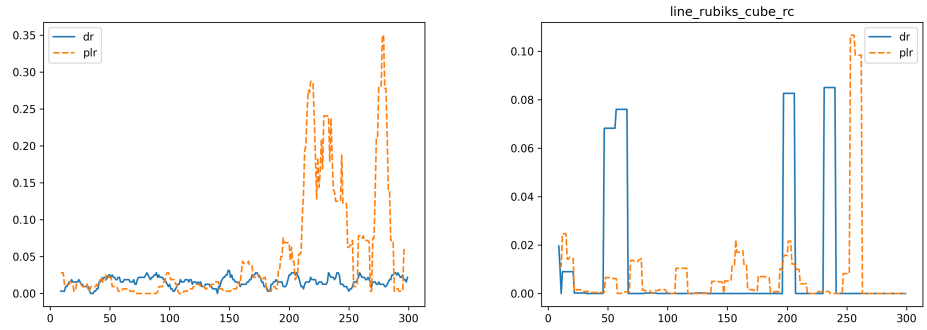

Figure 4: The training and evaluation learning curves of experiment 3

performed better than domain randomization. However, on the first glance it strongly contradicts with the scores on the figure 3. But it can be explained by the same argument as before: on the very beginning all UED algorithms generate instances of the same complexity, and then gradually adapt, what can't be said about domain randomization.

The goal of the third experiment was to determine the effect of the replay buffer while the MDP itself stayed mostly unchanged - agents were trained on the Rubik's Cube, and only initial and target states were sampled. As it can be seen on the figure 4, the results are similar to the previous ones - replay buffer allows to gradually increase complexity, however it doesn't help much in solving evaluation instance.

REFERENCES

Coward S. et al. Beukman M. Refining minimax regret for unsupervised environment design. *arxiv*, pp. 2402.12284, 2024.

Willems L. et al. Boisvert C. Minimalistic gridworld environment for ope- nai gym. *https://github.com/maximecb/ gym-minigrid*, 2018.

Deloup F. Bonfarte G. The genus of regular languages. *Mathematical Structures in Computer Science*, 28(1):14–44, 2018.

Chandra Book R. Inherently nonplanar automata. *A.K. Acta Informatica*, 6:89–94, 1976.

Dennis M. et al. Bruce J. Genie: Generative interactive environments. *arxiv*, pp. 2402.15391, 2024.

Eisenstat S. et al. Demaine E. Solving the rubik's cube optimally is np-complete. *In 35th Symposium on Theoretical Aspects of Computer Science (STACS 2018). Leibniz International Proceedings in Informatics (LIPIcs)*, 96:24:1–24:13, 2018.

Jaques N. Jaques et al. Dennis M. Emergent complexity and zero-shot transfer via unsupervised environment design. *arxiv.org*, pp. 2012.02096, 2021.

Schmidhuber J. Hochreiter S. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Dennis M. et al. Hughes E. Open-endedness is essential for artificial superhuman intelligence. *arxiv*, pp. 2406.04268, 2024.

Pin J. *Handbook of automata theory. Volume I. Theoretical foundations*, volume 1. Berlin: European Mathematical Society (EMS), 2021.

Dennis M. et al. Jiang M. Replay-guided adversarial environment design. *arxiv.org*, pp. 2110.02439, 2022.

Beukman M. et al. Matthews M. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. *arxiv.org*, pp. 2402.16801, 2024.

Jiang M. et al. Parker-Holder J. Evolving curricula with regret-based environment design. *arxiv*, pp. 2203.01302, 2024.

Bradley A. et al. Zhou H. What algorithms can transformers learn? a study in length generalization. *arxiv*, pp. 2310.16028, 2023.

A Appendix

You may include other additional sections here.