Curricula for Learning Robust Policies with Factored State Representations in Changing Environments

Panayiotis Panayiotou
Department of Computer Science
University of Bath
Bath, United Kingdom
pp2024@bath.ac.uk

Özgür Şimşek
Department of Computer Science
University of Bath
Bath, United Kingdom
o.simsek@bath.ac.uk

Abstract

Robust policies enable reinforcement learning agents to effectively adapt to and operate in unpredictable, dynamic, and ever-changing real-world environments. Factored representations, which break down complex state and action spaces into distinct components, can improve generalization and sample efficiency in policy learning. In this paper, we explore how the curriculum of an agent using a factored state representation affects the robustness of the learned policy. We experimentally demonstrate three simple curricula, such as varying only the variable of highest regret between episodes, that can significantly enhance policy robustness, offering practical insights for reinforcement learning in complex environments.

1 Introduction

Reinforcement learning has had remarkable success across a wide range of domains, including energy management [30], robotic control [26], and strategic board games [24]. However, in many applications, performance is evaluated solely on the training environment, often neglecting the importance of generalisation. This lack of emphasis contributes to some of the central challenges in reinforcement learning, including weak transferability between tasks and the brittleness of policies to small changes in environments or random seeds [32, 8, 14, 27]. Additionally, reinforcement learning algorithms often suffer from low sample efficiency, requiring large amounts of data to achieve robust performance.

Factored representations [20] decompose high-dimensional, unstructured state and action spaces into a few low-dimensional and high-level variables, each representing distinct and potentially independent aspects of the environment. This decomposition reduces the problem's dimensionality, possibly requiring fewer samples to learn a well-performing policy [29, 2]. Additionally, factored representations can enhance a policy's ability to generalise across different parts of the state space, making it more robust and transferable [1, 9].

Curriculum learning [4] is a training strategy that structures the learning process, such as by organising different subtasks in a particular sequence, with the goal of improving the learning speed or final performance. This can involve progressively increasing task difficulty or transferring knowledge between tasks of similar complexity. In reinforcement learning [16, 18, 17], this strategy involves training an agent on a sequence of different tasks, enabling it to leverage the knowledge gained from simpler tasks to tackle more challenging ones. This strategy can improve sample efficiency and enhance the robustness of the learned policies [25]. For example, Quick Chess is a simplified version of chess that starts with easier subgames and gradually introduces the player to the whole game [17]. As shown in Figure 1, early subgames can include only pawns to teach players how pawns move, attack and get promoted.

17th European Workshop on Reinforcement Learning (EWRL 2024).

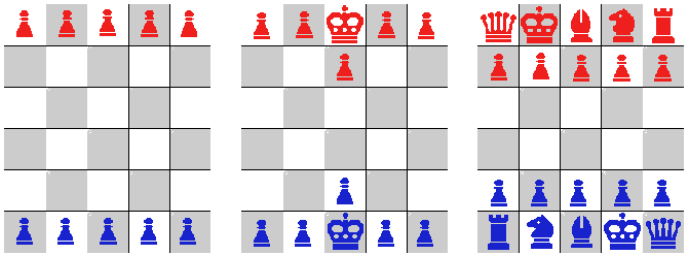

Figure 1: *Quick Chess subgames, increasing in complexity from left to right (image source: Narvekar et al. [17]).*

The real world is non-stationary and unpredictable, and we cannot capture all its variability in a static dataset or learning environment. In the real world, no two tasks are ever exactly the same, even if they may seem so in a simulation environment. Therefore, we aim to train robust policies that generalise effectively and adapt to unseen environments caused by distributional shifts [13]. For example, a domain shift might change the position of an object between different runs, or a task shift might change the designated endpoint in a navigation task.

While factored representations can help in learning more robust policies [1, 9], the role of curriculum learning in enhancing these policies remains underexplored. In this paper, we experimentally investigate how curriculum learning can improve the generalisation and adaptability of these policies to novel environments. We demonstrate the following:

1. Without factored representations, simple curricula are insufficient for training robust policies that generalise well to unseen environments.
2. Using factored representations, a curriculum of random shifts (domain randomisation) can enable learning robust policies.
3. Using factored representations, a curriculum of shuffling a few diverse examples can allow learning robust policies.
4. Using factored representations, we can design a curriculum for learning robust policies by identifying and adjusting the factors that cause the largest performance discrepancy (regret) when altered.

2 Preliminaries

Markov Decision Processes. A Markov decision process (MDP) is a mathematical framework used to model decision-making problems. An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

- $\mathcal{S}$ is a set of states.
- $\mathcal{A}$ is a set of actions.
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition probability function, where $P(s'|s, a)$ denotes the probability of transitioning to state s' from state s after taking action a.
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function, where $R(s, a, s')$ gives the expected reward for taking action a in state s and transitioning to state s'.
- $\gamma \in [0, 1]$ is a discount factor.

Reinforcement Learning. Most commonly, the reinforcement learning problem is modelled as a Markov Decision Process. In this framework, a policy $\pi(a|s)$ represents the probability of taking action a when the agent is in state s. The objective is to learn a policy that maximises the expected cumulative return $E_\pi[G_t]$, which is the sum of discounted rewards over time when following policy π. The return G_t from time step t is defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

where r_t indicates the reward at time step t. An agent interacts with its environment by taking actions based on its policy, receiving feedback in the form of rewards, and using this feedback to learn and improve its policy over time.

Dynamic Bayesian Networks. A Dynamic Bayesian Network is a probabilistic graphical model representing a set of variables and their conditional dependencies as a directed acyclic graph. It is specifically designed to model sequences of variables over time. In a Dynamic Bayesian Network, the state at time t, denoted $\mathbf{X}_t = (X_{1,t}, X_{2,t}, \ldots, X_{n,t})$, depends on the state at time $t-1$, denoted $\mathbf{X}_{t-1}$. The joint probability distribution over $\mathbf{X}_t$ is given by the product of the conditional probabilities of each variable given its parents in the graph, including temporal dependencies:

$$P(\mathbf{X}_t \mid \mathbf{X}_{t-1}) = \prod_{i=1}^{n} P(X_{i,t} \mid \text{Pa}(X_{i,t})),$$

where the parents of $X_{i,t}$, denoted as $\text{Pa}(X_{i,t})$, include variables from both $\mathbf{X}_{t-1}$ and $\mathbf{X}_t$.

Factored Representations. Factored representations can decompose the state and action spaces into sets of variables, each representing different components of the environment. Formally, an atomic state s is represented as a vector of high-level factors $s = (x_1, x_2, \ldots, x_n)$, and similarly an atomic action $a = (y_1, y_2, \ldots, y_m)$.

Factorisation of MDPs. A Factored Markov decision process is a type of MDP in which the state space, action space, transition model, and sometimes the reward function are represented in a factored form. Factored models leverage structure in the problem to manage complexity. They can make solving larger MDPs more computationally feasible without losing accuracy because they represent the MDP more compactly, reducing the number of parameters. They can also generalise better in environments with large state or action spaces, allowing for more efficient policy learning and planning.

The atomic state s and atomic action a can be represented as a factored representation of high-level factors. The transition probabilities $P(s'|s, a)$ depend on a subset of state and action variables and are often represented using a Dynamic Bayesian Network. Similarly, the reward function can be defined as the sum of local reward functions R_i, depending only on a subset of the state and action variables.

Distribution Shifts. Distribution shifts refer to changes in the data distribution encountered by an agent during different phases of learning, such as between training and testing. In reinforcement learning, the environment is often characterised by a set of variables that define its state and dynamics, such as the transition probabilities, reward functions, or physical properties (e.g. grid size in a grid-world task, friction coefficients in a robotic simulation, etc.). A distribution over environments refers to the probabilistic distribution of these variables. By sampling from this distribution, we obtain different instances of the environment, each with potentially different characteristics. Addressing distribution shifts is critical because real-world environments are typically non-stationary, meaning that the variables defining the environment can change over time.

We can distinguish between three different types of learning environments [13]. First, there are singleton environments where the training and testing environments are identical. Secondly, there are independent and identically distributed (IID) environments where training and testing environments are different but from the same distribution. Thirdly, there are out-of-distribution environments where the training and testing environments are from different distributions.

Low-Regret Policies. Regret is a measure of how much the performance of a policy (expected discounted cumulative reward) falls short of the optimal performance. Formally, the regret after T time steps of following policy π from an initial state s_0 can be defined as:

$$Regret_{\pi}(s_0, T) = \sum_{t=0}^{T} \left(V^*(s_t) - V^{\pi}(s_t) \right),$$

where π^* is the optimal policy, $V^*(s_t)$ is the value function of the optimal policy at state s_t representing the expected discounted cumulative reward from that state, and $V^{\pi}(s_t)$ is the value function of the current policy π at state s_t.

We consider low-regret policies robust because a low regret ensures that the performance difference compared to the optimal policy is minimised, demonstrating the policy's ability to handle various scenarios and adapt to changes effectively.

3 Background

> "No man ever steps in the same river twice."
>
> – Heraclitus

To effectively apply reinforcement learning in the real world, we must account for its non-stationary nature. Reflecting the idea of a constantly evolving environment, recent reinforcement learning research focuses on developing robust policies that can handle changing dynamics [12, 10, 11], highlighting the need for policies that work in varied settings. Distribution shifts can significantly impact performance, leading to poor generalisation and arbitrarily high errors [22, 21]. For reinforcement learning to be successful in the real world, we must consider robustness and how shifts (e.g. an object changing colour) can impact both the domain [7] and the task itself [31].

Factored state representations, which involve breaking down the environment into distinct components, are an active area of research [28, 15, 3]. These representations have been shown to improve the sample efficiency of reinforcement learning algorithms in both tabular and deep reinforcement learning methods [29, 2]. Additionally, they can help learn policies that are robust to domain shifts [1, 9]. It has also been proven that in scenarios where only the agent's decisions causally influence the reward (e.g. multi-armed bandits where the state does not affect the reward), all robust agents learn an approximate causal model [21], which implies a factored representation.

Curriculum learning in reinforcement learning structures an agent's learning process by strategically ordering tasks that the agent experiences [17]. It typically aims to enhance the agent's performance and learning speed by enabling the forward transfer of skills from simpler tasks to more challenging ones. A structured curriculum involves several key decisions: choosing the initial set of tasks, defining the progression of tasks, and establishing criteria for transitioning between them. Examples of such curricula include the work of Silva and Costa [23], where tasks are randomly generated and grouped based on their "transfer potential", and Narvekar et al. [17], where a set of source tasks is continuously refined to match the agent's current abilities using methods like mistake-driven subtasks, which help the agent correct erroneous behaviour. Similarly, unsupervised environment design [6] is a reinforcement learning training strategy that automatically generates a series of training environments to learn robust policies. Notable work in this area is ACCEL [19], which uses an evolutionary environment generator and regret-based feedback to make small edits to the environment and gradually introduce the agent to more complexity to train a robust policy.

4 The Shifting Frozen Lake

We define the Shifting Frozen Lake environment, where aspects of the environment can exhibit a shifting behaviour, allowing us to test for out-of-distribution generalisation.

Frozen Lake [5] is a grid-world environment where the agent navigates from a designated start cell (top-left) to a goal cell (bottom-right). The agent can move up, down, left, or right, and must avoid falling into holes along the way. Depending on the configuration, the actions can be either stochastic or deterministic. For an example, refer to Figure 2.

In the original Frozen Lake environment, the start location, goal location, hole locations, and grid size are kept constant throughout all the episodes. In Shifting Frozen Lake, the grid size N, the positions of the holes, the starting point, and the goal location can change from one episode to the next. For simplicity, we assume that these variables remain constant during an episode despite potential changes, such as warm weather that could cause the lake to start melting. Due to the changing nature of the environment (e.g. the start location might change), we will refer to different instances of the environment as "examples".

Below is a full specification of the task:

- Actions: Left, down, right, up with deterministic transitions.

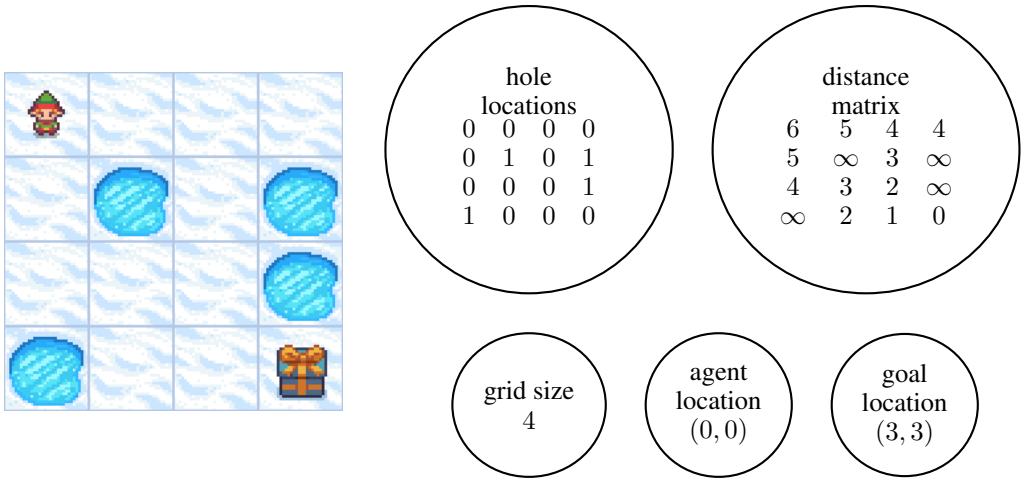

Figure 2: *A sample Frozen Lake environment. On the right hand side, we present a factored representation of the state. Using this factored representation, the transition function can be factorised using a Dynamic Bayesian Network (see Figure 12 in Appendix A).*

- State: This is an $N \times N$ matrix that shows where the agent, the goal, the holes and the frozen squares are. We can factorise this using the 5-tuple (grid size, hole locations, agent location, goal location, distance matrix) as seen in Figure 2.
- Start state: An initial location $[x_I, x_I]$ where $N \times N$ are the dimensions of the grid, and $0 \leq x_I, y_I < N$.
- Goal state: A location $[x_G, y_G]$ where $N \times N$ are the dimensions of the grid, and $0 \leq x_G, y_G < N$. All examples always include a possible path from the start state to the goal state.
- Rewards: -0.1 for each move, an additional $+10$ for reaching the goal, and an additional -10 for reaching a hole. The discount factor is $\gamma = 1$.
- The episode ends if the player moves into a hole or the goal state.

Environment shifts. The initial location, goal location, hole locations, and grid size can change from episode to episode. The environment supports the following shifting behaviours and the functionality to switch between them:

- **No Shifting**: The variables are sampled once upon the creation of the environment and remain constant for all episodes.
- **Random Shifting**: At the start of each new episode, the environment uniformly resamples all variables (start location, goal location, hole locations, grid size).
- **Single Preset Variable Shifting**: One variable is specified to shift. Upon the creation of the environment, all variables are sampled once. In each episode, only the chosen variable is resampled.
- **Single Random Variable Shifting**: Upon creation, all variables are sampled. In each episode, one randomly chosen variable is resampled, changed, and reverted at the end of the episode.
- **Stored Examples Shifting**: Upon creation, a sample of N examples is stored. For each new episode, one of these examples is randomly selected and used.

The state can be factorised by using variables that denote the grid size, hole locations, goal location, the current agent location, and a distance matrix from the goal location. However, this factorisation has redundancies, e.g. the hole locations can be inferred from the distance matrix. We can optimise the factored representation by retaining only the relevant variables, reducing redundancy and improving efficiency. For example, using only the distance matrix and the current agent location, an agent can learn an optimal and robust policy by always taking the shortest path to the end.

5 Experiments

Our experiments include the following agents:

- **Random Action Selection**: Selects action uniformly at random. Used as a baseline.
- **Optimal**: Achieves the highest possible performance by using breadth-first search to pick the direction with the smallest distance to the goal (without falling into a hole).
- **PPO**: Without using a factored representation, we apply a convolutional neural network to the grid, where each tile is one-hot encoded in a separate channel. We pad the grid with a special character so all grids have the same size.
- **PPO-F**: A PPO agent using an optimised factored representation, retaining only the immediate neighbourhood in the distance matrix, which is sufficient for the agent to act optimally in this task. The agent does not model the transition function or use the assumption that the transition function can be factorised.

We run all the experiments for five agents and plot the mean and standard error of the total undiscounted reward per epoch ($\gamma = 1$). Each epoch consists of 900 time steps, and each episode has a timeout of 100 time steps. Performance scores around -30 indicate "stuck" behaviour, where agents avoid losses by engaging in repetitive, looping movements, such as endlessly alternating between left and right actions. Scores higher than -30 but worse than optimal performance indicate an agent that solves some of the grids. For these experiments, we consider random shifting (resampling all variables at the start of each episode) as a test of deep understanding and generalisation of the task because it requires agents to know how to navigate to the goal from anywhere and avoid holes.

We explore the following curricula, with changes in curriculum phases indicated by vertical dotted lines in the figures:

(A) **No Shifting to Random Shifting**: Fit a single example, then shift all variables randomly to test generalisation.

(B) **No Shifting to Single Random Variable Shifting**: Fit a single example and then randomly shift only one variable per episode.

(C) **Random Shifting**: Test generalisation from diverse training (domain randomisation) by shifting all variables randomly from the start.

(D) **Stored Examples to Random Shifting**: Train a policy by shuffling a few pre-sampled examples and then test generalisation by shifting all variables randomly.

(E) **Single Preset Variable Shifting to Random Shifting**: Shift only one specified variable initially, then shift all variables randomly to test generalisation.

Curriculum (A): No Shifting to Random Shifting. We test generalisation from a single example and present the results in Figure 3. When fitting a single example, the methods show a significant standard error because the grid size can vastly change the reward per epoch. For instance, reaching the goal in 3 steps on a 4x4 grid gives 970 points per epoch, while 20 steps on a 10x10 grid give only 120 points per epoch. None of the trained methods demonstrate significant knowledge transfer after the shift, as their performance drops to around 0. After the shift, *PPO* exhibits "stuck" behaviour, repeatedly moving left/right or up/down. *PPO-F* is more active but only solves about 25% of the examples right after the shift. It loses in around 9% of the examples and displays "stuck" behaviour in the rest.

Curriculum (B): No Shifting to Single Random Variable Shifting. In Figure 4, we see that both *PPO* and *PPO-F* exhibit similarly low knowledge transfer and robustness as there is a big performance drop when random shifting starts. Notably, after a few epochs with random shifting, *PPO-F* adapts quickly to the new task distribution.

Curriculum (C): Random Shifting. In Figure 5, we evaluate how well the agents can generalise from diverse training. The test and train distributions of the environment here are identical, so this is IID generalisation. Note, however, that diverse training complicates the learning task. A closer examination of the *PPO* agent behaviour reveals that it often fails to reach the goal in any

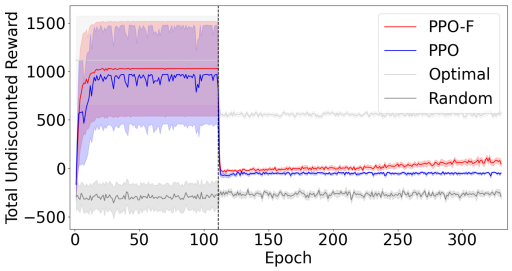

Figure 3: *Curriculum (A): No Shifting to Random Shifting.*

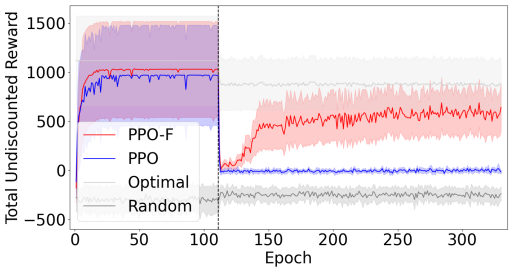

Figure 4: *Curriculum (B): No Shifting to Single Random Variable Shifting.*

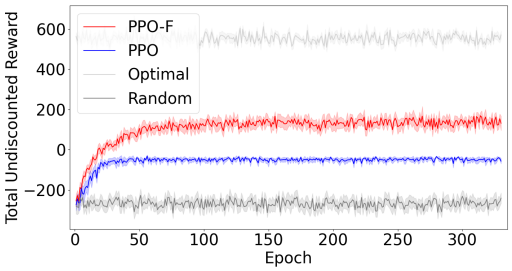

Figure 5: *Curriculum (C): Random Shifting.*

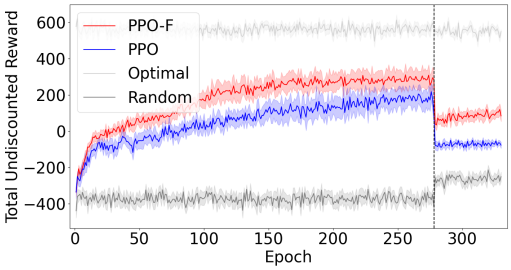

Figure 6: *Curriculum (D): 15 Stored Examples to Random Shifting.*

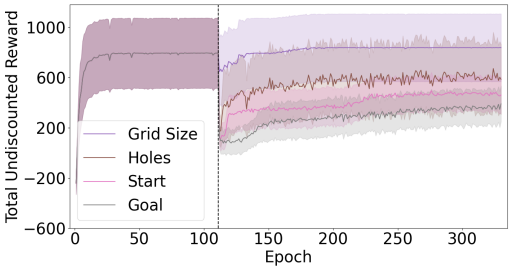

Figure 7: *Preliminary Experiment for Curriculum (E): Fit PPO-F to a single example, then shift only one of the four variables.*

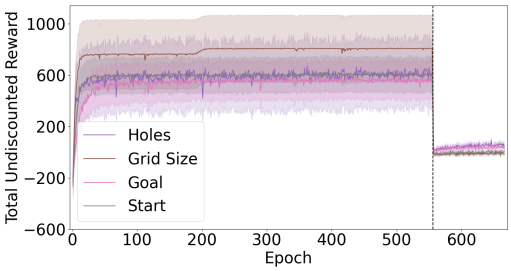

Figure 8: *Curriculum (E): Single Preset Variable Shifting to Random Shifting on PPO-F.*

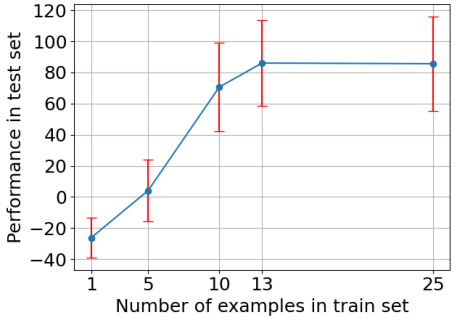

Figure 9: *PPO-F performance after training with different numbers of stored examples, followed by 50 epochs of random shifting (test).*

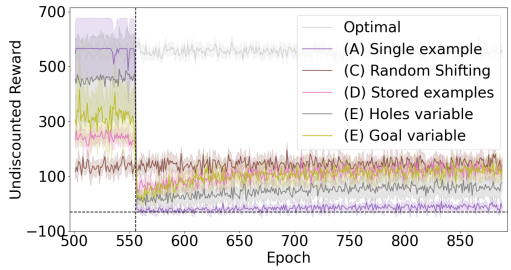

Figure 10: *Comparing Curricula for Factored Agents.*

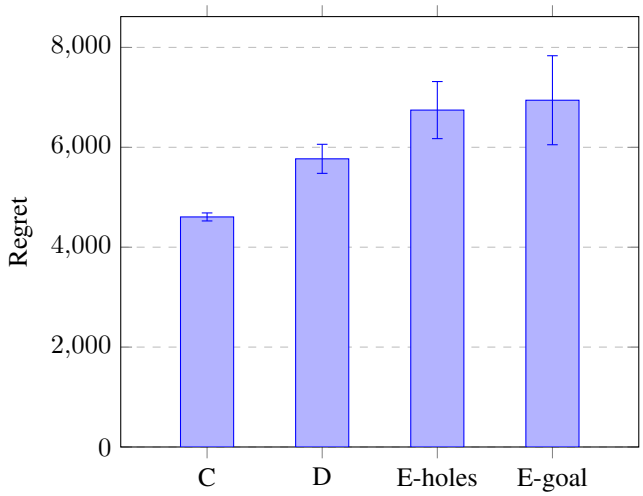

Figure 11: *Comparing the regret of robust policies learnt by following Curricula C, D, E with varying holes, and E with a varying goal location.*

episode, reverting to "stuck" behaviour. Factored variables, however, provide a significant advantage in discovering the task structure. The *PPO-F* agent identifies a robust policy within 75 epochs. But its policy is far from optimal. We examined the agent's performance over 50 epochs following stabilisation and found that 19% of its movements were repetitive, back-and-forth motions. It executed $3.6\times$ more moves than the optimal agent and, on average, fell into a hole 24.8 times per epoch.

Curriculum (D): Stored Examples to Random Shifting. In Figure 6, we evaluate how well the agent generalises using only 15 training examples. *PPO* exhibits "stuck" behaviour after the shift and shows no signs of knowledge transfer or robustness. *PPO-F* shows strong knowledge transfer and robustness, performing on par with diverse training after seeing only 15 examples. We examined the agent's performance over 50 epochs following stabilisation and found that 35% of its movements were repetitive, back-and-forth motions. It executed $4.5\times$ more moves than the optimal agent and, on average, fell into a hole 7.8 times per epoch. This experiment demonstrates that a few diverse examples are sufficient to build a robust policy over a factored state representation.

Curriculum (D) Follow-Up. We further investigate in Figure 9 how many stored examples are needed to achieve good test performance under Random Shifting. We train multiple *PPO-F* agents with different numbers of training examples and estimate their test performance by averaging over 50 epochs under Random Shifting. Generally, we expect more diverse training with more examples to correlate with improved performance. However, there are diminishing returns, as fitting more examples takes longer and does not necessarily result in better performance. Performance gains level off after fitting 13 examples, and training with more examples significantly increases the training time.

Preliminary Experiment for Curriculum (E). We train *PPO-F* on a single example and then shift only a specific variable on each episode (Single Preset Variable Shifting), as shown in Figure 7. We find that shifting the goal location, start location, and hole locations leads to high regret while shifting the grid size does not. In Curriculum (E), we then investigate if shifting only one of these during training is enough to learn a robust policy.

Curriculum (E): Single Preset Variable Shifting to Random Shifting. In Figure 8, we evaluate how well the *PPO-F* agent generalises when only one variable is shifted during training. We examine four training curricula, each shifting only one variable (holes, grid size, goal location, or start location). We test OOD generalisation by exposing the agents to random shifts. We find that varying just one variable, either hole locations or goal location, leads to learning a robust policy. Two of the three variables that cause high regret are sufficient by themselves when shifted to form a curriculum for training a robust agent. We examined the agents' performance over 50 epochs following stabilisation and found that when changing the holes, they executed $6.8\times$ more moves than the optimal agent and,

on average, fell into a hole 5.2 times per epoch. When changing the goal location, they executed $8.8\times$ more moves than the optimal agent and, on average, fell into a hole 5.4 times per epoch.

Comparing Curricula for Factored Agents. In Figure 10, we compare *PPO-F* agents trained with four robust curricula (C, D, E with varying holes, and E with varying goals) and a single-example curriculum (A). All robust curricula outperform the single-example curriculum after the start of random shifting. Training with random shifts gives the best immediate post-shift performance. However, the other robust curricula perform similarly. Curricula (D) and (E) (with a varying goal) adapt quickly and reach the same performance as (C). The horizontal dotted line represents the "stuck" behaviour observed by the single-example curriculum after the shift. Pre-shift performance is not comparable between agents because each curriculum exposes agents to different environments, and smaller grids lead to higher total rewards per epoch.

Regret Analysis of Robust Policies. In Figure 11, we rank the four robust curricula (C, D, E-holes, E-goals) by increasing regret and decreasing robustness (from left to right). This ranking also reflects decreasing risk-taking and falling in holes. Curriculum (C) benefits from test and train environments being identical (IID generalisation). The two (E) curricula only modify one variable at a time but learn robust policies. We suggest that changing the variable that shifts during training could further enhance policy robustness.

6 Discussion

First, our results demonstrate that methods using factored representations can help learn robust policies more easily. Agents using an atomic state representation usually fail to reach the goal when the environment has distribution shifts. While a tailored curriculum could help such agents to learn robust policies, simple curricula may be enough for agents that use a factored representation. Secondly, the curriculum used significantly impacted the robustness of the learned policy over a factored state representation. The agents learned comparably robust policies with either diverse training, shuffling a few stored examples, or by shifting a single variable that caused high regret when altered alone (true for two out of three variables). We also quantitatively compare the robustness of the learned policies following each of the curricula and point out the effect of the curriculum on the risk aversion and performance of the learned policies. Lastly, we believe that enabling agents to autonomously generate their own curricula by identifying and adjusting variables that require further exploration (such as those causing high regret) will lead to learning even more robust policies and better generalization across diverse environments.

Acknowledgements

This work was supported by the UKRI Centre for Doctoral Training in Accountable, Responsible and Transparent AI (ART-AI) [EP/S023437/1]. We thank Joshua B. Evans for useful discussions.

References

[1] Mahsa Baktashmotlagh, Masoud Faraki, Tom Drummond, and Mathieu Salzmann. Learning factorized representations for open-set domain adaptation. *arXiv preprint arXiv:1805.12277*, 2018.

[2] Bharathan Balaji, Petros Christodoulou, Xiaoyu lu, Byungsoo Jeon, and Jordan Bell-Masterson. Factoredrl: Leveraging factored graphs for deep reinforcement learning. In *NeurIPS 2020 Workshop on Deep Reinforcement Learning*, 2020.

[3] Yoshua Bengio. Deep learning of representations: Looking forward. In *International conference on statistical language and speech processing*, pages 1–37. Springer, 2013.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[6] 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. *Advances in neural information processing systems*, 33:13049–13061, 2020.

[7] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

[8] Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

[9] Fan Feng, Biwei Huang, Kun Zhang, and Sara Magliacane. Factored adaptation for non-stationary reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 31957–31971, 2022.

[10] Omer Gottesman, Fredrik Johansson, Joshua Meier, Jack Dent, Donghun Lee, Srivatsan Srinivasan, Linying Zhang, Yi Ding, David Wihl, Xuefeng Peng, et al. Evaluating reinforcement learning algorithms in observational health settings. *arXiv preprint arXiv:1805.12298*, 2018.

[11] Jixian Guo, Mingming Gong, and Dacheng Tao. A relational intervention approach for unsupervised dynamics generalization in model-based reinforcement learning. *arXiv preprint arXiv:2206.04551*, 2022.

[12] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[13] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76: 201–264, 2023.

[14] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning: State-of-the-Art*, pages 143–173. Springer, 2012.

[15] Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *International conference on machine learning*, pages 6348–6359. PMLR, 2020.

[16] Sanmit Narvekar. Curriculum learning in reinforcement learning. In *IJCAI*, pages 5195–5196, 2017.

[17] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Source task creation for curriculum learning. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, pages 566–574, 2016.

[18] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[19] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pages 17473–17498. PMLR, 2022.

[20] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[21] Jonathan Richens and Tom Everitt. Robust agents learn causal world models. *arXiv preprint arXiv:2402.10877*, 2024.

[22] Stephane Ross and J Andrew Bagnell. Agnostic system identification for model-based reinforcement learning. *arXiv preprint arXiv:1203.1007*, 2012.

[23] Felipe Leno Da Silva and Anna Helena Reali Costa. Object-oriented curriculum generation for reinforcement learning. In *Proceedings of the 17th international conference on autonomous agents and multiagent systems*, pages 1026–1034, 2018.

[24] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[25] Yeeho Song and Jeff Schneider. Robust reinforcement learning via genetic curriculum. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5560–5566. IEEE, 2022.

[26] Peter Stone, Richard S Sutton, and Gregory Kuhlmann. Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[27] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

[28] Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable factors. *arXiv preprint arXiv:1708.01289*, 2017.

[29] Yao-Hung Hubert Tsai, Paul Pu Liang, Amir Zadeh, Louis-Philippe Morency, and Ruslan Salakhutdinov. Learning factorized multimodal representations. *arXiv preprint arXiv:1806.06176*, 2018.

[30] Tianshu Wei, Yanzhi Wang, and Qi Zhu. Deep reinforcement learning for building hvac control. In *Proceedings of the 54th annual design automation conference 2017*, pages 1–6, 2017.

[31] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

[32] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

A Additional Figures

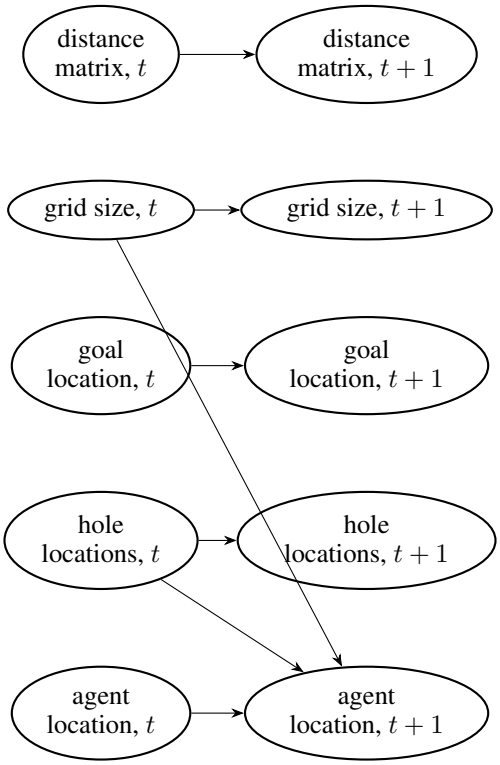

Figure 12: A Dynamic Bayesian Network for the factored MDP of the Shifting Frozen Lake. The distance matrix (from the goal location), the grid size, the goal location and the hole locations are constant throughout each episode.