Curriculum Generation and Sequencing for Deep Reinforcement Learning in StarCraft II

DANIEL HAO, PENNY SWEETSER, and MATTHEW AITCHISON, The Australian National University, Australia

Reinforcement learning has proven successful in games, but suffers from long training times when compared to other forms of machine learning. Curriculum learning, an optimisation technique that improves a model's ability to learn by presenting training samples in a meaningful order, known as curricula, could offer a solution for reinforcement learning. Due to limitations involved with automating curriculum learning, curricula are usually manually designed. However, due to a lack of research into effective design of curricula, researchers often rely on intuition and the resulting performance can vary. In this paper, we explore different ways of manually designing curricula for reinforcement learning in real-time strategy game, StarCraft II. We propose three generalised methods of manually creating tasks for curriculum learning and verify their effectiveness through experiments. We also experiment with different curricula sequences, in addition to the most commonly used easy-to-hard order. Our results show that all three of our proposed methods can improve a reinforcement learning agent's learning process when used correctly. We demonstrate that modifying the state space of the tasks is the most effective way to create training samples for StarCraft II and that reversed curricula can be beneficial to an agent's convergence process under certain circumstances.

CCS Concepts: • **Computing methodologies** → **Reinforcement learning**; • **Software and its engineering** → **Interactive games**; • **Applied computing** → **Computer games**.

Additional Key Words and Phrases: Game AI; Reinforcement Learning; Real-Time Strategy Games; StarCraft II

ACM Reference Format:

Daniel Hao, Penny Sweetser, and Matthew Aitchison. 2022. Curriculum Generation and Sequencing for Deep Reinforcement Learning in StarCraft II. In *ACSW '22: Australasian Computer Science Conference, February 14–18, 2022, Online.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Reinforcement learning (RL) is a type of machine learning in which an agent is placed into a problem environment and trained via basic trial-and-error actions. An RL agent continuously selects and performs actions that will affect its environment and, in turn, influence the RL agent's following course of action. In contrast to supervised learning, the agent is not provided with labelled data. Instead, rewards and punishments are designed and delivered to the agent to guide its learning process. The purpose of the agent is to learn to maximise the reward that it receives, thus 'learning' to solve a given problem.

Video games are a popular platform for developing and testing RL algorithms, as they are formal, well-defined, complex, and have dynamic environments that react to different actions. Recent state-of-the-art RL algorithms have achieved and surpassed human-level performance in various games, such as Atari [23] and Go [20]. RL algorithms such as Deep Q-learning (DQN) [11] and Advantage Actor-Critic (A2C) [28] have been widely deployed and have

Manuscript submitted to ACM

achieved promising results. However, in more complex tasks, such as real-time strategy (RTS) games, RL suffers from long training times, requiring hundreds of millions of training steps [24], which can take up to weeks to complete depending on the performance of the hardware. Due to the incredibly large state and action spaces and sparse reward nature of such tasks, vanilla RL agents with no further optimisations are incapable of mastering RTS games within a reasonable time [2]. As a result, shortening the training time has the potential to significantly benefit RL agents and the outcomes in complex environments. Previous research has employed techniques such as curriculum learning [15] to optimise RL. In this paper, we review and investigate previous research on curriculum learning in the field of RL and propose and evaluate new methods for effectively applying curriculum learning to RL in the RTS game, StarCraft II.

2 CURRICULUM LEARNING

Curriculum learning (CL) is a training strategy inspired by the human learning process, mimicking how humans learn new concepts in an orderly manner, usually based on the level of difficulty of the problems [9]. CL states that by exposing machine learning algorithms to tasks in meaningful orders, it can speed up convergence and can even lead to the discovery of better local minima [3]. RL agents trained on tasks with increasing difficulties, starting with easier environments and then scaling up to more challenging environments, can converge to an optimum more quickly than an agent trained without a curriculum [1]. CL is relatively easy to use, as it does not necessarily require expert knowledge [17] and has exhibited promising results in numerous applications [3].

Although the concept of CL is simple, there are two main challenges when applying it to RL. The first is identifying the most effective way of generating curricula and the second is determining the best curriculum sequence. For generating curricula, numerous criteria can serve as the measure of difficulty for problems, such as using the distance between the initial state and goal state [6] or the size of the environment [12]. However, there is a lack of previous research that compares the performance of different difficulty measures. Depending on the problem itself, various domain-specific measures could also be used for difficulty evaluation [12], further complicating the issue. Furthermore, all existing research efforts on generating tasks for CL in RL have focused on attempting to automate the process and all involve some degree of limitation [22].

The problem of curriculum sequencing is also non-trivial [13]. While the easy-to-hard sequence matches the analogy of how humans learn, it might not be the most suitable structure for RL agents. There have been cases where a reversed curriculum showed stronger convergence properties when applied to neural networks [26]. While this behaviour has yet to be observed in RL, there is no evidence to support the claim that an easy-to-hard sequence is always the best sequence to use. The problem of curriculum sequencing in RL for handcrafted curricula has not been previously explored; to address this problem, we evaluate RL agents trained under different curriculum sequences with the same set of tasks. We compare the easy-to-hard curriculum with other course sequences and draw conclusions from the results.

2.1 Curriculum Generation

While there are various approaches to automating task generation for CL (e.g., generative adversarial networks [5] and evolutionary generators [4]), of which many have exhibited exceptional performance, most involve some drawbacks in terms of efficiency or make assumptions to simplify problems. Curriculum generator approaches to task generation require the generators to be optimised prior to deployment [4, 5]. For real-world applications, it would be counter-intuitive if more time were spent on training an automated curriculum generator than on training the agent itself. However, other approaches such as reverse curriculum generation [6] or sub-task generation [19] are built on a set of

assumptions that may not hold true for all problems, such as assuming that there is a tangible distance between the goal state and the initial state, which fails if the goal does not require the agent to be in a specific state or configuration. As a result, in practice, it is more common to manually design tasks for RL agents than to use an automated approach. However, there is a lack of previous research that examines how to effectively handcraft curricula for RL agents and no clear guideline on the best practice to follow. When using CL to accelerate training, users often design curriculum tasks from an intuitive approach [15, 18], manually creating curriculum tasks guided by a rough understanding of the intended learning outcome of the agent.

In this paper, we propose three domain-independent measures for creating curricula, which are reward-based, state-based, and action-based measures of difficulty. We design and evaluate curricula with each measure to discover the most effective design methodology. We aim to discover the most suitable way to handcraft curricula so that RL agents can receive the maximum benefits from CL. We investigate effective generalised curriculum generation methods for handcrafting curricula and verify their effectiveness with experiments. The results of this paper are intended to provide guidance to practical implementations of CL by evaluating the performances of the proposed handcrafted curriculum structures.

2.2 Curriculum Sequencing

Given a set of curriculum tasks, the problem of ordering tasks effectively for training with CL is an active research area. However, in contrast to curriculum generation, the problem of curriculum sequencing has received relatively little scholarly attention. In most existing research, the curriculum tasks have been sequenced manually, either by domain experts [18] or non-expert users [17]. Most researchers have followed the most common easy-to-hard sequence when ordering their curriculum tasks and rarely considered other possible orders [27]. It is possible that other curriculum sequences may offer more benefits to an RL agent compared to an easy-to-hard curriculum sequence. Thus, we ask the question: Can a reversed or mixed curriculum sequence benefit a deep RL agent in an RTS game more than an easy-to-hard curriculum sequence?

One particular approach to automatic task sequencing is to formulate the design of a curriculum as a Markov decision process (MDP). This approach models the accumulation of knowledge as a curriculum agent in the form of an MDP [13], which actively selects tasks for the learning agent to train on. The goal of the curriculum agent is to discover an optimal policy that minimises the time required for the learning agent to converge. This approach uses Monte Carlo approximations to update and optimise the policy of the curriculum agent directly. The trained model is able to produce an agent-specific curriculum for three different learning agents with the same target task and exhibited improvements over learning with no curricula.

Another approach to selecting suitable source tasks for training is to model the transferability and expected benefits between source-target pairs. This method creates an agent that uses low-dimensional feature vector task descriptors to learn the expected benefit of transferring between source tasks and target tasks. The resulting CL agent has shown significant improvements over the baseline agent trained with no curricula [21]. However, this scenario only considers the transfer of knowledge from one single source task and does not apply to curricula involving multiple source tasks.

Both approaches to task sequencing in CL share the same limitation: they both require the collection of a large amount of experience in the source tasks [13, 21], in a similar way to how a generator needs to be trained a priori for automatic task generation. As a result, automating curriculum sequencing for a single agent is impractical and inefficient. In cases where CL is used for the sole purpose of elevating the training progress of an agent, it is still more

efficient to select the curricula manually. However, current manual designs of curricula in RL generally follow the easy-to-hard sequencing order [15], and other methods of ordering source tasks are usually not explored.

In this paper, we investigate the effectiveness of reversed and mixed curriculum sequences by comparing them to a baseline agent with no curricula. We use our curriculum generation methods to create additional curriculum tasks and sequence them in a mixed or reversed order. Previous works have compared an easy-to-hard sequence to a reversed sequence with a single difficulty evaluation method [25]. Our work differs from others in that we draw comparisons between multiple curriculum sequences across a variety of curriculum tasks generated by different design methodologies. We describe the curriculum settings in detail, present and analyse our results, and discuss their implications.

3 METHOD

We propose three domain-independent methods for manually generating curricula for CL. We use the game Starcraft II as our training and evaluation platform to measure the effectiveness of our proposed methods. Each proposed curriculum generation method is evaluated with two different problems (mini-games) to verify their effectiveness under different problem settings.

To investigate the effectiveness of different curriculum sequences, we train and compare the performance of agents with different curriculum sequences against a baseline agent trained with no curricula. We do this for each of our three three domain-independent curriculum generation methods. It is possible that certain curriculum generation methods might require a sequencing order other than the most commonly used easy-to-hard order. In this section, we describe our test environment, agent, scenarios, curriculum generation methods, and how the tasks are ordered for different sequences.

3.1 StarCraft II

Starcraft II (SC2) is a popular RTS game developed and published by Blizzard in 2010. It is a competitive game that is mostly played between two players (1v1), which is the game mode used in professional SC2 competitions. From the perspective of an RL agent, SC2 poses several challenges. First, it is an imperfect information game with only partially-observable states. The agent must learn to control units to explore unknown areas and make decisions based on incomplete information. Second, the action space and state space are extremely large and diverse; there are an estimated 10^8 possible actions for selection between every frame [24] and the branching factor for two consecutive states is estimated to be between $b \in [10^{50}, 10^{200}]$ even when only considering units [14]. Third, the nature of strategic games dictates that rewards for a strategic action may not be delivered until much later on, leading to credit assignment problems with sparse rewards. Many components of SC2 can be broken down into individual problems of their own, including build order optimisation [7], micromanagement [18], and macro-management [8].

In 2017, DeepMind and Blizzard collaboratively developed and released a library named PySC2, exposing Blizzard's SC2 machine learning API as a Python environment [24]. It provides an interface for RL agents to interact with SC2 by allowing the agent to receive spatial and non-spatial features that are similar to those that a human player would receive and thus perform actions similarly. In this paper, we use the SC2 game and the PySC2 library for our experiments. We have chosen SC2 as our platform as it is a complex and challenging task for RL agents. It is also scalable in difficulty and easy to use, as we can access and edit variables in the environment with the SC2 map editor and the Python PySC2 library.

Fig. 1. Initial game state for Defeat Roaches (left) and Collect Mineral Shards (right) mini-games.

3.2 Mini-Games

We selected two SC2 mini-games to test our proposed methods: Defeat Roaches and Collect Mineral Shards. A mini-game is a small stand-alone game created by extracting certain elements within the full game of SC2.

3.2.1 Defeat Roaches. The Defeat Roaches mini-game tasks the agent with controlling a group of nine marines to defeat four enemy roaches (see Figure 1). Whenever all four enemy roaches are defeated, a new group of four enemy roaches is spawned, and the agent is rewarded with five additional marines, while the remaining marines retain their existing health. Marines and roaches are randomly spawned on two opposite ends of the map, with their positions reset every time a group of roaches is defeated. All marines are initially selected for the agent when spawned. The camera is locked to the centre of the screen with no fog of war so that no camera movements are required. The game state is reset whenever all marines are killed or upon reaching the time limit of 120 seconds. The agent earns rewards every time a roach is killed and receives penalties for every marine lost. The optimal strategy requires the agent to micromanage the marines and focus fire on the roaches to minimise incoming damage. To create curriculum tasks with varying difficulties for different generation methods, we tuned parameters including the map size, marine and roach unit statistics, and the reward structure.

3.2.2 Collect Mineral Shards. In the Collect Mineral Shards mini-game, the agent needs to control two marines to collect as many mineral shards as possible (see Figure 1). The marines can collect the mineral shards on the map by stepping on them. The two marines are spawned in random locations on the map and are not preselected for the agent. There are initially 20 mineral shards on the map and whenever all 20 mineral shards have been collected a new group of 20 mineral shards are randomly spawned on the map, each being at least two units away from the marines. As with Defeat Roaches, the camera is locked to the centre of the map with the fog of war disabled, so that no camera movements are required. The game state is reset upon reaching the time limit of 120 seconds. Every time that the agent collects a mineral shard, it receives a reward, and there are no explicit penalties for losing marine units from friendly fire, although it would most likely receive a lower score due to the loss of collection units. The optimal strategy requires the agent to plan its movements efficiently, divide the marines, and move independently. To create curriculum tasks

with varying difficulties for different generation methods, we tuned parameters including the map size, number of marines, number of minerals, and reward structure.

3.3 Agent Structure

For our experiments, we used an Advantage Actor-Critic (A2C) RL agent structure with neural networks as function approximators, as implemented by [16]. The implementation of the agent is a close replica of the FullyConv agent described in the original publication of PySC2 [24]. The A2C agent network receives minimap and screen observations as input. The minimap is a simplified overview of the entire game environment displayed to the player at the bottom left corner of the screen and the screen observations are detailed images of the game environment that the agent is currently viewing. The A2C agent network assumes the minimap and the screen observations are of the same resolution. The observations are passed through two convolutional layers with 16 and 32 filters of sizes 5×5 and 3×3, respectively. The convolutional layers have no strides and use padding on every layer to preserve resolution. The state representation is formed by concatenating the outputs from passing the minimap and screen observations through the convolutional layers. We obtained non-spatial (categorical) actions by sending the state representation through a fully connected feedforward layer with 256 units and ReLU activations followed by fully connected feedforward linear layers. The policy over spatial actions is obtained using size 1×1 convolution over the state representation with a single output channel. In [16], modifications were made to the original FullyConv agent structure to simplify the agent network and accelerate training, including the discarding of non-spatial vectors entirely and the use of A2C instead of A3C [10], which is a non-asynchronous implementation of the algorithm.

3.4 Curriculum Generation Methods

Our three domain-independent methods for curricula generation were reward-based, state-based, and action-based. We modified and created curriculum tasks for each by altering the reward structures, state spaces, or action spaces of the environment, respectively. Domain-independent methods can be applied regardless of the problem's context, providing that the task is represented as a Markov decision process (MDP).

3.4.1 Reward-Based Curricula. In every RL problem defined as an MDP, there is a reward structure that is explicitly tailored to the learning targets of an agent. In reward-based curriculum generation, we create curricular tasks for the agent by relaxing or tightening the requirements for the agent to receive a positive reward. We defined the difficulty of the curriculum tasks as the likelihood of a random agent receiving a reward in the environment. In other words, the difficulty was measured by the probability that a random action would lead to a positive reward; the higher the probability, the easier the problem and vice versa.

We modified the difficulty of the two tasks by changing the values of certain variables. For the Defeat Roaches task, we increased the probability of receiving a reward through random actions by lowering the health of enemy units (see Table 1). Since a reward was delivered every time an enemy unit died, the lower its initial health, the more likely it was to die from random attacks. We could also achieve similar effects by increasing the attack damage of marines. However, this would also increase the chance of killing friendly units with friendly fire. As a result, decreasing the health of enemy roaches is a more reliable method, avoiding introducing unintended side effects. This approach of lowering difficulty through editing statistics of individual units has not been used before. Previous applications of CL in SC2 have created curriculum tasks in ways similar to our state-based or action-based methods, decreasing the size of the environment or number of actions the agent can perform [18]. For the Collect Mineral Shards mini-game, we modified

Table 1. Reward-based curriculum settings for Defeat Roaches (left) and Collect Minerals (right).

Difficulty	Roach Health
Easy	90
Medium	120
Target	145
Hard	160
Extreme	175

Difficulty	Mineral Shards
Easy	40
Medium	30
Target	20
Hard	14
Extreme	7

Table 2. State-based curriculum settings for Defeat Roaches (left) and Collect Minerals (right).

Difficulty	Map Size
Easy	13 x 12
Medium	18 x 14
Target	22 x 16
Hard	24 x 16
Extreme	26 x 16

Difficulty	Map Size	Shards
Easy	11 x 8	5
Medium	16 x 12	11
Target	22 x 16	20
Hard	24 x 16	23
Extreme	26 x 16	25

the likelihood of receiving a reward by increasing the density of mineral shards on the map. The higher the density in relation to the map size, the easier the problem (see Table 1).

3.4.2 State-Based Curricula. In state-based curriculum generation, we also use the properties of an MDP to create curriculum tasks. In this method, we used the size of the state space S to measure the difficulty of the tasks created. The larger the state space (i.e., size of the map), the harder the problem. For both mini-games, we decreased the map size to lower the difficulty of the tasks (see Table 2). For the Collect Mineral Shards mini-game, other than modifying the size of the map, we also needed to modify the number of mineral shards on the map, so that the density of the mineral shards remained constant. We did so to avoid changing the level of difficulty by other difficulty measures (e.g., reward-based difficulty). Otherwise, we would not be able to correctly identify which generation method had contributed to the change in results. As we lower the state-based difficulty by decreasing the map size, we also need to decrease the number of mineral shards on the map to maintain a consistent mineral density in relation to the map size (see Table 2).

3.4.3 Action-Based Curricula. The action-based curriculum generation method uses the definition of the action space A in an MDP to measure difficulty. By reducing the number of marines that the agent needed to control, we effectively reduced the size of the action space and the difficulty of the curriculum task. For Defeat Roaches, other than modifying the number of marines upon spawning and backup, we also needed to modify the strength of the individual units, so that the raw difficulty of the game did not increase as we decreased the number of friendly units (see Table 3). We also needed to make slight modifications to the reward structure as the quality of individual units varied. For the Collect Mineral Shards mini-game, we reduced the number of marines that the agent needed to control to reduce the action difficulty of the task. While reducing the number of units might affect the maximum performance of the agent, as it had less collecting power due to not being able to collect mineral shards simultaneously from two different locations, we did not consider this to be significant. First, reducing the number of units that the agent controlled did not have a substantial effect on the likelihood of it receiving a reward through random actions, as the update frequency of the algorithm limited the frequency of performing an action. Second, by observing the behaviour of the agent in test trials, we found that the agent was never able to simultaneously control two marines to collect mineral shards with different

Table 3. Action-based curriculum settings for Defeat Roaches.

Difficulty	Penalty	Marines	Backup	Health	Damage
Easy	-4	2	1	190	25
Medium	-3	5	3	80	10
Target	-1	9	5	45	6
Hard	-1	13	8	32	5
Extreme	-1	18	10	20	3.4

Table 4. Action-based curriculum settings for Collect Mineral Shards.

Difficulty	Marines	Selected
Easy	1	1
Medium	2	1
Target	2	0
Hard	4	0
Extreme	6	0

paths. The agent always unified the control of the two marines, giving the same commands to both marine units, and thus never utilised the advantages of having two marines over one. As a result, for the Collect Mineral Shards mini-game, we only modified the number of marines and the initial selection of units (see Table 4).

3.5 Curriculum Sequencing

We tested five different conditions for curriculum sequencing (see Table 5). We created a baseline for comparison by training an agent directly on each mini-game with no curricula. For our standard curriculum, we used an easy-to-hard training order, as in previous research [15, 18]. Training on each curriculum task was updated whenever the performance of the agent converged. Our reversed curriculum initially trained the agent on a harder task than that of the target task, and slowly lowered the difficulty as the agent learned, as opposed to a simple reversal of the training order of curriculum tasks (from target to medium to easy). For this purpose, we needed to create additional tasks that are more difficult than the original target task. We designed harder, more difficult curriculum tasks. For reversed curricula, we followed the training order of Extreme to Hard to Target. We did not use the tasks that were easier than the target task in training for reversed curricula.

In addition to experimenting with reversed curricula, we also experimented with mixed curricula, whereby we constructed curricula with tasks with non-conventional mixed orderings. Mixed curricula served as a control for our results. We wanted to know whether the improvements of utilising curriculum tasks came from training in a meaningful order or that only training on curriculum tasks would be sufficient to aid the overall convergence process. We also want to explore different ways of ordering tasks other than the commonly used easy-to-hard sequence. For mixed curricula, we created two curriculum orders, one starting with an easier curriculum task followed by a harder one, and another curriculum order starting with a harder task followed by an easier one.

4 RESULTS AND ANALYSIS

The results from training each curriculum on the two mini-games were generated using data collected during training and presented, for clarity, with a smoothing value of 0.98. Figures 2 (Defeat Roaches) and 3 (Collect Mineral Shards)

Table 5. Curriculum sequences for all curricula.

Curriculum	Sequence
Baseline	Target
Standard	Easy->Medium->Target
Reversed	Extreme->Hard->Target
Mixed 1	Medium->Hard->Target
Mixed 2	Hard->Medium->Target

show a comparison of all Standard, Reversed, Mixed 1, and Mixed 2 curricula and the baseline agents. Final score at the end of an episode is plotted against the number of episodes trained.

4.1 Curriculum Generation Analysis

In this paper, we proposed three different methods for manually creating curriculum tasks for RL and evaluated them against baseline RL agents in two different mini-games in SC2. In this section, we analyse the results of our curriculum generation experiments in Defeat Roaches and Collect Mineral Shards.

4.1.1 Defeat Roaches. We found that action-based curricula offered no improvements over the baseline agent and performed worse than the baseline at various stages of training. For the Defeat Roaches problem, reducing the number of units that the agent controlled (and thus reducing the action space) did not help to accelerate the training process. The ideal strategy for the Defeat Roaches task is to focus fire on enemies one-by-one. We initially predicted that an action-based curriculum would be helpful to the agent, as it lowers the difficulty in controlling multiple units. However, despite controlling fewer units, the agent failed to recognise that focused fire was the ideal strategy to minimise damage taken. We theorise that this could be due to the increased strength of individual marines as we reduced their numbers, leading to the reduced frequency of penalties from the death of friendly units. This created a sparser penalty structure for the agent, which was not beneficial to the agent's learning process and did not help the agent recognise that minimising friendly unit loss was an important goal.

In the reward-based curriculum for Defeat Roaches, minor improvements were evident when compared to the baseline. The agent trained with a reward-based curriculum was able to converge slightly faster than the baseline, with minor improvements in the final result. The high episode score in the first 20 thousand episodes of training was the result of training on easier curriculum tasks. When the training task was updated to the target task, the episode score of the reward agent quickly fell to within range of the baseline agent. From observing behaviours after training, we realised that both agents used a very similar strategy. Both moved their units to either the top-left or top-right corners of the map, depending on their initial location. When moved to a corner, the marines automatically attacked the closest roach due to attack range limitations. This resulted in an imperfect focus of fire on enemies, as some marines had more than one roach within their attack range. The minor improvement of the reward agent was to position the marines in a slightly more effective formation when moved to the corners of the map. First training the agent on maps with lower reward difficulty against enemies with less health meant that the agent was able to quickly recognise that attacking was the best action to earn rewards. The agent was able to benefit from this previous knowledge and optimised its actions for attacking faster than the baseline. The reward-based curriculum was able to improve the convergence of the agent by elevating its initial learning process, leading it to discover ways of earning rewards faster.

The agent trained on state-based curriculum for Defeat Roaches performed much worse during training in the earlier stages, but ultimately converged to a much stronger performance. By observing the state-based agent after training, we

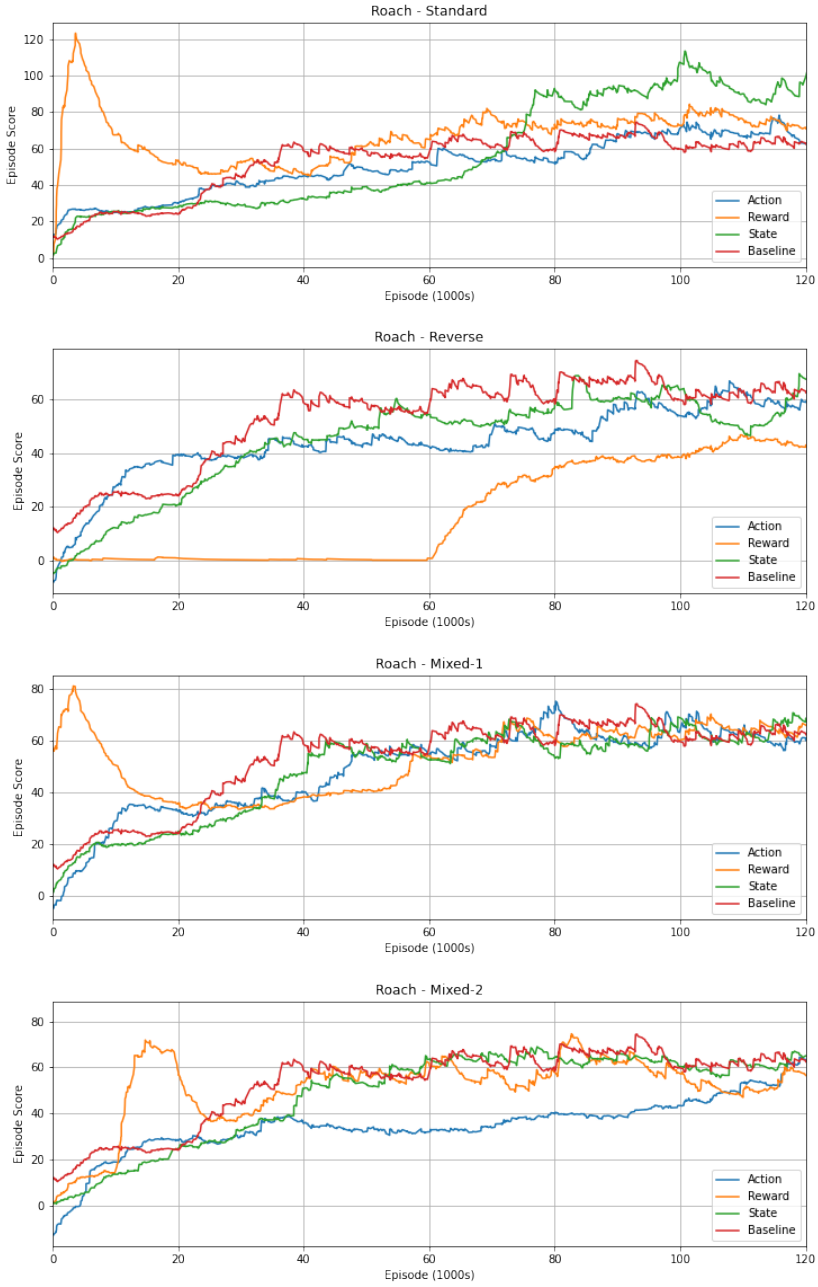

Fig. 2. Training plots for Defeat Roaches for all curricula and baseline agents.

realised that it learned the optimal strategy of focused attack commands on roaches one-by-one, unlike the baseline agent that learned the sub-optimal strategy. As a result, all marines were able to consistently attack the same target,

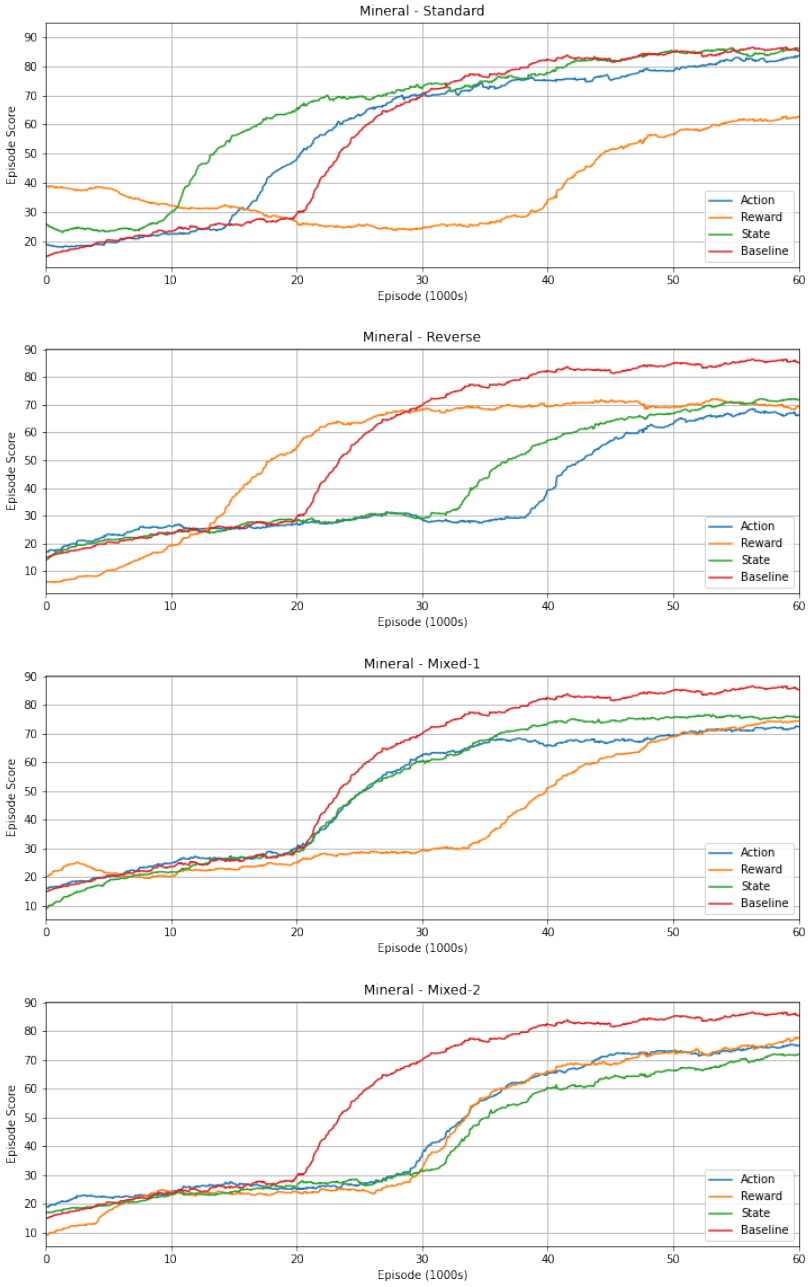

Fig. 3. Training plots for Collect Mineral Shards for all curricula and baseline agents.

minimising incoming enemy fire by killing enemies faster. In the early stages of the state-based curriculum, the agent was forced to attack enemies head-on due to map size limitations. When playing on a smaller map, the number of

redundant states was reduced and the small map size meant that the agent was unable to use sub-optimal strategies, such as moving to the corners of the map without being spotted by enemy roaches. This may have contributed to the worsened performance in the middle stages of training, as fighting an entire group of enemies head-on would be much more difficult than fighting a smaller group of enemies from corners of the map. However, this also led the agent to discover the optimal strategy as it repeatedly fought enemy roaches head-on, thus achieving a stronger final performance. The state-based curriculum was beneficial to the agent as it reduced the number of redundant states in the environment, thus avoiding local minimums and leading the agent to discover the optimal strategy faster.

4.1.2 Collect Mineral Shards. We observed some key differences between Collect Mineral Shards and Defeat Roaches. In contrast to Defeat Roaches, a reward-based curriculum was not helpful for the agent in the Collect Mineral Shards mini-game. In the reward-based curriculum, we increased the number of mineral shards on the map for the agent, reducing the sparseness of rewards, hoping that it would aid the learning process of the agent. However, as the results indicate, this was not the case. The agent trained on the reward-based curriculum failed to quickly draw correlations between moving the marines to the mineral shards and receiving a reward, despite the higher density of mineral shards. We theorise that there may have been too many mineral shards on the map, allowing the agent to receive high rewards from performing random actions. As a result, the reward-based curriculum with more mineral shards was harmful to the agent's training. This hypothesis can be tested by training the agent on a reversed reward-based curriculum with a decreased number of mineral shards.

Unlike Defeat Roaches, the action-based curriculum was able to offer improvements on the baseline for Collect Mineral Shards. In the action-based curriculum for Collect Mineral Shards, we also reduced the number of marines that the agent needed to control. However, in this mini-game, having fewer units to control was helpful to the learning process of the agent. By having fewer units under its control, the agent was able to draw correlations between its actions and receiving a reward efficiently. As there were fewer units on the map, it was easier for the agent to distinguish which of its actions led to rewards. As a result, an action-based curriculum that assigned tasks to the agent with fewer units to control was beneficial to the agent's overall learning process. Again, we can test this hypothesis by training the agent on a reversed action-based curriculum with an increased number of units under the agent's control.

For the state-based curricula, we observed similar results for Collect Mineral Shards to those of Defeat Roaches. The state-based curriculum for Collect Mineral Shards was able to accelerate the convergence of the agent compared to the baseline, although in this case, it did not improve on its final performance. The state-based curriculum was again beneficial to the learning process of the agent, as it omitted redundant states from the task. During the early training periods of the baseline agent, the agent spent much time moving around pointlessly as it failed to locate the last mineral shard with random movements. By having a reduced state space, the agent was more likely to retrieve the final mineral shard with random movements, allowing the game to spawn the next set of mineral shards to continue meaningful training. As a result, the agent gathered knowledge more efficiently in the state-based curriculum and thus converged much faster than the baseline. The state-based curriculum was able to consistently improve the agent's performance by removing redundant states in both Defeat Roaches and Collect Mineral Shards.

4.2 Curriculum Sequencing Analysis

In this paper, we proposed three alternative curriculum sequences to a standard easy-to-hard order and evaluated them against baseline RL agents in two different mini-games in SC2. In this section, we analyse the results of our curriculum sequence experiments in Defeat Roaches and Collect Mineral Shards.

4.2.1 Defeat Roaches. Training agents with reversed curricula provided no benefits for the agent's learning process in the Defeat Roaches mini-game, as demonstrated in Figure 2. All curriculum generation methods converged slower than the baseline. While the reversed action-based curriculum had slightly better performance at the beginning, it quickly fell behind the baseline as we updated the training tasks to the target task. Out of all three reversed curricula, reward-based performed the worst, having detrimental effects on the agent's learning process. By observing the reward-based agent's behaviour during training, we noted that the agent resorted to a passive strategy of hiding from enemies to minimise friendly unit loss. This was due to the agent consistently receiving negative scores from losing friendly units without defeating any roaches at the beginning of the training. As we increased the roaches' health for the reward-based curriculum, it was less likely to receive any rewards with random actions and thus more likely to end the training with a negative episode score. As a result, the agent learned to 'maximise' its score by avoiding all possibility of receiving a penalty, while at the same time, losing its ability to receive any positive rewards. It is only through random explorations that the agent was able to escape this local minimum later in training. For the Defeat Roaches task, a reversed curriculum offered no improvements for the agent and instead had adverse effects on its convergence.

Mixed 1 curriculum produced much closer results to the performance of the baseline. Figure 2 demonstrates that despite still converging slower than the baseline, action-based and state-based mixed curricula followed the baseline agent closely, with only reward-based mixed curricula showing significant slowdowns. While reward-based and action-based mixed curricula showed better performance at the beginning of the training, the agent was unable to utilise the knowledge learned from these curriculum tasks and failed to maintain its lead over the baseline agent after switching to the target task. Considering the overall convergence process of all four agents, it is clear that this mixed curriculum offered no benefits to the agent for the Defeat Roaches task.

Similarly, in the results for Mixed 2 curriculum (presented in Figure 2), no improvements over the baseline were evident when considering the overall convergence process. While the reward-based mixed curriculum followed the baseline very closely after switching to the target task, as indicated by the sudden drop in performance, it still failed to surpass the baseline. All other mixed curricula also showed adverse effects, especially the action-based mixed curriculum, as its performance staggered for the majority of training time. It is clear from the results presented in Figure 2 that training with mixed orders did not offer any benefits for the agent.

For the Defeat Roaches task, all curriculum sequences showed no improvements over the baseline and often had adverse effects. We can conclude that for Defeat Roaches, any curriculum sequence other than the easy-to-hard sequence offers no benefits to the learning process of the agent.

4.2.2 Collect Mineral Shards. In contrast to Defeat Roaches, where none of the reversed curricula exhibited improvements over the baseline, the reward-based reversed curriculum resulted in faster convergence, as can be seen in Figure 3. Having a decreased probability of receiving a reward with random actions did not adversely affect the agent's learning process; instead, it was quite beneficial. This result is consistent with our hypothesis in Section 4.1.2 that having too many mineral shards on the map might confuse the agent's ability to associate receiving rewards with collecting mineral shards. As benefits result from a reversed reward-based curriculum and adverse effects from a standard reward-based curriculum, we can confirm our hypothesis for Collect Mineral Shards. In addition, the action-based reversed curriculum had significantly worse performance than the baseline, again confirming our hypothesis (in Section 4.1.2) that decreasing the number of units under the agent's control helps to associate specific actions with earning rewards correctly. From the results in Collect Mineral Shards, it is clear that a reversed curriculum might also aid the convergence of an agent,

depending on the design of the curriculum tasks. Although, at the same time, it exhibited adverse effects on the agent's final performance.

The two mixed curriculum sequences in Collect Mineral Shards produced very similar results as in Defeat Roaches, in that they failed to offer any improvements over the baseline. As Figure 3 shows, the two mixed curriculum sequences - with all three generation methods - performed worse than the baseline. In various cases, they not only converged slower, but also ended with weaker performance. For mixed curricula, the agent was unable to learn any useful knowledge that was transferable to the target task. As a result, spending time on curriculum tasks only slowed down their overall convergence speed. It is clear from the results in Collect Mineral Shards that curricula with mixed orders do not aid the learning process of RL agents.

5 CONCLUSIONS

In this paper, we proposed three different methods for manually generating curriculum tasks for RL problems formulated as an MDP and evaluated them through experiments in RTS game StarCraft II. We presented reward-based, action-based, and state-based methods for creating curriculum tasks for CL, and measured their performance via experiments. We demonstrated the effectiveness of our proposed methods under different tasks and provided an in-depth analysis of our results. We showed that our proposed methods can accelerate the learning process of a deep RL agent and can even improve its final performance. Out of all of the curricula that we investigated, only state-based curricula produced consistent results in both performance and application. Other curriculum generation methods, such as reward-based or action-based curricula, did not provide consistent improvements and might require further adjustments in order for them to be effective.

We also experimented with different curriculum sequences and verified the effectiveness of various training orders. Our results suggest that it is only beneficial to train RL agents with curricula in a meaningful order, namely either from easy-to-hard or reversed. Our results demonstrate that reversed curricula can be beneficial to an RL agent's convergence process under certain circumstances. However, mixed curriculum training with non-conventional orders is unlikely to support learning, regardless of the curriculum task. We conclude that only well-designed curriculum tasks that are sequenced in meaningful orders - either an easy-to-hard or reversed order - can accelerate the rate of convergence or elevate the final performance of an RL agent.

CL is a useful optimisation technique that can be easily applied with good results. However, without proven guidelines for manually designing curricula, most researchers create curricula intuitively depending on the problem that they are trying to solve [18]. Based on our results, we can conclude that modifying the state space and training the agent from smaller to larger state spaces is an effective approach for CL applied to games modelled as an MDP. We have presented valid methods for manually designing curricula rather than relying on intuition. However, as we have conducted our experiments on the game Starcraft II and no other deep RL domains, such as robotics or marketing, our results are restricted to certain scenarios in Starcraft II.

5.1 Future Work

The work in this paper not only serves as a guideline for handcrafting curricula, but also as a stepping stone towards the more effective automation of CL. The current automated methods for CL in RL involve various shortcomings in various areas. By gaining a better understanding of designing curricula, we could utilise this knowledge to create less restrictive automated methods. Another possible future direction in this field is to explore generalised curriculum design methods for areas other than RL. CL has also been successfully applied to neural networks, but a generalised

method for applying it has yet to be proposed. This would be an extremely challenging task, as neural networks cover a much wider range of problems and applications, from simple time-series predictions to complex, human-like speech and image recognition.

In addition to the experiments conducted in this paper, we would also like to investigate the application of our methods to different games and other RL problems. It would be useful to determine whether the performance of our proposed methods hold in other RL problems or if they are limited to a specific, but as yet unknown, set of tasks. By gaining a better understanding of the limitations of our work, we could expand our current research or explore other ways to more effectively apply CL to RL, addressing the long training time problem of RL agents.

REFERENCES

[1] Marcus Adamsson. 2018. *Curriculum learning for increasing the performance of a reinforcement learning agent in a static first-person shooter game.* Ph. D. Dissertation. Kth Royal Institute of Technology, Stockholm, Sweden.

[2] Khan Adil, Feng Jiang, Shaohui Liu, Worku Jifara, Zhihong Tian, and Yunsheng Fu. 2017. State-of-the-Art and Open Challenges in RTS Game-AI and Starcraft. *International Journal of Advanced Computer Science and Applications* 8, 12 (2017). https://doi.org/10.14569/IJACSA.2017.081203

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada) *(ICML '09)*. Association for Computing Machinery, New York, NY, USA, 41–48. https://doi.org/10.1145/1553374.1553380

[4] Michael Cerny Green, Benjamin Sergent, Pushyami Shandilya, and Vibhor Kumar. 2019. Evolutionarily-Curated Curriculum Learning for Deep Reinforcement Learning Agents. *arXiv preprint arXiv:1901.05431* (2019).

[5] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. 2018. Automatic Goal Generation for Reinforcement Learning Agents. In *International Conference on Machine Learning*. 1515–1528.

[6] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse Curriculum Generation for Reinforcement Learning. In *Proceedings of the 1st Annual Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 78)*, Sergey Levine, Vincent Vanhoucke, and Ken Goldberg (Eds.). PMLR, 482–495.

[7] Niels Justesen and Sebastian Risi. 2017. Continual Online Evolutionary Planning for In-Game Build Order Adaptation in StarCraft. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Berlin, Germany) *(GECCO '17)*. Association for Computing Machinery, New York, NY, USA, 187–194. https://doi.org/10.1145/3071178.3071210

[8] N. Justesen and S. Risi. 2017. Learning macromanagement in starcraft from replays using deep learning. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. 162–169. https://doi.org/10.1109/CIG.2017.8080430

[9] Kai A. Krueger and Peter Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition* 110, 3 (2009), 380 – 394. https://doi.org/10.1016/j.cognition.2008.11.014

[10] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. 1928–1937.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[12] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. 2016. Source Task Creation for Curriculum Learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems* (Singapore, Singapore) *(AAMAS '16)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 566–574.

[13] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. 2017. Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 2536–2542.

[14] S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. 2013. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games* 5, 4 (12 2013), 293–311. https://doi.org/10.1109/TCIAIG.2013.2286295

[15] Zhen-Jia Pang, Ruo-Ze Liu, Zhou-Yu Meng, Yi Zhang, Yang Yu, and Tong Lu. 2019. On Reinforcement Learning for Full-Length Game of StarCraft. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 4691–4698. https://doi.org/10.1609/aaai.v33i01.33014691

[16] Pekaalto. 2017. *sc2aibot.* https://github.com/pekaalto/sc2aibot

[17] Bei Peng, James MacGlashan, Robert Loftin, Michael L Littman, David L Roberts, and Matthew E Taylor. 2016. An Empirical Study of Non-Expert Curriculum Design for Machine Learners. *Proceedings of the Interactive Machine Learning Workshop (at IJCAI 2016)* (2016).

[18] K. Shao, Y. Zhu, and D. Zhao. 2019. StarCraft Micromanagement With Reinforcement Learning and Curriculum Transfer Learning. *IEEE Transactions on Emerging Topics in Computational Intelligence* 3, 1 (2 2019), 73–84. https://doi.org/10.1109/TETCI.2018.2823329

[19] Felipe Leno Da Silva and Anna Helena Reali Costa. 2018. Object-Oriented Curriculum Generation for Reinforcement Learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1026–1034.

[20] 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. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[21] Jivko Sinapov, Sanmit Narvekar, Matteo Leonetti, and Peter Stone. 2015. Learning Inter-Task Transferability in the Absence of Target Task Samples. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (Istanbul, Turkey) *(AAMAS '15)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 725–733.

[22] Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. 2017. Automatic Curriculum Graph Generation for Reinforcement Learning Agents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (San Francisco, California, USA) *(AAAI'17)*. AAAI Press, 2590–2596.

[23] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2094–2100.

[24] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. 2017. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* (2017).

[25] Ravi Kumar Vuddagiri, Hari Krishna Vydana, and Anil Kumar Vuppala. 2018. Curriculum learning based approach for noise robust language identification using DNN with attention. *Expert Systems with Applications* 110 (2018), 290 – 297. https://doi.org/10.1016/j.eswa.2018.06.004

[26] Edward Wong. 2018. *Investigation on the Effects of Curriculum Learning through a Simulation Study.* Master's thesis. UC San Diego.

[27] Yuxin Wu and Yuandong Tian. 2017. Training Agent for First-Person Shooter Game with Actor-Critic Curriculum Learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=Hk3mPK5gg

[28] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. 2019. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*.