Designing Curriculum for Deep Reinforcement Learning in StarCraft II

Daniel Hao, Penny Sweetser, and Matthew Aitchison

The Australian National University, Canberra ACT 2601, Australia

Abstract. Reinforcement learning (RL) 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. Curricula are usually designed manually, due to limitations involved with automating curricula generation. However, as there is 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 RL in real-time strategy game StarCraft II. We propose four generalised methods of manually creating curricula and verify their effectiveness through experiments. Our results show that all four of our proposed methods can improve a RL agent's learning process when used correctly. We demonstrate that using subtasks, or modifying the state space of the tasks, is the most effective way to create training samples for StarCraft II. We found that utilising subtasks during training consistently accelerated the learning process of the agent and improved the agent's final performance.

Keywords: Game AI · Reinforcement Learning · Real-Time Strategy Games · StarCraft II · Curriculum Learning.

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 cumulative 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 and Go. RL algorithms such as Deep Q-learning (DQN) [11] and Advantage Actor-Critic (A2C) [10] have been widely deployed and have 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 [20], which can take up to weeks to complete depending on the performance of the hardware. Due to the incredibly large environment and sparse reward nature of such tasks, 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 [14] 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 [16] 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. In this paper, we focus on the first problem. 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 [19].

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 [5, 4]. 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 [18] 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 [14, 17], manually creating curriculum tasks guided by a rough understanding of the intended learning outcome of the agent.

In this paper, we propose three intuitive domain-independent measures for creating curricula: reward-based, state-based, and action-based. We design and evaluate curricula with each difficulty measure to discover the most effective design methodology. We also investigate using domain-dependent subtasks as curricula and draw comparisons between domain-dependent and domain-independent curricula. Domain-dependent subtasks may not always be a viable option, unlike domain-independent measures. As a result, subtasks are considered separately to other measures. 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 for practical implementations of CL by evaluating the performance of the proposed handcrafted curriculum structures.

3 Method

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

3.1 StarCraft II

Starcraft II (SC2) is a competitive RTS game that is mostly played between two players (1v1). 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 [20] and the branching factor for two consecutive states is estimated to be between $b \in [10^{50}, 10^{200}]$ even when only considering units [13]. 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 [8], micromanagement [17], and macro-management [7].

In 2017, DeepMind and Blizzard collaboratively developed and released a library named PySC2, exposing Blizzard's SC2 machine learning API as a Python environment [20]. 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. 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.

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. In Defeat Roaches, the agent controls a group of nine marines to defeat four enemy roaches. When all four enemy roaches are defeated, a new group of four enemy roaches is spawned. The agent is rewarded with five additional marines and its 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 preselected for the agent when spawned. The camera is locked to the centre of the screen with no fog of war. The game state is reset when all marines are killed or after 120 seconds. The agent earns rewards when a roach is killed and 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.

In Collect Mineral Shards, the agent controls two marines to collect as many mineral shards as possible (by stepping on them). The marines are spawned in random locations and are not preselected. The map starts with 20 mineral shards, with another 20 being spawned after all shards have been collected. Each shard is spawned at least two units away from the marines. The camera is locked to the centre of the map with no fog of war. The game state is reset upon reaching the time limit of 120s. Every time the agent collects a mineral shard, it receives a reward, and there are no explicit penalties for losing marine units from friendly fire. 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.

3.2 Curricula 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). The domain-dependent method that we propose is called subtask curriculum generation. We define a subtask as a subset of the full task, where we extract certain aspects and challenges of the full task to create a simpler sub-problem. We can design a subtask that only trains the agent to tackle and learn a single sub-problem. Subtask creation is highly dependent on the original task and may not exist for certain problems.

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. 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 Defeat Roaches, we increased the probability of receiving a reward from random actions by lowering the health of enemy units (Easy=90; Medium=120; Target=145). We could achieve similar effects by increasing the attack damage of marines. However, this would also increase the chance of killing friendly units with friendly fire. 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 [17]. For Collect Mineral Shards, we modified 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 (Easy=40; Medium=30; Target=20).

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 Defeat Roaches, we decreased the map size to lower the difficulty of the tasks (Easy=13x12; Medium=18x14; Target=22x16). For Collect Mineral Shards, 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 (Difficulty=map-size(shards); Easy=11x8(5); Medium=16x12(11); Target=22x16(20)). 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.

Table 1. Action-based settings for Defeat Roaches.

	Penalty	Marines	Backup	Health	Damage
Easy	-4	2	1	190	25
Medium	-3	5	3	80	10
Target	-1	9	5	45	6

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 the agent controls, we effectively reduce the size of the action space and the difficulty of the task. For Defeat Roaches, we also needed to modify the strength of individual units, so that the raw difficulty did not increase as we decreased the number of friendly units (see Table 1). We also made 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 controls to reduce the action difficulty of the task. We considered whether this might affect the maximum performance of the agent, as it had less collecting power. However, we found that the update frequency of the algorithm limited the frequency of performing an action and we observed that the agent was never able to simultaneously control two marines to collect mineral shards with different paths. As a result, we only modified the number of marines and the initial selection of units (Difficulty=marines(selected); Easy=1(1); Medium=2(1); Target=2(0)).

Subtask Curricula Subtask curriculum generation is quite different from the previous methods in that it does not rely on the definition of an MDP. A subtask is a sub-problem of the target task, so we needed to create a new curriculum task that was tailored to the specific needs of the target task rather than modifying the original task. The creation of curriculum subtasks is highly dependent on the original target problem. Thus, we consider this method to be domain-dependent, as there is no formally generalised method for creating a subtask.

For the Defeat Roaches task, we created a new mini-game called 'Destroy Target'. The purpose of this mini-game was to encourage the agent to learn to focus fire on enemies and reduce redundant actions. In Destroy Target, the agent needs to control nine marines to destroy an enemy structure (a Pylon). In a similar setup to Defeat Roaches, the agent received a reward (10) for destroying the enemy structure and received penalties (-1) for the death of friendly units. All marines were also initially selected for the agent when spawned. The camera was locked to the centre of the screen with no fog of war so that no camera

movements were required. The game state was reset upon reaching the time limit (120s). At any given time, there was only one Pylon on the map, and upon its destruction, a new enemy Pylon was created. However, in this mini-game, the agent did not receive new marine units upon destroying the enemy Pylon. We modified the properties of the Pylon so that it had no shield and increased health (10) and health regeneration (100). We also disabled auto-attack for marines. Since the Pylon was able to regenerate its health, if the agent decided to cease attacking or perform other redundant actions, the Pylon would quickly return to full health, rendering the agent's previous efforts useless.

For the Collect Mineral Shards mini-game, we used the 'Move to Beacon' mini-game created by Blizzard [20] as a subtask, as it is very similar to Collect Mineral Shards. It requires the agent to control a single marine to move to an indicated area (the beacon). In contrast to the Collect Mineral Shards mini-game, there is only one target beacon on the map at any given time. Upon reaching the beacon, the agent receives a reward and a new beacon is randomly spawned on the map. The game state is also reset upon reaching the time limit (120s). We selected the Move to Beacon mini-game tailored to the 'collecting' aspect of Collect Mineral Shards. We wanted to encourage the agent to draw connections between receiving a reward and moving to indicated areas. Collect Mineral Shards can be understood as an extension of Move to Beacon with multiple target areas and more units under command.

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 Pekalto [15]. The agent is a close replica of the FullyConv agent described in the original publication of PySC2 [20]. 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 5x5 and 3x3, 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 1x1 convolution over the state representation with a single output channel. Pekalto [15] modified the original FullyConv agent structure to simplify the agent network and accelerate training, including discarding non-spatial vectors entirely and using A2C instead of A3C [10], which is a non-asynchronous implementation of the algorithm.

Table 2. Stable score and total episodes to converge.

	Defeat Roaches		Collect Minerals	
Agent	Score	Episodes	Score	Episodes
Baseline	60	60k	85	40k
Action	60	100k	85	50k
Reward	70	70k	65	60k
State	100	120k	85	50k
Subtask	110	120k	100	35k

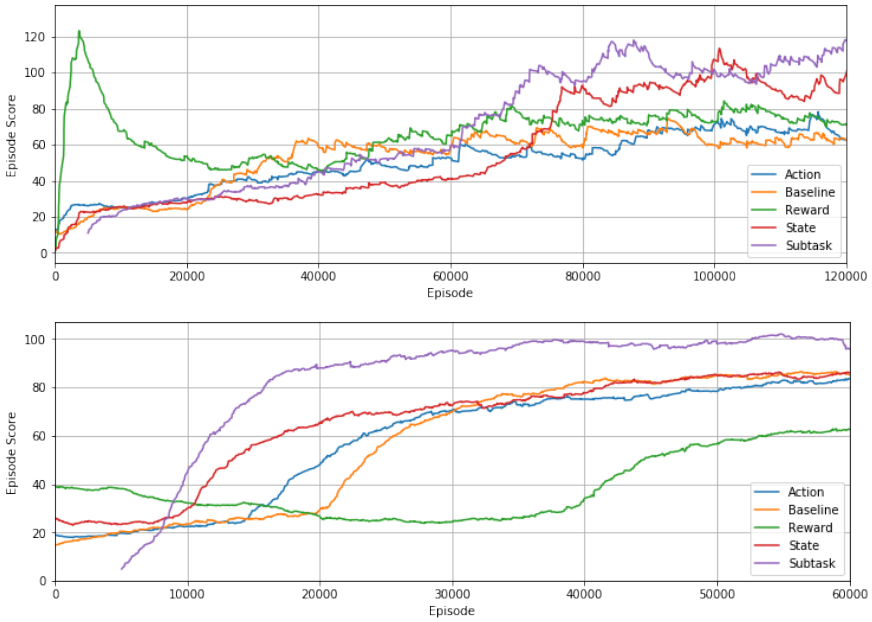

Fig. 1. Training plots for Defeat Roaches (top) and Collect Mineral Shards (bottom).

4 Results

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. We created a baseline for comparison by training an agent directly on each mini-game with no curricula. Figure 1 shows a comparison of all curricula and the baseline agents, with final score at the end of an episode plotted against the number of episodes trained. The training graph of the subtask curriculum agent begins at five thousand episodes in Figure 1 to exclude the Move to Beacon subtask, as it is an entirely different problem with different reward structures and not comparable to other episode scores. Table 2 shows the stable score and number of episodes to converge for each curriculum.

5 Discussion

In this paper, we proposed four different methods for manually creating curriculum tasks for RL and evaluated them against baseline RL agents in two different mini-games in SC2. 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 Defeat Roaches, reducing the number of units that the agent controlled (and thus reducing the action space) did not help to accelerate the training process. We initially predicted that an action-based curriculum would help the agent by lowering 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 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. For Collect Mineral Shards, the action-based curriculum offered improvements on the baseline. We reduced the number of marines that the agent needed to control in both mini-games. However, controlling fewer units in Collect Mineral Shards allowed the agent to draw correlations between its actions and receiving a reward efficiently. With 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.

In the reward-based curriculum for Defeat Roaches, minor improvements were evident when compared to the baseline. We observed that both agents 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. In contrast, a reward-based curriculum was not helpful for the agent in Collect Mineral Shards. For this curriculum, we increased the number of mineral shards on the map to reduce the sparseness of rewards. However, 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 minerals. We theorise that the large number of mineral shards on the map allowed the agent to receive rewards for performing random actions. This hypothesis could be tested by training the agent on a reversed reward-based curriculum with a decreased number of mineral shards.

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. We observed that the state-based agent learned the optimal strategy of focused attack commands on roaches one-by-one, unlike the baseline agent. As a result, all marines were able to consistently attack the same target, 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 might 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. We observed similar results for Collect Mineral Shards. The state-based curriculum was 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, it spent much time trying 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.

In the subtask curriculum for Defeat Roaches, the agent was able to converge to a stronger performance when compared to the baseline. We observed that the subtask agent used the same strategy as the state-based agent of efficiently focusing fire on enemy units. The subtask agent was first trained on the Destroy Target mini-game, which encouraged the agent to perform focused attacks on the target structure. From the results, we can conclude that the subtask curriculum helped the agent master a sub-problem of the target task at a faster rate and accelerated the final learning process for the target task. The subtask curriculum performed the best among all curricula. The results were similar for Collect Mineral Shards, where the agent achieved the best performance among all curricula and converged faster than the baseline. Although the subtask curricula utilised very different subtasks for the two mini-games, they achieved the same goal of helping the agent to master a sub-problem at a much faster rate. The subtask curriculum first trained the agent on the Move to Beacon mini-game, which can be viewed as a simplified version of Collect Mineral Shards with only one 'mineral shard' (the beacon) on the map. From the results, we can conclude that the agent was able to effectively utilise the knowledge that it had learned from the subtask and successfully apply it to the target task. By comparing the performances of all curricula against the baseline, it is evident

that the subtask curriculum agent again outperformed all other agents, followed by the state-based curriculum agent.

6 Conclusion

CL is a useful optimisation technique that can be easily applied with good results. However, without proven guidelines for manually designing curricula, most CL users create curricula intuitively depending on the problem that they are trying to solve [17]. In this paper, we presented curriculum design methodologies for manually designing curricula to improve an agent's learning process. We proposed four effective methods for generating curriculum tasks for RL problems and evaluated them through experiments in RTS game StarCraft II. We demonstrated the effectiveness of our proposed methods under different tasks, provided an in-depth analysis of our results, and verified different ways that CL can aid an RL agent's learning process. Out of all of the curricula that we investigated, only subtask and 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. To our knowledge, at the time of writing, we are the first to experiment and present generalised methods for handcrafting curricula in RL. Based on our results, we recommend that future applications of CL in Deep RL applied to games modelled as an MDP use subtasks as their curriculum tasks. In cases where subtasks are not available or cannot be constructed, modifying the state space and training the agent from smaller to larger state spaces is also an effective option.

References

1. Adamsson, M.: Curriculum learning for increasing the performance of a reinforcement learning agent in a static first-person shooter game. Ph.D. thesis, Kth Royal Institute of Technology, Stockholm, Sweden (10 2018)
2. Adil, K., Jiang, F., Liu, S., Jifara, W., Tian, Z., Fu, Y.: 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. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. p. 41–48. ICML '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1553374.1553380
4. Cerny Green, M., Sergent, B., Shandilya, P., Kumar, V.: Evolutionarily-curated curriculum learning for deep reinforcement learning agents. arXiv preprint arXiv:1901.05431 (2019)
5. Florensa, C., Held, D., Geng, X., Abbeel, P.: Automatic goal generation for reinforcement learning agents. In: International Conference on Machine Learning. pp. 1515–1528 (2018)

6. Florensa, C., Held, D., Wulfmeier, M., Zhang, M., Abbeel, P.: Reverse curriculum generation for reinforcement learning. In: Levine, S., Vanhoucke, V., Goldberg, K. (eds.) Proceedings of the 1st Annual Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 78, pp. 482–495. PMLR (11 2017)
7. Justesen, N., Risi, S.: Learning macromanagement in starcraft from replays using deep learning. In: 2017 IEEE Conference on Computational Intelligence and Games (CIG). pp. 162–169 (8 2017). https://doi.org/10.1109/CIG.2017.8080430
8. Justesen, N., Risi, S.: Continual online evolutionary planning for in-game build order adaptation in starcraft. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 187–194. GECCO '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3071178.3071210
9. Krueger, K.A., Dayan, P.: Flexible shaping: How learning in small steps helps. Cognition **110**(3), 380 – 394 (2009). https://doi.org/https://doi.org/10.1016/j.cognition.2008.11.014
10. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. pp. 1928–1937 (2016)
11. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
12. Narvekar, S., Sinapov, J., Leonetti, M., Stone, P.: Source task creation for curriculum learning. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems. p. 566–574. AAMAS '16, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2016)
13. Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., Preuss, M.: A survey of real-time strategy game ai research and competition in starcraft. IEEE Transactions on Computational Intelligence and AI in Games **5**(4), 293–311 (12 2013). https://doi.org/10.1109/TCIAIG.2013.2286295
14. Pang, Z.J., Liu, R.Z., Meng, Z.Y., Zhang, Y., Yu, Y., Lu, T.: On reinforcement learning for full-length game of starcraft. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4691–4698 (2019)
15. Pekaalto: sc2aibot (2017), https://github.com/pekaalto/sc2aibot
16. Peng, B., MacGlashan, J., Loftin, R., Littman, M.L., Roberts, D.L., Taylor, M.E.: An empirical study of non-expert curriculum design for machine learners. Proceedings of the Interactive Machine Learning Workshop (at IJCAI 2016) (2016)
17. Shao, K., Zhu, Y., Zhao, D.: Starcraft micromanagement with reinforcement learning and curriculum transfer learning. IEEE Transactions on Emerging Topics in Computational Intelligence **3**(1), 73–84 (2 2019). https://doi.org/10.1109/TETCI.2018.2823329
18. Silva, F.L.D., Costa, A.H.R.: Object-oriented curriculum generation for reinforcement learning. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. p. 1026–1034. AAMAS '18, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2018)
19. Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., Stone, P.: Automatic curriculum graph generation for reinforcement learning agents. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. p. 2590–2596. AAAI'17, AAAI Press (2017)
20. Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A.S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al.: Starcraft ii: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782 (2017)