Shaping the Behavior of Reinforcement Learning Agents

George Sidiropoulos
Athena-Research and Innovation Center in Information, Communication and Knowledge Technologies
Xanthi, Greece
georsidi@athenarc.gr

Chairi Kiourt
Athena-Research and Innovation Center in Information, Communication and Knowledge Technologies
Xanthi, Greece
chairiq@athenarc.gr

Vasileios Sevetlidis
Athena-Research and Innovation Center in Information, Communication and Knowledge Technologies
Xanthi, Greece
vasiseve@athenarc.gr

George Pavlidis
Athena-Research and Innovation Center in Information, Communication and Knowledge Technologies
Xanthi, Greece
gpavlid@athenarc.gr

ABSTRACT

With the advent of machine learning and agent-based approaches, behavior-shaping in environments composed of several autonomous entities has become a popular and active research field for the development of unique realistic behaviors. Realistic simulations have been particularly studied in the fields of crowd management, swarm behavior analysis and civilization simulation. In this study, we present a new dynamic rewarding approach for shaping the behavior of reinforcement learning agents in mixed (cooperative and competitive) multi-agent environments. The evaluation of the proposed rewarding approach is tested in a developed 3D environment of two groups of ancient Greek warriors fighting inside an octagonal arena, testing different agent behaviors in various scenarios. Interestingly, the results reveal that the trained agents' behaviors vary based on the situations and the constraints of the environment, resembling realistic behavior variations.

CCS CONCEPTS

• **Computing methodologies → Multi-agent reinforcement learning**.

KEYWORDS

reinforcement learning, behavior-shaping, self-playing agents, multi-agent environments

ACM Reference Format:
George Sidiropoulos, Chairi Kiourt, Vasileios Sevetlidis, and George Pavlidis. 2021. Shaping the Behavior of Reinforcement Learning Agents. In *25th Pan-Hellenic Conference on Informatics (PCI 2021), November 26–28, 2021, Volos, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3503823.3503905

1 INTRODUCTION

Reinforcement Learning (RL) environments typically consist of a single agent that aims to learn actions that yield the highest rewards, according to a specific reward policy [17]. This area of Machine Learning (ML) has proven to be very effective in various scenarios, even against human competition [6, 14, 18]. Very common applications of RL are games, in which the agent learns to act within the rules of a specific environment by competing with other agents, in order to gain insight into how to face progressively more challenging opponents. Other applications of RL focus on training agents that develop realistic behaviors within specific environments [1, 16]. Behavior-shaping and analysis aims to influence and study the behaviors of autonomous agents in various situations and scenarios. Nowadays, agent behavior-shaping is a very active field of research and development, especially in the case of multi-agent environments [9, 10].

A common usage of RL is in multi-agent (MA) RL (MARL) scenarios (cooperative, competitive or mixed scenarios). Multi-agent scenarios were studied in the past, but they have recently reemerged with the advent of new RL techniques [20] and powerful computational resources. MARL scenarios are typically more complicated than single-agent scenarios, mainly due to the fact that the agent has to learn complex behaviors within more dynamic environments. In addition to that, the environment is changing dynamically (non-stationary), with its level of complexity depending on the number of the agents [9, 10].

In these directions, in the last few years, important attention has been given to the analysis of the generated behaviors by the trained agents. The agents learn to generate the required behavior usually through either a reward function (reward-shaping) [2, 13], where the agent is rewarded positively for the correct behaviors and/or negatively otherwise, or imitation learning [19], where in its simplest form the agent's model is trained on data consisting of the actions/behaviors to be generated (behavioral cloning). On top of these kinds of behavior shaping processes, further analyzes of the behaviors have been conducted, focusing on the collaboration of agents in collaborative MA environments [3, 8] and even generating human-like behaviors [4].

In this work, we focus on the application of MARL agents in battle game scenarios, where the agents learn to cooperate and to fight as a group (team) against other groups. The ultimate goal of the agent's learning process is to act autonomously, performing realistic behaviors in various scenarios, in order to prevail in the context of the game. For this purpose, we present a complex cooperative *and* competitive multi-agent environment consisted of two groups of agents. The ultimate goal of this work in progress, is to experiment on the process of behavior-shaping in autonomous agents, implementing a dynamic rewarding approach. After the training, in the inference phase, the agents are evaluated in various scenarios, battling against varying numbers of opponents and teammates, generating unique behaviors and outcomes, making each battle unique in itself.

2 RELATED WORKS

In the last decade, there have been several studies focusing on agent's behavior-shaping, through the application of various training algorithms, with most of them exploiting reinforcement learning approaches. Through reward-shaping, Matheus R. F. Mendonça et al. [13] developed a fighting game called Boxer and aimed to train an RL agent for human-like behaviors through three different reward functions. The reward functions focused on the victory, hitting the enemy and following a recommended strategy. Moreover, they compared these approaches to the usage of an ANN trained on recorded sessions. With a more general aim, A. Barreto et al. [2] proposed the policy improvement and policy evaluation operations, through which they shaped the agent's behaviors by using a modulated preference vector, defining the preferred actions the agent should consider. E. Langlois and T. Everitt [10] used a Modified-Action Markov Decision Process to produce asymptotic behaviors that differ from the agent's policy, with agents both ignoring modifications and others avoiding those that decrease their reward.

Other studies follow different approaches for behavior modeling, such as different model architecture or training process and parameters. For example, S. Loiacono et al. [11] analyzed overtaking behaviors of Non Playable Characters (NPCs) in a racing car simulator, such as fast opponents or in tight bend situations, by applying Q-learning. C. Kiourt and D. Kalles [8] analyzed the behaviors of agents through opponent-learning, where differently trained agents (Good, Moderate, Bad) chose their opponents depending on their characteristics, with each category of agents generating different behaviors depending on the complexity of the environment.

With regard to the aforementioned related works (and many other), and focusing on the development of more complex environments, our study has incorporated the following key elements in the design of the experiments:

(1) Design a complex 3D battle environment as a multi-agent system in terms of action-state space exploration.
(2) Train an agent in a cooperative approach, as being a part of a group of agents focusing on the same goal, with the aim of learning cooperative behaviors.
(3) Evaluation of the trained agent in the context of different battle scenarios.
(4) Create various agent behaviors through the application of action-based dynamic rewarding approaches.

3 METHODOLOGY

In this section we present the adopted methodology, algorithms as well as the proposed behavior shaping approach.

3.1 Agent setup

The setup of the agents' learning algorithms/parameters, the actions, the rewarding policy, as well as the state space representation, are challenging in RL experiments. In our case, where we focused on shaping the agent's behaviors, the adopted setup is presented in the following paragraphs.

3.1.1 *Character setup:*
The character's attributes consist of the following key elements:

(1) Health Points (HP): The life points of the agent with a maximum (mHP) of 200 HP
(2) Attack Damage (AD): $light = 20$ and $heavy = 2 * light$ HP
(3) Defence Capacity: Takes $0.5 * AD$ of an incoming attack

The values of the above attributes were kept simple without focusing our attention on the balancing of the values so as to make the AD or Defence Capacity more "fair" towards the player/agents. As the study focused on shaping the behaviors, the values were chosen so that they made sense regarding their functionality, e.g. a heavy attack does more damage than a light attack (without thinking *how much* heavier it is).

3.1.2 *Action space:*
The agent's action space includes the following four action branches:

(1) Horizontal movement: $h\epsilon\{-1, 0, 1\}$, with -1 corresponding to left and 1 to right.
(2) Vertical movement: $v\epsilon\{-1, 0, 1\}$, with -1 corresponding to backward and 1 to forward.
(3) Attack (melee attack): $a\epsilon\{0, 1, 2\}$, with 1 corresponding to light attack (will cause light damage) and 2 to heavy attack (will cause heavy damage).
(4) Defence: $d\epsilon\{0, 1\}$, with 1 corresponding to defence.

In all cases, the value of 0 corresponds to no action. Moreover, a heavy attack, although it is expected to inflict heavy damage, lasts twice as longer compared to the light attack, making the attacker more susceptible to counter-attacks. Moreover, it should be noted that for an attack to hit, the enemy has to be hit by the spear during the attack animation. In both attack type cases, the agent has to be at least 1.25m close to the enemy for the spear to reach them. On the other hand, to defend an attack, the agent simply has to be on a defensive stance (holding the shield in front of them, covering their body), defending attacks from all sides and taking half the damage from the opponent's attacks. Additionally, the agent can take actions from all the above branches, but only one action will be performed (except for the movement, where the agent can move horizontally *and* vertically at the same time). For example, if the agent chooses to attack *and* defend, the defence will override the attack action, if the agent is moving *and* attacks or defends, the movement action will be blocked (movement will stop to perform the other action).

3.1.3 *Observations:*
A single agent collects 135 observations (state-space representation) in real-time, specifically:

(1) 80 of the 135 observations are calculated using 20 rays of 20m length, evenly spread around the agent. Each ray detects objects that belong to two object layers, differentiating between allies and enemies, returning as a value the normalized distance it travelled from the origin (the agent's position),
(2) 5 of the observations are the agent's status and include:
  (a) Health percentage $H \epsilon [0, 1]$
  (b) If the agent attacked during the last action it took [True, False]
  (c) If the agent defended during the last action it took [True, False]
  (d) Its normalized x and z position inside the arena
(3) The remaining 50 observations contain information about the enemy team's agents, with 5 observations for each one, up to 10 agents. For each agent, the following 5 observations are collected:
  (a) Enemy's health percentage $H \epsilon [0, 1]$
  (b) If the enemy is attacking [True, False]
  (c) If the enemy is defending [True, False]
  (d) The normalized x and z distance between the agent and the enemy

It should be noted that in case there are less than 10 opponents, the agent fills the remaining values with zeros (zero padding), for the total 50 expected observations. This helps the agent with receiving observations for various sizes of opponent groups, as well as to test its generalization in non-stable environments. Moreover, it is obvious that the agents do not differentiate between light and heavy attacks, due to the more general focus of the study.

3.1.4 Rewards: A time penalty of $-1/MaxSteps$ is applied for every action taken, forcing the agents to explore the state-action space with an inclination towards fast wins. When an agent hits an opponent, it receives a reward of +0.1 and when it manages to defend an enemy attack it receives a reward of +0.05. This rewarding approach produces more active agents, by pushing them to attack their opponents, while also trying to defend their attacks, disfavouring, at the same time, constant defensive strategies. In addition, by exploiting a multi-agent credit assignment algorithm, the cooperative behaviour of the agents is boosted. Regarding group rewards, a −1 indicates loss of a battle, a +1 indicates a win and 0 indicates a draw, which was awarded at the end of an episode. The result is regarded a draw when the episode has ended after $MaxSteps = 15,000$ steps, with at least one warrior surviving in both groups. An agent is considered out-of-battle ("dead") when its HP have reached 0 and so, a group wins the battle when *all* the opponent's agents are out-of-battle. The reasoning behind this credit assignment approach is so that the agents of a group cooperate to receive the positive group reward, despite their individual status, even when some agents of a group are expected to die during a battle. It should be noted that all of the above rewards were normalized.

3.2 Behavior-shaping

We tested three different behavior-shaping approaches based on the rewarding of an agent. This included the comparison of the training outcomes in each Test-Case (TC), as well as the monitoring and analysis of the behavior-shaping of the agents in the arena. In all three approaches, the agents received the same observations and rewards mentioned in previous paragraphs, with an additional reward depending on the approach (which focuses on the behavior-shaping of the agents), calculated as follows:

TC-1 (center): Distance of the agent from the center of the arena; this gives the agent more space to act more freely and accurately, "*precise behavior*".

TC-2 (midpoint): Distance of the agent from the midpoint of the enemy group; this pushes the agent directly towards the opponents, "*selfish behavior*".

TC-3 (both): Distance of the agent from the midpoint between the arena and the agent's closest opponent; this pushes the agent directly towards an opponent by trying to have more space for more accurate actions, "*wiser behavior*".

The distance reward was calculated using 1, taking into account only the normalized x and z axes:

$$dist = \begin{cases} \dfrac{(d_i - d_c) \cdot \frac{d_c}{d_i}}{MaxSteps}, & \text{if } d_c > d_i \\ \dfrac{(d_i - d_c) \cdot (1 - \frac{d_c}{d_i})}{MaxSteps}, & \text{if } d_c < d_i \end{cases} \qquad (1)$$

where $MaxSteps = 15,000$, d_c is the current normalized distance between the agent and the target position (different for each TC) and $d_i = 0.0875$ is the maximum normalized distance for an agent to be able to hit an opponent. The resulting value of Equation 1 was then clipped to $[-0.5, +0.5]$, so that the total reward is maintained in the range of $[-1, +1]$, preventing the agents on maximizing this reward only. It is worth noting that the distance reward was applied every time the agent took an action and for this reason it was divided by MaxSteps, resulting in the same minimum and maximum values at the end of an episode.

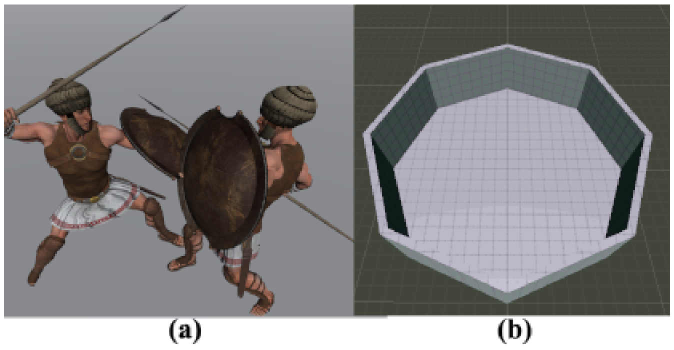
Figure 1: Two agents (ancient Greek warriors) during a melee action (a) and the training environment, the arena (b).

4 EXPERIMENTAL SETUP

The training environment and agents were implemented using the Unity Game Engine, using the ML-Agents toolkit [7] for the training process of the agents. The training was done by using an 8 core 16 threads CPU, 16GB RAM and a RTX 3070 Laptop GPU, with the training process including 16 separate environments running concurrently, using a headless (no graphics enabled) executable.

4.1 Environment

Figure 1 depicts two 3D avatars representing two opponent agents in melee action (a), and the environment, the arena (b). A special care is taken so that the environment and agents be of natural dimensions, thus the arena is an octagonal region of a diameter of 20m, enclosed with walls of 5.25m height, whereas the height of the avatars of the agents is fixed to 1.8m for all the agents. The arena confines the movement of the agents and they eventually have to face each other.

4.2 Training Approach

Two groups of 4 agents were battling among each other during the training process. In every episode a new battle started with a new random position and rotation for each agent inside the arena. Each group was spawning on a separate half of the arena (top and bottom), with the battle lasting a maximum of 15,000 steps or until a group won. After that, the entire environment was reset.

4.3 Learning Parameters

For the experiments we used the Proximal Policy Optimization (PPO) [15] RL algorithm, which is considered to be a very effective approach in large scale complex environments. The setup of the neural network consists of 2 hidden layers each one with 256 neurons. For the optimization algorithm we used: $learning\ rate = 0.0003$, $\beta = 0.005$, $\epsilon = 0.2$, $\lambda = 0.95$, $epochs = 3$, $batch\ size = 1024$ and $memory\ size = 20480$. The agents' extrinsic reward parameters were: $strength = 1.0$ and $\gamma = 0.99$. The values of those learning parameters were chosen after testing a large number of different parameters, from larger networks and values to small networks and even smaller *learning rate*, *beta* etc. The *batch size* and *memory size* parameters, were set to higher values as the length of the episode was large, requiring the collection of larger memory and (as a result) for a larger *batch size* during the model training.

Moreover, for the multi-agent credit assignment algorithm, we used the MultiAgent POsthumous Credit Assignment (MA-POCA) algorithm provided by the ML-Agents toolkit, which utilizes a centralized neural network for the whole group of agents, giving rewards for the whole group. With MA-POCA, agents eventually learn how to cooperate together so as to earn the reward, while also helping each other to earn their individual rewards. The algorithm is built upon other cooperative learning algorithms, such as [5, 12]

The total duration of the training was 15 million steps for each TC. We used self-play parameterization for the agents in the training process, to allow them to train against various models (older versions) of opponents, resulting in a more stable learning process, while also producing variable opponent behaviors, increasing (as a result) the complexity of the non-stationarity of the environment. Thus, the parameters used where the following:

(1) *save_steps* = 25000: number of steps between model snapshots that are saved
(2) *team_change* = 150000: after this number of steps the agent changes teams
(3) *swap_steps* = 50000: number of steps before swapping the enemy teams model

5 DISCUSSION

In the inference phase of the experiment, we performed a number of different scenarios regarding the sizes of each agent group for each TC. Each scenario was performed 500 times, with the same configuration and the following group sizes: *1v1*, *2v5*, *4v4* and *6v3*. With these setups, we tested the generalization of the agents' performance under different constraints (e.g. lower number of allies or opponents).

Table 1: Win-rates of the first team for each TC scenario.

	Team Sizes			
Behavior type	**1vs1**	**2vs5**	**4vs4**	**6vs3**
TC-1	53.80%	39.80%	54.40%	64.40%
TC-2	60.80%	37.60%	55.80%	61.40%
TC-3	46.60%	27%	43.80%	51.80%

Based on the data gathered during the experiments, we created heat map representations for each TC, as shown in tabular form in Figure 2. In these heat maps, the concentration of heat corresponds to the most busy regions in the arena. Overall, the battles took place in various regions of the arena and therefore had different duration and outcomes. For example the battles of TC-1 "Center" ended quite fast and the agents utilized quite large areas of the arena in almost all cases. In the TC-2 "Midpoint", both groups rushed onto each other, ending the battle fast by trying to squeeze each other at a corner/edge of the arena. In most cases the fastest acting group (squeezing the other group) won the battle. In the TC-3 "Both" the agents of each group split, each one fighting its closest enemy while trying to keep the battle in the middle of the arena.

The analysis of the win-rates revealed some important and expected outcomes. As shown in Table 1, in the cases in which the group sizes were the same for all TC scenarios, a win rate of 44 – 60% was observed, which is reasonable as both groups had the same size and same configuration. Similarly, in most cases of imbalanced group sizes, in which the larger group had a much higher win rate than the smaller one, with about 27 – 40% win rate for the smaller group in the scenarios of *2v5* and 35 – 40% win-rate for the scenario

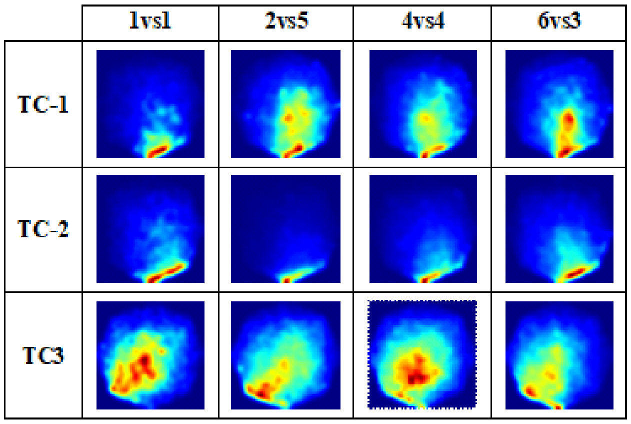

Figure 2: Heat maps of agents' activity for each TC scenario.

of *6v3*. At this point, an important exception should be highlighted: in the TC-1 "Both" with group size *6v3* the smaller group's win-rate was 48.2%, an average difference of about 15.7% to the other two *6v3* TC scenarios. This is due to the fact that a smaller group size results in better cooperation ("wiser behaviors") among the agents. Simply put, they act more freely/wisely by focusing more on their opponents instead of their co-warriors, as it happens in large groups of agents, where one may leave his fighting position to protect his co-warrior resulting in a loss of a battle.

An important conclusion that can be derived from the heat maps is that there is a slight bias towards the lower part of the arena, especially in the TC-2 "Midpoint". This was happening due to the fact that during the training process, the team that was spawning at the lower half of the arena (and most of the times losing the episode) was running away from the enemy team, so as to stay alive for as much time as possible. Therefore, this resulted in the agent model to become biased towards that part of the arena.

6 CONCLUSIONS

In this work in progress, we presented a complex mixed multi-agent (cooperative and competitive) environment, represented by groups of warriors in a melee game, briefly analyzing the RL agents' behaviors under some constraints. The overall goal was to experiment in the behavior-shaping of autonomous agents. In order to implement the behavior-shaping process, we proposed a dynamic rewarding approach, with quite large self-adapted rewards and penalties.

In our preliminary experiments, the agents were tested and evaluated in different Test-Case scenarios. The results show that the agents were able to learn the corresponding behaviors that were favored by the behavior-shaping algorithm. Additionally, the analysis showed encouraging results regarding the behavior-shaping of the agent in mixed environments, through the dynamic reward shaping approach, paving the road for further study on this subject.

The study implemented a cooperative and competitive multi-agent environment, utilizing the MultiAgent POsthumous Credit Assignment training algorithm which has shown to perform well in the literature in this type of environments. Although the episodes lasted long at the beginning of the training process, eventually they lasted much less than the maximum number of actions per episode, meaning that smaller values for the aforementioned parameters should be tested. Lastly, the training approach required training for a long time (10,5 hours on average), despite using multiple instances of the environment. The training duration can be attributed to the fact that the environment is complex with a large action space, despite it being small in size, while also having a large observation space. Therefore, the training duration and results of a smaller observation space should be researched.

As for future works, the study will focus on:

- Comparing the performance of a different training algorithm (such as SAC) and so, changing the action space from a discrete to continuous one;
- increasing the complexity of the environment, along with the available actions in the action space of the agents, such as jumping, crouching and different weapon usage;
- designing a reward function so that more actions can be incorporated to the behavior-shaping process, making an agent defend more, or prefer a type of attack (heavy or light) more than another;
- analyzing the behaviors in different environments and arenas, such as in unlimited and unconstrained open spaces;
- analyzing the resulting behaviors with regards to the realism of those behaviors in different kinds of situations and battle scenarios.

ACKNOWLEDGMENTS

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call "Special Actions Aquaculture - Industrial Materials - Open Innovation in Culture" (project code:T6YBP-00166).

REFERENCES

[1] Péter Almási, Róbert Moni, and Bálint Gyires-Tóth. 2020. Robust Reinforcement Learning-based Autonomous Driving Agent for Simulation and Real World. *2020 International Joint Conference on Neural Networks (IJCNN)* (July 2020), 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207497 arXiv: 2009.11212.
[2] André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. 2020. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences* 117, 48 (Dec. 2020), 30079–30087. https://doi.org/10.1073/pnas.1907370117
[3] Sean L. Barton, Nicholas R. Waytowich, Erin Zaroukian, and Derrik E. Asher. 2018. Measuring collaborative emergent behavior in multi-agent reinforcement learning. *arXiv:1807.08663 [cs]* (July 2018). http://arxiv.org/abs/1807.08663 arXiv: 1807.08663.
[4] Linqin Cai, Binbin Liu, Jimin Yu, and Jianrong Zhang. 2017. Human behaviors modeling in multi-agent virtual environment. *Multimedia Tools and Applications* 76, 4 (Feb. 2017), 5851–5871. https://doi.org/10.1007/s11042-015-2547-z
[5] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual Multi-Agent Policy Gradients. arXiv:1705.08926 [cs.AI]
[6] Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. 2019. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *Science* 364, 6443 (May 2019), 859–865. https://doi.org/10.1126/science.aau6249 arXiv: 1807.01281.
[7] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. 2020. Unity: A General Platform for Intelligent Agents. arXiv:1809.02627 [cs.LG]
[8] Chairi Kiourt and Dimitris Kalles. 2016. Synthetic learning agents in game-playing social environments. *Adaptive Behavior* 24, 6 (Dec. 2016), 411–427. https://doi.org/10.1177/1059712316679239
[9] Chairi Kiourt, Dimitris Kalles, and George Pavlidis. 2019. Rating the skill of synthetic agents in competitive multi-agent environments. *Knowledge and Information Systems* 58, 1 (2019), 35–58.
[10] Eric D. Langlois and Tom Everitt. 2021. How RL Agents Behave When Their Actions Are Modified. In *AAAI*.
[11] Daniele Loiacono, Alessandro Prete, Pier Luca Lanzi, and Luigi Cardamone. 2010. Learning to overtake in TORCS using simple reinforcement learning. In *IEEE Congress on Evolutionary Computation*. 1–8. https://doi.org/10.1109/CEC.2010.5586191
[12] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6382–6393.
[13] Matheus R. F. Mendonça, Heder S. Bernardino, and Raul F. Neto. 2015. Simulating Human Behavior in Fighting Games Using Reinforcement Learning and Artificial Neural Networks. In *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 152–159. https://doi.org/10.1109/SBGames.2015.25
[14] OpenAI. 2018. OpenAI Five. https://blog.openai.com/openai-five/.
[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[16] Hartmut Surmann, Christian Jestel, Robin Marchel, Franziska Musberg, Houssem Elhadj, and Mahbube Ardani. 2020. Deep Reinforcement learning for real autonomous mobile robot navigation in indoor environments. *arXiv:2005.13857 [cs]* (May 2020). http://arxiv.org/abs/2005.13857 arXiv: 2005.13857.

[17] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA.

[18] Xiangjun Wang, Junxiao Song, Penghui Qi, Peng Peng, Zhenkun Tang, Wei Zhang, Weimin Li, Xiongjun Pi, Jujie He, Chao Gao, Haitao Long, and Quan Yuan. 2021. SCC: an efficient deep reinforcement learning agent mastering the game of StarCraft II. *arXiv:2012.13169 [cs]* (May 2021). http://arxiv.org/abs/2012.13169 arXiv: 2012.13169.

[19] Ziyu Wang, Josh Merel, Scott Reed, Greg Wayne, Nando de Freitas, and Nicolas Heess. 2017. Robust Imitation of Diverse Behaviors. *arXiv:1707.02747 [cs]* (July 2017). http://arxiv.org/abs/1707.02747 arXiv: 1707.02747.

[20] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. *arXiv:1911.10635 [cs, stat]* (April 2021). http://arxiv.org/abs/1911.10635 arXiv: 1911.10635.