Multi-Agent Competitive and Cooperative Reinforcement Learning

David Winderl **Camilo Requena Witzig** **Marco Hövekenmeier**
Tecnical University of Munich (TUM)
{david.winderl, camilo.requena, m.hoevekenmeier}@tum.de

Abstract

In this project, we investigated learned strategies of multi-agent reinforcement learning settings in terms of emergent behaviour. We focused specifically on cooperation and competition between the agents. To achieve this, three new and more complex environments were created based on OpenAI Gym. In these environments, agents were trained with the algorithm Deep Determinstic Policy Gradient (DDPG), as well as its extensions for multi-agent scenarios MADDPG and DE-MADDPG. Multiple experiments were carried out and benchmarked. Our results suggest that agents are only able to learn one complex behavior at a time and that individual rewards are best suited for the agents to learn in complex multi-agent settings.

1 Introduction

In the past years, reinforcement learning (RL) has been mainly used in various games and robotics applications. Here, the focus often lied on single agent systems, for which algorithms were continuously developed in combination with Deep Learning techniques (François-Lavet et al., 2018). However, with respect to future applications in robotics, for example within autonomous driving as well as mobile robotics, in production or home environments, multiple agents will need to perform cooperative actions simultaneously. Examples for purely technical agents in this context are automated lane merging maneuvers on the road, but cooperation between humans and machines will also be required more frequently in a wide variety of scenarios in the future (Bdiwi et al., 2017). The RL methodology of deriving suitable policies through positive and negative reinforcements from the environment is often more difficult in multi-agent systems (Lowe et al., 2017). For this reason, the *Multi-Agent Deep Deterministic Policy Gradient* (MADDPG) algorithm was developed for multi-agent systems (Lowe et al., 2017). In the paper and implementations, the algorithm was tested on different, simple environments based on the *OpenAI Gym* toolkit. In our project, we focused on validating the performance of the algorithm in terms of the strategies learned by the agents with respect to cooperation and competition in more complex environments. For this, three new environments were developed and a large number of experiments with different settings were performed and analyzed. We investigated the best possibilities to achieve cooperation and competition in multi-agent scenarios and explored possibilities to incentivize specific behavior.

In the next section, related work will be presented. In Section 3, our developed environments will be explained and our experiments as well as their results will be shown in Section 4. The report ends with some conclusions and ideas for possible future work.

2 Related Work

In this section, we provide a short overview of some technical details of the algorithms we used and our environment setup. For a more basic introduction to RL concepts we recommend the reader to have a look at this educational resource provided by OpenAI (OpenAI). Our reinforcement learning setup consists of multiple agents that interact in discrete timesteps with its environment E. Since we work in the multi-agent setting, this environment is modeled as a Markov game, an extension of the Markov decision process (Littman, 1994). It consists of the state space $\mathcal{S}$, which includes all possible configurations of the N agents, the action spaces $\mathcal{A}_1, \ldots, \mathcal{A}_N$ and deterministic state transitions $s_{t+1} = s'_t = f(s_t, a_t) \in \mathcal{S}$. In each timestep, each agent observes the environments state $o_{i,t} \subseteq \mathcal{S}$ and chooses an action $a_{i,t}$ based on its deterministic policy $\pi_i : \mathcal{S} \rightarrow \mathcal{A}_i$ and receives a scalar reward $r_t(s_t, a_t)$. Introducing the discount factor γ and time horizon T, the discounted reward

function of the i-th agent reads $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ (Lowe et al., 2017). The Action-Value (or Q-) function $Q^{\boldsymbol{\pi}}(s, a) = \mathbb{E}[R \mid s_t = s, a_t = a]$ describes the expected reward when performing action a in state t and forever after act according to the policy π (Lowe et al., 2017). In deep RL methods, this Q-function is approximated by a neural network, parametrized by θ with the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}\left[(Q_\theta^*(s,a)) - y)^2\right], \quad (1)$$

$$\text{where} \quad y = r + \gamma \max_{a'} \bar{Q}^*\left(s', a'\right) \quad (2)$$

Here, the function $\bar{Q}$ denotes the target-network, whose parameters only slowly follow those of the main network through polyak averaging $\theta_{targ} \leftarrow \rho\, \theta_{targ} + (1-\rho)\,\theta$ (Lillicrap et al., 2015). Thus, the overestimation bias as studied by Hado van Hasselt, Arthur Guez and David Silver (van Hasselt et al., 2015) can be reduced, which stabilizes the learning.

We use the Q-function to ultimately find a policy that maximizes the expected reward. When approximating this policy with a neural network, we simply get the loss function in equation 3.

$$J(\phi) = -\mathbb{E}[Q(s, \pi_\phi)] \quad (3)$$

In machine learning settings, we often assume that the data is independent and identically distributed, which is clearly not the case for sequences of experiences in Markov games. Therefore, a finite replay buffer is introduced, in which the actions that led to state transitions and its rewards are stored. When updating the network weights, mini-batches of this replay buffer are then sampled to make the learning more robust.

The DDPG algorithm is an actor-critic method, because we do not only optimize the Q-function (critic), but also the policy (actor) simultaneously (Lillicrap et al., 2015). The input space of the Q-network for each agent in this case has the dimension $dim(o_i) + dim(a_i)$. However, since in each time step N agents act simultaneously upon the environment, it becomes unstationary for each agent which makes learning an optimal Q-Function and policy very hard (Lowe et al., 2017). Furthermore, it is not easy to learn cooperative or competitive policies, as the agents do not know about the other agents' observations and policies. Therefore, an extension of the DDPG algorithm – the so-called MADDPG – was introduced (Lowe et al., 2017).

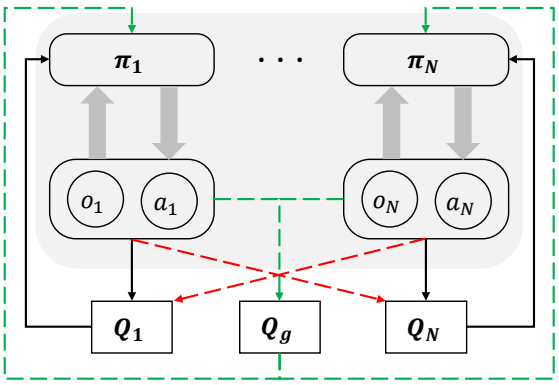

Figure 1: Overview of the learning process for DDPG (black arrows) and its extensions MADDPG and DE-MADDPG (additional red and green dashed lines, respectively). Q_g denotes the global critic in DE-MADDPG.

Here, each critic is augmented with information of the other agents' observations and actions. Thereby, the environment becomes stationary for each agent again. Furthermore, it is easier to learn cooperative strategies since the other agents' actions can be inferred. One downside of this approach is, that each critics' input size becomes $\sum_{i=1}^{N} dim(o_i) + dim(a_i) \approx N(dim(o_i) + dim(a_i))$. As this leads to higher computational complexity, Hassam Ullah Sheikh and Ladislau Bölöni proposed the decomposed MADDPG (DE-MADDPG) (Sheikh and Bölöni, 2020). Here, similarly to the classic DDPG algorithm, every agent has its own local critic. Additionally, one global critic is introduced which has access to the information of all N agents' observations and actions, which leads to a less steep increase in parameters with an increasing number of agents. The policy of each agent therefore tries to maximize the global and local expected reward. The methodology of the three introduced algorithms is illustrated in Figure 1.

3 Environments

In order to evaluate different strategies developed by the different algorithms, more complex environments with a higher dimensionality were created. Lowe *et al.* (Lowe et al., 2017) provide, aside from the algorithm, a set of various implemented scenarios, which involve communication, cooperation and also competitive aspects. The environments we implemented in this project are called *Tragedy of commons*, *Team Fight* and *Hunter Gatherer*. These are all fully observable and deterministic environ-

ments with continuous state and discrete time. In the following sections, the different environments will be described briefly and their intentions as well as evaluation goals within the scenarios will be pointed out. A more detailed description of every parameter of the environments can be found at the accompanying git repository for all environments.

3.1 Tragedy of commons

This environment is based on the idea of the tragedy of the commons. The tragedy of commons describes the social dilemma of an open system that may collapse due to overconsumption by the participants, for example overfishing in a common pond (Hardin, 1968). Its abstraction in game theory is the Common-Costs-Private-Profit-Game (CC-PP-Game). The theoretic outcome of the CC-PP-Game is that the common resource will be overused by all participants. Nevertheless, studies have shown that a self governance of common resources is possible (Ostrom et al., 1992). The overall research question of this environment was to explore whether the MADDPG algorithm was capable to develop a policy which includes the self-governance of a pool of resources.

Description In order to model this scenario, a forest with food orbs was created. Each food orb gets deactivated for a certain period of time if an agent collects it. The agent collecting the food orb will get a positive reward and one can optionally define a positive team reward for all other agents in case of such an event. After the period expires, the food orb is active again for repeated collection. If all food orbs in the forest are in a timeout, the forest cannot recover. Then, no food orbs can respawn in the forest until the end of the game. Optionally, one can define a penalty for this situation. In order to give the agents more options, houses were introduced. Here the agent could acquire an additional reward if it had first intersected with a food orb and then with his house.

Observations The agent observes the position of all other agents, the relative position of his home [1], the relative food position and its state (whether or not the food object is active).

Reward The environment was designed, so that one could specify different configurations of hyperparameters, leading to different behaviour types of the agents. Two parameters turned out to be highly relevant, namely P_DESERT and COLLECTION_REWARD_DIST. P_DESERT is the penalty that gets subtracted from the overall reward in case that the agents rob the forest. COLLECTION_REWARD_DIST describes the distribution of the collection reward.

Cooperation and competition The cooperation and competition aspects in this scenario can be summarized as follows: On the one hand, the agent wants to *cooperate* with the other agents in order to avoid an empty forest. On the other hand, the goal of an intelligent agent is to maximize his reward. To do this, the agent is in *competition* with the other agents to collect as many food orbs as possible.

3.2 Hunter Gatherer

Description The *Hunter Gatherer* environment is based on the *Simple Tag* environment by Lowe *et al.* (Lowe et al., 2017), but expands it by introducing two teams consisting of the so called hunters and gatherers. Hunters have to catch gatherers from the opposite team and gatherers have to collect food items that are randomly distributed around the environment while evading the hunters. The overall research question of this environment was to find out if we were able to incentivize specific behavior and identify which rewards were key for the agents to properly learn strategies to act in the environment.

Observations The agents can observe the position of all other agents, food items and landmarks. The gatherers additionally observe the velocities of other agents, as was the case for similar environments by Lowe *et al.* (Lowe et al., 2017).

Reward For this scenario we introduced the following possible rewards as variable parameters: Every individual agent had an individual reward for achieving its' goal, so for gatherers it was collecting food items and a penalty for being hunted and for hunters it meant catching other gatherers. Additionally, every group in a team, e.g. the hunters or the gatherers, had collective rewards, meaning that if one of the hunters caught a gatherer, all of the hunters in the team would get a reward. The same was done for the gatherers. We further introduced team rewards, so any agent of a team would be rewarded for the actions of another teammate, regardless of the group. Additionally, we used distance-based rewards, so that hunters would try to be close to the gatherers and gatherers would

[1] relative position means the vector difference: $p_a - p_h$

try to be far away from hunters and closer to food items. Lastly, we found that penalties applied to inactive agents were needed. Other parameters we varied during testing were the algorithms used by the teams, the number of food items, hunters, gatherers, their speed and whether gatherers would respawn in one corner of the environment or simply keep collecting food after being caught.

Cooperation and competition There are multiple levels of cooperation between the hunters of a team and the gatherers, but also between all of the agents of the same team, across hunters and gatherers. Similarly, there is competition between the hunters of one team and the gatherers of the opposite team, but there is also the possibility of competition between the hunters of one team or the gatherers, especially if there is a high individual reward for catching a gatherer or collecting food items.

3.3 Team Fight

Description This environment is heavily based on the *Hunter Gatherer* environment as most aspects are the same. The main difference is that it combines the role of a hunter and gatherer in one agent. Thus, two agents of one team can catch an agent of the opposite team, but all agents can collect food items. This means that the possibilities for interaction and the levels of cooperation and competition are even more complex. The overall research question of this environment was to explore whether the agents were able to properly learn to follow both behavior types and correctly act within the environment.

Observations The observations in this environment are identical to those in the *Hunter Gatherer* environment.

Reward For this environment, we primarily used individual rewards for achieving a goal, since the action of catching an opposite teams' agent implicitly introduced team rewards as two agents had to work together to achieve this goal. We also reintroduced the distance-based rewards to try and incentivize a teammate closing the distance to an adversary when a teammate is close to them. The other variable parameters are mostly identical to the *Hunter Gatherer* environment and can be found in our repository.

Cooperation and competition Agents have to follow two different strategies simultaneously, namely collecting food items and catching the opposite teams' agents. To achieve the latter, they must cooperate with other teammates, as only two together can catch another agent. This makes the environment much more complex.

4 Experiments & Results

In the experiments, the goal was to test the enviroments and follow up on their specific research question. In order to achieve this, different environments where trained simultaneously in different configurations and evaluated later on. The results of the experiments will be presented in the following.

4.1 Tragedy of commons

Problems learning complex behaviour Firstly, we found that, in terms of the achieved reward, even though it would have been a better strategy to bring food orbs home after collection, agents were not able to learn this specific behaviour. This occurred even when it was highly incentivized by setting the corresponding reward distribution via COLLECTION_REWARD_DIST as can be seen in Figure 2. One can ascribe this behaviour to the fact that the agents seem to only be able to learn one specific type of behaviour. Another possible reason could be that the algorithms are stuck on local optima due to sparse occurrence of the agents intersecting with their home and having already collected a food orb. Since the agents were not able to learn the full desired behaviour of the environment, the focus of the environment shifted towards only collecting food orbs.

Learning of cooperation As stated in the description of the environment, agents are in conflict of trying to collect as much as possible while trying to preserve the forest. In Figure 3, one can see a bar-plot of the collected food orbs per agent for the MADDPG and DDPG algorithm. For P_DESERT$= 0$, both algorithms tend to just rob the whole forest. With an increasing P_DESERT, using DDPG one agent always tends to show competitive behaviour. The agent collects as many food orbs as possible, but also tries not to rob the forest. Meanwhile, the other agents stay out of the forest in order to avoid the high penalty. With MADDPG, actually two agents tend to share the available food orbs, while one agent still stays out of the forest in order to avoid the high penalty. Consequently, one can clearly see that due to the augmented critic

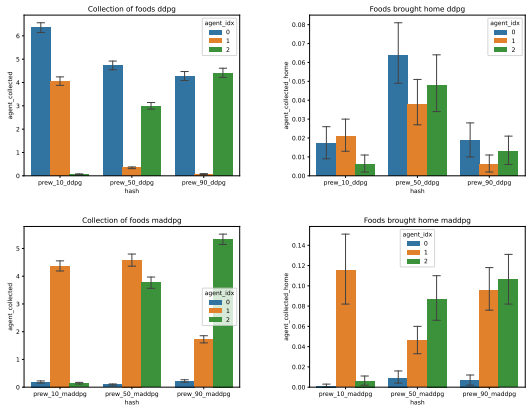

Figure 2: The plots show the average amount of food orbs collected per round or brought home. On the X-Axis, the COLLECTION_REWARD_DIST is plotted from 0.1 to 0.9. One can clearly see that the agents favored collecting (which occurred around 4-6 times on average per round) over bringing the foods to their home (which occurred close to 0 times on average). Hence, the agents were not able to learn the desired behaviour of this environment.

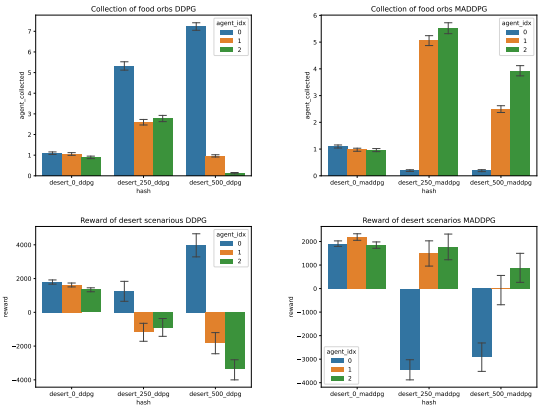

Figure 3: One can see the collection of the food orbs for DDPG and MADDPG for different desert penalties. Using MADDPG, two agents share the available food objects, while using DDPG only one agent tends to collect all of the food orbs while leaving one out.

of MADDPG, the agents tend to follow a different kind of behaviour.

4.2 Team Fight

Similarly to the *Tragedy of commons* environment, the agents were not able to follow the two specified behaviors simultaneously, namely collecting food items and catching other agents. In our experiments, we were only able to have the agents either only collect food items or only try to catch each other, so they would simply pile up and stay grouped all together without trying to collect any food items. Because of this, we decided not to test any further on this environment, but rather focus on the *Hunter Gatherer* environment. Future work could focus on implementing a more complicated reward system, a visual field for the agents to see only items and other agents in a closer proximity, or applying a different algorithm to help the agents learn both behavior types.

4.3 Hunter Gatherer

Learning complex behaviors One of the goals of the project and specifically of this environment was to research the learning of complex incentivized behavior. Through the experiments, we found multiple possibilities of achieving this goal. With the introduction of team rewards, we observed that one of the agents of each group would sometimes stop moving. This could be explained by the fact that the agent would be rewarded by the work of their teammates. We described this as the learning of a 'lazy' behavior. Consequently, we introduced a negative reward punishing inactivity, which exactly counteracted this 'lazy' behavior and resulted in all agents moving according to the environment again. This showed that we were able to enforce behavior using the reward system, but we were also able to make the agents learn a specific behavior by changing the environment. As we introduced respawning of gatherers if they were caught, the hunters of the opposite team learned to wait by those spawn points and trap the gatherers. With regard to future use of RL to train cooperation of intelligent autonomous systems with humans, such behavior types could be even be characterized as some sort of learned moral behavior.

DDPG and MADDPG In the course of our project, we were able to validate the results from Lowe *et al.* (Lowe et al., 2017) for the MADDPG algorithm. For this environment, we compared a base setting using four agents and changing only the algorithm used. The results can be seen in figure 4. In average, the total rewards were much higher using the MADDPG algorithm, depicted on the left. One of the reasons for this might be that the hunters are able to cooperate better and thus catch more gatherers. This would also explain why the reward for the gatherers is lower using MADDPG, since it is more difficult for them to get away from the hunters and they have the disadvantage of being hunted while trying to collect the food items.

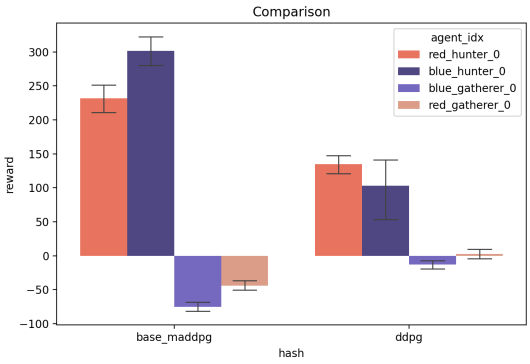

Figure 4: Comparison of average reward using the MADDPG (left) and DDPG (right) algorithm in the *Hunter Gatherer* environment.

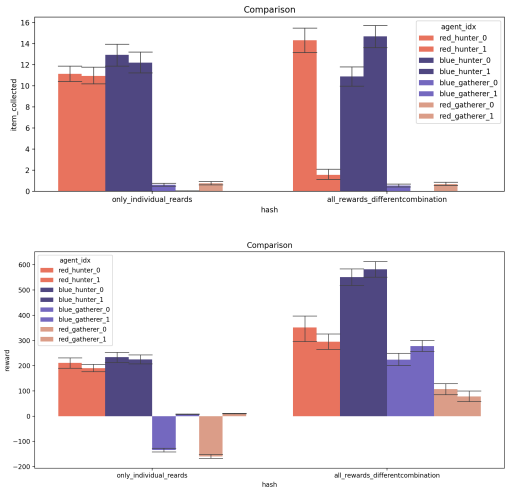

Figure 5: Average amount of items (upper graphs) and rewards (lower graphs) collected per round using only individual rewards (left) or all rewards (right) in the *Hunter Gatherer* environment.

Learning strategies Another goal of this environment was to find the best strategies to use in order for the agents to learn cooperative and competitive behavior. We focused on identifying which rewards were key for the agents to properly learn complex strategies. Here, we tested different combinations of the described rewards in a setting with eight agents. This led to very mixed results, as many configurations would result in agents running around without a rational policy. We found configurations using individual, group and team rewards that showed good results, but at least one agent would always either move randomly or stand still, even with the use of penalties for inactiveness. Thus, we tested the same environment once using only individual rewards, then using only group rewards and then focusing on team rewards. We found that for the group and team reward setting, the results stayed the same. In contrast, the results for the individual reward configuration were much better. Here, all of the agents would move according to their role in the environment.

The upper graphs in Figure 5 show the average amount of items collected per round for every agent, so for gatherers the amount of food items collected and for hunters the amount of gatherers caught. It can be observed that the total amount of items collected is higher using only individual rewards, mainly because all hunters hunt almost the same number of gatherers. When using all rewards, one of the hunters collects less gatherers than the others. This was also shown in the environment, as the overall behavior of the groups was much more coherent in the individual reward setting. Looking at the food items collected, it also becomes clear that for each team, one of the gatherers learned to collect the food items, while the other learned to only run away from hunters. This is also visible when comparing the average reward for each actor, shown in the lower graphs. The gatherers that actually collected food items had a large negative reward, as they were also the gatherers being caught by the other team, while the gatherers that only ran away had a positive average reward per round. Once again, this shows that the agents were only able to learn one specific complex behavior type.

From these results, we concluded that the use of individual rewards is better fitting to achieve meaningful learning in very complex environments, with many possibilities for interaction. It seems that the use of group and team rewards often led to confusing learning for the actors in the setting with eight total agents, while the use of specific individual rewards showed better learning according to the environment and resulted in an emerging cooperative or competitive behavior. The hunters for instance would still work together to catch one other agent and the gatherers split up to better collect food items without being caught, which can be seen in Figure 6.

4.4 Experiments carried out with DE-MADDPG

Experiments were also performed with DE-MADDPG in the environments *Tragedy of commons* and *Hunter Gatherer*. In both settings, it could be observed that the learned strategies were

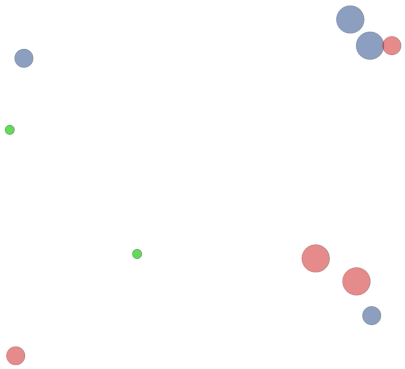

Figure 6: *Hunter Gatherer* environment using only individual rewards.

better in terms of the collected reward than those strategies learned with the basic DDPG algorithm. This shows that the introduction of one shared global critic with the information of all agents in fact makes it possible to learn more cooperative strategies, needed for the higher reward we observed. However, we were not able to reproduce the results from the paper (Sheikh and Bölöni, 2020), where a better performance compared to training with the MADDPG algorithm could be observed. Since we were not able to explain these discrepancies, it would make sense to perform a more in-depth analysis of the algorithm on the presented environments, which was out of the scope of our project.

5 Conclusion & Future Work

In the course of this project, we verified that specific cooperative and competitive behaviour could be enforced by applying specific rewards or changing the environment in the multi-agent reinforcement learning setting. We developed three complex environments and found that the use of individual rewards was most suited to train specific behavior. Further, we showed that MADDPG responded better to such changes and proved to be more stable than DDPG. This reproduces the findings by Lowe *et al.* (Lowe et al., 2017).

Nevertheless, we were also able to elaborate the weaknesses of the MADDPG algorithm. Agents trained with this algorithm were not able to learn multiple complex behaviours simultaneously. This was clearly visible in the *Team Fight* environment and a possible explanation for the inability of agents to learn the full expected behaviour in the *Tragedy of commons* environment. Progress in the direction of hierarchical or multi-objective RL was already made in order to tackle such problems (Yang et al., 2019). Secondly, MADDPG shows a drastic increase of parameters when scaling the number of agents. Here, further elaboration of the DE-MADDPG algorithm by Sheikh and Bölöni (Sheikh and Bölöni, 2020) could show interesting results. Lastly, further experiments with our environments and other algorithms could be used to verify our results with and would further contribute to this field of research.

References

Mohamad Bdiwi, Marko Pfeifer, and Andreas Sterzing. 2017. A new strategy for ensuring human safety during various levels of interaction with industrial robots. *CIRP Annals*, 66(1):453–456.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. 2018. An introduction to deep reinforcement learning. *CoRR*, abs/1811.12560.

Garrett Hardin. 1968. The tragedy of the commons. *Science*, 162:1243–1248.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning.

Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*, pages 157–163. Elsevier.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *CoRR*, abs/1706.02275.

OpenAI. Key concepts in rl.

Elinor Ostrom, James Walker, and Roy Gardner. 1992. Covenants with and without a sword: Self-governance is possible. *American Political Science Review*, 86(2):404–417.

Hassam Ullah Sheikh and Ladislau Bölöni. 2020. Multi-agent reinforcement learning for problems with combined individual and team reward. *CoRR*, abs/2003.10598.

Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *CoRR*, abs/1908.08342.