Action Markets in Deep Multi-Agent Reinforcement Learning

Kyrill Schmid ✉, Lenz Belzner, Thomas Gabor, and Thomy Phan

Mobile and Distributed Systems Group, LMU Munich, Germany
{kyrill.schmid,belzner,thomy.phan,thomas.gabor}@ifi.lmu.de
http://www.mobile.ifi.lmu.de

Abstract. Recent work on learning in multi-agent systems (MAS) is concerned with the ability of self-interested agents to learn cooperative behavior. In many settings such as resource allocation tasks the lack of cooperative behavior can be seen as a consequence of wrong incentives. I.e., when agents can not freely exchange their resources then greediness is not uncooperative but only a consequence of reward maximization. In this work, we show how the introduction of markets helps to reduce the negative effects of individual reward maximization. To study the emergence of trading behavior in MAS we use Deep Reinforcement Learning (RL) where agents are self-interested, independent learners represented through Deep Q-Networks (DQNs). Specifically, we propose *Action Traders*, referring to agents that can trade their atomic actions in exchange for environmental reward. For empirical evaluation we implemented action trading in the Coin Game – and find that trading significantly increases social efficiency in terms of overall reward compared to agents without action trading.

1 Introduction

The success of combining reinforcement learning (RL) and artificial neural networks (ANNs) in single agent settings has also respawned interest in multi agent reinforcement learning (MARL) [8, 20, 9, 16]. In so called independent learning each agent is represented by a neural network which is trained according to a specific learning rule such as Q-learning [12]. When agents are self-interested the emergent behavior is often suboptimal as agents learn behavior w.r.t. their individual reward signal. In tasks such as resource allocation problems this leads to first-come, first-served strategies. The resulting allocations from such strategies are in general inefficient. An allocation is said to be inefficient, if there is another allocation under which at least one agent has higher reward and all other agents have at least equally high rewards compared to the former allocation.

While some work tries to mitigate greedy behavior based on game theoretic strategies such as Tit-for-Tat [9] we argue that inefficiency can also be seen as a consequence of market failure. Specifically, many settings provide no incentives for agents to increase efficiency. I.e., as long as an agent's best alternative in terms of utility is being greedy then the learned behavior is rational rather than

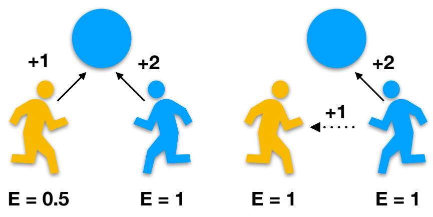


Fig. 1: Two agents competing for a coin: while pure self-interested behavior without trading incentivizes agents to act greedily the introduction of a market can help to increase both agents' expected value.

uncooperative. However, individual utility maximization can originate efficiency when agents are enabled to incentivize other agents. We call such a mechanism a market for behavior as it enables agents to trade behavior in exchange for other resources e.g. environmental reward. In the presence of a behavior market a utility maximizing agent can invest to stimulate behavior.

Figure 1 illustrates how the introduction of a behavior market helps to overcome inefficiency in a stylized scenario. Suppose two agents are competing for a coin where agent 1 (yellow) gains a reward of +1 while agent 2 (blue) gains a reward of +2 from gathering the coin. When there is a probability of 0.5 for both agents to get it when they step forward then they will have an expected value of 0.5 (agent 1) and 1 for agent 2. As each agent only considers it's own reward there will be no incentive for agent 1 to dedicate the coin to agent 2 while this would maximize the overall outcome. This changes when agents are enabled to exchange reward for behavior. When being able to trade, agent 2 could propose agent 1 a reward +1 when agent 1 steps back. In this case, expected values are both 1 which increases overall reward.

The main contributions of this paper are:

– A definition of action trading as a realization of a behavior market.
– Empirical evidence that the introduction of markets is sufficient in order to increase efficiency in MAS.

The rest of this paper is organized as follows: Section 2 gives an overview about related work. Section 3 describes the learning methods. Section 4 introduces action trading. Finally, in Section 5 we evaluate action trading in two experiments comparing self-interested agents with and without action trading in a matrix game and the Coin Game.

2 Related Work

Independent and cooperative RL in multi-agent systems has been researched for decades [21, 10, 14]. Recent successes of both model-free and model-based deep RL extending classical approaches with learned abstractions of state and action spaces [12, 17, 19] motivated the use of deep RL also in multi-agent domains [1, 3].

The tensions of cooperation, competitiveness, self-interest and social welfare have traditionally been researched in the framework of game theory [13]. Game theory has also been a central theoretic tool for studying multi-agent systems [18]. A recent line of research investigates game-theoretic considerations in multi-agent (deep) RL settings, extending the idea of classical games into the setting of sequential decision making under uncertainty [8, 20, 9, 16].

In particular, to bring the concept of social dilemmas closer towards real-world problems the authors of [8] propose sequential social dilemmas (SSDs) where cooperation and competition cannot be seen as an atomic action but are represented through complex policies. In different experiments the authors show how learned behavior depends on the choice of environmental factors such as resource abundance. Through variation of these external properties the authors train different policies and classify these as cooperative or competitive respectively. In this work we adopt the idea of SSDs with multiple independent agents each represented through deep Q-networks. Still, in our analysis we do not focus on the emergence of cooperative policies through variation of environmental factors. Instead we were interested in answering the question whether in a system of autonomous, self-interested agents the chance to make economical decisions leads to efficient allocation of resources and hence increases social welfare.

In [20] the authors demonstrated how cooperative behavior emerges as a function of the rewarding scheme in the classic video game Pong. Agents, represented by autonomous Deep Q-Networks, learned strategies representing cooperation and competition respectively through modification of the reward function. In our approach we do not specify the rewarding scheme as a static property of the environment but rather as a changing structure through which agents can express their willingness to cooperate.

To deal with resource allocation in MARL the authors in [11] propose resource abstraction where each available resource is assigned to an abstract group. Abstract groups build the basis for new reward functions from which learning agents receive a more informative learning signal. Whereas the building of abstract resource groups and hence the shaping of rewards is done at design time, in this work the transformation of reward schemes is part of the learning process.

An approach to carry the successful Prisoner's Dilemma strategy tit-for-tat into complex environments has been recently made by Lerer and Peysakhovich [9]. In their work they construct tit-for-tat agents and show through experiments and theoretically their ability to maintain cooperation while purely reactive training techniques are more likely to result in socially inefficient outcomes. The analysis of reward trading agents is more interested in emergent properties than in implementing a fixed strategy. We therefore make no other assumption than agents maximizing their own returns.

3 Reinforcement Learning

For the purpose of this work we follow the line of descriptive approaches similar to [8]. Rather than asking what learning rule agents should use we model each agent as a specific learner and observe the emergent system behavior. In this sense we model agents as independent learners, i.e., agents cannot observe each other but only recognize a changing environment which is the result of the learning of other agents. We apply methods from the framework of reinforcement learning where it is known that indepenent learning results in non-stationarity as well as to the violation of the Markov property [4, 7]. However, as [8] points out in the descriptive approach this can be considered as a feature rather than a bug as it is an aspect of the real environment that the model captures.

Reinforcement learning (RL) are methods where an agent learns a policy π from repeated interaction with an environment. If multiple agents are involved the problem can be described with a so called Stochastic Game (SG). Formally, a SG is a tuple $(\mathcal{S}, \mathcal{N}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ where: $\mathcal{S}$ is a finite set of states, $\mathcal{N}$ is a finite set of $\mathcal{I}$ players, $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_\mathcal{I}$ describes the joint-action space where $\mathcal{A}_i$ is the finite action set of player i, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition function and $\mathcal{R} = r_1, ..., r_\mathcal{I}$ where $r_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function for player i. [4]

An agent's goal is to maximize its expected return which is $\mathcal{R}_t := \sum_{t=1}^{\infty} \gamma^{t-1} R_t$. An agent decides which actions to take in a certain state according to a policy π which is a function $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ from states to probability distributions over $\mathcal{A}$. Over the course of training the agent is supposed to learn a policy that maximizes the expected return. One way to obtain a policy is to learn the action value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that gives the value of an action in a certain state. A popular way to learn the action value function is Q-learning where an agent i updates its values according to: $Q_i(s, a) \leftarrow Q_i(s, a) + \alpha\big[r_i + \gamma \max_{a' \in \mathcal{A}^i} Q_i(s', a') - Q_i(s, a)\big]$ where α is the learning rate and γ is a discount factor. From Q a policy π can be derived by using e.g. ϵ-greedy action selection where with probability $1 - \epsilon$ the agent selects an action with $argmax_{a \in \mathcal{A}} Q(s, a)$ and with probability ϵ the agent selects an action random uniform from the available actions.

In this work, we model agents as independent Q-Learners. Deep RL refers to methods that use deep neural networks as function approximators. In deep multi-agent RL each agent can be represented by a deep Q-network (DQN) [12]. For independent learners, each agent stores a function $Q_i : \mathcal{S} \times \mathcal{A}_i \to \mathbb{R}$ that approximates the state-action values.

4 Action Trading

This section formally introduces action trading which is realized through extending agents' action spaces. The idea of action trading is to let agents exchange environmental reward for atomic actions. Learning then comprises two parts: policies for the original action space of the stochastic game and a trading policy that represents an agent's trading behavior. To keep notation simple we define action trading for the two agent case i.e., $\mathcal{N} = \{1, 2\}$.

For a given stochastic game $(\mathcal{S}, \mathcal{N}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, action trading is realized through extending action spaces $\mathcal{A}_1$ and $\mathcal{A}_2$ in the following way: $\mathcal{A}'_1 = \mathcal{A}_1 \times (\mathcal{A}_2 \times [0, .., N])$ and $\mathcal{A}'_2 = \mathcal{A}_2 \times (\mathcal{A}_1 \times [0, .., N])$. I.e., action spaces $\mathcal{A}'_i$ comprise the original actions $a_{orig} \in \mathcal{A}_i$ and also trading actions $a_{trade} \in \mathcal{A}_j \times [0, .., N]$. A trading action a_{trade} is a tuple (a_{ij}, p) that defines the amount of reward $p \in [0, ..., N]$ that agent i is offering agent j for an action a_{ij}. p therefore is the price an agent pays and is transferred from agent i to agent j if a trade is established.

In this work we require a successful trade to satisfy two conditions. Firstly, agent i made an offer to agent j at time-step t for action a written as a_{ij}. Secondly, also at time-step t agent j actually chose action a, written as a_j. Thus, a trade will only be established if offer and supply match at the same time step. The resulting rewards at time-step t in the two agents scenario for agent 1 are $r_t^1 = \mathcal{R}^1 + p_2 * \delta_{21} - p_1 * \delta_{12}$ and for agent 2 $r_t^2 = \mathcal{R}^2 + p_1 * \delta_{12} - p_2 * \delta_{21}$ where $\mathcal{R}^i$ represents the original environmental reward and δ_{ij} are boolean values to define successful trades i,.e., $\delta_{ij} = \begin{cases} 1, & \textit{if } a_{ij} = a_j, \\ 0, & \textit{otherwise} \end{cases}$.

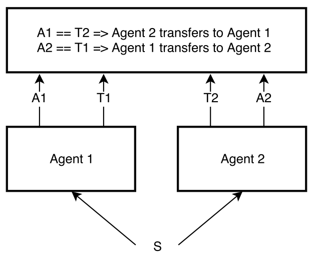

A1 == T2 => Agent 2 transfers to Agent 1
A2 == T1 => Agent 1 transfers to Agent 2
A1
T1
T2
A2
Agent 1
Agent 2
S

Fig. 2: Action trading describes a mechanism to offer other agents environmental reward in exchange for specific actions. Agents therefore choose in addition to their original actions also trading actions. A trade is realized when an offer matches an actual action.

Figure 2 visualizes how action trading is realized. Agents select actions from their original action space and from the trading action space. Trading actions describe agents' offers towards other agents for specific actions. Whenever an offer matches an actual performed action a trade is realized i.e., a fixed amount of reward is transferred between the two involved agents.

5 Experiments

In this section, we describe two experiments. The first experiment is an iterated matrix game that has been extended to enable agents to trade actions. The second experiment is the Coin Game, which is used for studying sequential social dilemmas in the recent literature for multi-agent learning [9, 2]. In all experiments we compared action traders with self-interested agents.

To measure the social outcomes of multi-agent learning, it is necessary to define a metric as the value function cannot be used as a performance metric like in single agent RL. To measure efficiency, we use the total sum of rewards obtained by all agents over an episode of duration T, also called the Utilitarian metric (U), which is defined by [15]: $U = \mathbb{E}[\frac{\sum_{i=1}^{N} R^i}{T}]$ where $R^i = \sum_{i=1}^{T} r_i^t$ is the return for agent i for a sequence of rewards $\{r_t^i | t = 1, ..., T\}$ over an episode of duration T. For the Coin Game the Utilitarian is complemented by the total number of collected coins, and the share of correctly collected coins within one episode.

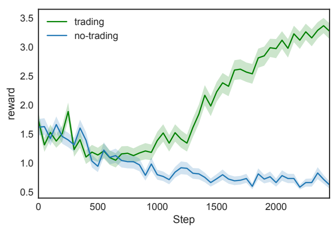
5.1 Iterated Matrix Game

To study the effects of action trading in a simple matrix game, we used a game with pay-offs as given in Figure 3a. Action trading in the matrix game was realized by extending action spaces $\mathcal{A}_i = \{1, 2\}$ to $\mathcal{A}_i = \{(1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)\}$, i.e., each agent decides what action to take from the original action space in combination with a trading action. The price in terms of reward is fixed with $p = 1$ for all actions. As learning rule we used tabular Q-learning with learning rate $\alpha = 0.001$. For action selection we used the ϵ-greedy Q-Function with ϵ decaying from 1.0 to 0.1 over 2500 steps.

The results from 100 runs each comprising 2500 steps are shown in Figure 3. Independent learners without trading (blue) start to select the dominating action $(1, 1)$ with high probability which is reasonable as agent 2 only ever receives reward when choosing action 1. Likewise, agent 1 learns to choose action 1 as a best response to the selection of agent 2. In contrast, independent learners with action trading (green) have decreasing reward for around 1000 steps. Afterwards overall reward constantly increases.

Agent 2 / Agent 1	**1**	**2**
1	0.5 / 0.5	4.0 / 0.0
2	0.0 / 1.0	1.0 / 0.0

(a) Payoffs



(b) Overall reward

Fig. 3: 100 runs of the iterated matrix game with payoffs as given in the table (left). Whereas non-trading agents (blue) fail to find a global optimum agents with action trading (green) eventually learn to maximize overall and indiviudal reward

5.2 The Coin Game

To study the effects of action trading in a problem with sequential decision making we adopt the Coin Game first proposed in [9]. The Coin Game is a 2-dimensional grid world that includes 2 agents (blue and yellow) and their respective coins. The task is to collect the coins and agents get a reward of $+1$ for collecting coins of any color. However, whenever an agent collects a coin that is different from its own color the other agent loses two points. To evaluate the performance of action trading for $n > 2$ we also tested an extended version of the Coin Game comprising 4 agents. The 4 agents Coin Game works in the same way, i.e., agents have their associated coins and impose costs on a fellow agent whenever they collect a differently colored coin.

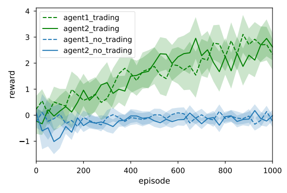
From the perspective of this work, the Coin Game can be seen as a task where resources (coins) need to be allocated to agents. When efficiency is measured as overall reward then it would be best if agents only collected their own coins to prevent imposing costs on the other agent. As a consequence agents have an incentive to pay the other agent for not collecting their own coins. Consider the situation, when agent 1 (yellow) is about to collect the blue coin. This will bring agent 1 a reward of $+1$ and -2 for agent 2 (blue). Consequently, agent 2 would be willing to pay a price $p \leq 3$ to agent 1 in exchange for the coin.

Action spaces A^i in the Coin Game have four actions: $A^i = \{North, South, East, West\}$. To reduce the trading options for agents at any step, we decided to define a single tradeable action *StepBack* which is any action that increases the distance between an agent and the current coin. The trading decision an agent has to make is whether to offer another agent the fixed price p in exchange for a *StepBack* action. I.e., each agent i chooses actions from: $A^i \times \prod_{j \neq i} s^j$ where A^i describes the original action space of agent i and $s^j = \{0, 1\}$ describes the binary choice to trade with any other agent j.

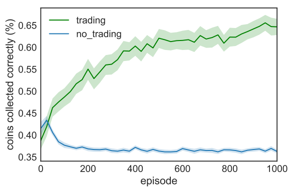
Learning Agents in the Coin Game were represented as deep Q-Networks (DQNs). During learning, exploration was encouraged by using a linear Boltzman policy, defined by: $\pi(s) = \text{argmax}_a(V_a)$, where V_a is sampled from $V_a \sim \frac{\exp(q_t(a)/\tau)}{\sum_{i=1}^{n} \exp(q_t(i)/\tau)}$ for each $a \in A$. All agents updated their policies from a stored batch of transitions $\{(s, a, r_i, s')_t : t = 1, ..., T\}$ [6]. For the Coin Game experiments, the batch size was limited to $50k$ transitions, where older transitions are discarded after inserting new transitions. The network was trained with the Adam optimization algorithm with a learning rate of $1e^{-3}$ [5]. Coin Game episodes lasted for 100 steps and after 25 episodes we logged 50 test episodes. The discount rate γ was 0.99.

Modeling trade in the Coin Game required to set a couple of trading related parameters. Firstly, the price p for an action a. In our experiments, we set $p = 1.25$ as it exceeds an agents profit from collecting a coin and is less than the designated owner of the coin would lose if the other agent collected the coin.

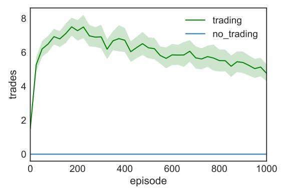
The second parameter of interest is the trading budget m i.e., the available budget until the current coin is collected. We experimented with different budgets and chose m to be 2.5 which allowed for a maximum of 2 trades when $p = 1.25$. A third critical question was whether agents should be allowed to accumulate wealth over steps or even episodes. Although this seems an interesting aspect we decided not to let agents gather their earnings and leave the analysis of such a setting for future work.



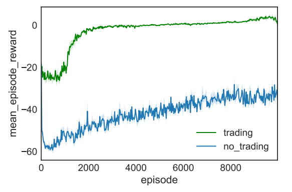
(a) Rewards



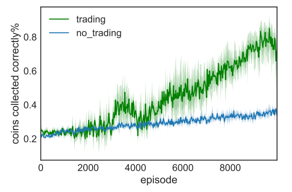
(b) Share collected coins



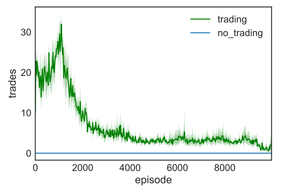
(c) Number of trades



(d) Overall reward



(e) Share collected coins



(f) Number of trades

Fig. 4: Coin Game results for 2 agents (upper row) and 4 agents (lower row). Results comprise 1000 (2 agents) and 10000 (4 agents) episodes and show mean values and confidence intervals from 80 runs for 2 agents and 10 runs for 4 agents. Each plot shows results for agents with action trading (green) and without trading (blue). Action traders show increasing individual and overall rewards (left column) along with an increasing share of correctly collected coins (middle column). The number of trades (third column) decreases after a steep rise during the early learning period (best viewed in color).

Results Figure 4 shows Coin Game results for 2 agents (upper row) and 4 agents (lower row) respectively. Experiments involve agents without trading (blue) and trading (green) for 80 runs (2 agents) and 10 runs (4 agents) where runs last for 1000 episodes (2 agents) and 10000 episodes (4 agents). Shaded areas show .95 confidence intervals. The left column shows the overall reward and the individual rewards in the 2 agents setting. While non-trading agents' reward never increases, action traders manage to increase individual and overall reward. This comes from an increasing share of correctly collected coins (middle column). The number of trades sharply increase during the first 200 episodes and continuously decrease afterwards.

6 Discussion

Action trading in the iterated matrix game outperformed pure self-fish agents. Nevertheless, prices for actions were given at design time which renders the question on the ability of agents to find prices on their own.

The results from the Coin Game clearly confirm that action trading effectively increases social welfare, measured through overall increase of reward for all agents. It also shows that a given number of available resources (coins) are allocated more efficiently as the proportion of correctly collected coins also constantly increases. This is the consequence of agents' trading activity that increases sharply at early learning phases and is kept at a high level afterwards. In learning to trade, agents realize Pareto improvements and empirically confirm the first fundamental theorem of welfare economics according to which competitive markets will tend towards Pareto efficiency. From the experiments we realized that the trading budget is a critical parameter with respect to the problem of interest which will be left for future work.

An interesting point seems the slow decrease in the number of trades. This might be caused by an agent speculating for short-term profits by not offering a trade in the hope that the other agent might be doing the expected action anyway. This could cause distrust which threatens future trades.

We recognize that trading actions in MARL presumes that a trade can be controlled, i.e., agents cannot cheat on each other by making offers which they do not hold afterwards. While this seems like a strong assumption, it appears less restrictive from a practical point of view. The only extension with respect to the environment is that agents' rewards need to include the net earnings that where realized by their trading activity. I.e., the environment adopts the role of an neutral auctioneer that matches supply and offer and returns the resulting rewards for each agent.

References

1. Foerster, J., Assael, Y.M., de Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 2137–2145 (2016)
2. Foerster, J.N., Chen, R.Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., Mordatch, I.: Learning with opponent-learning awareness. arXiv preprint arXiv:1709.04326 (2017)
3. Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative multiagent control using deep reinforcement learning. In: Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2017) (2017)
4. Hernandez-Leal, P., Kaisers, M., Baarslag, T., de Cote, E.M.: A survey of learning in multiagent environments: Dealing with non-stationarity. arXiv preprint arXiv:1707.09183 (2017)
5. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Lange, S., Gabel, T., Riedmiller, M.: Batch reinforcement learning. In: Reinforcement learning, pp. 45–73. Springer (2012)

7. Laurent, G.J., Matignon, L., Fort-Piat, L., et al.: The world of independent learners is not markovian. International Journal of Knowledge-based and Intelligent Engineering Systems 15(1), 55–64 (2011)
8. Leibo, J.Z., Zambaldi, V., Lanctot, M., Marecki, J., Graepel, T.: Multi-agent reinforcement learning in sequential social dilemmas. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 464–473. International Foundation for Autonomous Agents and Multiagent Systems (2017)
9. Lerer, A., Peysakhovich, A.: Maintaining cooperation in complex social dilemmas using deep reinforcement learning. arXiv preprint arXiv:1707.01068 (2017)
10. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the eleventh international conference on machine learning. vol. 157, pp. 157–163 (1994)
11. Malialis, K., Devlin, S., Kudenko, D.: Resource abstraction for reinforcement learning in multiagent congestion problems. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems. pp. 503–511. International Foundation for Autonomous Agents and Multiagent Systems (2016)
12. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015)
13. Osborne, M.J., Rubinstein, A.: A course in game theory. MIT press (1994)
14. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. Autonomous agents and multi-agent systems 11(3), 387–434 (2005)
15. Perolat, J., Leibo, J.Z., Zambaldi, V., Beattie, C., Tuyls, K., Graepel, T.: A multi-agent reinforcement learning model of common-pool resource appropriation. arXiv preprint arXiv:1707.06600 (2017)
16. Peysakhovich, A., Lerer, A.: Prosocial learning agents solve generalized stag hunts better than selfish ones. arXiv preprint arXiv:1709.02865 (2017)
17. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
18. Shoham, Y., Leyton-Brown, K.: Multiagent systems: Algorithmic, game-theoretic, and logical foundations. Cambridge University Press (2008)
19. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. Nature 550(7676), 354–359 (2017)
20. Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., Vicente, R.: Multiagent cooperation and competition with deep reinforcement learning. PloS one 12(4), e0172395 (2017)
21. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning. pp. 330–337 (1993)