Experimental Evaluation of a Method for Simulation based Learning for a Multi-Agent System Acting in a Physical Environment

Kun Qian, Robert W. Brehm and Lars Duggen
SDU Mechatronics, Mads Clausen Institute, University of Southern Denmark, Denmark

Keywords: Cooperative Multi-Agent Systems, Multi-Agent Reinforcement Learning, Multi-Agent Actor-Critic, Cooperative Navigation, Simulation Based Learning.

Abstract: A method for simulation based reinforcement learning (RL) for a multi-agent system acting in a physical environment is introduced, which is based on Multi-Agent Actor-Critic (MAAC) reinforcement learning. In the proposed method, avatar agents learn in a simulated model of the physical environment and the learned experience is then used by agents in the actual physical environment. The proposed concept is verified using a laboratory benchmark setup in which multiple agents, acting within the same environment, are required to coordinate their movement actions to prevent collisions. Three state-of-the-art algorithms for multi-agent reinforcement learning (MARL) are evaluated, with respect to their applicability for a predefined benchmark scenario. Based on simulations it is shown that the MAAC method is most applicable for implementation as it provides effective distributed learning and suits well to the concept of learning in simulated environments. Our experimental results, which compare simulated learning and task execution in a simulated environment with that of task execution in a physical environment demonstrate the feasibility of the proposed concept.

1 INTRODUCTION

Multi-agent systems (MASs) have been considered as one of the most promising technologies for Industry 4.0 applications (Sycara et al., 1996; Xie and Liu, 2017). In a MAS, several agents act autonomously in a shared environment, in order to follow and fulfill specific objectives (Wooldridge, 2009). RL, concerned with how agents learn by trial-and-error interaction with an environment, is closely coupled to the concept of an agent (Neto, 2005). The agent interacts with the environment by executing specific actions, which result in a state change of the environment. The agent learns by accumulated rewards, which it receives for a series of executed actions (Wooldridge, 2009; Neto, 2005; Sutton et al., 2018).

In many applications, agents act individually to achieve a given objective. However, if several agents are acting within the same environment, there is a need to cooperate, coordinate, and negotiate with one another to cope with shared resources, data, knowledge or coordination of given tasks. Integrating RL methods into a MAS has attracted increasing attention in recent years (Stone and Veloso, 2000; Busoniu et al., 2008), since the complexity of some tasks make it hard to coordinate between agents using pre-programmed agent behaviors. Applications for MARL ranges from game playing to industrial applications. In (Wang and De Silva, 2006; Yang and Gu, 2004), a multi-agent robot scenario is introduced, in which learning is required to specify optimal actions for all states that each robot might encounter. A set of MARL systems for traffic lights control is presented in (Wiering, 2000; Bakker et al., 2010), which help to optimize driving policies. A method for optimization of distributed energy resources using MARL is introduced in (Raju et al., 2015).

Formation control is one of the most popular problems considered in MASs. The aim is to form a prescribed geometrical shape in a given environment while preventing collisions with obstacles and other agents. In (Lowe et al., 2017), a cooperative navigation problem is presented, in which agents are situated in an environment and are given the objective to navigate to a set of locations. Agents must visit all locations without colliding with each other. The methods presented in (Matignon et al., 2007; Wang and De Silva, 2006; Lowe et al., 2017; Li et al., 2008; Foerster et al., 2017) can easily be applied to solve the given cooperative navigation problem. However, the validations of the proposed methods are based on simulations, a validation and evaluation of MARL and the applicability in real environments is missing. In this paper we present the evaluation and application of MARL on the bases of a laboratory benchmark setup, as shown in Figure 1.

Qian, K., Brehm, R. and Duggen, L.
Experimental Evaluation of a Method for Simulation based Learning for a Multi-Agent System Acting in a Physical Environment.
DOI: 10.5220/0007250301030109
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 103-109
ISBN: 978-989-758-350-6

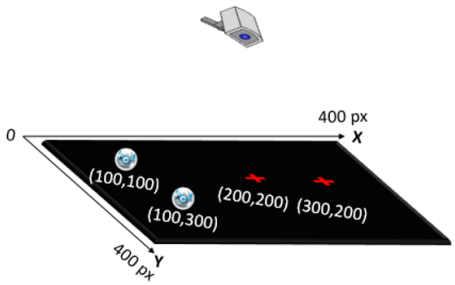

Figure 1: Laboratory benchmark model for MARL evaluation. The length of each axis is 400 pixels, which corresponds to the physical length of 1 meter.

In the laboratory benchmark setup, spherical robots are able to move within a predefined area. A camera, mounted on the ceiling, monitors the movement of the robots and continuously monitors the robots positions. The robots manoeuvring range is selected to be restricted to a specific area which is visible to the camera. There are two goal landmarks at specific locations inside the area. Further the spherical robots are originally positioned at two base stations inside the given area. The robots know the relative position of each other and also the positions of the landmarks and the base stations. In the benchmark scenario, the robots will be requested to simultaneously move to one of these goal landmarks without colliding into each other, stay at that location for a short period of time, and then move back to the base station.

In the remaining of this paper, in Section 2, an evaluation and comparison of three suitable MARL algorithms based on a simulation of a self-defined environment is given. In Section 3 the detailed setup of the laboratory benchmark setup and the design of the final MAS is introduced. For validation of the proposed simulation based learning method, in Section 4, the implementation of the MAS and the experimental results are presented. The experimental results are compared to results from the simulated environment. The paper ends with concluding remarks and suggestions for future work.

2 MULTI-AGENT REINFORCEMENT LEARNING

In RL, an agent that is situated in an environment learns which action to take for a particular environmental state in order to maximize its total received reward. The agent discovers the best actions for an environmental state, by trying them. Finite Markov decision processes (MDP) are mathematically idealized forms of RL problems. The agent perceives its environment, and after a decision, it takes an action, which leads to an environment state transition and a reward for the agent. The introduced MARL frameworks are based on the MDP. However, the difference to single-agent RL is that actions of other agents will have an effect on the environment as well. This leads to a non-deterministic interaction of an agent with the environment it acts in. Following the assumptions for MASs as stated in (Poole and Mackworth, 2017), the exsiting approaches integrate developments in the areas of single agent RL, game theory, and direct policy search techniques (Busoniu et al., 2008).

In (Matignon et al., 2007), a comparison of basic Q-learning algorithms is presented. Centralized Q-learning shows good performance but there is a high information demand and a larger state-action space to be maintained. In decentralized Q-learning the state-action space is reduced. Noticeably, an agent can get punished even if it takes a correct action. The reason for this is that other agents may take wrong actions and the joint action then leads to punishment. This can be avoided by distributed Q-learning method, which restricts Q-values to only increment. A key issue with distributed Q-learning is that it does not guarantee to convergence to the optimal joint policy in difficult coordination scenarios. For this reason, hysteretic Q-learning has been proposed (Matignon et al., 2007). This learning method is decentralized in the sense that each agent builds its own Q-table whose size is independent of the number of agents in the environment and a linear function of its own actions. According to (Matignon et al., 2007), the performance of hysteretic Q-learning is similar to centralized algorithms while much smaller Q-value tables are used.

Apart from adapting Q-learning to multi-agent scenarios, policy gradient based methods have also been applied, especially the actor-critic method (Lowe et al., 2017; Li et al., 2008; Foerster et al., 2017). To ease training, a framework based on centralized training with decentralized execution is applied. The critic is based on extra information, such as the policies of other agents, while the actor only uses the local observations to choose actions. In a fully cooperative environment, there is only one critic for all actors since all always have the same reward. However, in a mixed cooperative-competitive environment, there is one critic for each actor.

In the remainder of this section, centralized Q-learning, hysteretic Q-learning and the MAAC method with linear function approximation will be introduced. Further, these three methods are evaluated with respect to applicability in the introduced laboratory benchmark setup.

2.1 Multi-Agent Task Description

Figure 2 shows the simulated environment used for the evaluation of the MARL methods. Similar to the laboratory benchmark setup, two robot agents are initially positioned at two base stations, which are located in x/y-direction at $(7,7)$ and $(7,21)$. Further, there are two goal locations at $(14,14)$ and $(21,14)$. The maneuvering area of the two agents is restricted to 30×30 units. The objective of the agents is to simultaneously move to one of the goal locations without colliding with the other agent. The current position of each agent represents the environments state. When both agents are at the same location, the interaction is terminated with a reward of $r = -100$. If each agent is occupying one goal location or it is in close proximity of less then two pixels, the interaction is terminated with a reward of $r = 10$. Apart from the mentioned terminal states, the agents will get a reward of $r = -1$ each time they interact with the environment. The rewards are generally defined as positive for desired actions while negative for poor actions. The reward $r = -100$ indicates that the collision is the most undesired situation for the agents. The reward $r = -1$ means that the agents are consuming time. The reward $r = 10$ encourages the agents to reach the goal.

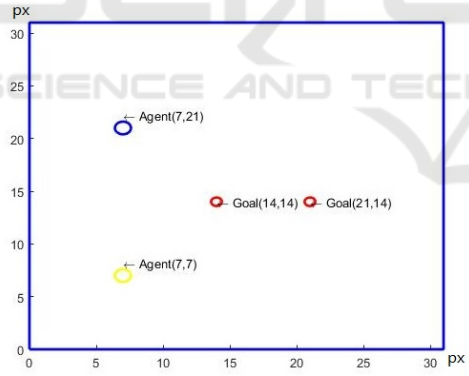

Figure 2: Benchmark MAS environment for MARL methods evaluation.

The set of possible agent actions is given by $\mathcal{A} = \{up, down, right, left\}$. These actions will take the agent into the chosen direction, except if the chosen heading will take the agent out of the restricted area, in that case, the agents position will stay unchanged.

2.2 MARL Methods Evaluation

2.2.1 Centralized Q-learning

For the introduced scenario, in the environment state s, an action consists of the two agent actions, $a = (a_1, a_2)$ with $a_1, a_2 \in \mathcal{A}$. The extracted state features are given by horizontal and vertical distances $\boldsymbol{\phi}(s) = (\phi_1, \phi_2, ..., \phi_{10})$ summarized in Table 1.

Table 1: Summery of the features for a state. Here h_d denoted the horizontal and v_d the vertical distance, respectively.

Feature	Description
ϕ_1	h_d from the 1^{st} **agent** to the 1^{st} **goal**
ϕ_2	v_d from the 1^{st} **agent** to the 1^{st} **goal**
ϕ_3	h_d from the 1^{st} **agent** to the 2^{nd} **goal**
ϕ_4	v_d from the 1^{st} **agent** to the 2^{nd} **goal**
ϕ_5	h_d from the 2^{nd} **agent** to the 1^{st} **goal**
ϕ_6	v_d from the 2^{nd} **agent** to the 1^{st} **goal**
ϕ_7	h_d from the 2^{nd} **agent** to the 2^{nd} **goal**
ϕ_8	v_d from the 2^{nd} **agent** to the 2^{nd} **goal**
ϕ_9	h_d from the 1^{st} **agent** to the 2^{nd} **agent**
ϕ_{10}	v_d from the 1^{st} **agent** to the 2^{nd} **agent**

Similar to what is mentioned in (Geramifard et al., 2013), the features for the state action pair can be re-encoded in a manner such that the 10 features for each possible a_1, a_2 combination are indexed accordingly, yielding a re-encoded state action set $\boldsymbol{\phi}(s, a_1, a_2) = (\phi_1(s), \phi_2(s), ..., \phi_{160}(s))$ of size 160 ($\boldsymbol{\phi}(s, a_1, a_2) \in \mathcal{A}^{16} \times \mathbb{R}^{10}$). From this, the action values can be approximated by:

$$q_{\boldsymbol{\omega}}(s, a_1, a_2) = \omega_0 + \sum_{i=1}^{160} \omega_i \phi_i(s, a_1, a_2) \quad (1)$$

with $\boldsymbol{\omega}$ being the parameter vector to be learned, which is updated by:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \alpha \left[r + \gamma \max_{a_1', a_2'} q_{\boldsymbol{\omega}_t}(s', a_1', a_2') - q_{\boldsymbol{\omega}_t}(s, a_1, a_2) \right] \boldsymbol{\phi}(s, a_1, a_2). \quad (2)$$

Here, s, a_1, and a_2 denote the state and actions chosen at time step t, while r, s', a_1', and a_2' are the reward, state, and available actions at time step $t+1$.

2.2.2 Hysteretic Q-learning

Hysteretic Q-learning provides decentralised RL in deterministic multi-agent environments. In a certain state of the above-defined environment, the action space for an agent is only of size 4. For the defined environment this shrinks down the size of features representing the state-action pair from 160 (feature size of centralized Q-learning) to just 40. To approximate the action values for two agents, two parameter vectors $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2$ are required, which need to be learned.

The update rule for the parameter vectors is given by:

$$\delta = r + \gamma \max_{a'} q_i(s', a') - q_i(s, a_i), \quad (3)$$

$$\boldsymbol{\omega}_{i,t+1} = \begin{cases} \boldsymbol{\omega}_{i,t} + \alpha\delta & \text{if } \delta \geq 0 \\ \boldsymbol{\omega}_{i,t} + \beta\delta & \text{else} \end{cases}. \quad (4)$$

Here, i denotes the index of the agent, with $a_i, a' \in \mathcal{A}$ and α, β are the increase and decrease rate for the parameter updates.

2.2.3 Multi-Agent Actor-Critic

In the MAAC method a centralized critic is learned to critique the actors. The critic approximates the values for a state with a parameter vector $\boldsymbol{\omega}$ based on feature vector $\widetilde{\boldsymbol{\phi}}(s) = (\widetilde{\phi}_1, \widetilde{\phi}_2, \widetilde{\phi}_3)$. With $\widetilde{\phi}_1$ being the Euclidean distance from the first agent to a goal, $\widetilde{\phi}_2$ being the Euclidean distance from the second agent to the other goal and $\widetilde{\phi}_3$ being the Euclidean distance from the first agent to the second agent. Thus, the state value $v(s)$ is approximated by:

$$v_\omega(s) = \omega_0 \cdot 1 + \omega_1 \widetilde{\phi}_1(s) + \omega_2 \widetilde{\phi}_2(s) + \omega_3 \widetilde{\phi}_3(s). \quad (5)$$

To calculate the policy $\pi_i(a_i \mid s)$ for each actor, parameterized numerical preferences $h_i(s, a_i, \boldsymbol{\theta}_i)$ need to be formed for each state-action pair:

$$h_i(s, a_i, \boldsymbol{\theta}) = \boldsymbol{\theta}_i^\top \boldsymbol{\phi}(s, a_i). \quad (6)$$

Here, i denotes the index of the agent, with $a_i \in \mathcal{A}$. $\boldsymbol{\theta}_i$ is the parameter vector for parametrising the policy and $\boldsymbol{\phi}(s, a_i)$ is found based on the features listed in Table 1. For the interaction between agents and environment the parameter vectors are updated using:

$$\delta_t = r + \gamma v_{\boldsymbol{\omega}_t}(s') - v_{\boldsymbol{\omega}_t}(s), \quad (7)$$

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \alpha_\omega \delta_t \widetilde{\boldsymbol{\phi}}(s), \quad (8)$$

$$\boldsymbol{\theta}_{i,t+1} = \boldsymbol{\theta}_{i,t} + \alpha_\theta \delta_t \nabla_{\boldsymbol{\theta}_i} \ln \pi(a_i \mid s, \boldsymbol{\theta}_i). \quad (9)$$

Here, s and a_i are the state and action chosen at time step t, while r and s' are the reward and state at time step $t+1$.

2.2.4 Methods Evaluation

For a comparative evaluation of these three methods, each is simulated for a 1000 episodes, consisting of the steps from initial position to a terminal state, and the sum of rewards for each episode is collected. The hyperparameters used for the simulations are shown in Table 2.

The mean value and standard deviation for every 100 episodes is shown in Figure 3, with the error bar showing the standard deviations. It can be observed that in centralized Q-learning, the parameters are learned more slowly, due to the larger action space. Futher, Hysteretic Q-learning shows good performance while MAAC can solve the task with best performance. In centralized Q-learning the action space grows exponentially with the number of agents. In compliance with the concept of decentralized decision making in a MAS, hysteretic Q-learning and MAAC are more applicable. However, in MAAC at least one centralized critic is needed, which means, during the execution process, the parameters will not be able to be updated. And this will only work if the dynamics of the environment are stable. In the coordinated multi-agent scenario as studied herein, the dynamics will not change. Therefore, it is proposed to use a concept in which avatar agents learn in a simulated model of the given environment using the MAAC method with a central critic. The learned parameters are then to be used by the real agents in a real environment. The feasibility of this concept will be proven based on the experimental benchmark setup, which is described in the following section.

Table 2: Hyperparameters for the above three methods.

Methods	Hyperparameters
Centralized Q-learning	Learning rate α = 0.05, discount factor γ = 0.99, ε-soft policy with ε decreasing slowly from 0.8 to 0.1
Hysteretic Q-learning	Increasing rate α = 0.05, decreasing rate β = 0.005, discount factor γ = 0.99, ε-soft policy with ε decreasing slowly from 0.8 to 0.1
MAAC	Learning rate $\alpha_\omega = 0.0025$, $\alpha_\theta = 0.05$, discount factor $\gamma = 0.99$

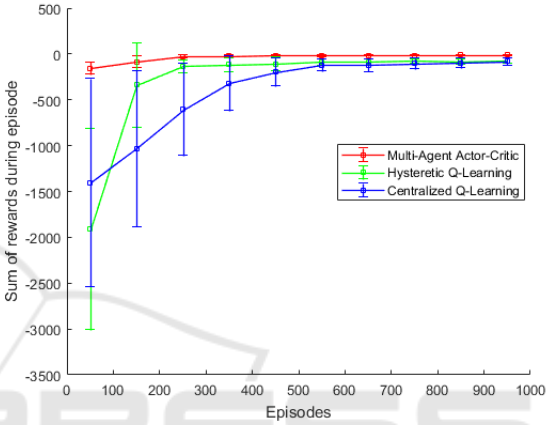

Figure 3: Mean values and standard deviations of rewards.

3 EXPERIMENTAL BENCHMARK SETUP

3.1 Multi-Agent System Architecture

The JADE framework is used for the implementation of the MAS in the benchmark setup. Agents communication is based on FIPA compliant messaging. There are two ways of designing the software architecture: coupled design and embedded design. The coupled design is currently popular in automation scenarios, while the embedded design promotes the decoupling of agents logically and geographically, effectively enabling the creation of plug-and-produce entities comprising the artifact being controlled, the controller, and the agent (Leitão and Karnouskos, 2015). In the implementation of the benchmark scenario, an embedded design is used. Here, the agents and the low-level control system are situated on the same embedded platform. Each robot agent is deployed on a single board computer (SBC), which controls a spherical robot (Sphero) using a wireless Bluetooth link. A simplified architectural diagram for used MAS is shown in Figure 4. The current position of the

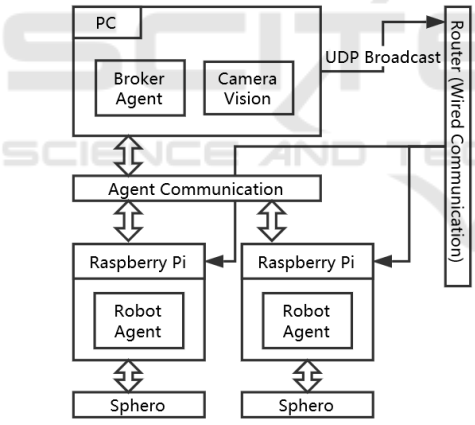

Figure 4: Structural diagram.

Spheros is monitored using a camera mounted on the ceiling. The video stream is send to a computer vision program on a central server which detects the current position of the individual *Spheros*. The server streams (via UDP broadcasting) the individual positions to the robot agents, which are executed on the SBC. Based on the experimental evaluation, MAAC will be used for the benchmark scenario. And as motivated in Section 2.2.4, the goal is to demonstrate the concept of learning in a simulated, and execution in a physical environment.

3.2 Benchmark Scenario

The benchmark scenario consists of three agents, two robot agents and a broker agent. The broker agent is able to send FIPA compliant requests to the robot agents. This requests the robot agents to move to a goal location (e.g. to pick something up) and return to the base station.

Two simplified models of the real environment are used to train the agents. The first model is used to train the agents to move from the base stations to the goal location, while the second model is used to train the agents to move from the goal locations back to the base stations. In the model, as for the physical environment as shown in Figure 1, the two agents are restricted to move only within an area of the size of 400×400 pixels. There are two base stations which are located in x/y-direction at $(100, 100)$ and $(100, 300)$ while there are two landmarks located at $(200, 200)$ and $(300, 200)$. Terminal states with a reward of $r = -1000$ are defined around these locations if an agent is within a circle, of a radius of 10 pixels, around the other agent. A reward of $r = 100$ is returned, and the interaction is terminated, if each agent occupies each target location with a distance of less than 5 pixels. If an action takes the agent out of the restricted area, the position will stay unchanged. The parameter vectors for policies which are obtained by the agent avatars during the training in the simulated environment will be used for the real agents on the SBC, which control the *Sphero* robots in the physical environment.

3.3 Experimental Results

Experiments to prove the concept of learning in a simulated environment and usage of the learned experience for tasks in a physical environment have been carried out. The presented and discussed results are based on four episodes in which, simultaneously, two agents are requested to each move to one goal location, as described in Section 3.2.

Shown in Figure 5 and 6 are the traces of the two robots as they simultaneously move through the benchmark environment. For each executed episode, the traces of robots are identified by two different colors as denoted in the legend of the figures.

Figure 5 shows the movement of the robots based on learning in a simulated environment and also execution of the task in the simulated environment. In contrast, shown in Figure 6, are four episodes for the movement of the real robots in the real environment based on learning in the simulated environment. Comparing the results shown in Figure 5 and 6, it can

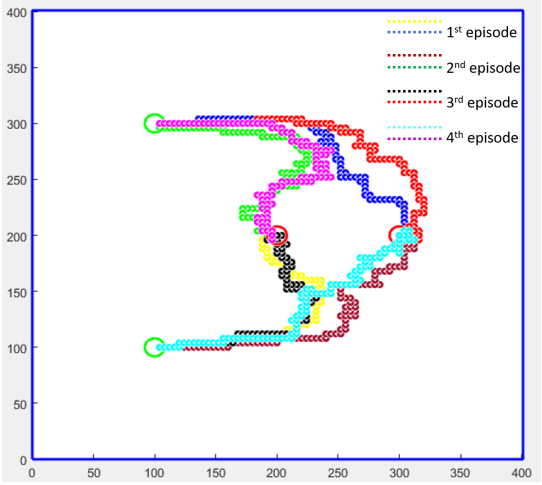

Figure 5: Simulation in MATLAB.

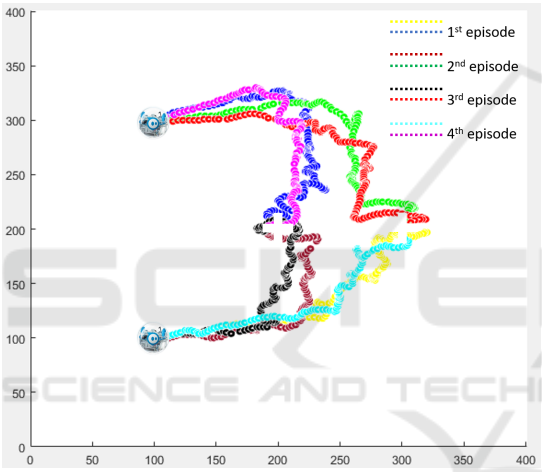

Figure 6: Experimental result.

be observed that for both cases agents coordinate with each other and move to one goal landmark each, without colliding with each other. The Figure is in fact representative of approximately 20 episodes, in which no collision has been observed. Though, it can also be observed that the paths are not always optimal since it is hard to perceive the real environment and control the agents precisely. For clarity, only four episodes are shown. From this results it is evidenced that using learned parameters from a simulated environment in a real environment is applicable for the benchmark setup. Further if agents reach their goal landmarks, they are each able to move back to a base station without collision, for clarity this return path is not shown in the presented results.

4 DISCUSSION AND FUTURE WORK

An introduction to applications for RL in MAS has been given, with a focus on formation control and coordinated movement of robots in a shared two dimensional environment like a factory floor. A benchmark scenario, in which two agents are requested to simultaneously pick up a virtual good at two goal landmark locations and deliver this to two base stations has been introduced. In order to prevent collisions between agents, while simultaneously moving through the environment, MARL has been used. Suitable algorithms for MARL have been evaluated on the basis of a predefined scenario. For effective and fast learning it has been proposed to use avatar agents which learn in a simulated environment while the learned parameters are then used by agents in the real environment. It has been shown, based on simulations of the given benchmark scenario that, by using the MAAC method agents learn effectively and that MAAC is applicable for the given benchmark scenario. The laboratory benchmark setup to implement the given scenario and the agent architecture has been introduced. With the experiment conducted and on the basis of the presented results, it is evidenced that the MAAC method, in combination with learning in simulated environment and usage of the learned parameters in a real environment, is well applicable for the given use case scenario.

Since real environments are not always static in time, as in our example, further studies have to be carried out in order to investigate the applicability of avatar agents for learning. Nevertheless our results stipulate the feasibility of the MAAC method for these general scenarios.

REFERENCES

Bakker, B., Whiteson, S., Kester, L., and Groen, F. C. a. (2010). Traffic light control by multiagent reinforcement learning systems. *Interactive Collaborative Information Systems*, pages 475–510.

Busoniu, L., Babuska, R., De Schutter, B., and Schutter, B. D. (2008). A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2017). Counterfactual Multi-Agent Policy Gradients. *ArXiv e-prints*.

Geramifard, A., Walsh, T. J., Tellex, S., Chowdhary, G., Roy, N., and How, J. P. (2013). A tutorial on linear function approximators for dynamic programming

and reinforcement learning. *Foundations and Trends in Machine Learning*, 6(4):375–451.

Leitão, P. and Karnouskos, S. (2015). *Industrial Agents: Emerging Applications of Software Agents in Industry*.

Li, C.-G., Wang, M., and Yuan, Q.-N. (2008). A Multi-agent Reinforcement Learning using Actor-Critic methods. In *2008 International Conference on Machine Learning and Cybernetics*, volume 2, pages 878–882.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*.

Matignon, L., Laurent, G. J., and Le Fort-Piat, N. (2007). Hysteretic Q-Learning : An algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *IEEE International Conference on Intelligent Robots and Systems*, pages 64–69.

Neto, G. (2005). From single-agent to multi-agent reinforcement learning: Foundational concepts and methods learning theory course.

Poole, D. L. and Mackworth, A. K. (2017). *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, New York, NY, USA, 2nd edition.

Raju, L., Sankar, S., and Milton, R. S. (2015). Distributed optimization of solar micro-grid using multi agent reinforcement learning. In *Procedia Computer Science*, volume 46, pages 231–239.

Stone, P. and Veloso, M. (2000). Multiagent systems: a survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383.

Sutton, R. S., Barto, A. G., and Klopf, H. (2018). *Reinforcement Learning : An Introduction Second edition*. MIT Press, 2nd edition.

Sycara, K., Pannu, A., Williamson, M., Zeng, D., and Decker, K. (1996). Distributed intelligent agents. *IEEE Expert-Intelligent Systems and their Applications*, 11(6):36–46.

Wang, Y. and De Silva, C. W. (2006). Multi-robot box-pushing: Single-agent Q-learning vs. team Q-learning. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3694–3699.

Wiering, M. A. (2000). Multi-agent reinforcement learning for traffic light control. *In Machine Learning: Proceedings of the Seventeenth International Conference*, pages pp. 1151–1158.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems [Paperback]*.

Xie, J. and Liu, C.-C. (2017). Multi-agent systems and their applications. *Journal of International Council on Electrical Engineering*, 7(1):188–197.

Yang, E. and Gu, D. (2004). Multiagent reinforcement learning for multi-robot systems: A survey. *University of Essex Technical Report CSM-404, ...*, pages 1–23.