Application of Multi-Agent Reinforcement Learning Techniques in Sequential Games

Vanya Markova, Ventseslav Shopov

Institute of Robotics
Bulgarian Academy of Sciences
Plovdiv
Bulgaria
Email: markovavanya@yahoo.com, vkshopov@yahoo.com

***Abstract*—This article focuses on the application of multi-agent deep reinforcement learning techniques in sequential games. The main hypothesis is that deep reinforcement learning and collective behaviour approach demonstrate better performance than classic reinforcement learning. So autonomous agents are capable of discovering good solutions to the problem at hand by cooperate with other learners.**

***Index Terms*—autonomous agents; deep reinforcement learning; multi-agent systems; collective behaviour**

I. Introduction

We employ deep multi-agent reinforcement learning to model the emergence of cooperation. The new notion of sequential social dilemmas allows us to model how rational agents interact, and arrive at more or less cooperative behaviours depending on the nature of the environment and the agents cognitive capacity. The research may enable us to better understand and control the behaviour of complex multi-agent systems.

The mathematical framework for question of unsupervised learning and autonomous decision making: is Reinforcement Learning (RL). In RL, a software agent interacts with an environment and occasionally perceives rewards to learn an optimal behavioural policy. Deep reinforcement learning combines established reinforcement learning techniques with the ability of deep neural networks to capture structure in complex environments and generalize over large state spaces. While deep reinforcement learning is still in its infancy, researchers and practitioners are rapidly exploring new applications. We want to develop this framework in sequential decision making in game theory. The aim of this paper is to apply this techniques of Multi-Agent Reinforcement Learning (MARL) and Multi-Agent Deep Reinforcement Learning (MADRL) and evaluate results of process of learning in sequential games. In real life, cooperating require complex behaviours, involving difficult sequences of actions that agents need to learn to execute by deep multi-agent reinforcement learning [1] [2].

Since the work on learning ATARI TV games by Google DeepMind [3], end-to-end reinforcement learning or deep reinforcement learning is garnering attention. This approach extends reinforcement learning to the entire process from sensors to motors by forming it using an artificial neural network especially a deep network without designing state space or action space explicitly [4]–[6].

Pursuit-evasion is a problem area in computer science in which one group of agents attempts to catch members of another group in an environment [7]. The reinforcement learning techniques have been used in some of recent studies in the field of pursuit-evasion games [8] [7]. A survey of actor-critic reinforcement learning is given in [9].

Sequential decision making under uncertainty is always a challenge for autonomous agents populating a multi-agent environment, since their behaviour is inevitably influenced by the behaviour of others. Further, agents have to constantly struggle to find the right balance between exploiting current information regarding the environment and the rest of its inhabitants, and exploring so that they acquire additional information. Moreover, they need to profitably trade off short-term rewards with anticipated long-term ones, while learning through interaction about the environment and others, employing techniques from RL, a fundamental area of study within Artificial Intelligence (AI) [10]–[13].

Coalition formation is a problem of great interest within game theory and AI, allowing autonomous individually rational agents to form stable or transient teams (or coalitions) to tackle an underlying task. Agents participating in realistic scenarios of repeated coalition formation need to successfully negotiate the terms of their participation in coalitions often having to compromise individual with team welfare effectively [14].

In our work, we assume that the environment dynamics or the types (capabilities) of other agents are not known, and thus the agents have to account for this uncertainty, in a Bayesian way [2], when making decisions. Handling type uncertainty allows information about others acquired within one setting to be exploited in possibly different settings in the future. The core of our contributions lies in the area of coalition formation under uncertainty. We studied several aspects of both the cooperative and non-cooperative facets of this problem, coining new theoretical concepts, proving theoretical results, presenting and evaluating algorithms for use in this context, and proposing a Bayesian RL framework for repeated coalition formation under uncertainty.

An essential quality of a cognitive being is its ability to learn, that is, to gain new knowledge or skills, as well as to improve existing knowledge or skills based on experience. Cognitive beings are able to cope with situations they have been previously confronted with as well as they are able to adapt to new situations sensibly. Thus, when designing an artificial agent that shall exhibit cognitive qualitieswhich we refer to in short as a cognitive agentone central task is to develop computational means that enable the agent to learn. In this paper, we subscribe to one of the most influential paradigms of machine learning, reinforcement learning (RL) [15]. Reinforcement learning is very valuable when the characteristics of the underlying system are not known and/or difficult to describe or when the environment of an acting agent is only partially known or completely unknown.

In recent years, there are many application of reinforcement learning techniques in multi-agent systems. There are a representative selection of algorithms for the different areas of multi-agent reinforcement learning research [2] [16].

Agents situated in the real world can perceive a great variety of information. Two situations are unlikely to lead to the same perception even if the situations are very similar. Learning requires the agent to acknowledge important details in a perception, to recognise commonalities across situations, and, most importantly, to disregard irrelevant information. In other words, the ability to learn involves the ability to abstract. Abstraction enables us to conceptualise the surrounding world, to build categories, and to derive reactions from these categories that can adapt to different situations. Complex and overly detailed circumstances can be reduced to much simpler concepts and not until then it becomes feasible to deliberate about conclusions to draw and actions to take. Abstract concepts explicate commonalities in different scenes and, thus, can cover new situations [17]–[21].

The question of how the solution, inferred by RL can be used to enable the cognitive agent to adapt the behavior to new tasks or similar situations is a great challenge in engineering machine learning. For this kind of reuse of knowledge, the term transfer training has become popular [22].

The main goal of this paper is to apply transfer learning trough reinforcement learning in area of building autonomous agent behaviour. In the second part of this paper we will briefly describe the underlying theory of Autonomous Agent, Markov Decision Process, Reinforcement Learning, Transfer Learning, Deep Learning and Sequential Games. In addition, we will describe the implementation of our approach. In the third part of our study we describe the experiments and gathers evidence to support our hypothesis.

II. Methods and Materials

A. Theory

1) Autonomous Agent behaviour: The information about past and current states of the agent and environment allow the agents to estimate its own progress. Moreover, this information allow the agent to make the corrections in existing pans if any needed and even to ma make new plans if it is necessary.

However if the agent makes corrections in the existing plans too often then this could lead to poor overall performance. Moreover, the frequently dropping and building plans could make the things even worse. Hence, it is desirable to reduce (or completely avoid) situations in which the agent should changes its mind. There are two main approaches to do that: the first is to make the changing of the plans less recourse consuming task and the e second is to make such plans that are able to deal with volatile environment behaviour. This material is concerning the second approach.

As a key issue in building more efficient plans in rapidly changing environment we can point the ability of the agent to makes its plans in accordance not only with past and current states of the environment but also bearing in mind the future. To do that the agent needs to predict or forecast the future states of the environment. So, if we describe the states of the agent and environment as a time series then the task of making efficient plans will be significantly aided if the agent could forecasts the future with desirable accuracy.

An n-tipple (vector) is a result of one cycle of a work of the agent. It consists of the parameters of the behaviour of the agent: $b(b1, b2, \ldots, bn)$. The data from environment are collected and transformed into time series in the knowledge base of the agent.

2) Markov Decision Process: We formulate the transfer learning problem in sequential decision making domains using the following framework of Markov Decision Process. We use the following definition of MDP as a 5-tuple

$$< S, A, P, R, \gamma > \tag{1}$$

where the set of states, set of actions, transition function and reward function are described. And

$$P : S \times A \rightarrow \Pi(S) \tag{2}$$

is a transition function that maps the probability of moving to a new state given an action and the current state,

$$R : S \times A \rightarrow R \tag{3}$$

is a reward function. that gives the immediate reward of taking an action in a state.

And

$$\gamma \in [0, 1) \tag{4}$$

is the discount factor.

So, the MDP of the agent is described in (1), where S is the set of states, A is the set of actions, P is transition function and R is a reward function. The transition function P maps the the probability of moving to a new state given an action and the current states is shown in (2). The reward functions R that gives the immediate reward of taking an action is described in (3). An the discount factor γ is bounded as is shown in (4).

3) Reinforcement learning: Reinforcement learning [15] is a popular and effective method to solve an MDP.

In our work we implement the reinforcement learning algorithm Q-learning as is described in [23]. At each moment of time, the agent is in a given state $s \in S$, and the agents view is represented by a feature vector. Upon this information the agent makes the decision which action a from a set of all possible actions A to take in order to reach its goal. The outcome of Q-learning is a Q-function

$$Q(s,a) \tag{5}$$

that attaches to any state-action tuple (s,a) the expected reward over time. We discuss here the overall reward when starting in s and executing action a. From that Q-function, one can derive the policy π by always choosing the action with the highest Q-value:

$$\pi(s) = \operatorname*{argmax}_{a \in A}(Q(s,a)) \tag{6}$$

Under these conditions, Q-learning should converge to an optimal Q-function

$$Q* = \pi(s) = \operatorname*{argmax}_{a \in A}(Q(s;a)) \tag{7}$$

that returns the highest reward for any state-action tuple (s,a). Hence, in this way we establish an optimal policy $\pi*$.

4) Transfer learning: Machine learning and data mining techniques have been used in numerous real-world applications. An assumption of traditional machine learning methodologies is the training data and testing data are taken from the same domain, such that the input feature space and data distribution characteristics are the same. However, in some real-world machine learning scenarios, this assumption does not hold. There are cases where training data is expensive or difficult to collect. Therefore, there is a need to create high-performance learners trained with more easily obtained data from different domains. This methodology is referred to as transfer learning [24].

There is a hierarchical Bayesian framework for transfer in sequential decision making tasks of transferring two basic kinds of knowledge [1] [2].

In our paper, we uses meta-data (e.g., attribute-value pairs) associated with each task to learn the expected benefit of transfer given a source-target task pair. An example of such a metadata is given in [25].

5) Deep Learning: In reinforcement learning, an agent interacting with its environment is attempting to learn an optimal control policy. At each time step, the agent observes a state s, chooses an action a, receives a reward r, and transitions to a new state s'. Q-Learning is an approach to incrementally estimate the utility values of executing an action from a given state by continuously updating the Q-values using the following rule:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{8}$$

Where $Q(s;a)$ denotes the utility of taking action a from state s. Q-learning can be directly extended to deep reinforcement learning by using a deep neural network function approximator $Q(s,a|\theta)$ for the Q-values, where are the weights of the neural network that parametrize the Q-values. We update the neural network weights by minimizing the loss function:

$$\mathcal{L}(s,a|\theta_i) = (r + \gamma \operatorname*{argmax}_{a} Q(s',a|\theta_i) - Q(s,a|\theta_i))^2. \tag{9}$$

The backpropogation algorithm is used to update the network weights at iteration $i+1$ by performing the computation: $\theta_{i+1} = \theta_i + \alpha\nabla_\theta \mathcal{L}(\theta_i)$.

In this work, the stochastic optimization method ADAM is applied. Deep neural networks are used to approximate the value function. In addition, using a target Q-network to calculate the loss function significantly accelerates convergence. The experience replay dataset contains a fixed number of transition tuples in it that contain $(s;a;r;s')$ where r is the reward obtained by performing action a from state s, and s' is the state the agent transitions to after performing that action.

The experience tuples are sampled in mini batches, and are used to update the network. The experience replay have to prevent correlation between training samples, and helps to improve convergence. The target network in the loss function $Q(s,a|\theta)$ is kept fixed for a certain number of iterations and is updated periodically.

We outline outline an extenion of multi-agent deep reinforcement learning (MADRL) approach presented in [26]. We identify three major MADRL-related challenges and offer three solutions that make this approach possible.

The first challenge is to present the problem in such a way that it is possible to develop an effective implementation. In other words, the problem is to present the problem in such way, that it can be used by any number of agents without changing the deep Q-network architecture. To solve this problem, there have to be imposed several assumptions: time and space are discrete quantities, the agent's agent is 2D and the agents are divided into two groups of pursuers and evaders. Moreover, two types of agents are two competing groups (competing groups of agents).

These assumptions allow us to present the state of the global system as an image-like tensor. So that each image channel contains an agent and environmental information. This presentation allows us to take advantage of the convolutional neural networks that are proven to work well for image processing tasks [26].

6) Sequential Games: By highlighting some of the important issues introduced by learning in a multilingual environment, the traditional framework of game theory can not represent the complete complexity of learning with multiple agents. An important part of the problem is to make consistent decisions in a state of transition. This aspect can not be described by standard normal-form games, as they allow only stationary, stochastic features that depend solely on the actions of the players. That is why we are now looking at an expanded framework that summarizes both sequential games and MDP.

Introducing multiple agents in MDP significantly complicates the problem. Both rewards and transitions in the environment now depend on the actions of all agents that are present in the system. That is why all agents need to learn in a space for joint action. Moreover, as agents may have different goals, there may not exist an optimal solution that maximizes the rewards for all agents at the same time.

As the name suggests, Markov's game still implies that state transitions are branded, but both the probabilty of transition and the anticipated rewards now depend on the joint action of all agents. Markov's ganes can be seen as an extension of MDP to the case of many agents, as well as repetitive games to multiple state case.

B. Implementation:

We generate a discrete map with predefined dimensions. Then randomly put obstacles on the map. The next stage generates two lists: one with the pursuers and one with the prey. We study the impact of the number of pursuers and booty on the speed of reinforcement learning. We also investigate the impact of the number of obstacles on the speed of learning. And also we study the impact of magnitude of reward on non catching moves of pursuers on the speed of training.

In our case group of predator pursue a group of preys (intruders). So, as in a classic Pursue-evasion process we study our problem as a MDP task. The all members of either group act after all members of the other group have made their moves. So, we could describe our approach as a classic sequential game. Both pursuers and evader have a short range of view so they have to move continuously.

We define a stochastic behaviour of both of the groups imposing some additional rules. With a small probability α_{evader} will miss opportunity to move out and will give some handicap to the pursuer. From the other hand the pursuer with small probability $\alpha_{pursuer}$ will lose the evader from site and thus will give a chance to evader to evade.

In general, predators have a small negative reward for every empty step and the prey have small positive reward for every evasion. If a pursuer catch a prey the its reward increases considerably (at almost two orders of magnitude) and the prey's reward will be reduced by the same amount.

The groups are implemented by two lists: one for the predators and another one for the evaders. And a new prey is generated in random place on the map but out of sight of the pursuers. In our implementation, we claim that if the values of the use of MADRL will superapss MARL approach in mater of maximum reward. So, we will reach an optimal policy for a final number of epochs (steps) faster.

In order to speed up the training, it is good that the coefficients of the P matrix are somewhat closer to the desired policy. This can be achieved through a TL in a simpler environment (or just a part of the environment). The classic reinforcement learning consists of finding an optimal policy for the whole area with high details.

Our approach is based on following: In our case, group of predator pursue a group of preys (intruders).

- Loading the whole map and scraping all details but geometric obstacles
- Find a reinforcement learning solution for this plain map
- Use the MADRL and MARL to train both groups
- Load full map and use learned knowledge to study the impact of chosen factors in learning speed

Notation and transfer learning: Let G be the set of all possible tasks. Let $G_{source} \subset G$ be a set of source tasks for which the pursers and evaders has already learned a policy and let $G_{target} \subset G$ be another set of target tasks that have to be learned by the agents. For each task

$$g_i \in G, letD_i inR^n \tag{10}$$

is a descriptor of features for the given agent(either pursuer or prey). We assume that g_i and D_i that are known to the all agents.

So, we define a target task $g_j \in G_{target}$, as the goal of the agents. In both groups this should lead to higher summary rewards.

We assume that for each pair of tasks (g_i, g_j) such that $g_i, g_j \in G_{source}$, the agents could reliable estimate $f_u(g_i, g_j)$. E.g. the pursuer "catch" a prey and respectively the evader "evades". So both groups of agents can use these similar policies estimates to predict the expected transfer benefit between tasks in G_{source} and tasks in G_{target}.

III. Experiments and results

We gather evidence to support the hypothesis that using the Deep Learning will significantly speed up the training process of Multi Agent Reinforcement Learning. Hence we claim that building of Multi Agent Deep Reinforcement Learning for Autonomous Agent behaviour building is more efficient that applying the MARL direct approach.

We perform the following experiment: for a given map we should find an optimal autonomous agent group behaviour. The map is described by its size nxn and complexity rate R_c.

We have two method: Multi Agent Reinforcement Learning and Multi Agent Deep Reinforcement Learning. And four cases:

- case study I - Study the impact of the number of pursuers and booty on the speed of reinforcement learning
- case study II - The impact of the number of obstacles on the speed of learning
- case study III -Impact of magnitude of reward on non catching moves of pursuers on the speed of training
- case study IV - The impact of the map size on the speed of learning

We study following algorithms:

- case I - Multi Agent Reinforcement Learning (MARL)
- case II - Multi Agent Deep Reinforcement Learning (MADRL)

We do the following task: for a given map we need to find optimal behaviour of pursuers. The agent's task is to travel on a chased the maximum preys for given amount of time. The environment is represented as a two-dimensional obstacle

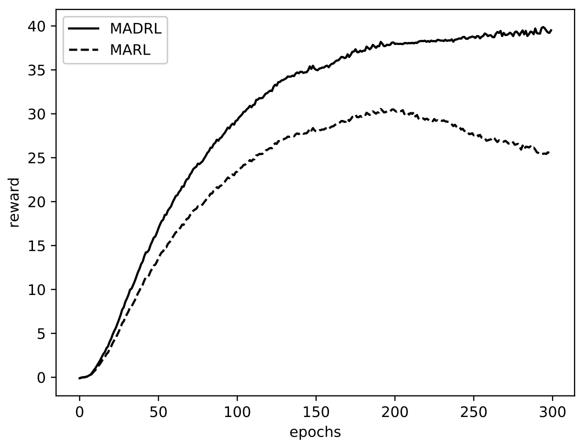

Fig. 1. We study the impact of the number of pursuers and booty on the speed of reinforcement learning.

map. The map is described by its size nxn and the rate of complexity R_c.

From Figure 1 one can see that when nuber of pursuers and preys are at the same magnitude then MADRL has a better performance than MARL.

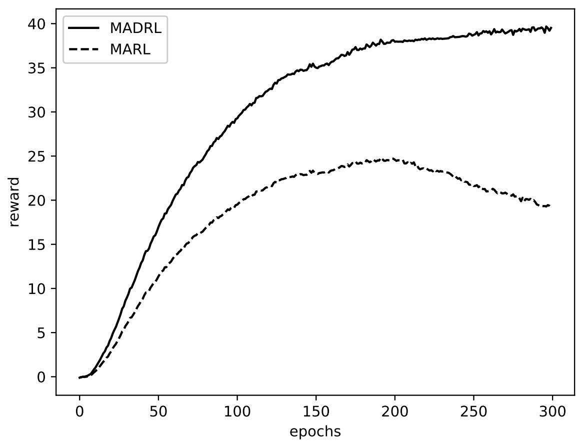

Fig. 2. We study the impact of the number of obstacles on the speed of learning.

hen the number of pursuers and prey is roughly the same then the MADRL is significantly better than the direct MARL. Moreover, with the rise in the number of ages, the quality of the gauze is significantly reduced while the deep approach is weakly affected by re-education. On a map similar to the first case, but with twice as many obstacles 1, the deep approach keeps practically the same total payout, while the direct approach has a considerably lower rewards. In the third case, there is a significant increase in the dispersion of the maximum reward MARL, while the deep training has a relatively stable dispersion.

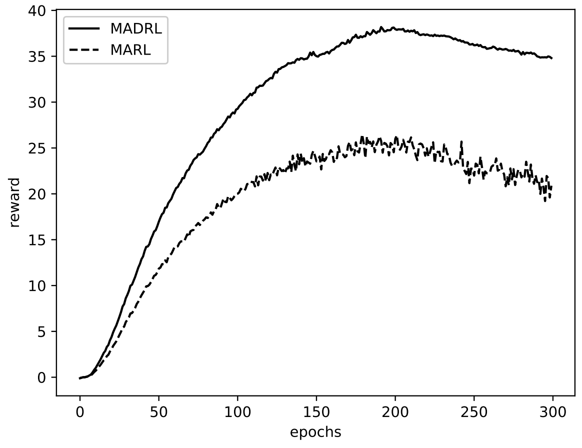

Fig. 3. We study impact of magnitude of reward on non catching moves of pursuers on the speed of training.

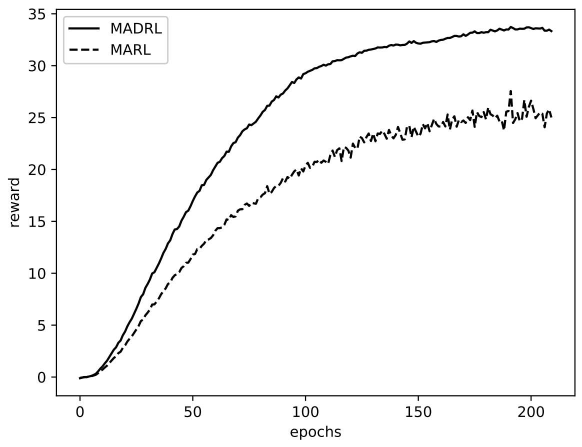

Fig. 4. We study the impact of the map size on the speed of learning.

From the Figure 4, it can be seen that for smaller maps the two approaches have a more productive performance.

IV. Conclusion and future work

The impact of different factors for building of Multi Agent behaviour is discussed in this paper. Two different approaches are presented: Multi Agent Reinforcement Learning and Multi Agent Deep Reinforcement Learning. The impact of four factors on Reinforcement Learning performance has studied. The summary reward is used as a measure of performance. In all case studies the Multi Agent Deep Reinforcement Learning demonstrate significantly better performance than Multi Agent Reinforcement Learning.

References

[1] A. Wilson, A. Fern, and P. Tadepalli, "Transfer learning in sequential decision problems: A hierarchical bayesian approach," in Proceedings of ICML Workshop on Unsupervised and Transfer Learning, 2012, pp. 217–227.

[2] A. Wilson, A. Fern, S. Ray, and P. Tadepalli, "Multi-task reinforcement learning: a hierarchical bayesian approach," in Proceedings of the 24th international conference on Machine learning. ACM, 2007, pp. 1015–1022.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, 2015, p. 529.

[4] Y. Li, "Deep reinforcement learning: An overview," arXiv preprint arXiv:1701.07274, 2017.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971, 2015.

[6] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in Advances in neural information processing systems, 2016, pp. 3675–3683.

[7] R. B. Borie, C. A. Tovey, and S. Koenig, "Algorithms and complexity results for pursuit-evasion problems." in IJCAI, vol. 9, 2009, pp. 59–66.

[8] S. Barrett, P. Stone, and S. Kraus, "Empirical evaluation of ad hoc teamwork in the pursuit domain," in The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 567–574.

[9] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 6, 2012, pp. 1291–1307.

[10] C. Dimitrakakis, F. Jarboui, D. Parkes, and L. Seeman, "Multi-view Sequential Games: The Helper-Agent Problem," Feb. 2017, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01408294

[11] P. J. Gmytrasiewicz and P. Doshi, "A framework for sequential planning in multi-agent settings." J. Artif. Intell. Res.(JAIR), vol. 24, 2005, pp. 49–79.

[12] P. Doshi and P. J. Gmytrasiewicz, "A framework for sequential planning in multi-agent settings," arXiv preprint arXiv:1109.2135, 2011.

[13] R. Smorodinsky and M. Tennenholtz, "Sequential information elicitation in multi-agent systems," in Proceedings of the 20th conference on Uncertainty in artificial intelligence. AUAI Press, 2004, pp. 528–535.

[14] Y. Han and P. Gmytrasiewicz, "Learning others' intentional models in multi-agent settings using interactive pomdps," in 28th Modern Artificial Intelligence and Cognitive Science Conference, MAICS 2017. MAICS, 2017.

[15] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press Cambridge, 1998, vol. 1, no. 1.

[16] A. Nowé, P. Vrancx, and Y.-M. De Hauwere, Game theory and multi-agent reinforcement learning. Springer, 2012, pp. 441–470, chapter 14, pp. 441–470, in Wiering, M., Reinforcement Learning State-of-the-Art, ISBN: 978-3-642-44685-6.

[17] V. Markova, "Adaptive behaviour approach for autonomous mobile sensor agent," in Proceedings of the International Conference on Information Technologies (InfoTech-2012), 2012, pp. 1314–1023.

[18] ——, "Autonomous agent design based on jade framework," in Proceedings of the International Conference on Information Technologies (InfoTech-2013), 2013, pp. 19–20.

[19] V. Markova and V. Shopov, "An approach to build fuzzy cognitive map for time series," in Proceedings of the International Conference on Information Technologies (InfoTech-2014), 2014, pp. 207–211.

[20] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, J. Perolat, D. Silver, T. Graepel et al., "A unified game-theoretic approach to multiagent reinforcement learning," in Advances in Neural Information Processing Systems, 2017, pp. 4193–4206.

[21] M. J. Hausknecht, "Cooperation and communication in multiagent deep reinforcement learning," Ph.D. dissertation, 2016.

[22] L. Frommberger and D. Wolter, "Structural knowledge transfer by spatial abstraction for reinforcement learning agents," Adaptive Behavior, vol. 18, no. 6, 2010, pp. 507–525.

[23] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[24] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," Journal of Big Data, vol. 3, no. 1, 2016, p. 9.

[25] J. Sinapov, S. Narvekar, M. Leonetti, and P. Stone, "Learning inter-task transferability in the absence of target task samples," in Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 725–733.

[26] M. Egorov, "Multi-agent deep reinforcement learning," 2016, report. [Online]. Available: http://cs231n.stanford.edu/reports/2016/pdfs/