Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 126 (2018) 517–526

Procedia Computer Science

www.elsevier.com/locate/procedia

22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Cooperative Multi-Agent Systems Using Distributed Reinforcement Learning Techniques

Wiem Zemzem[b], Moncef Tagina[1]

[a]*COSMOS laboratory, National School Of Computer Sciences (ENSI), University of Manouba,Tunisia*
[b]*zemzem.wiem@gmail.com*

Abstract

In this paper, the fully cooperative multi-agent system is studied, in which all of the agents share the same common goal. The main difficulty in such systems is the coordination problem: how to ensure that the individual decisions of the agents lead to jointly optimal decisions for the group? Firstly, a multi-agent reinforcement learning algorithm combining traditional Q-learning with observation-based teammate modeling techniques, called $TM\_Qlearning$, is presented and evaluated. Several new cooperative action selection strategies are then suggested to improve the multi-agent coordination and accelerate learning, especially in the case of unknown and temporary dynamic environments. The effectiveness of combining $TM\_Qlearning$ with the new proposals is demonstrated using the hunting game.

Keywords: fully cooperative multi-agent system; coordination; cooperative action selection strategies; unknown and temporary dynamic environments

1. Introduction

Reinforcement learning (RL) studies the problem of an agent that learns by interacting with its environment and observing the results of these interactions and it has been applied successfully in many single-agent systems [1, 2, 3, 4]. In RL, the learning occurs iteratively and is performed through trial and errors process, which make it a robust way when dealing with dynamic and unknown environments [5].
Given RL properties, a growing interest was developed these last years in order to extend reinforcement learning to distributed and cooperative multi-agent systems (MASs) [6, 7, 8, 9, 10]. The studies focused on this domain can be

* Wiem Zemzem. Tel.: +216 99171757.
E-mail address: zemzem.wiem@gmail.com

divided into two main categories: The independent learners (ILs) where each agent only knows its own actions and the joint actions learners (JALs) where each agent collects information about its own choice of action as well as other agents' choices [11].
Many existing works are interesting in developing MASs using ILs. Examples include PA (Policy Averaging) [12], EC (Experience Counting) [13], D-DCM-Multi-Q (Distributed Dynamic Correlation Matrix based MultiQ) [14] and CMRL-MRMT (Cooperative Multi-agent learning approach based on the Most Recently Modified Transitions) [15]. All these methods use the assumption that all learners can be in the same state at the same time. These methods succeed to establish a cooperative learning but become inapplicable when considering the collision between learners, like the case in reality of mobile and autonomous robotic systems. As explained in [11], many problems are detected when dealing with ILs, especially when agents are learning synchronously. In such cases, the agents' movement makes the environment neither stationary nor markovian since a past action can have an effect on the current experiences of other agents. Therefore, the theoretical convergence of single-agent RL algorithms is no longer guaranteed. Another difficulty is the multi-agent coordination problem: how to make sure that each IL coherently chooses its own action so that the resulting joint action is optimal?
In the contrary, methods using JALs are often applied to stochastic games [16, 17]. These methods can ensure distributed learning while avoiding collisions between agents. They solve ILs' problems but suffer from other limitations, especially the combinatorial explosion of the size of the state-action space with the number of agents. This is because each JAL learns the value of joint actions contrary to ILs that provide a state space size independent of the number of agents.
To remedy to problems mentioned above, a new kind of multi-agent reinforcement learning algorithm, called $TM\_Qlearning$, was proposed in [18], which combined traditional Q-learning with observation-based teammate modeling techniques. Although experimental results on the hunting game have shown its effectiveness, the problem of cooperative learning in distributed systems is not fully resolved and there are still areas for improvement, especially in terms of the curse of dimensionality, the coordination level and the exploration/exploitation dilemma.
The trade-off between exploration and exploitation is indeed a popular challenge of RL. Each agent faces the following dilemma: either it chooses a random action in order to explore new areas and enlarge its knowledge about the environment (exploration), or it chooses an action which looks optimal given the past observations and rewards (exploitation)[1, 4, 19, 2]. Algorithms such as "ϵ-greedy, Boltzmann distribution, Simulated annealing, Probability Matching and Optimistic Initial Values are proposed to solve this dilemma [19]. However, they are designed for simply assuming the environment is stationary and are primarily dedicated to single-agent systems.
In this paper, two new MARL methods are proposed to better solve the coordination problem and the exploration/exploitation dilemma in the case of non-stationary environments. The same concept of $TM\_Qlearning$ is used here. The novelty consists in promoting the selection of actions so that the chosen action not only relies on the present experience but also on an estimation of possible future ones. Experimental results have shown the effectiveness of the new proposals. It is also demonstrated that favoring least recently visited states accelerates learning and improves multi-agent coordination.
The rest of the paper is organized as follows. Some basic reinforcement learning concepts are introduced in section 2. In section 3, the proposals are presented. Section 4 is dedicated to simulation experiences showing the efficiency of it. Some concluding remarks and future works are discussed in section 5.

2. Reinforcement Learning

Most RL problems are modelled as Markov Decision Processes (MDPs). MDPs are useful for studying a wide range of optimization problems solved via dynamic programming and reinforcement learning. They are used in order to model situations in which an agent has to decide how to act based on the observation of the current state.
More precisely, a Markov model is defined as a 4-tuple (S, A, T, R), where S is a discrete set of environmental states, A is a discrete set of agent actions, $R : S \times A \rightarrow \Re$ is a reward function and $T : S \times A \rightarrow \Pi(S)$ is a state transition function ($\Pi(S)$ is a probability distribution over S). $T(s, a, s')$ is written as the probability of making a transition from s to s' taking action a. The action-value function $Q^*(s, a)$ is defined as the expected infinite discounted sum of rewards that the agent will gain if it chooses the action a in the state s and follows the optimal policy. Given $Q^*(s, a)$ for all state/action pairs, the optimal policy π^* will be developed as the mapping from states to actions such that the future

reward is maximized [1].
One of the most important breakthroughs in RL was the development of an off-policy Temporal Difference (TD) control algorithm known as Q-learning [20] in which the agent maintains a matrix Qvalue storing what he has learned through experience. Despite its multiple successful results in single agent cases, several new challenges arise when scaling it to multi-agent systems (MASs), especially, the loss of convergence hypotheses of the single agent framework and the multi-agent coordination problem.
An extension of the Q-learning method was made by Zhou and Shen [18] and led to the development of a cooperative multi-agent reinforcement learning approach based on the following three modules:

- TM: teammate module, which generates a teammate model to estimate its teammate agents' behaviors strategies
- ASM: action selection module, which uses the greedy policy to select an action for the learning agent
- LM: learning module, which implements the traditional Q-learning algorithm

Experiences illustrated in [18] on a hunting game shows that a system of two predators succeeds to catch a moving prey. However, it is assumed that predators can share the same positions at the same time, which isn't always the case of real applications. In another hand, the successful adaptation of the multi-agent system to the environmental changes isn't only ensured by the learning method and the greedy policy, as outlined in [18], but the random movements of the target can accelerate the system convergence. More clearly, the scenario adopted in [18] assumes that, at each time step, the target has the equal probability to move to one of its neighboring cells or stay at the original one and that it is only captured when the vertically or horizontally neighboring cells are occupied by the two predators. As the environment is small (a 5×5 grid environment) and the target is able to move at any time step, even if predators fail to catch the target, this latter ends up by finding itself in the catch situation.
Consequently, by expanding the environment's size and decreasing the movement of the target over time, the learning performance of $TM\_Qlearning$ may decrease. In what follows, new learning methods that better solve non-stationarity and coordination problems will be proposed.

3. The Proposed Learning Approaches For Distributed And Cooperative Multi-agent Systems

The proposed learning approaches uses the similar Teammate Model and Learning Model adopted in [18]. The novelty lies in the action selection Model.

3.1. The Teammate Model

Consider a system of N learning agents in a joint state s. At a learning step t, a learning agent i under consideration executes an action a_i^* and observes its partners' actions $(a_1^*, ..., a_{i-1}^*, a_{i+1}^*, ..., a_N^*)$, the new obtained state s' and the resulted reward r. Then, it updates its teammate model for each possible action a_i in the state s. In the case of two-agents systems, the teammate model can be updated as follows:

$$P_{ij\_t}(s, a_i) = \begin{cases} P_{ij\_(t-1)}(s, a_j) + \beta^{T-t+1} \sum_{a_t \in A_j - \{a_j\}} P_{ij\_(t-1)}(s, a_t), & a_j = a^* \\ (1 - \beta^{T-t+1}) P_{ij\_(t-1)}(s, a_j), & elsewhere \end{cases} \quad (1)$$

where i and j are respectively the learner under consideration and its partner, $P : S \times A_i \to \Re$ is the teammate model of agent i, S is the set of all possible joint states ($S = S_i \times S_j$), A_i and A_j are respectively the action set of agent i and j, $\beta \in [0, 1]$ is the learning rate that determines the effect of previous action distribution, and T is the number of iterations needed for the task completion. s and s' are joint states that present the instantaneous positions of agents (i, j) in the environment.

3.2. The Learning Model

After experimenting the transition $(s, a_i, a_j) \rightarrow (s', r)$, the learner i can update its Q function at the corresponding entry using Eq.2:

$$Q_{ij}\left(s, a_i, a_j\right) = (1-\alpha)\, Q_{ij}\left(s, a_i, a_j\right) + \alpha \left(r + \gamma max_{a' \in A_i} Q_{ij}(s', a_i^{'}, a_j^{'})\right) \quad (2)$$

where

$$a_j^{'} = argmax_{b \in A_j} P_{ij}(s', b) \quad (3)$$

Thus, the teammate model is exploited by the learning model. More precisely, the best displacement in the next state s' depends on the next action $a_i^{'}$ of the agent i under consideration and the next action $a_j^{'}$ of its partner j. Given that the Q function of this later isn't known by agent i, a prediction of its action is done using the memorized teammate model. For the examined agent, the action that should be preferably executed in the next state is determined by the exploitation of its Q function.

3.3. The Action Selection Model

The main idea is to incorporate the results of the teammate model and the learning model in the action selection procedure so that the final decision takes into account the information already learned by this agent. Two new cooperative action selection strategies are then proposed.

3.4. First Proposition: The $TM\_maxNextQ$ Policy

For more useful action choice, the chosen action is not only whose Qvalue is the highest (as in [18]) but also which leads to a more promising state, i.e, the state whose its highest Qvalue is greater than that of the current state.
To this end, a new action selection policy (ASP) exploiting the teammate model and favoring more promising next states are proposed. This ASP, called $TM\_maxNextQ$ (a policy exploiting Teammate model and maximizing Next Qvalues), is summarized by algorithm 1. Note that the list L_j stores all actions $a_j \in A_j$ that maximize Pvalues in the current state s while the list L_i stores all actions $a_i \in A_i$ that maximize Qvalues in s. However, the list L stores all actions $a_i \in L_i$ that lead to a next state $s'(s' \leftarrow T(s, a_i, a_j))$ having a Qvalue greater than that of the current state s $(maxQ(s, a_s, a_j) < maxQ(s', a_i', a_j'))$.
As noted earlier, for the $TM\_Qlearning$ method, only the Pvalues and Qvalues of the current state are exploited by the ASM. The action selection policy is then restricted to steps [1-9;19].

3.5. Second Proposition: the $TM\_maxNextQ\_LRVS$ Policy

In order to promote the exploration of new areas, it is possible to alternate between a random selection and a selection favoring the least recently tested states. To do that, each agent memorizes a table $Anc : S \rightarrow R$, storing the seniority of each visited state. $Anc(s) = t$ means that the last visit of state s occurred at time t.
This new alternative, called $TM\_maxNextQ\_LRVS$ (the $TM\_maxNextQ$ policy favoring Least Recently Visited States) is summarized by algorithm 2.

3.6. The Proposed Learning Algorithm

The objective of this paper is to find a distributed learning approach letting several agents cooperate in order to perform a common task efficiently and without colliding with each other. To this end, a new learning method that

Algorithm 1 Algorithm $TM\_maxNextQ$ for two-agent systems

```
Require:
    $a_i \in A_i$: actions of the learner under consideration
    $a_j \in A_j$: actions of the teammate agent
    $s \in S$: possible joint states
    $T(s, a_i, a_j)$: the new state after executing $a_i$ and $a_j$ in the current state s
Ensure:
 1: Initialize $L_j$, $L_i$ and $L$ to be empty lists
 2: for $a_j \in A_j$ do
 3:     if $a_j = argmax_{b \in A_j} P\_(s, b)$ then
 4:         $L_j \leftarrow a_j$
 5:     end if
 6: end for
 7: for $(a_i, a_j) \in A_i \times L_j$ do
 8:     if $a_i = argmax_{b \in A_i} Q\_(s, b, a_j)$ then
 9:         $L_i \leftarrow a_i$
10:     end if
11: end for
12: for $(a_i, a_j) \in L_i \times L_j$ do
13:     $s' \leftarrow T(s, a_i, a_j)$
14:     $maxNextQ = maxQ(s', a'_i, a'_j)$ where $a'_i$ and $a'_j$ are determined following steps [2-5;7-9], respectively
15:     if $Q(s, a_i, a_j) < maxNextQ$ then
16:         $L \leftarrow a_i$
17:     end if
18: end for
19: if L is not empty then
20:     Choose a random action from L
21: else
22:     Choose a random action from $L_i$
23: end if
```

Algorithm 2 Algorithm $TM\_maxNextQ\_LRVS$ for two-agent systems

```
 1: Initialize $L_i$, $L_j$ and $L$ to be empty lists
 2: Initialize $Anc$ to be zero, $\forall s \in S$
 3: Initialize $explore \in [0, 1]$
 4: Performing steps [2-18] of algorithm 1
 5: Choose a random value $prob \in [0, 1]$
 6: if $prob < explore$ then
 7:     Choose a random action from L
 8: else
 9:     Choose an action $a_i \in L_i$ such as $\exists a_j \in L_j$ where: $T(s, a_i, a_j) = argmin_{s' \in S} Anc(s)$
10: end if
```

implements the three modules explained earlier is proposed. It's summarized by algorithm 3.

Note that all the proposed algorithms can be applied to N-agents systems ($N >= 2$) but with replacing the partner j by $(N-1)$ partners, $a_j \in A_j$ by $(a_1, ..., a_{i-1}, a_{i+1}, ..., a_N) \in A_1 \times ... \times A_{i-1} \times A_{i+1}, ... \times A_N$ and $s_j \in S_j$ by $(s_1, ..., s_{i-1}, s_{i+1}, ..., s_N) \in S_1 \times ... \times S_{i-1} \times S_{i+1} \times ... \times S_N$.

Algorithm 3 The learning algorithm for two-agent systems

```
1:  $Anc(s) \leftarrow 0, \forall s \in S$
2:  $P(s, a_j) \leftarrow \frac{1}{|A_j(s)|}, \forall s \in S, \forall a_j \in A_j$
3:  $Q(s, a_i, a_j) \leftarrow \frac{1}{|A_i(s)||A_j(s)|}, \forall s \in S, \forall a_i \in A_i, \forall a_j \in A_j$
4:  while learning in progress do
5:    $t \leftarrow 0$
6:    Initialize $s_t$
7:    for each step per episode do
8:      Observe the current state $s_t$
9:      Select an action $a_i$ using algorithm 1 or 2
10:     Execute $a_i$, the teammate learner simultaneously performs an action $a_j^*$
11:     Observe the new state $s_{t+1}$ and the resulted reward $r_{t+1}$
12:     $\forall a_j \in A_j$, update $P(s_t, a_j)$ according to eq.1
13:     Update $Q(s_t, a_i, a_j^*)$ according to eq.2
14:     $Anc(s) \leftarrow t$
15:     if The new state $s_{t+1}$ is the goal state then
16:       The episode is ended, go to step 4
17:     else $\{s_{t+1} \neq goal\_state\}$
18:       $t \leftarrow t + 1$, go to step 7
19:     end if
20:   end for
21: end while
```

4. Simulation Experiments

In this section, the impact of the new proposed learning strategies on a distributed multi-agent system is evaluated. For that, a cooperative hunting task with a temporary moving target is considered. It's a popular case study where some agents, called predators, are trying to capture one other agent, called the prey [11].

4.1. Benchmark

The testing environment is a 10×10 grid world which is initially unknown, surrounded by walls and contains obstacles. In this maze, the predators are the learners. They have to coordinate themselves in order to capture a single prey. At each time step, they are able to move in four directions: north (Up), south (Down), east (Right) and west (Left) or stand still (NoMove). They can't share the same position and can't cross obstacles. The case study, described in [18, 11], is modified such as the prey will no longer move at each time step but performs several successive displacements each N iterations.
More precisely, the environment is initially in the form of Fig.1-a. After 1000 episodes (captures), it changes to the the form of Fig.1-b. The red ball refers to the prey. This latter is captured when the vertically or horizontally neighboring cells are occupied by the two predators. Then, predators are relocated at the top right corner of the maze (Fig.1) and the next episode starts.

4.2. Experiments Results

The prey is temporary moving and predators have to capture it without colliding with each other. Through learning, every predator should be able to build an optimal path from its starting position to the target while avoiding obstacles and other partners as well as to correct this path after each environmental change. An episode ends when the prey is captured or that 1000 iterations are exceeded.
For the parameter settings, the decline factor γ and the learning rate α are defined as 0.8 for all the learners. The reward is defined as -0.05 for each regular action without hitting any obstacle or other agents (the prey or the learning

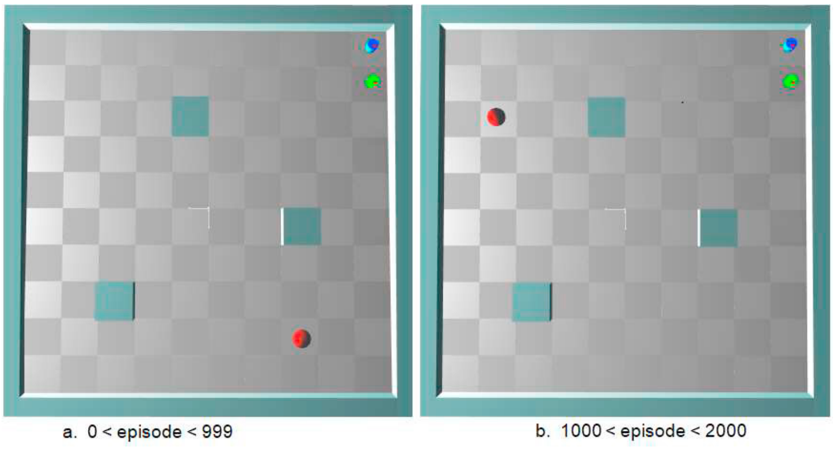

Fig. 1. The testing environment

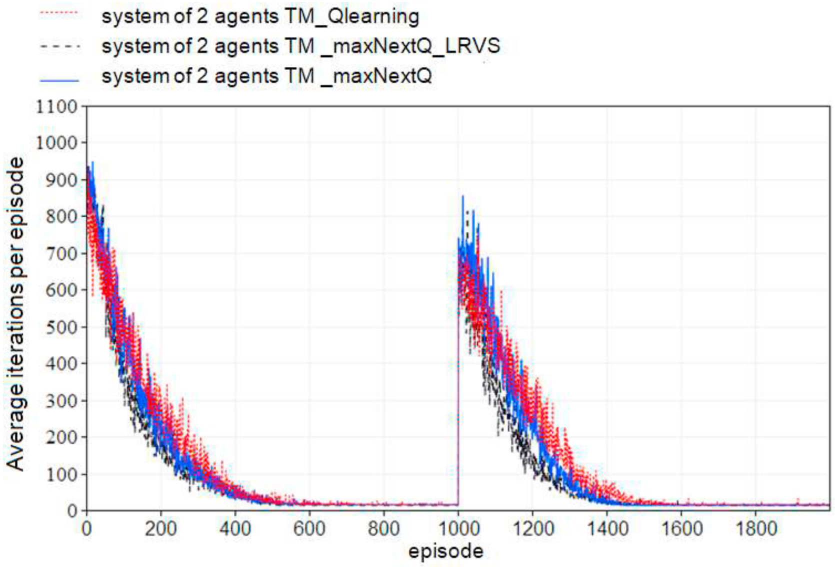

Fig. 2. The average number of time steps (learning steps) needed on each episode over time

partner). If a predator hits a wall or another agent, it gets a penalty of -3 and if the prey is successfully captured, a reward of 3 is assigned to both predators. Finally, the values of tables P, Q and Anc are initially set to 0.2,0.04 and 0, respectively. All the results given for the following experiments are the average value of 30 distinct runs and each run consists of 2000 episodes.

In what follows, the two proposed learning strategies ($TM\_maxNextQ$ and $TM\_maxNextQ\_LRVS$) will be compared with $TM\_Qlearning$ [18]. Fig.2 and Fig.3 describes the learning and the collision graphs of the three tested methods respectively and Table.1 shows the average number of time steps and collisions performed by each learner during its learning experience.

From Fig.2, it can be seen that the system convergence is ensured by the three methods and that the learners succeed to rebuilt a new optimal path after the environmental change. However, the proposed methods accelerate learning compared to $TM\_Qlearning$ and especially the $TM\_maxNextQ\_LRVS$ that outperforms both other learning methods and this is before (episodes [0-999]) and after (episodes [1000-2000]) the displacement of the prey.

Table 1. The average learning results of different learning algorithms

Learning method	Total iterations		Total collisions	
	Episodes [0-999]	Episodes [1000-2000]	Episodes [0-999]	Episodes [1000-2000]
TM_Qlearning	144645	133963	32879	29476
TM_maxNextQ	136591	123922	6826	6212
TM_maxNextQ_LRVS	119988	97192	6656	5499

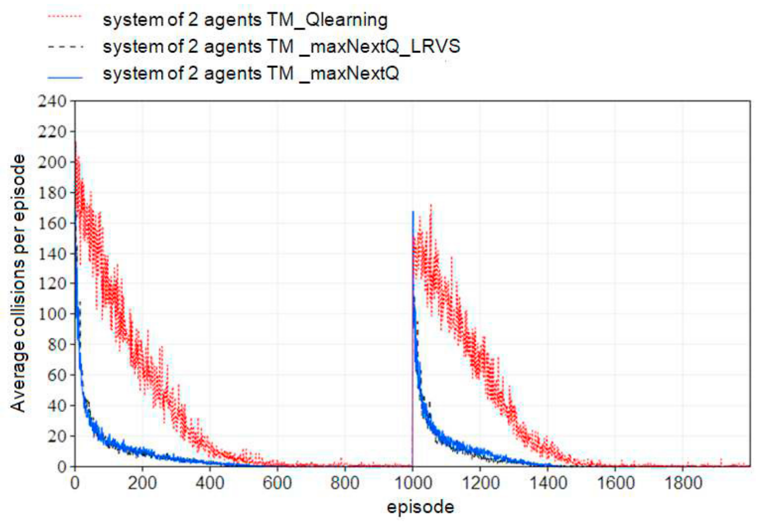

Fig. 3. The average number of collisions detected on each episode over time

The strength of the new proposed methods raises when dealing with collisions. As shown in Fig.3 and Table.1, the number of collisions done during [0-999] episodes falls from 32879 collisions with *TM_Qlearning* to only 6826 collisions by *TM_maxNextQ* and 6656 collisions using *TM_maxNextQ_LRVS*. This result remains true after the environmental change. Furthermore, after convergence, the *TM_maxNextQ_LRVS* learners converge to an optimal path free of collisions. However, few collisions are observed after the convergence of the *TM_maxNextQ* learners and a more important number with the *TM_Qlearning* learners. Fig.4 shows this result; it describes the average number of collisions detected after the two system convergences (episodes [599-600] and [1600-2000]).

5. Conclusion

In this paper, the problem of cooperative reinforcement learning for distributed decision-making in multi-agent systems has been studied. For that, two new cooperative MARL methods, called *TM_maxNextQ* and *TM_maxNextQ_LRVS*, have been proposed.
It is demonstrated using the hunting game that these methods improve coordination between agents and that they outperform the *TM_Qlearning* method which deals with the same problem. This is by ensuring the convergence to an optimal path with the minimum of collisions and in a timelier manner. These new methods are also a good alternative to solve the exploration/exploitation dilemma. This is shown through the increasing of the learning speed compared to the *TM_Qlearning* method. Moreover, it is demonstrated that favoring least recently visited states considerably accelerates learning and provides the convergence to an optimal path free from collisions. However, these methods don't solve the curse of dimensionality since they implement joint states and joint actions. New improvements can be then proposed to overcome this issue.
The present work can be further ameliorated by extending it with capabilities for solving more complex scenarios. Possible test cases include more agents, several targets, as well as continuous state spaces. Real-world applications

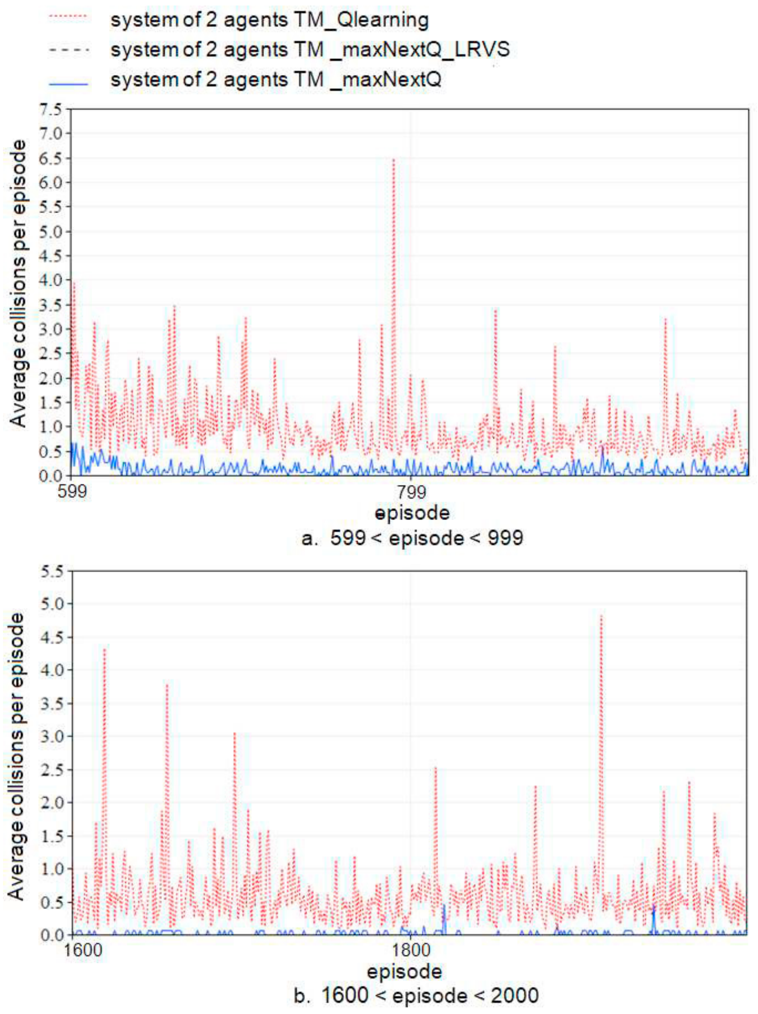

Fig. 4. The average number of collisions detected after the system convergence

are usually characterized by these complex elements. In addition to that, the current Markov game model assumes that all agents can observe the global state space, while this might not be feasible in practice. It would be worthwhile further investigating how agents can still coordinate effectively when only partial state information is available.

Acknowledgements

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

References

[1] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, IEEE Transactions on Neural Networks 9 (5) (1998) 1054–1054.
[2] K. Chen, F. Lin, Q. Tan, Z. Shi, Adaptive action selection using utility-based reinforcement learning, in: 2009 IEEE International Conference on Granular Computing, IEEE, 2009, pp. 67–72.
[3] W. Zemzem, M. Tagina, A New Approach for Reinforcement Learning in non Stationary Environment Navigation Tasks, International Review on Computers and Software 7 (5) (2012) 134–143.
[4] W. Zemzem, M. Tagina, A novel exploration/exploitation policy accelerating learning in both stationary and non stationary environment navigation tasks, International Journal of Computer and Electrical Engineering 7 (3) (2015) 149–158.

[5] E. Yang, D. Gu, Multiagent reinforcement learning for multi-robot systems: A survey, Tech. rep., 2004.
[6] L. Matignon, G. J. Laurent, N. Le, Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems, Knowledge Engineering Review 27 (1) (2012) 1–31.
[7] X. Chen, G. Chen, W. Cao, M. Wu, Cooperative learning with joint state value approximation for multi-agent systems, Journal of Control Theory and Applications 11 (2) (2013) 149–155.
[8] J. Hao, D. Huang, Y. Cai, H.-f. Leung, The dynamics of reinforcement social learning in networked cooperative multiagent systems, Engineering Applications of Artificial Intelligence 58 (2017) 111–122.
[9] W. Zemzem, M. Tagina, Cooperative multi-agent reinforcement learning in a large stationary environment, in: 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), IEEE, 2017, pp. 365–371.
[10] W. Zemzem, M. Tagina, Multi-agent Coordination using Reinforcement Learning with a Relay Agent, in: Proceedings of the 19th International Conference on Enterprise Information Systems, SCITEPRESS - Science and Technology Publications, 2017, pp. 537–545.
[11] L. Matignon, G. J. Laurent, N. L. Fort-Piat, Hysteretic q-learning :an algorithm for decentralized reinforcement learning in cooperative multi-agent teams, in: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2007, pp. 64–69.
[12] M. Tan, Multi-agent reinforcement learning: independent vs. cooperative agents, in: the tenth international conference on machine learning, Morgan Kaufmann Publishers Inc., 1997, pp. 330–337.
[13] B. Cunningham, Y. Cao, Non-reciprocating Sharing Methods in Cooperative Q-Learning Environments, in: 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, IEEE, 2012, pp. 212–219.
[14] H. Guo, Y. Meng, Distributed Reinforcement Learning for Coordinate Multi-Robot Foraging, Journal of Intelligent and Robotic Systems 60 (3-4) (2010) 531–551.
[15] W. Zemzem, M. Tagina, Cooperative multi-agent learning in a large dynamic environment, in: Lecture Notes in Computer Science, Springer, 2015, Ch. MDAI, pp. 155–166.
[16] H. M. Schwartz, Multi-Agent Machine Learning: A Reinforcement Approach, John Wiley and Sons, 2014.
[17] M. Wiering, M. van. Otterlo, Reinforcement learning : state-of-the-art, Springer-Verlag Berlin Heidelberg, 2012.
[18] P. Zhou, H. Shen, Multi-agent cooperation by reinforcement learning with teammate modeling and reward allotment, 8th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2011 2 (4) (2011) 1316–1319.
[19] M. Yogeswaran, S. G. Ponnambalam, Reinforcement learning: exploration-exploitation dilemma in multi-agent foraging task, OPSEARCH 49 (3) (2012) 223–236.
[20] C. Watkins, P. Dayan, Q-learning, Machine Learning 8 (1992) 279–292.