Hysteretic Q-Learning : an algorithm for Decentralized Reinforcement Learning in Cooperative Multi-Agent Teams.

Laëtitia Matignon, Guillaume J. Laurent and Nadine Le Fort-Piat
Laboratoire d'Automatique de Besançon UMR CNRS 6596
ENSMM , Université de Franche-Comté
24 rue Alain Savary, 25000 Besançon, France
E-mails : (laetitia.matignon,guillaume.laurent,nadine.piat)@ens2m.fr
Web site : www.lab.cnrs.fr

***Abstract*— Multi-agent systems (MAS) are a field of study of growing interest in a variety of domains such as robotics or distributed controls. The article focuses on decentralized reinforcement learning (RL) in cooperative MAS, where a team of independent learning robots (IL) try to coordinate their individual behavior to reach a coherent joint behavior. We assume that each robot has no information about its teammates' actions. To date, RL approaches for such ILs did not guarantee convergence to the optimal joint policy in scenarios where the coordination is difficult. We report an investigation of existing algorithms for the learning of coordination in cooperative MAS, and suggest a Q-Learning extension for ILs, called Hysteretic Q-Learning. This algorithm does not require any additional communication between robots. Its advantages are showing off and compared to other methods on various applications : bi-matrix games, collaborative ball balancing task and pursuit domain.**

I. INTRODUCTION

Learning in multi-agent systems (MAS) are a field of study of growing interest in a wide variety of domains, and especially in multi-robot systems [1]. Indeed, a decentralized MAS point of view offers several potential advantages as speed-up, scalability or robustness [2].

In this paper, we are interested in learning in MAS thanks to reinforcement learning (RL) methods, where an agent learns by interacting with its environment, using a scalar reward signal called reinforcement as performance feedback [3]. Over the last decade, many approaches are concerned with the extension of RL to MAS [4], *e.g.* team of soccer robots [5] or distributed control of a robotic manipulator [6].

We investigate the case of *cooperative MAS* where all agents share the same goal and the common return can be jointly maximized. As pointed out by Boutilier [7], a cooperative MAS could be solved by classical RL algorithms in *a centralized view* in which one agent represents the whole team. But the size of the state-action space should quickly become too big for RL. So we investigate the case of agents which learn their own behavior in a decentralized way.

In this framework, Claus & Boutilier [8] distinguish two cases of reinforcement learners : the case of agents that get information about their own choice of action as well as their partners' choices, called "*joint action learners*" (JALs), and the case of agents which only know their own action, called "*independent learners*" (ILs). The former case suffers from combinatorial explosion of the size of the state-action space with the number of agents given that each agent learns the value of joint actions, while the latter one brings the benefit of a state space size independent of the number of agents. We focus on ILs which is a more realistic assumption and don't require any communication between agents.

The use of *ILs in cooperative MAS* induces *three major difficulties*. The first one is the fact that other learning agents are *unpredictable elements* of the environment because of the local view of each agent. Secondly, from the viewpoint of any agent in MAS, the environment is not any longer markovian since a past action can have an effect on the other agents current behaviors. This fact destroys the theoretical convergence guarantees of single-agent RL algorithms. Finally, the third difficulty is the *multi-agent coordination problem* : how to make sure that all ILs coherently choose their individual action such that the resulting joint action is optimal?

In this paper, we present a decentralized RL algorithm for ILs which computes a better policy in a cooperative MAS without additional information or communication between agents than existing algorithms, presented in Section II. Our algorithm, called Hysteretic Q-Learning, is a variant of Q-learning [9] and is compared with existing algorithms designed for the cooperation/coordination of ILs. First, we study the case of cooperative repeated matrix games where the coordination is difficult (Sect. III). Then, we extend the comparison to identical payoff stochastic games (Sect. IV) through a collaborative ball balancing task and a multi-agent pursuit benchmark (Sect. V). The study is lastly extended to a partial observable benchmark with 4 players (Sect. VI).

II. RELATED WORKS

In this section, related works dealing with RL algorithms in cooperative MAS are reviewed, with an emphasis on research dealing with Q-learning [9] and Q-learning variants for ILs. It was one of the first algorithms applied to multi-agent environments [10]. Such works focus on game theory - more particularly repeated games (Sect. III) - and stochastic games (Sect. IV).

Claus & Boutilier [8] compare *the performance of JALs and ILs* on repeated coordination game (Tab. I) and found that even though JALs have much more information at their disposal, they do not perform much differently from ILs in the straightforward application of Q-learning to MAS. Notably, they show that convergence to global optimum is not always achieved in these games even if each agent can immediately perceive the actions of all other agents in the environment.

They investigate too the crucial effect the applied *exploration policy* has on the performance of the learning when the single-agent RL methods are utilized in MAS. Notably, they try to compute an equilibrium point by continuously reducing the exploration frequency, so as to avoid *concurrent exploration*. This problem is also investigated in [11].

Lauer & Riedmiller [12] introduced the *Distributed Q-Learning* algorithm. Optimistic independent agents neglect the penalties due to a non-coordination of agents in their update. They show that their algorithm will find the unique optimal equilibrium solution in deterministic cooperative MAS. Nevertheless, this algorithm used without any additional coordination mechanism is not able to manage the coordination in case of multiple optimal joint actions.

Kapetanakis & Kudenko [13] point out another flaws in Lauer's approach when dealing with stochastic environments, and present a modified exploration strategy. In their algorithm, the Q-value of an action in the Boltzmann exploration strategy is changed by an *heuristic value*, taking into account how frequently an action produces its maximum corresponding reward. This heuristic, which is called *FMQ*, works only in repeated games. It solves the reward uncertainty issue due to the other agents' actions, but does not overcome the difficulty of games strongly noised.

Besides the theoretical examinations, several successful applications of decentralized RL have been reported, like in the control of a group of elevators [14] or in the task of multi-robot box-pushing [15]. They use ordinary Q-Learning without any consideration of the existence of other agents.

III. Fully cooperative repeated games

The studies of learning algorithms in MAS are based on game theory and more particularly on repeated games. In this section, we first setup this framework. Then, we present a version of the Hysteretic Q-Learning for repeated games and some results on two usual cooperative games with regard to other algorithms.

A. Definition

A *matrix game*[1] (MG) is a multiple-agent, single state framework. It is defined as a tuple $< n, A_1, ..., A_n, R_1, ..., R_n >$ where n is the number of players, A_i is the set of actions available to player i (and A is the joint action space $A_1 \times ... \times A_n$) and R_i is player i's payoff function $A \rightarrow \Re$.

[1]also called *strategic game*

TABLE I

Two Cooperative bi-matrix games

		Agent 2		
		A	B	C
Agent 1	A	11	-30	0
	B	-30	7	6
	C	0	0	5

Climbing game

		Agent 2		
		A	B	C
Agent 1	A	10	0	K
	B	0	2	0
	C	K	0	10

Penalty game

If $R_1 = ... = R_n = R$, the MG is *fully cooperative*. We are interested in *repeated games* which consist of the repetition of the same MG by the same agents. Among matrix games, bi-matrix games are often used to formulate the 2-agents case.

Table I shows 2 cooperative MG : the Climbing game and the Penalty game, introduced in [8] for the study of *coordination* in cooperative MG. Each agent have 3 actions and the table specifies the joint rewards. Each of these games is challenging due to mis-coordination penalties. In the Climbing game, the optimal joint action is (a, a) but if an agent chooses its individual optimal action a when the other agent chooses action b, a severe penalty is received. However, there are no mis-coordination penalties associated with action c, potentially making it tempting for the agents. In the Penalty game, K is usually chosen inferior to 0. This game introduces another mis-coordination issue due to the presence of two optimal joint actions (a, a) and (c, c) : simply choosing its individual optimal action does not guarantee that the other agent will choose the same optimal.

In the case of only one state, Q-Learning is reduced to the update equation of the Q-value function [8] :

$$Q_i(a_i) \leftarrow Q_i(a_i) + \alpha(r - Q_i(a_i)) \tag{1}$$

where a_i is the agent's chosen action, r the reward it receives, $Q_i(a_i)$ is the value of action a_i for the agent i and $\alpha \in]0; 1]$ is the learning rate.

B. Hysteretic Q-learning

In a MAS, the reinforcement received by an agent relies on actions chosen by the team. So an agent can be punished because of a bad choice of the team even if it has chosen an optimal action. Then the agent had better to attach less importance to a punishment received after the choice of an action which has been satisfying in the past. This is the idea of *the optimistic agents* used in the Distributed Q-Learning algorithm. Distributed Q-Learning update equation is [12] :

$$\begin{aligned} \delta &\leftarrow r - Q_i(a_i) \\ Q_i(a_i) &\leftarrow \begin{cases} Q_i(a_i) + \alpha\delta & \text{if } \delta \geq 0 \\ Q_i(a_i) & \text{else} \end{cases} \end{aligned} \tag{2}$$

However, the key issue with Distributed Q-Learning algorithm is that optimistic agents do not manage to achieve

TABLE II

PERCENTAGE OF TRIALS CONVERGING TO THE OPTIMAL JOINT ACTION (AVERAGE ON 3000 TRIALS).

	Climbing game	Penalty game ($K = -100$)
Decentralized Q-Learning	12.1%	64%
Distributed Q-Learning	95.8%	50.6%
FMQ	99.8%	99.9%
Hysteretic Q-Learning	99.5%	99.8%

the coordination between multiple optimal joint actions [12]. Agents must not be altogether blind to penalties at the risk of staying in sub-optimal equilibrium or mis-coordinating on the same optimal joint action. That's why we suggest to use *two learning rates* according to the result of a joint action.

Thus, the update equation (1) is modified :

$$\begin{aligned} \delta &\leftarrow r - Q_i(a_i) \\ Q_i(a_i) &\leftarrow \begin{cases} Q_i(a_i) + \alpha\delta & \text{if } \delta \geq 0 \\ Q_i(a_i) + \beta\delta & \text{else} \end{cases} \end{aligned} \tag{3}$$

where α and β are the increase and decrease rates of Q-values. The Hysteretic Q-learning is decentralized : each ILs builds its own Q-table whose size is independent of the agents number and linear in function of its own actions.

C. Experimentations

We tested different algorithms on both cooperative repeated games (Tab. I) : decentralized Q-Learning, FMQ, Distributed Q-Learning and Hysteretic Q-Learning. Decentralized Q-Learning is the straightforward application of update equation 1 to ILs.

A trial consists of 7500 repetitions of the game. At the beginning of a trial, Q-tables are initialized at 0. At the end of each trial, we determine if the last chosen joint action is optimal. We take $\alpha = 0.1$ for all methods, $\beta = 0.01$ for Hysteretic Q-Learning and $c = 10$ for the weight in the FMQ. Concerning the action selection method, we have chosen an action selection according to Boltzmann distribution where T is a temperature parameter. T is setting up at $T = T \times 0.99$ with $T_{init} = 5000$. Results are shown in Table II.

First, decentralized Q-Learning is inefficient to reach the optimal joint action in both games. Indeed, all rewards are equally considered. So, high penalties for mis-coordination caused the agents to be attracted by safer sub-optimal equilibrium in both games. Distributed Q-Learning performs rather well in the Climbing game because of agents omission of penalties. But in the Penalty game, they don't overcome the issue of coordination : each agent has two optimal individual actions so four greedy policies can be generated (a, a), (c, c), (c, a) and (a, c), and only the first two are optimal. That's why the agents choose the two optimal with a probability of 50%. Both algorithms FMQ and Hysteretic Q-Learning discovered the optimal joint action more than 99% of the time. Anyway, it is important to notice that FMQ requires larger memory size than Hysteretic Q-Learning, storing only the Q_i-values.

So in cooperative MG, *Hysteretic Q-Learning manages to solve the coordination issue and requires agent's internal status smaller than FMQ.* Indeed, FMQ augments the agent's internal status by maintaining 3 values for each of its actions so as to carry the information of how frequently an action produces its maximum corresponding reward.

IV. STOCHASTIC GAMES

A. Definition

Stochastic games (SG) can be seen as the extension of matrix game to the multi-states framework. Specifically, each state of a SG can be viewed as a matrix game. They were first examined in the field of game theory and more recently in the field of multi-agent RL.

A *stochastic game*[2] is defined as a tuple $< n, S, A_1, ..., A_n, T, R_1, ..., R_n >$ where :

- n is the number of agents;
- S is a finite set of states;
- A_i is the set of actions available to the agent i (and $A = \prod A_i$ the joint action space);
- $T : S \times A \times S \rightarrow [0, 1]$ a transition function that defines transition probabilities between states;
- $R_i : S \times A \rightarrow \Re$ the reward function for agent i.

In a SG framework, all agents have access to the complete observable state s. The joint actions of the agents determine the next state and the rewards received by each agent. If all agents receive the same rewards, the SG is fully cooperative. It is then defined as an *identical payoff stochastic game* (IPSG)[3]. The objective of each agent is then to find the optimal policy maximizing the expected sum of the discount rewards in the future.

The straightforward extension of centralized Q-Learning to SG considers joint actions in the computation of Q-values. Thus, the update equation in a centralized view is :

$$\begin{aligned} Q(s, a_1, ..., a_n) \leftarrow (1 - \alpha)Q(s, a_1, ..., a_n) + \\ \alpha \left[r + \gamma \max_{a'_1, ..., a'_n} Q(s', a'_1, ..., a'_n) \right] \end{aligned} \tag{4}$$

where s' is the new state, α the learning rate and $\gamma \in [0; 1[$ the discount factor.

In a decentralized framework, the Q-learning update equation for ILs is :

$$Q_i(s, a_i) \leftarrow (1 - \alpha)Q_i(s, a_i) + \alpha \left[r + \gamma \max_{a'} Q_i(s', a') \right] \tag{5}$$

It is obvious that such Q_i tables for ILs are smaller. But each agent has only a local view because it has no access to the actions of the others.

[2] also called Markov game.

[3] also called Multi-agent Markov Decision Process (MMDP).

B. Hysteretic Q-learning

We extend the equation (3) to SG. Then, the Hysteretic Q-Learning update equation for an agent i executing the action a_i from state s ending up in state s' is :

$$\delta \leftarrow r + \gamma \max_{a'} Q_i(s', a') - Q_i(s, a_i)$$
$$Q_i(s, a_i) \leftarrow \begin{cases} Q_i(s, a_i) + \alpha\delta & \text{if } \delta \geq 0 \\ Q_i(s, a_i) + \beta\delta & \text{else} \end{cases} \quad (6)$$

V. Experiments on Stochastic Games

We propose to compare the performance of Hysteretic Q-Learning in SG tasks with a centralized Q-Learning view (update equation 4), a decentralized Q-Learning framework (update equation 5) and the Distributed Q-Learning algorithm (update equation 2).

A. Experiments on a collaborative ball balancing task

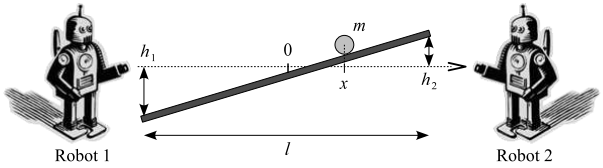


Fig. 1. Overview of a collaborative ball balancing task

1) Benchmark: We take up an example task whose purpose is to keep balance of a rolling ball in the center of a flat table holding by two robots at the extremities [16]. Each robot control h_i values. The dynamics are given by :

$$m\ddot{x} = -c\dot{x} + mg\left(\frac{h_1 - h_2}{l}\right) \quad (7)$$

with $m = 0.5$, $g = 9.8$, $l = 2$ and $c = 0.01$.

2) State space: The complete plant state is given by $\mathbf{x} = (x, \dot{x})$. If centralized learning is used, the command is $\mathbf{a} = (h_1, h_2)$; for decentralized learning with two robots controlling each extremity of the table, the robot commands are $a_1 = h_1$ and $a_2 = h_2$. To apply RL in the form presented previously, the time axis and the continuous state and action must be first discretized. The sample time is 0.03 seconds. For state space, a 100×50 discretization is chosen. For each control, 15 values are used equally distributed between -1 and 1. So, this yields a Q-table of size $100 \times 50 \times 15^2$ in a centralized view, to compare with two Q_i-tables of size $100 \times 50 \times 15$ in a decentralized view.

Each trial starts from an initial state $\mathbf{x} = (0.5, 0.1)$ and goes on at the most 20 seconds. A trial ends if the ball is fallen from the table.

3) Reinforcement: According to [17], the chosen reward function is :

$$r = 0.8e^{-\frac{x^2}{0.25^2}} + 0.2e^{-\frac{\dot{x}^2}{0.25^2}}. \quad (8)$$

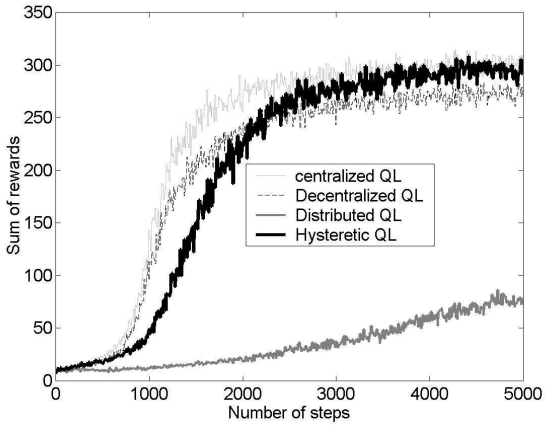


Fig. 2. Sum of rewards (averaged over 20 runs) with $\alpha = 0.9$ $\beta = 0.1$ $\gamma = 0.9$ $\epsilon = 0.1$.

4) Results: All trials use a discount factor $\gamma = 0.9$ and follow the ϵ-greedy action selection method[4] ($\epsilon = 0.1$). We have plotted the sum of rewards obtained at each trial over 5000 successive trials. Results are shown Fig. 2.

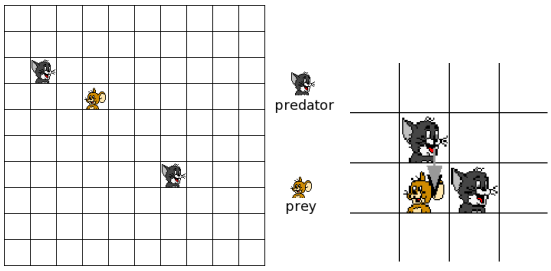
First, Distributed Q-Learning shows a very slow convergence. Otherwise, with decentralized Q-Learning, results are only slightly less successful than with centralized Q-Learning, although Q-tables sizes are smaller and no-coordination mechanisms are setting up. So this confirms the fact that ordinary decentralized Q-Learning could be applied to some cooperative MAS systems. Anyway, only Hysteretic Q-Learning reaches the same convergence as centralized algorithm. *The best coordination in this decentralized framework is obtained with the Hysteretic Q-Learning algorithm.*

B. Experiments on a pursuit domain

The pursuit problem is a popular multi-agent domain in which some agents, called predators, have to capture one other agent, called the prey [18].

1) Benchmark: We use a discrete 10×10 toroidal grid environment in which two predators have to explicitly coordinate their actions in order to capture a single prey (Fig. 3a). At each time step, all predators simultaneously execute one of the 5 possible actions : move north, south, east, west or stand still. The prey moves according to a randomized policy : it remains on its current position with a probability of 0.2, and otherwise moves to ones of its free adjacent cells with uniform probability. A prey is captured when both predators are located in cells adjacent to the prey, and one of the two predators moves to the location of the prey while the other predator remains, for support, on its current position (Fig. 3b).

[4]the probability of taking a random action is ϵ and, otherwise, the selected action is the one with the largest Q-value in the current state.



(a) (b)

Fig. 3. Example pursuit problem. (a) Complete 10×10 toroidal grid. (b) Possible capture position.

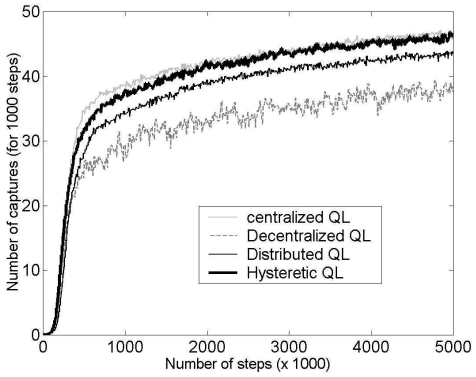


Fig. 4. Number of capture for 1000 steps (averaged over 20 runs) with $\alpha = 0.1$ $\beta = 0.01$ $\gamma = 0.9$ $\epsilon = 0.1$.

2) State space: The state space is represented by the relative position of the two predators to the prey. The complete state-action space then consists of all combinations of the two predators relative position. In total this yields $99 \times 98 \times 5^2$, that is $242,550$ state-action pairs in a centralized view stored in the unique Q function. This is to compare with Q_i tables for each ILs in a decentralized framework, corresponding to $99 \times 98 \times 5$, *i.e.* $48,510$ different state-action pairs for each of both agents.

3) Reinforcement: When two predators end up in the same cell, they are penalized ($R_i = -10$) and are moved to a random empty position on the grid. A capture results in a reward $R_i = 37.5$ for each agent, and predators and prey are replaced to random positions. Furthermore, each predator receives a reward of -25 when one of them moves to the prey without support and both are moved to random, empty cell.

4) Results: All trials use a discount factor $\gamma = 0.9$, a learning rate $\alpha = 0.1$, a decrease rate (Hysteretic Q-Learning) $\beta = 0.01$ and follow the ϵ-greedy method ($\epsilon = 0.1$). Results are shown Fig. 4.

The Decentralized Q-Learning performs worst. Indeed, the learned policy oscillates because it stores Q_i tables based on the individual actions of the agents, so they update the same Q_i-value both after successful and unsuccessful coordination with the other agent. For example, when both predators are located next to the prey and one predator moves to the prey position, this predator is not able to distinguish between the situation in which the other remains on its current position or moves. Thus the same Q_i-value is updated in both cases, although a positive reward is received in the first case and a large negative one in the second case. So Decentralized Q-Learning is not able to perform the coordination between both predators. Distributed Q-Learning performs better but does not manage to reach the same performance as centralized Q-Learning. Finally, *Hysteretic Q-Learning presents the same convergence as centralized method, although five times less Q-values are used* !

C. Conclusion

To conclude, these two experiments confirms that Hysteretic Q-Learning is able to manage the coordination between agents in cooperative MAS better than other tested algorithms. Without any additional communication, this method performs as well as a centralized algorithm and moreover, uses smaller Q table. On the other hand, it seems that the straightforward application of decentralized Q-Learning to cooperative MAS is able to perform approximately successful coordination, but this is not guaranteed.

VI. Partially observable Stochastic Games.

Finally, we extend our study to partially observable framework. Especially, we test our algorithm in a pursuit domain where 4 agents have partially observable perceptions.

A. Definition

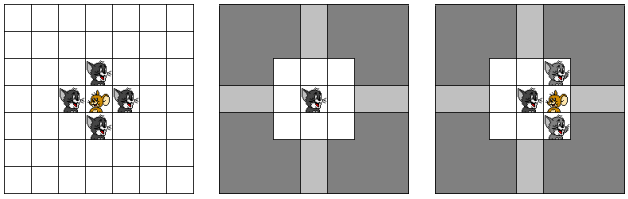
A *partially observable stochastic game* (POSG) is a tuple $< n, S, A_1, ..., A_n, \Gamma_1, ..., \Gamma_n, O, T, R_1, ..., R_n >$ where $< n, S, A_1, ..., A_n, T, R_1, ..., R_n >$ is a SG and:

- Γ_i is a finite set of observations for agent i (and $\Gamma = \prod \Gamma_i$ is the set of joint observations);
- $O : S \times A \times \Gamma \rightarrow [0, 1]$ defines the observations probabilities.

B. Experiments on a pursuit domain

In this section, we use a pursuit domain [18] different from the one previously presented. The task consists in 4 predators situated in a 7×7 toroidal grid. The objective is always to achieve coordination in order to capture a prey by surrounding it (Fig. 5a). Predators and prey are the same as in the other pursuit domain, but predators perceptions differs. Indeed, a predator perceives something according to the 8 cardinal directions and a close or distant criterion (Fig. 5b), so 16 perceptions. Given that each predators perceives its 3 teammates plus the prey, there are 16^4 possible observations per agent. Each agent has 5 possible actions, *i.e.* a Q_i-table of size $16^4 \times 5$ for each predator. An agent receives independently $R_i = 25$ in the situation in Fig. 5c.

We do not test centralized Q-Learning because the size of the state-action space is too big, *i.e.* $16^4 \times 5^4$ Q-values.



(a) Capture (b) Perceptions (c) Reinforcement

Fig. 5. a) The prey is captured. b) $(2 \times 8)^4$ perceptions per agent. c) The reinforcement is attributed in an individual way and is only function of local perceptions (and similar situations obtained by rotation of $90°$).

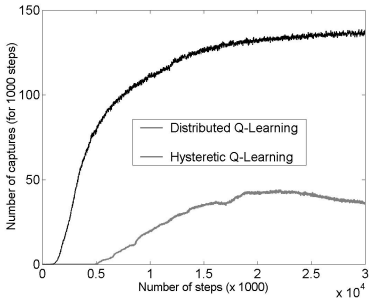


Fig. 6. Number of captures for 1000 steps (averaged over 20 runs) with $\alpha = 0.3$ $\beta = 0.03$ $\gamma = 0.9$ $\epsilon = 0.05$.

Anyway, we experimented with Decentralized Q-Learning and observed that the agents are not able to learn even a sub-optimal strategy after 10.10^6 steps. But with the Hysteretic Q-Learning, the predators manage to achieve coordination to surround the prey much more often than with Distributed Q-Learning (Fig. 6). The coordination learning stabilizes around 5.10^6 steps with 115 captures for 1000 steps.

So Hysteretic Q-Learning manages the coordination in this partially observable environment.

VII. CONCLUSIONS

In this paper, we investigated the issue of developing an algorithm for decentralized RL on the basis of Q-Learning. The major difficulty remains the coordination problem. First, we investigate the case of ILs so as to obtain sizes of Q_i-tables independent of the number of agents and linear in function of their own actions. Furthermore, we are looking for a method which does not require any additional communication.

To overcome the issue of coordination, two learning rates are introduced for the increase and decrease of Q_i-values. Thanks to these learning rates, hysteretic agents are chiefly optimistic to reduce oscillations in the learned policy. Indeed, we have studied here the case of an increase rate largely superior to the decrease rate ($\alpha > \beta$). Anyway, they are not absolutely blind to the received penalties to avoid mis-coordination in case of multiple optimal joint actions.

It has been shown on various multi-agent benchmarks that Hysteretic Q-Learning algorithm achieves successfully coordination's purpose. Actually, Hysteretic Q-Learning's convergence is closed to centralized Q-Learning's. Moreover, computed policies are better than results with existing algorithms. Therefore, Hysteretic Q-Learning is an attractive decentralized RL algorithm for the learning of the coordination in cooperative MAS.

In perspectives, we intend to widen the idea of hysteretic agents to other RL algorithms, especially TD(λ). Indeed, we assume that the idea of hysteretic agents could be implemented in various RL algorithms to achieve coordination in cooperative MAS. Besides, it could be interesting to study the influence of the two learning rate parameters.

REFERENCES

[1] E. Yang and D. Gu, "Multiagent reinforcement learning for multi-robot systems: A survey," Department of Computer Science, University of Essex, Tech. Rep., 2004.

[2] P. Stone and M. M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, 1998.

[4] L. Busoniu, R. Babuska, and B. D. Schutter, "Multi-agent reinforcement learning: A survey," in *Proc. of the 9th ICARCV*, December 2006, pp. 527–532.

[5] A. M. Tehrani, M. S. Kamel, and A. M. Khamis, "Fuzzy reinforcement learning for embedded soccer agents in a multi-agent context," *Int. J. Robot. Autom.*, vol. 21, no. 2, pp. 110–119, 2006.

[6] L. Busoniu, R. Babuska, and B. D. Schutter, "Decentralized reinforcement learning control of a robotic manipulator," in *Proc. of the 9th ICARCV*, Singapore, Dec. 2006, pp. 1347–1352.

[7] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Theoretical Aspects of Rationality and Knowledge*, 1996, pp. 195–201.

[8] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems." in *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998, pp. 746–752.

[9] C. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[10] M. Tan, "Multiagent reinforcement learning: Independent vs. cooperative agents," in *10th International Conference on Machine Learning*, 1993, p. 330 337.

[11] S. Kapetanakis and D. Kudenko, "Improving on the reinforcement learning of coordination in cooperative multi-agent systems," Second Symposium on Adaptive Agents and Multi-Agent Systems(AISB/AAMAS-II), Imperial College, London, April 2002.

[12] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc. 17th ICML*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 535–542.

[13] S. Kapetanakis and D. Kudenko, "Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems," in *Proc. of AAMAS '04*, 2004, pp. 1258–1259.

[14] R. H. Crites and A. G. Barto, "Elevator group control using multiple reinforcement learning agents," *Machine Learning*, vol. 33, no. 2-3, pp. 235–262, 1998.

[15] Y. Wang and C. W. de Silva, "Multi-robot box-pushing: Single-agent q-learning vs. team q-learning," in *Proc. of IROS*, 2006, pp. 3694–3699.

[16] T. Taniguchi and T. Sawaragi, "Adaptive organization of generalized behavioral concepts for autonomous robots: schema-based modular reinforcement learning," in *Proc. of Computational Intelligence in Robotics and Automation*, June 2005, pp. 601–606.

[17] L.Matignon, G. J. Laurent, and N. LeFort-Piat, "Improving reinforcement learning speed for robot control," in *Proc. of IROS*, 2006, pp. 3172–3177.

[18] M. Benda, V. Jagannathan, and R. Dodhiawala, "On optimal cooperation of knowledge sources - an experimental investigation." Boeing Advanced Technology Center, Boeing Computing Services, Seattle,Washington, Tech. Rep. BCS-G2010-280, 1986.