ICIC Express Letters
Volume **10**, Number **12**, December **2016**
 ISSN 1881-803X
pp. **2853–2859**

EVENT-TRIGGERED Q-LEARNING FOR MULTI-AGENT SYSTEMS

Wenxu Zhang, Lei Ma and Xiaodong Wang

School of Electrical Engineering
Southwest Jiaotong University
No. 111, 1st Northern Section, Second Ring Road, Chengdu 610031, P. R. China
wenxu_zhang@163.com

Received April 2016; accepted July 2016

Abstract. *This paper investigates an event-triggered problem for multi-agent systems in reinforcement learning. An event-triggered multi-agent Q-learning algorithm, called ET-MAQL is proposed. Most existing multi-agent reinforcement learning algorithms have high communication cost and computational complexity. To address these problems, we focus on event-triggered multi-agents at the learning strategy layer. During interaction with the environment, the agents trigger communication and learning using the variation rate of observed information. Thus, there is no need for agents to communicate or learn in real time or periodically, resulting in fewer data transmissions. Meanwhile, agents do not need to perform trial and error or iterations at each time instance, thereby reducing resource consumption. Finally, the convergence of the proposed algorithm is analysed. Simulation results show feasibility and validity of the proposed algorithm.*
Keywords: Event-triggered, Multi-agent system, Q-learning, Decentralized partially observable Markov Decision Processes (DEC-POMDPs)

1. **Introduction.** In recent years, event-triggered methods receive more and more attention from multi-agents [1, 2, 3]. The ability of agents to update status periodically using measurement errors reduces transmissions and computational complexity. The use of event-triggered strategies for collaboration in multi-agent systems was pioneered in [4] and later extended to non-linear adaptive dynamic planning [5, 6, 7]. However, most event-triggered reinforcement learning is currently focused on how to design multi-agent controllers based on measurement error [8, 9]. Little attention is paid to their combination with the strategy layer of multi-agent learning. The learning process has two problems because each agent carries resource-limited communication devices and microprocessors. First, information interaction between agents consumes large network bandwidth. Second, trial and error and iterative processes are computationally intensive. In order to solve these problems, this paper focuses on the learning strategy layer by using the advantage of event-triggered, and a distributed Markov model is used to propose a novel event-triggered multi-agent Q-learning algorithm. Unlike traditional multi-agent learning algorithms, there is no need for agents to communicate or learn in real time in learning process by self-triggering and joint-triggering. The optimal strategy of agents is calculated while reducing resource consumption and data transmissions. Finally, the convergence of proposed algorithm is analysed.

2. Problem Statement and Preliminaries.

2.1. Distributed Markov model.

The decentralized partially observable Markov Decision Processes (DEC-POMDPs) problem is concerned with the decision process of a group of agents. It is a tuple: $\langle I, S, A_i, P, \Omega_i, O, R, b^0 \rangle$, where I is a finite set of the agents, S is a finite set of states with designated initial state distribution b^0, A_i is a finite set of actions, $P : S \times \vec{A} \to \triangle S$ is a Markovian transition function, $P\left(s^{'}|s, \vec{a}\right)$ denotes the probability

of a transition from state s to state $s^{'}$, Ω_i is a finite set of observations, $\vec{\Omega} = \otimes_{i \in I} \Omega_i$ is the set of joint observations, $O : \vec{A} \times S \to \triangle\vec{\Omega}$ is an observation function, $O\left(\vec{o}|\vec{a}, s\right)$ denotes the probability of observing joint observation $\vec{o}$ given that the joint action $\vec{a}$ is taken and leads to state $s^{'}$, and $R : \vec{A} \times S \to \mathbb{R}$ is a reward function.

2.2. **Q-Learning.** Q-Learning [10] is a model-independent method of reinforcement learning. The basic form of Q-learning is as follows: $Q^*(s,a) = R\left(s,a\right) + \gamma \Sigma_{s' \in S} P\left(s,a,s^{'}\right) \max Q^*\left(s^{'},a^{'}\right)$ where $Q^*(s,a)$ represents sum of the discount rewards obtained with action a under state s, γ stands for the discount factor, $P\left(s,a,s^{'}\right)$ is the probability function. The main problem with Q-learning is that agents need to find the optimal strategy through trial and error, which consumes significant computational resources.

3. Design of Triggering Rules.

3.1. **Design of self-triggering.** In DEC-POMDP, each agent obtains local information through independent observation and then broadcasts this to the whole team. After the observation at time t, each agent triggers itself to determine whether it should broadcast its observations based on the rate of variation. For agent i, the rate of variation from the observation at t and $t-1$, is defined as: $e_i(t) = \left|o_i(t) - o_i(t-1)\right| / o_i(t-1)$, where $o_i(t)$ is the observed value at t. Let $0 < C < 1$ be a self-triggering threshold. Agent i determines whether to communicate based on the variation rate of observation. There is no need for agents to communicate at every moment in the self-triggering process; this reduces communication overhead.

3.2. **Design of joint-triggering.** Joint triggering is performed on the team of agents, taking account of the variation of a joint observation $O(t) = (O_1(t), O_2(t), \ldots, O_n(t))$ at time t. The variation rate of joint observations at $t-1$ and t for the team is defined as: $E(t) = (e_1(t), e_2(t), \ldots, e_n(t))$. The variation is used to compute the bias error of two moments. Let $F(t) = \Sigma_{i=1}^{n} e_i(t)/n$ be the mathematical expected rate of joint observations, we compute the variance as: $\sum_1^n = \left[e_i(t) - F(t)\right]^2 \cdot p$, where $p = 1/n$ is the distribution law of $e_i(t)$, let $G(t) = \left|G(t) - F(t)\right| / F(t)$. Let $0 < H < 1$ be the joint trigger threshold of the team. When $G(t)$ is above H, the status of the team has changed rapidly, highlighting the need for Q-value lookup and iterations to compute a new joint strategy. Otherwise, agents will continue to use the joint strategy from the previous moment.

Remark 3.1. *Difference between self-trigger and joint-trigger.*

First, self-trigger is enabled by the variation rate of independent observations made by the agents, and the triggered action is to broadcast communication to reduce consumption of communication resources. Joint-trigger takes account of the variation rate of joint observations made by the team of agents, and the triggered action is to compute a joint strategy to reduce the consumption of resources required for calculations.

Second, when the observation variations for an individual agent are larger than the threshold, the variation of joint observations made by the team of agents may not necessarily be over the threshold. When the overall environment changes, the observation of each agent changes correspondingly. However, it is possible for the joint strategy to remain unchanged, as the relative variation rates of all agents are almost the same, making it unnecessary to drive the agents.

4. Event-triggered Q-Learning.

4.1. Proposed algorithm.

An event-triggered DEC-POMDP is composed by a tuple $\langle I, S, A_i, P, \Omega_i, O, R, b^0, e\rangle$ where e denotes the event trigger function. When the trigger function of the agent is above the threshold, the agent will be triggered and state transition will occur based on the transition function P. As shown in Figure 1, unlike classical Q-learning, the agent decides whether to trigger according to the trigger function e, and then executes an action to influence the environment.

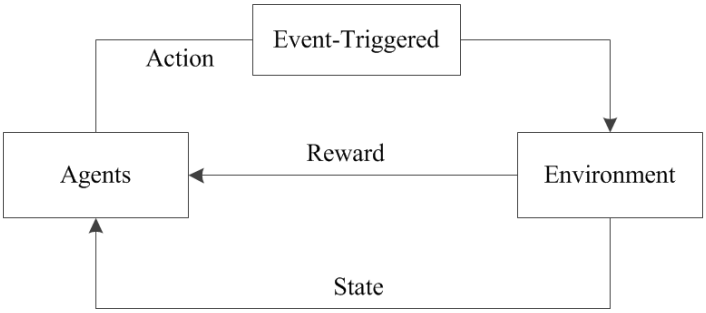

FIGURE 1. The frame of reinforcement learning with event-triggered

The key to Q-learning is to find a strategy that enables the team of agents to maximize the reward. For all states, status-joint action of the optimal strategies has the same optimal function for $(s, \vec{a})$, which we denote as Q^*. Solving DEC-POMDPs for state s^0 can be seen as finding a policy q which maximizes the expected cumulative reward Q:

$$Q\left(s, \vec{q}\right) = R\left(s, \vec{a}\right) + \Sigma_{s' \in S} \Sigma_{\vec{o} \in \vec{\Omega}} P\left(s' | s, \vec{a}\right) O\left(\vec{o} | s', \vec{a}\right) Q\left(s', \vec{q}_{\vec{o}}\right) \tag{1}$$

where $R\left(s, \vec{a}\right)$ is the reward function, $\vec{a}$ is a joint action from strategy $\vec{q}$, $O\left(\vec{o} | s', \vec{a}\right)$ denotes probability of the observation after state s and reaching state s'. For ET-MAQL, the Q value does not need to be computed iteratively unless the agents are driven. Here, we define the Q function as the accumulation of discounted reinforcement values while triggering event e at s_t, executing the joint action $\vec{a}_t$. That is,

$$Q_{t+1}\left(s_t, \vec{a}, e\right) = \gamma_t \cdot \max_{\vec{a}_t} \left\{Q_{t+1}\left(s_t, \vec{a}, e\right) | \vec{a}_t \in A\right\} \tag{2}$$

Given any strategy and the next state, the relationship between values of s and the next state can be expressed as:

$$\begin{aligned} Q^* &= E\left\{r_t \cdot Q_{t+1}\left(s_t, \vec{a}, e\right) | s_t = s, \vec{a} = \vec{a}, e_t = e\right\} \\ &= \textstyle\sum_{s'} P^{\vec{a}}_{ss'} \left[R^{\vec{a}}_{ss'} + \gamma \cdot \max_{\vec{a}} Q^*\left(s', \vec{a}, e\right)\right] \end{aligned} \tag{3}$$

If the agent is not triggered, we directly use the previous Q value as the current Q value.

$$\Delta Q\left(s_t, \vec{a}_t, e\right) = r_t + \gamma \max_{\vec{a} \in A} Q_t\left(s_{t+1}, \vec{a}_t, e\right) - Q\left(s_t, \vec{a}_t, e\right) \tag{4}$$

$$\begin{aligned} Q\left(s_t, \vec{a}_t, e\right) &= Q\left(s_t, \vec{a}_t, e\right) + \alpha Q\left(s_t, \vec{a}_t, e\right) \\ &= Q\left(s_t, \vec{a}_t, e\right) + \alpha \left[r_t + \gamma \max_{\vec{a} \in A} Q_t\left(s_{t+1}, \vec{a}_t, e\right) - Q_t\left(s_{t+1}, \vec{a}_t, e\right)\right] \end{aligned} \tag{5}$$

Equations (4) and (5) show iterations of traditional Q-learning. Similarly, in event-triggered Q-learning, the Q value iteration can be expressed as:

$$Q\left(s_t, \vec{a}_t, e\right) = (1 - \alpha) Q_{t-k}\left(s_t, \vec{a}_t, e\right) + \alpha \left[r_t + \gamma \max_{\vec{a} \in A} Q_t\left(s_{t+1}, \vec{a}_t, e\right)\right] \tag{6}$$

where k is the difference between the previous and current time.

Algorithm 1 Event-triggered Multi-agent Q-Learning

procedure
  Input: $b^0, \alpha, \gamma, C, H$
  foreach $i \in I$
    For $t-1$ to $T-1$ **do**
      o_i^t // receive the local observation from environment
      compare the o_i^{t-1}
      if self-triggering **then**
        $\vec{a} \leftarrow o_i^t$ // broadcast
        if joint-triggering **then**
          $b^t(h^t)$ // calculate the joint belief states by h^t
          $Q^t \leftarrow (s, \vec{a}_t)$ // lookup table
          $\vec{a}_t = (a_i, \ldots, a_n)$ // the joint action
          $\vec{q}_t(\vec{a}_t)$ // the joint strategy
          update Q // learning
        else return to $\vec{q}_{t-1}$ // the previous strategy
      else return to $\vec{q}_{t-1}$
    return q // the optimal strategy
end procedure

4.2. **Analysis of algorithm convergence.** In ET-MAQL, agents do not evaluate the strategy unless their observations change. Consider that at t, the agent is not triggered by an event. Then, the agent will directly use the iterated Q value of the previous moment, rather than iterating in Equation (6). Hence, in the process of finding the optimal strategy, instead of iterating at each moment, the number of Q value iterations is reduced by only iterating when it is triggered by events.

$$\pi_0 \to Q^{\pi_0} \to \pi_1 \to Q^{\pi_1} \to \pi_2 \to Q^{\pi_2} \to \pi_3 \to Q^{\pi_3} \to \cdots \pi^* \tag{7}$$

$$\pi_0 \to Q^{\pi_0} \to \pi_1 \to Q^{\pi_1} \to \pi_2 \to Q^{\pi_2} \to \pi_2 \to Q^{\pi_2} \to \cdots \pi^* \tag{8}$$

As shown in Equation (7), the Q value converges gradually from the initial value to the optimal value Q^*. The iteration process from $t-1$ to t brings the Q value closer to the optimal value. As Equation (8) shows, the Q value at t is not iterated when agents are not driven, and it directly uses the Q value of t, reducing the computational resource of Q value iterations.

Lemma 4.1. *Convergence theorem: Let χ be an arbitrary set and assume that B is the space of bounded functions over χ, $B(\chi)$ i.e., $T : B(\chi) \to B(\chi)$. Let v^* be a fixed point of T and let $\tau = (T_0, T_1, \cdots)$ approximate T at v^* and for initial values from $\mathcal{F}_0(v^*)$, and assume that $\mathcal{F}_0$ is invariant under τ. Let $V_0 \in \mathcal{F}_0(v^*)$, and define $V_{t+1} = T_t(V_t, V_t)$. If there exist random functions $0 \le F_t(x) \le 1$ and $0 \le G_t(x) \le 1$ satisfying the conditions below w.p.1, then V_t converges to v^* w.p.1 in the norm of $b(\chi)$:*

(1). for all U_1 and $U_2 \in \mathcal{F}_0$, and $x \in \chi$, $|T_t(U_t, v^*)(x) - T_t(U, V)(x)| \le G_t(x)|U_1(x) - U_2(x)|$;

(2). for all U and $V \in \mathcal{F}_0$, and all $x \in \chi$, $|T_t(U_t, v^*)(x) - T_t(U, V)(x)| \le F_t(x)(\|v^* - V\| + \lambda_t)$, where $\lambda_t \to 0$ w.p.1. as $t \to \infty$;

(3). for all $k > 0$, $\prod_{t+k}^{n} G_t(x)$ in x $n \to \infty$; and,

(4). there exists $0 \le \gamma < 1$ such that for all $x \in X$ and large enough t, $F_t(x) \le (1 - G_t(x))$.

Corollary 4.1. *The convergence is not influenced by ET-MAQL.*

Proof: In the ET-MAQL, let $T = (T_0, T_1, \ldots T_k, T_{k+1} = T_k, T_t, \ldots)$ be an action sequence, which is a mapping from current state to next state after an action is operated, where $(\ldots T_{k+1}, T_k, \ldots)$ means the T_{k+1}th action equals T_kth action. The iterative process is $f_{t+2} = T_{k+1}(f_{t+1}, f_{t+1}) = T_k(f_t, f_t)$. Let $V, U_0, V_0 \in B_\chi$, $U_{t+1} = T_t(U_t, V)$, $V_{t+1} = T_t(V_t, V^*)$, and $\delta_t(x) = |U_t(x) - V_t(x)|$. According to convergence theorem, we have:

$$\begin{aligned}
\delta_{t+1} &= |U_{t+1}(x) - V_{t+1}(x)| \\
&= |T_t(U_t, v^*)(x) - T_t(V_t, V_t)(x)| \\
&\leq |T_t(U_t, v^*)(x) - T_t(V_t, v^*)(x)| + |T_t(V_t, v^*)(x) - T_t(V_t, V_t)(x)| \\
&= G_t(x)\,\delta_t(x) + F_t(\|v^* - V_t\| + \lambda_t) \\
&\leq G_t(x)\,\delta_t(x) + F_t(\|v^* - V_t\| + \|U_t - V_t\| + \lambda_t) \\
&= G_t(x)\,\delta_t(x) + F_t(x)\,\|v^* - V_t\| + \lambda_t
\end{aligned} \tag{9}$$

Equation (9) meets conditions (1) and (2), although the action sequence T contains same actions T_k and T_{k+1}, it still meets the Lipschitz condition. So the convergence is not influenced.

5. **Simulation Results.** This section studies the multi-agent coverage problem. Two agents are randomly placed in a 10×10 grid, as shown in Figure 2. Each agent has four movement options (forward, backward, left, and right) and it can observe two cells (shadowed part) along each of the four directions. The observed cell is labeled "passed", "not passed", and "obstacle", the probability that the observation is true is 0.9 for each agent, and their corresponding reward are 30, -5 and -10. The boundary of grid is regarded as obstacle. Each agent can broadcast messages. The task of agents is to traverse (i.e., cover) all cells as soon as possible. This task is considered to be accomplished successfully when over 90 of the cells are passed within 1000 steps; otherwise, the task is failure. The learning rate is 0.5, and the discount factor is 0.2. The self-triggering $C = 0.3$. The joint-triggering function $H = 0.05, 0.1, 0.2, 0.3$, respectively.

In Figure 3, the success rate of two algorithms is basically accorded, but the convergence speed of ET-MAQL is slower, because the Q iterative number is reduced. And we can see the relationship between joint triggered function and convergence speed of ET-MAQL.

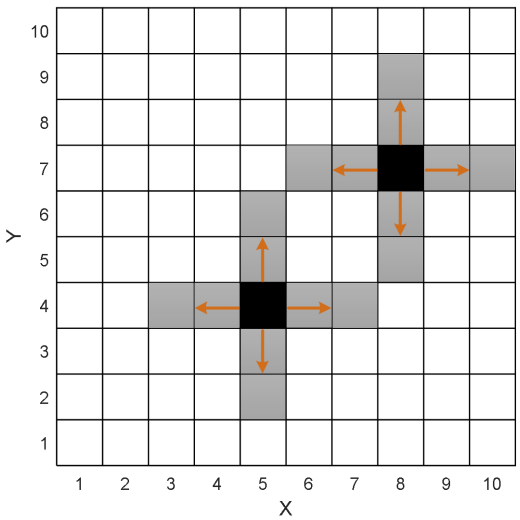

FIGURE 2. The coverage problem of multi-agent

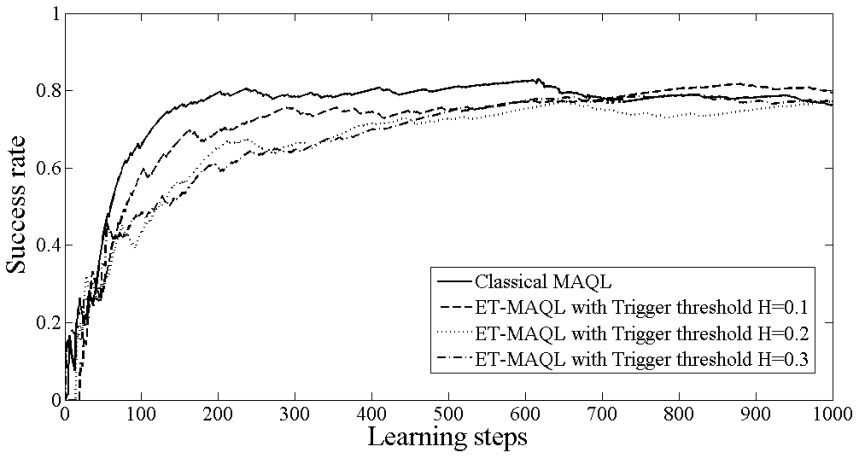

FIGURE 3. The success rate of event-triggered MAQL and classical MAQL

TABLE 1. The number of traverse of classical MAQL and ET-MAQL with $H = 0.1$

Steps	Classical-MAQL	ET-MAQL	Reduced
50	$\approx 2^{29} \times 50$	$\approx 2^{29} \times 41$	$\approx 2^{32.2}$
100	$\approx 2^{29} \times 100$	$\approx 2^{29} \times 77$	$\approx 2^{33.5}$
200	$\approx 2^{29} \times 200$	$\approx 2^{29} \times 151$	$\approx 2^{34.6}$
300	$\approx 2^{29} \times 300$	$\approx 2^{29} \times 218$	$\approx 2^{35.4}$
500	$\approx 2^{29} \times 500$	$\approx 2^{29} \times 376$	$\approx 2^{36}$

The smaller threshold H, the lower convergence speed, because the iterative number is reduced.

In the process of learning, the agent team has to traverse all $(3^8 \times 4)^2 \approx 2^{29}$ Q value to find an optimal one. Table 1 shows that the number of traverse is greatly reduced by ET-MAQL. The above results indicate that, as compared with the classical MAQL, ET-MAQL obtains the same optimal strategy while effectively saving computing resource.

6. **Conclusion and Future Work.** This paper proposed an event-triggered multi-agent Q-learning algorithm, called ET-MAQL. During interaction with the environment, the agents trigger communication and learning using the variation rate of observed information. Thus, there is no need for agents to communicate or learn in real time or periodically, thereby reducing resource consumption and data transmissions. Future work will include extending the proposed algorithm to hierarchical reinforcement learning, and analyze the relationships between ET-MAQL and the computational complexity.

REFERENCES

[1] W. Zhu, Z. P. Jiang and G. Feng, Event-based consensus of multi-agent systems with general linear models, *Automatica*, vol.50, no.2, pp.552-558, 2014.

[2] D. Ding, Z. Wang, B. Shen et al., Event-triggered consensus control for discrete-time stochastic multi-agent systems: Input-to-state stability in probability, *Automatica*, vol.62, pp.284-291, 2015.

[3] D. Yang, W. Ren, X. Liu et al., Decentralized event-triggered consensus for linear multi-agent systems under general directed graphs, *Automatica*, vol.69, pp.242-249, 2016.

[4] P. Tabuada, Event-triggered real-time scheduling of stabilizing control tasks, *IEEE Trans. Automatic Control*, vol.52, no.9 pp.1680-1685, 2007.

[5] H. J. Wang and Q. Q. Xue, Event-triggered guaranteed cost control for a class of nonlinear networked control systems, *ICIC Express Letters, Part B: Applications*, vol.6, no.6, pp.1727-1732, 2015.

[6] S. S. Kia, J. Cortés and S. Martínez, Distributed event-triggered communication for dynamic average consensus in networked systems, *Automatica*, vol.59, pp.112-119, 2015.

[7] A. Wang and T. Dong, Event-triggered synchronization strategy for complex dynamical networks with the Markovian switching topologies, *Neural Networks*, vol.59, pp.52-57, 2016.

[8] X. Zhong, Z. Ni, H. He et al., Event-triggered reinforcement learning approach for unknown nonlinear continuous-time system, *Proc. of International Joint Conference on Neural Networks*, Beijing, China, pp.3677-3684, 2014.

[9] H. Xu and S. Jagannathan, Near optimal event-triggered control of nonlinear continuous-time systems using input and output data, *Proc. of the 11th IEEE World Congress on Intelligent Control and Automation*, Shenyang, China, pp.1799-1804, 2014.

[10] C. Watkins and P. Dayan, Distributed event-triggered control for multi-agent systems, *Machine Learning*, vol.8, no.3, pp.279-292, 1992.

[11] C. Szepesvari and M. L. Littman, A unified analysis of value-function-based reinforcement-learning algorithms, *Neural Computation*, vol.11, no.8, pp.2017-2060, 1999.