Multi-agent Cooperation by Reinforcement Learning with Teammate Modeling and Reward Allotment

ZHOU Pucheng
Department of Information Engineering,
Hefei New Star Applied Technology Research Institute,
Hefei, P.R.China

SHEN Huiyan
School of Environmental and Energy Engineering,
Anhui University of Architecture,
Hefei, P.R.China

***Abstract*—How to coordinate the behavior of different agents through learning is a challenging problem within multi-agent domains. This paper addressed a kind of reinforcement learning algorithm to learn coordinated actions of a group of cooperative agents. This algorithm combines advantages of teammate modeling and reward allotment mechanism in a multi-agent Q-learning framework. The effectiveness of the proposed algorithm is demonstrated using the hunting game.**

Keywords-multi-agent cooperation; reinforcement learning; Q-learning; teammate modeling; reward allotment

I. Introduction

A multiagent system (MAS) consists of a group of agents that can potentially interact with each other, essentially constitutes a complex system. In this paper, the fully cooperative MAS is concerned, in which all of the agents share the same common goal. One of the key problems in such systems is the problem of coordination: how to ensure that the individual decisions of the agents result in jointly optimal decisions for the group [1]. Learning enables MAS to be more flexible and robust, and makes them better able to handle uncertain and changing circumstances. Thus how to coordinate different agents' behaviors by learning so as to achieve the common goal is an important theme for cooperative MAS [2].

Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions with dynamic environments, and it has been applied successfully in many single-agent systems [3~6]. Learning from the environment is robust because agents are directly affected by the dynamics of the environment. Because of these characters, reinforcement learning can provide a robust and natural means for agents to learn how to coordinate their action choices in MAS, and it has become one of the important learning approaches for MAS [7, 8].

However, the application of reinforcement learning to multi-agent domains is not that straightforward. One simple approach is to model the MAS as a single agent and thus treat the joint actions of the agents as single actions. Although this approach would lead to optimal solution, it is infeasible for problems with many agents because of the curse of dimensionality, since the state space and action space grows exponentially in terms of the number of agents. Another approach is that each agent has its own learning mechanism, and just treats other agents as parts of the environment. Due to individual learning, each agent modifies its own policies and behaviors asynchronously, thus the environment is no longer stationary.

In this paper, to remedy these problems mentioned above, a new kind of multi-agent reinforcement learning algorithm is proposed, which combined traditional Q-learning with teammate modeling and reward allotment mechanism. Experimental results have shown the effectiveness of it.

II. Reinforcement Learning

Markov Decision Process (MDP) is generally regarded as the mathematical model of reinforcement learning. A MDP is a quadruple (S, A, T, R), where S is a finite set of states, A is a set of actions, T: $S\times A\times S\rightarrow[0,1]$ is the state transition function that describes the probability $p(s'|s,a)$ of ending up in state s' when performing action a in state s, and R: $S\times A\rightarrow\mathbb{R}$ is reward function that returns a numeric value after taking action a in state s.

An agent's policy is a mapping π: $S\rightarrow A$. The goal of the agent is to find an optimal policy π that could maximize the expected sum of discounted rewards

$$V(s,\pi)=\sum_{t=0}^{\infty}\gamma^t E(r_t\mid\pi,s_t) \tag{1}$$

Where r_t is the scalar reward at time step t, $\gamma\in[0,1]$ is discount factor. Equation (1) can be rewritten as

$$V(s,\pi)=r(s,a_\pi)+\gamma\sum_{s'\in S}p(s'\mid s,a_\pi)V(s',\pi) \tag{2}$$

Where a_π is the action determined by policy π.

According to Dynamic Programming (DP) theory [9], there exists at least a deterministic optimal policy π^*, that for any $s\in S$, it satisfies

$$V(s,\pi^*)=\max_{a\in A}\{r+\gamma\sum_{s'\in S}p(s'|s,a)V(s',\pi^*)\} \tag{3}$$

Q-learning [10] is one kind of model-free reinforcement learning algorithms that based on DP. Q-learning directly computes the approximation of an optimal action-value function, called Q-value, by using the following update rule

$$Q(s_t,a_t)\leftarrow Q(s_t,a_t)+(1-\alpha)\left(r_t+\gamma\max_a Q(s_{t+1},a)-Q(s_t,a_t)\right) \tag{4}$$

Here $\alpha\in[0,1]$ is learning rate.

III. PROPOSED REINFORCEMENT LEARNING ALGORITHM

The following architecture (shown in Fig. 1) for each learning agent is used, which consists of three modules, that is, LM: learning module, which implements traditional Q-learning algorithm; TM: teammate module, which generates a teammate model to estimate its teammate agents' behaviors strategies; ASM: action selection module, which uses a simple decision procedure to select an action for the learning agent.

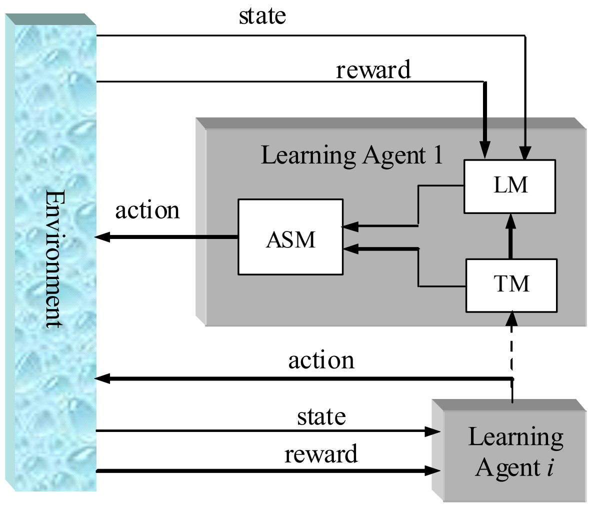


Figure 1. The proposed learning architecture

A. Teammate modeling based on observation

When multiple agents exist in the same environment and work for the same task, the success of an agent's actions depends not only on the environment, but also on the actions of other agents. Thus, it is necessary to predict what other agents are going to do in the future. Such knowledge can help an agent determine which of its current action options are most likely to help it achieve its goal.

There is a long history in adversarial game playing of using a model of an opponent which assumes that is always acts optimally, for example, the minimax search algorithm [11] and the recursive modeling method (RMM) [12]. A limitation of both minimax and RMM is that they rely on knowing the state of other agents and their action capabilities in order to construct payoff matrices, thus are designed for adversarial domains with complete information.

In this paper, a kind of teammate modeling approach that only based on observation is adopted, so as to predict other teammate agents' actions, in cases when communication between agents is either unreliable or even impossible. Since at the initial period of learning, the action of the teammate agent is arbitrary, thus the teammate model should be less accurate initially. After some periods of learning, the action of the teammate agent would be more stable, thus it is more accurate to predict its action of next time. The proposed teammate modeling is described as follows.

At time step *t*, when learning agent *i* under consideration observes that its teammate agent executes an action a^* in state *s*, then agent *i* updates the teammate model for every action a_o executable in state *s* as

$$P_t(s,a_o)=\begin{cases}P_{t-1}(s,a_o)+\beta^{T-t+1}\sum\limits_{a_t\in A_o(s)-\{a_o\}}P_{t-1}(s,a_t), & a_o=a^*\\ \left(1-\beta^{T-t+1}\right)P_{t-1}(s,a_o) & \text{,otherwise}\end{cases}\qquad(5)$$

Where $\beta\in[0,1]$ is learning rate that determines the effect of previous action distribution, and *T* is the time step that the task is finished.

For two learning agents, the teammate modeling process can be described as follows.

(1) At time *t*, the agent under consideration takes action a_s. Simultaneously, the teammate agent performs action a_o. The environment changes from state s_t to a new state s_{t+1}.

(2) After that, the agent under consideration recognizes the action a_o^* performed by its teammate agent at time *t*, and updates its corresponding teammate model for every action a_o executable in state s_t according to equation (5).

B. Reward allotment mechanism

For many multi-agent reinforcement learning problems, they will often spend long time to finish learning, especial for those inherently cooperative tasks, because it is usually difficult to reach successful states at the beginning of learning.

We distinguish two kinds of reward for reinforcement learning, the first one we call it immediate reward, that is, after each action executed, the learning agent will get some scalar reward, and the other one we call it goal reward, that is, when the learning agent reaches the goal state, the reward it will be given. Immediate reward will cause the learning agent to choose other actions, since usually it will be punished if it doesn't reach the goal state using current action. Goal reward will lead the learning agent to the goal state, because it will be rewarded if it using current action selection strategy.

To speed up multi-agent learning process, we propose a kind of reward allotment mechanism, which allot goal reward among relevant state-action pairs (SAPs), so as to speed up the learning process. The proposed reward allotment mechanism is described as follows.

At time step *t*, the learning agent observes current state s_t, an action a_t is then selected from available action set. After executing an action, the agent determines whether a goal reward *R* is received. If not, the agent stores the SAP (s_t, a_t) in its episodic memory, and repeats this cycle until a goal reward is acquired. When a goal reward *R* is given to the agent, all of the SAPs stored in the episodic memory are reinforced by

$$Q(s_t,a_t)\leftarrow Q(s_t,a_t)+D(R,t)\qquad(6)$$

Where *D*(.,.) is the reward allotment function that assigns a goal reward *R* among SAPs in the episodic memory.

To meet with rationality theorem [13] that will avoid those SAPs forming loop, the following reward allotment function is adopted:

$$D(R,t)=\lambda^{T-t-1}R\qquad(7)$$

Where $0<\lambda\leq 1/\max|A(s)|$ is a discount rate, and *R* is the goal reward after the task is finished.

C. Action selection Stragety

Action selection module uses some kind of decision-making procedure to incorporate the results of learning module with teammate modeling, so as to determine a final decision for the corresponding agent to perform. Here we use the following greedy strategy

$$a_t = \arg\max_{a_s \in A_s} \quad V(a_s | a_o^*), \tag{8}$$

Where $V(a_s|a_o^*)$ is the conditional expectation value of action a_s of learning agent when it considers the teammate model:

$$V(a_s | a_o^*) = \begin{cases} \sum_{a_o \in A_o} P(s,a_o) Q(s,a_s,a_o), & t \le T_B \\ \left(\arg\max_{a_o^* \in A_o} P(s,a_o^*) \right) Q(s,a_s,a_o^*), & t > T_B \end{cases} \tag{9}$$

Here T_B is denoted as the episode threshold, and in this paper we set T_B=0.75 × total episodes.

D. Proposed learning algorithm

Let $a_s \in A_S$ and $a_o \in A_o$ are actions of the learning agent under consideration and the teammate agent, respectively. A_s and A_o are the sets of actions available for the learning agent and the teammate agent, respectively. The proposed learning algorithm for each learning agent is followed.

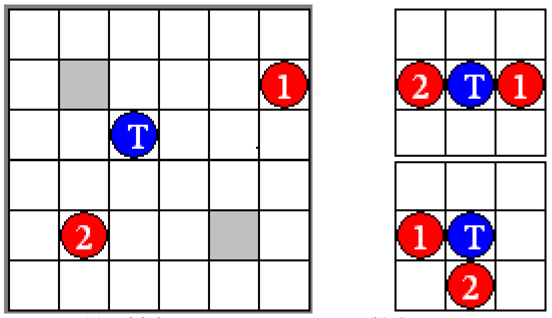
```
1. Initialize. For all $s \in S$, $a_s \in A_s(s)$, $a_o \in A_o(s)$:
       $Q(s,a_s,a_o) \leftarrow \frac{1}{|A_s(s)||A_o(s)|}$
2. Repeat (for each episode)
  1) $t \leftarrow 0$. Empty the episodic memory, initialize the state $s_t$;
  2) Repeat (for each step per episode)
    (1) Observe the current state $s$;
    (2) According to action $a_s$ selected by ASM, save ($s_t$, $a_s$) into
        episodic memory;
    (3) The learning agent performs the action $a_s$. At the same
        time, the teammate agent executes an action $a_o^*$. The
        environment changes to the state $s_{t+1}$, and the learning
        agent gets a immediate reward $r_{t+1}$ from the environment
    (4) If state $s_{t+1}$ is the goal reward state, $t \leftarrow t+1$, go to step 3);
    (5) Update the Q-value, $Q(s_t, a_s, a_o)$:
        $Q(s_t,a_s,a_o) \leftarrow (1-\alpha)Q(s_t,a_s,a_o)$
            $+\alpha\left[ r_{t+1} + \gamma \max_{a' \in A_s} Q(s_{t+1},a',a'_o) \right]$
        Where $a._o = \arg\max_{b \in A_o} P(s_{t+1},b)$ .
    (6) $t \leftarrow t+1$. Go back to (1)
  3) For each SAP in episodic memory, update its Q-value as
     follows
        $Q(s_t,a_t,a_o) \leftarrow Q(s_t,a_t,a_o) + D(R,t)$
     Where $R$ is the goal reward after the task is finished.
```

IV. Simulation Experiments

To evaluate the effectiveness of the proposed multi-agent learning approach, the hunting game which is derived from the well-known pursuit problem is chosen.

A. The hunting game

In a 6×6 grid world, one prey agent and two hunter agents are placed at random positions in the environment, as shown by Fig. 2. Hunters are learning agents, and try to capture the moving prey. At each time step, agents synchronously select and perform one out of five actions without communicating with each other, that is, staying at the current position or moving up, down, left, or right. The prey and hunters cannot share a cell, but two hunters can be at the same position. Also, an agent is not allowed to move off the environment. The gray cells in the world are referred as obstacles. Agents cannot pass through any obstacles, but they can see thru. Each agent has a unique identifier. A hunter can locate the relative position and recognize the identifier of any other agents. The prey is captured, when the vertically or horizontally neighboring cells are occupied by the two hunters, as shown in Fig. 2(b). Then all of the agents are relocated at new random positions and the next trial starts.



(a) Initial state (b) Some capture states

Figure 2. The pursuit problem in a grid world.

B. Experiments results

The simulation run consists of a series of trials. Each trial begins with one prey and two hunter agents placed at random positions, and ends when the prey is captured. A trial is aborted when the prey is not captured within 1000 time steps. Upon capturing the prey, each hunter agent immediately receives a reward of 1.0. On the other hand, hunters receive a reward of –0.05 in any other cases.

The following parameters are used for Q-learning: learning rate α=0.8, discount factor γ=0.9, initial Q-values for individual SAP is set to 0.04, and initial value of each teammate model function is 0.2. All the results given for the following experiments are the average value of 20 distinct runs, and each run consists of 2000 trials.

To test the proposed multi-agent reinforcement learning approach, which is denoted as VPTMQ, two kinds of other reinforcement learning algorithms are selected for comparison, that is, individual Q-learning, denoted as Individual Q, which means each learning agent uses traditional Q-learning algorithm to learn, without taking any other agents into consideration; Q-learning with teammate modeling, denoted as TMQ, which means the learning agent takes its teammate agent action strategies into account using teammate model. Here the unique difference between TMQ and VPTMQ is that the latter uses reward allotment mechanism. The discount rate

λ for reward allotment function is 0.2, and the control factor β for teammate modeling is 0.8.

Fig. 3 gives the result of learning curves of the steps required to capture the prey, where the x axis indicates the number of trials and the y axis indicates the average number of steps for each trial.

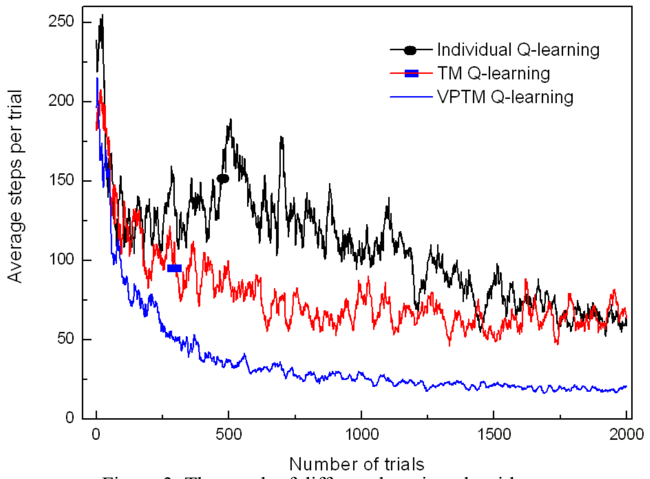


Figure 3. The graph of different learning algorithms

Fig. 4 is the average results of the three learning algorithms within 2000 trails, where the x axis indicates different learning algorithm, and the y axis indicates the average steps of each algorithm required to capture the prey within 2000 trial.

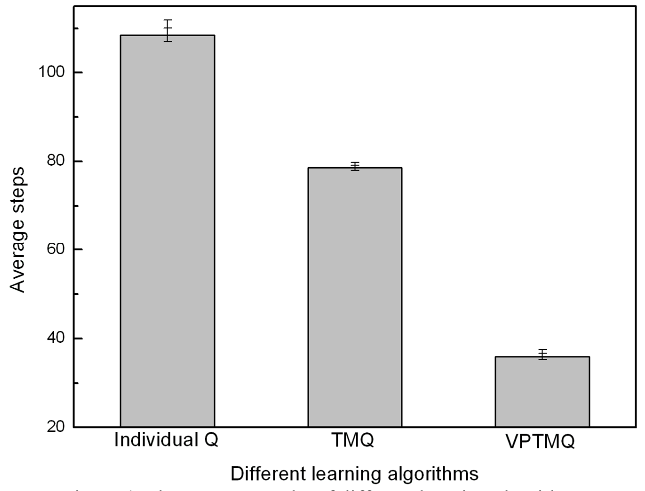


Figure 4. The average results of different learning algorithms

As seen from these figures, it has been demonstrated that the proposed multi-agent reinforcement learning approach at least has two advantages over those with standard Q-learning. First of all, since the other agent's action policy is taken into account by using teammate modeling technique, the acquired policy performs better than Q-learning without using teammate modeling. Secondly, Q-learning combined with reward allotment mechanism can remarkably speed up the learning process, since once the goal reward is obtained, all of the relevant SAPs will be updated. These advantages have shown that the proposed learning approach can be used more effectively to achieve nearer-optimal solutions much faster than standard Q-Learning.

V. CONCLUSIONS

In this paper, to obtain cooperative behavior of different agents through learning, a kind of multi-agent reinforcement learning approach is proposed, which integrated traditional Q-learning with observation-based teammate modeling technique and reward allotment mechanism, so as to deal with various problems encountered by learning agents within cooperative multi-agent domains. Experimental results of the hunting game have shown the effectiveness of the proposed approach.

REFERENCES

[1] L. Panait, S. Luke, "Cooperative multi-agent learning: The state of the art", Autonomous Agents and Multi-Agent Systems, 2005, 11(3): 387-434.

[2] C. Jonqeun, O. Songhwai, H Roberto, "Distributed learning and cooperative control for multi-agent systems", Automatica, 2009, 45(12): 2802-2814.

[3] L. P. Kaelbing, M. L. Littman, A. W. Moore, "Reinforcement learning: A survey", Journal of Artificial Research, 1996, 4(1): 237-285.

[4] R. C. Hsu, C. T. Liu, "A reinforcement learning agent for dynamic power management in embedded systems", Journal of Internet Technology, 2008, 9(4): 347-353.

[5] P. G. Balaji, X. German, D. Srinivasan, "Urban traffic signal control using reinforcement learning agents", IET Intelligent Transport Systems, 2010, 3(3): 177-188.

[6] C. Mahsa, C. S. Woo, "Software agent with reinforcement learning approach for medical image segmentation", Journal of Computer Science and Technology, 2011, 26(2): 247-255.

[7] G. Chen, Zh. Yang, "Coordinating multiple agents via reinforcement learning", Autonomous Agents and Multi-Agent Systems, 2005, 10(3): 273-328.

[8] Y. M. De Hauwere, P. Vrancx, A Nowe, "Generalized learning automata for multi-agent reinforcement learning", AI Communications, 2010, 23(4): 311-324.

[9] Bellman R. Dynamic programming. Princeton, NJ: Princeton University Press, 1957

[10] C. J. Watkins, P. Dayan, "Technical note: Q-learning", Machine learning, 1992, 8: 279-292.

[11] M. Rovatsou, M. G. Laqoudakis, "Minimax search and reinforcement learning for adversarial tetris", Lecture Notes in Computer Science, 2010, Vol. 6040 LNAI, p: 417-422.

[12] A. Ledezma, R. Aler, A. Sanchis, D. Borrajo, "OMBO: An opponent modeling approach", AI Communications, 2009, 22(1): 21-35.

[13] K. Miyazaki, S. Kobayashi, "On the rationality of profit sharing in partially observable markov decision process", Proceedings of the fifth International Conference on Information Systems Analysis and Synthesis. Orlando, USA. 1999: 190-197.