http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 5 (6): 1006-1011, 2006
ISSN 1812-5638

Hybrid Multiagent Reinforcement Learning Approach: The Pursuit Problem

Zhou Pu-Cheng, Hong Bing-Rong, Huang Qing-Cheng and Javaid Khurshid
Research Center of Multiagent and Robotics,
School of Computer Science and Technology
Harbin Institute of Technology, Harbin 150001, People's Republic of China

Abstract: How to coordinate the behaviors of the agents through learning is a challenging problem within cooperative multi-agent domains. As a generic machine learning method, reinforcement learning can suit the needs of multi-agent learning in many aspects. This study presents a novel approach, which using reinforcement learning to learn the coordinated actions of a group of cooperative agents, without requiring explicit communication among them. The proposed approach combines advantages of the modular architecture, profit-sharing learning and opponent modeling technique in a single multi-agent framework. The effectiveness of the technique is demonstrated using the pursuit problem.

Key words: Multi-agent learning, Q-learning, profit-sharing learning, modular architecture, opponent modeling

INTRODUCTION

Multi-agent Systems (MAS) in which there are a number of autonomous agents interacting, with each affecting the actions of the others essentially constitute a complex system. Performing and completing tasks in such an environment can be extremely difficult. In this study we focus on cooperative MAS, in which several agents attempt, through their interaction, to jointly solve tasks or to maximize their utility. 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.

Reinforcement Learning (RL) is the problem faced by an agent that must learn behavior through trial-and-error interactions with environments (Sutton and Barto, 1998). Learning from the environment is robust since agents are directly affected by the dynamics of the environment. For these characters, RL has become one of the important learning approaches for MAS.

In this study, we propose a novel hybrid multiagent RL approach, which combines the modular architecture together with profit-sharing learning and opponent modeling technique in a single multi-agent framework. Experimental results obtained on the pursuit problem show the effectiveness of the proposed learning approach.

THE PURSUIT PROBLEM

The well-known pursuit problem is an appropriate one for illustration of cooperative MAS. In this study, a variant of the pursuit problem is considered, that is, in a 10×10 grid world, one prey agent (T) and four hunter agents (1), (2), (3) and (4) are placed at random positions in the environment, as shown in Fig. 1a. Hunters are learning agents and try to capture the randomly moving prey. At each time, agents synchronously select and perform one out of five actions without communicating with each other: staying at the current position or moving north, south, west, or east. The prey and hunters cannot share a cell, but more than one hunter can be at the same position. Also, an agent is not allowed to move off the environment. Every agent can see objects within a certain distance, which is called the visual depth of the agent. An agent having visual depth d can see all the cells inside the (2d+1)×(2d+1) square around it and these cells are called the visual environment of the agent. Each agent has a unique identifier. A hunter can locate the relative position and recognize the identifier of any other agents in its

Corresponding Author: Zhou Pu-Cheng, Research Center of Multiagent and Robotics,
School of Computer Science and Technology Harbin Institute of Technology,
Harbin 150001, People's Republic of China

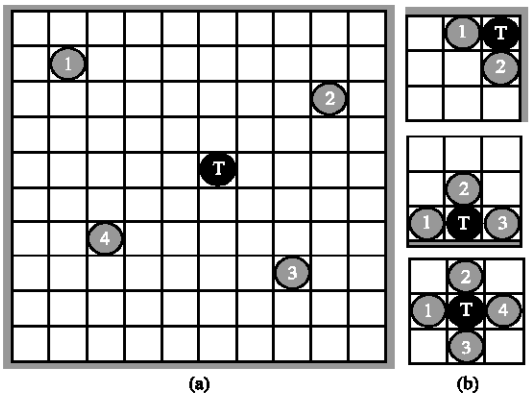

Fig. 1: The pursuit problem in a grid world. (a) The initial state. (b) Examples of capturing states

visual environment. The prey is captured, when all of its neighbour cells are occupied by hunters, as shown in Fig. 1b.

REINFORCEMENT LEARNING

MDP and Q-learning: A Markov Decision Process (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: $S\times A\rightarrow\Re$ 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 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 written 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.

Dynamic Programming (DP) theory (Bellman, 1957) guarantees that there exists at least one policy π^*, for any $s\in S$, it has

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

Q-learning (Watkins and Dayan, 1992) is one model-free RL algorithm that based on DP. Q-learning computes the approximation of an optimal action-value function, called Q-value, by using the following update rule

$$Q(s_{t},a_{t})\leftarrow Q(s_{t},a_{t})+\alpha\times\left(r_{t}+\gamma\max_{a}Q(s_{t+1},a)-Q(s_{t},a_{t})\right) \tag{4}$$

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

Profit-sharing learning: Profit-sharing (PS) learning (Grefenstette, 1988) is one of the RL methods that allow an agent to learn effective behaviors from its experiences within dynamic environments. At time t, the learning agent observes a state s_t, an action a_t is then selected from the action set. After executing an action, the agent determines whether a reward R is generated. If there is no reward, the agent stores the State-action Pair (SAP) so-called rule (s_t, a_t) in its episodic memory and repeats this cycle until a reward is acquired. The process of moving from an initial state to the final reward state is called an episode. When a reward R is given to the agent, the rules stored in the episodic memory are reinforced by

$$w(s_{t},a_{t})\leftarrow w(s_{t},a_{t})+f(R,t) \tag{5}$$

where $w(s_t, a_t)$ is the weight of rule at time t and f is called the reinforcement function that assigns a reward R among rules in the episodic memory.

Rationality theorem (Miyazaki and Kobayashi, 1999) guarantees that ineffective rules, which evermore make up the loop, are always given smaller weight than effective ones, if f satisfies

$$f(R,t)>H\sum_{i=0}^{t-1}f(R,i) \tag{6}$$

where H is the maximum number of conflicting effective rules in the same state.

PROPOSED RL APPROACH

For cooperative MAS, since multiple learning agents have to jointly perform a coordinated task, the state space for each agent grows exponentially in the number of its partners. To overcome the combinatorial explosion of the state space, we adopt a kind of modular learning architecture, shown as Fig. 2. This architecture is based on the modular-Q model presented by Ono and Fukumoto (1996), it is also similar to that of Kaya and Alhajj (2004). However, different learning algorithms are introduced here which will be discussed in detail later on.

The modular architecture in each learning agent consists of three kinds of modules.

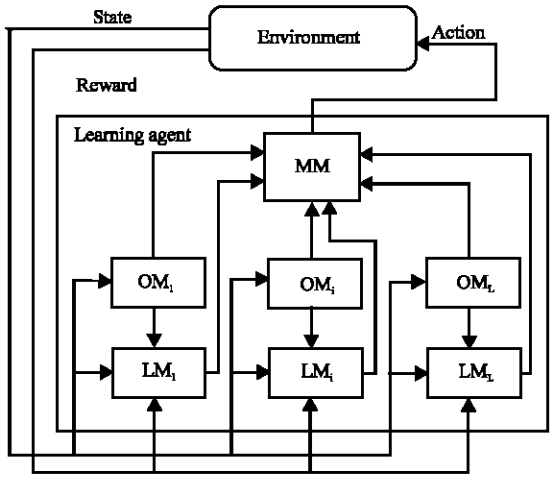

Fig. 2: The proposed modular learning architecture

LM: Learning Module. In each learning module, learning concentrates on a specific agent and carries out RL algorithm in the environment. The number of learning modules, L is equal to the number of learning agents involved in the task.

OM: Opponent Module. The opponent module generates an opponent model, which is the action distribution estimation of the other agent, by observation of its actions and for the learning agent to evaluate its action value.

MM: Mediator Module. The mediator module arbitrates the results of learning modules and opponent modules using a simple decision procedure and determines a final decision by the learning agent.

Opponent modeling: When there are multiple agents in the same environment, the success of the agent's actions depends not only on the environment, but also on the actions of the other agents. In order to succeed, an agent should consider other agents' actions. Opponent modeling studies the prediction of other agents' actions and the generated opponent model can be used by each agent to predict the moves of the others.

In this study, we adopt an opponent modeling approach that based on observation for cooperative agents, in cases when communication between agents is unreliable or even impossible. At time step t, when the learning agent i under consideration observes that the other agent executes an action a^* in state s, then i updates the opponent model for every action a_o executable in state s as follows

$$P_t(s,a_o)=\begin{cases}1-\dfrac{|A_o(s)|-1}{|A_o(s)|}k, & a_o=a^*\\ kP_{t-1}(s,a_o) & \text{, otherwise}\end{cases} \quad (7)$$

where $k\in[0, 1]$ is control factor that determines the effect of previous actions, $|A_o(s)|$ is the number of actions available for the other agent in state s.

The opponent modeling process for two agents, e.g., M (the agent under consideration) and O (the other agent), is followed.

- At time t, M takes action a_m. Simultaneously, O takes action a_o. The environment changes from state s_t to a new state s_{t+1}.
- After that, M recognizes the action a_o^* performed by O at time t by a simple heuristic method: if at two consecutive time step t and t+1, O both can be observed by M, then a_o^* is obtained according to the change of O's position. Otherwise, a_o^* is UNCERTAIN.
- If a_o^* is not UNCERTAIN, then M updates the corresponding opponent model for every action executable in state s_t according to Eq. 7.

Hybrid RL algorithm: Within the modular architecture discussed above, since each learning module only focuses on a part of the environment, it easily leads to perceptual aliasing problem. Besides, owing to multiple learning agents coexisting in the environment and interacting with each other, each agent may change its future action according to other agents' actions. Therefore, the multi-agent environment will no longer obey Markovian property.

PS learning uses trial-and-error experiences and reinforces effective rules, instead of estimating values using the sequential state's value. Therefore, it utilizes this policy to escape states susceptible to perceptual aliasing. It has been proven that PS learning is more suited for dynamic or multi-agent domains than Q-learning (Arai *et al*., 2000). However, for PS learning, a reward is given only after the episodic task is finished. Then all the rules stored in episodic memory which leading to the successful state is reinforced. In other words, PS learning cannot change its knowledge during the learning process until the end of a task.

In order to remedy these problems, we present a kind of hybrid RL approach that combines Q-learning with PS learning and use it as the basic algorithm for each learning module. At the beginning of learning, Q-learning is applied to update Q-value of each SAP. Once the successful state is achieved, PS learning is implemented

to reinforce each SAP which leads to the reward state. After that, Q-learning is adopted again. Let $a_s \in A_s$ and $a_o \in A_o$ and are actions of the learning agent under consideration and the other agent, respectively. A_s and A_o are the sets of actions that are available for the learning agent and the other agent, respectively. The proposed hybrid RL algorithm for each learning module is followed.

Initialize. For all: $s \in S$, $a_s \in A_s(s)$ $a_o \in A_o(s)$:
$Q(s, a_s, a_o) \leftarrow$ a small constant.
Repeat (for each episode)
1) $t \leftarrow 0$. Empty the episodic memory
2) Repeat (for each step per episode)
(1) Observe the current state s_t
(2) Save (s_t, a_s) into episodic memory, where a_s is the action selected by the mediator module.
(3) The learning agent performs action a_s. At the same time, the other agent executes action a_o^*. The environment changes to state s_{t+1} and the learning agent gets a reward r_{t+1}.
(4) If the new state s_{t+1} is the reward state, $T \leftarrow t+1$, go to step 3)
(5) Update the Q-value, $Q(s_t, a_s, a_o)$:
If $a_o \neq$ UNCERTAIN

$$Q(s_t,a_s,a_o) \leftarrow (1-\alpha)Q(s_t,a_s,a_o) + \alpha\left[r_{t+1} + \gamma \max_{a' \in A_s} Q(s_{t+1},a',a'_o)\right]$$

where $a'_o = \arg\max_{b \in A_o} P(s_{t+1}, b)$.
If a_o = UNCERTAIN
For aua$_o \in A_o$, update as follows:

$$Q(s_t,a_s,a_o) \leftarrow (1-\alpha)Q(s_t,a_s,a_o) + \alpha\left[r_{t+1} + \gamma \max_{a' \in A_s} EQ(s_{t+1},a')\right]$$

where $EQ(s_{t+1},a') = \sum_{b \in A_o} P(s_{t+1},b)Q(s_{t+1},a',b)$

(6) $t \leftarrow t+1$. Go back to (1)
3) For each SAP (s_t, a_t) in episodic memory, update its Q-value as follows:

$$Q(s_t,a_t,a_o) \leftarrow Q(s_t,a_t,a_o) + f(R,t)$$

where R is the received reward after the task is finished, that is, $R = r_T$. To satisfy Eq. 6, the following function is adopted:

$$f(R,t) = \lambda^{T-t-1}R \quad (8)$$

where $0 < \lambda \leq 1/\max|A(s)|$ is discount rate.

Multi-agent learning process: The multi-agent learning process is followed:

1. Each learning module and opponent module observes the current state s_t.
2. Each opponent module sends action estimation value of the other agent and each learning module sends Q-values to mediator module.
3. Mediator module selects an action a_s by
$$a_s = \arg\max_{a \in A_s(s_k)} \sum_{k=1}^{L} \sum_{a_o \in A_o(s_k)} P(s_k,a_o)Q(s_k,a,a_o)$$
where s_k is the state of the kth learning module.
Then, each learning module saves the chosen SAP (s_t, a_s) in episodic memory.
4. The agent under consideration executes action a_s. Simultaneously, the other agent executes an action a_o. The environment changes to a new state s_{t+1} and the learning agent receives a reward r_{t+1}.
5. The learning agent under consideration recognizes the action a'_o of the other agent executed just now and updates its opponent model $P(s_t, a'_o)$.
6. If state s_{t+1} is the terminal state, then go to step 8.
7. The learning agent under consideration updates the Q-function, $Q(s_t, a_s, a'_o)$. $t \leftarrow t+1$. Go to step 1.
8. If the state s_{t+1} is a reward state, then assign the received reward R with PS learning.
9. If the algorithm satisfies a stopping condition, the learning process is over. Otherwise, terminate the current trial and go back to step 1.

SIMULATIONS

To evaluate the proposed hybrid multiagent RL approach, the pursuit problem is chosen, as shown in Fig. 1. The simulation run consists of a series of trials. Each trial begins with one prey and four hunter agents placed at random positions and ends when the prey is captured. A trial is aborted when the prey is not captured within 1500 time steps. Upon capturing the prey, each hunter immediately receives a reward of 1.0 and accordingly all of its constituent learning modules uniformly receive the same reward. On the other hand, hunters receive a reward of -0.1 in any other cases.

The visual depth of each hunter is set to d = 3. Each learning module only considers the relative position of the prey and that of a partner, so the size of the state space is about $|S| = ((2d+1)^2+1)^2 = 2500$ and the maximum size of the state space for each hunter is 7500. The following

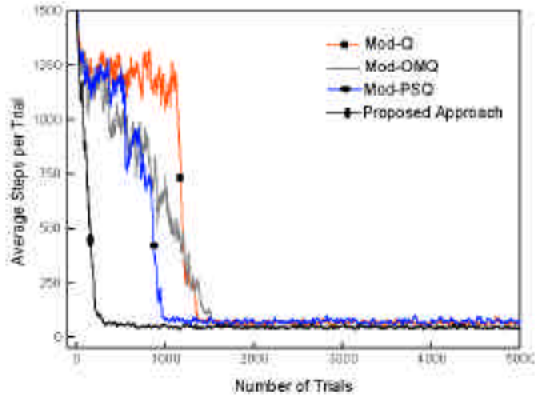

Fig. 4: Comparison of different algorithms

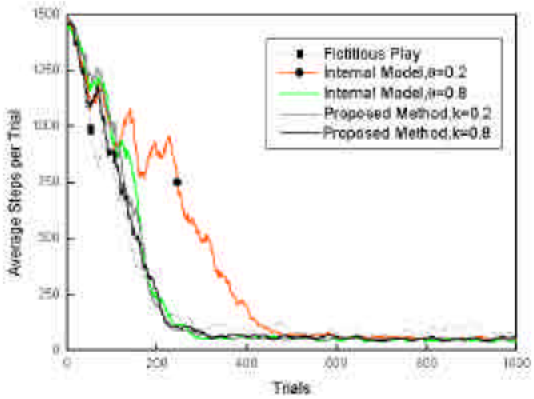

Fig. 5: Comparison of different opponent modeling methods

Table 1: Comparison of learning results

Learning interval (Trials)	Learning algorithm			
	Mod-Q	Mod-OMQ	Mod-PSQ	Proposed approach
1 ~ 500	1243.5	1139.6	1207.8	393.4
500 ~ 1000	1194.5	827.7	687.4	53.7
1000 ~ 2000	294.8	202.7	75.8	49.1
2000 ~ 3000	68.6	50.1	75.5	48.1
3000 ~ 4000	68.5	49.8	74.6	47.2
4000 ~ 5000	67.9	49.3	72.1	46.3

parameters are used: learning rate $\alpha = 0.8$, discount factor $\gamma = 0.9$, the initial Q-values for individual SAP is 0.1 and initial value of each opponent model function is 0.2. The discount rate λ for PS learning is set to 0.2 and the control factor k for opponent modeling is 0.8. All the results given for the following experiments are the average value of 10 distinct runs and each run consists of 5000 trials.

Firstly, we compare our multi-agent modular RL approach with three other algorithms, i.e., modular Q-learning proposed by Ono, denoted as Mod-Q, modular Q-learning with opponent modeling, denoted as Mod-OMQ and modular Q-learning with PS learning, denoted as Mod-PSQ, respectively. Here Mod-OMQ is derived from Kaya's work, but without using fuzzy logic. Figure 4 is the learning curves of the four learning algorithms by time. Table 1 is the average steps for capturing the prey within different learning intervals.

As seen from the figure and the table given above, it has been demonstrated that the proposed modular RL approach at least has two important advantages over those with standard Q-learning. Firstly, Q-learning combined with PS learning can remarkably speed up the learning process. Experiments with the proposed approach have improved the convergence speed by approximately four to five times, when compared with modular Q-learning with or without opponent model. Similarly, Mod-Q with PS learning also has improved convergence speed by about 25%, compared with Mod-Q. Secondly, since the other agent's policy is also taken into account by opponent modeling technique, the acquired policy is much better than that of those works without using opponent modeling, such as Ono's work. For example, the number of steps required to capture the prey has demonstrated a decrease of up to 27-31% in our experiments. All these advantages have shown that the proposed learning approach can be used more effectively to achieve nearer-optimal solutions.

To evaluate the given opponent modeling method, two other methods are integrated with the proposed modular RL architecture, i.e., fictitious play based opponent modeling Q-learning (Claus and Boutilier, 1998) and the internal model (Nagayuki *et al*., 2000). Parameter settings are the same as above and the obtained learning results for the first 1000 trials are shown in Fig. 5.

From Fig. 5 it can be seen that the proposed opponent modeling method is more robust than the internal model, since it is less sensitive to the change of the parameter. Besides, despite that fictitious play based opponent modeling Q-learning has approximate convergence speed, the acquired policy is inferior to that of the proposed approach.

CONCLUSION

In this study, a novel hybrid RL approach is proposed to handle various problems encountered by learning agents in cooperative multi-agent domains. The proposed approach uses modular architecture to cope with the combinatorial explosion of the state space. In order to remedy the non-Markovian property inherently for multi-agent learning and to speed up the convergence speed, PS learning is introduced into standard Q-learning. Since modeling other learning agents coexisting in the environment is unrealistic in real applications, an

opponent modeling technique, which is based on the observation of other agents' past actions is used to predict other agents' actions distribution probability. Experimental results have shown the effectiveness of the proposed approach.

REFERENCES

Arai, S., K.P. Sycara and T.R. Payne, 2000. Experience-based reinforcement learning to acquire effective behavior in a multi-agent domain. Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence, pp: 125-135.

Bellman, R., 1957. Dynamic Programming. Princeton: Princeton University Press.

Claus, C. and C. Boutilier, 1998. The dynamics of reinforcement learning in cooperative mutliagent systems. Proceedings of AAAI-98, pp: 746-752.

Grefenstette, J.J., 1988. Credit assignment in rule discovery systems based on genetic algorithms. Machine Learning, 3: 225-245.

Kaya, M. and R. Alhajj, 2004. Modular fuzzy-reinforcement learning approach with internal model capabilities for multiagent systems. IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, 34: 1210-1223.

Miyazaki, K. and S. Kobayashi, 1999. On the rationality of profit sharing in partially observable markov decision process. Proceedings of the 5th International Conference on Information Systems Analysis and Synthesis, pp: 190-197.

Nagayuki, Y., S. Ishii and K. Doya, 2000. Multi-agent reinforcement learning: An approach based on the other agent's internal model. Proceedings of IEEE ICMAS, pp: 215-221.

Ono, N. and K. Fukumoto, 1996. Multi-agent reinforcement learning: A modular approach. Proceedings of the 2nd International Conference on Multi-agent Systems, pp: 252-258.

Sutton, R.S. and A.G. Barto, 1998. Reinforcement Learning: An Introduction. Cambridge: MIT Press.

Watkins, C.J.C.H. and P. Dayan, 1992. Technical note: Q-learning. Machine Learning, 8: 279-292.