Optimal Tracking Agent: A New Framework for Multi-Agent Reinforcement Learning

Weihua Cao, Gang Chen, Xin Chen, Min Wu
School of Information Science and Engineering, Central South University, Changsha 410083 China
Email: chen_gang@csu.edu.cn

Abstract—To cope with the curse of dimensionality, an ubiquitous problem in multi-agent reinforcement learning, this paper deals with the multi-agent learning in a new perspective and proposes a new algorithm, the optimal tracking agent (OTA). The OTA treats the other agents as a part of the system and uses an estimator to track the dynamics of the system. Thus, it obtains the dynamic model with limit accuracy and uses the model-based reinforcement learning to react optimally to the system. All the processes are just from one agent’s perspective, then the searching space for action is just its own and not exponential with the number of agents any more. Thus, the curse of dimensionality is relieved from action space. Experiment illustrates the validity and efficiency of the proposed method.

Keywords—multi-agent system; curse of dimensionality; optimal tracking agent; estimator.

I. INTRODUCTION

Multi-Agent Systems (MASs) consist of a group of agents, which interact with each other autonomously in a common environment to optimize a performance measure [1], [2]. MASs can be applied to many variety of domains including robotic teams [3], air traffic management [4] and product delivery [5]. They provide various new perspectives of looking at MAS and push forward its development rapidly.

This paper focuses on fully cooperative multi-agent systems, in which all agents have the same reward function. A key problem in such systems is coordination: the process that ensures that the individual behaviors of the agents result in optimal joint behaviors for the group [6]. The last few years, we have witnessed increasing interest and progresses in extending reinforcement learning (RL) to multi-agent systems in the powerful framework of Markov games, proposed kinds of useful multi-agent reinforcement learning (MARL) architectures, such as FFQ [7], OAL [8], Rmax [9]. Most of these architectures can obtain the coordinated optimal joint behavior and provide certain convergence guarantees as well.

However, to ensure the stationary of environment and obtain the optimal joint behavior, most algorithms employ the joint state and joint action form and suffer from the "curse of dimensionality", which means that the learning and storing space grows exponentially as the number of agents increases [10]. Such enlargement in RL causes the learning speed to decrease drastically and a large amount of memory to be required [6]. A broad spectrum of approaches to tackle this problem has been studied, which can be coarsely classified as indirect and direct approaches.

Indirect approaches attempt to decrease the state space through joint state space and (or) joint action space. Such as hierarchical MARL, by decomposing the overall task into a hierarchy of subtasks, these algorithms limit the state space in each subtask [11]. However, the hierarchical structure usually demands enough prior knowledge or communication. Direct approaches try to reduce the state space through the number of agents. An example of such algorithm is Coordination Graph (CG) [6]. In the graph, each node represents an agent, and an edge means that the corresponding agents have to coordinate their behaviors. However, the graph constructed by Variable Elimination (VE) is complex and time-consuming. To relieve the problem, [12] and [13] introduce hierarchy and communication respectively. Frequency Maximum Q-value (FMQ) is not exponential with the number of agents but only works in deterministic environment.

Shoham proposes “AI Agenda” as a more appropriate and promising direction for MARL algorithms, which assumes that the other agents in the system following a stationary behavior policy, and the goal is to learn the agenda’s own optimal behavior in the presence of other agents. Wernber and Rosenschein [14] extend this method to the agents whose policies are non-stationary with a limit. Though, these methods do not need to consider equilibrium selection problem based on some specific assumptions, they still fall into the curse of dimensionality.

Inspired by the “AI Agenda”, this paper proposes Optimal Tracking Agent (OTA), which does not need any prior knowledge or communication like several aforementioned works. This method treats the other agents and the environment as a whole, namely a generalized environment. However, the method does not need to assume that the other agents’ policies are stationary or no-stationary with a limit like [14], [19]. Based on estimating the other agents’ policies and tracking the dynamics of the generalized environment, this method relives the curse of dimensionality to some extent. Hereafter, OTA is our proposed method and also the learning agent since the agent adopts the OTA method.

The paper is organized as follows. In Section II, literature
review is discussed on reinforcement learning in single-agent and multi-agent settings. Section III details how the OTA is derived and implemented. In Section IV, we verify the proposed algorithm in classical cooperative multi-agent systems and conclude in Section V with some final remarks.

II. RELATED BACKGROUND

In this section, the necessary background on single-agent and multi-agent RL is introduced. First, the single-agent RL task and its solution are described. Then, the multi-agent RL is defined with necessary game-theoretic concepts.

A. Reinforcement Learning

In single-agent case, the RL usually can be described as the Markov decision process, in which an agent must choose the sequence of actions that maximize some reward-based optimization criterion \cite{15}.

**Definition 1**: A finite Markov decision process (MDP) is a 4-tuple \( M = (S, A, T, R) \), where

1. \( S \) represents a finite set of states;
2. \( A \) represents a finite set of actions;
3. \( T: S \times A \times S \rightarrow [0,1] \) represents the transition probabilities from state \( s \) to state \( s' \) when action \( a \) is taken;
4. \( R: S \times A \times S \rightarrow \mathbb{R} \) is a reward function giving the immediate reward or reinforcement received under each transition.

The agent must choose its actions so as to maximize the reward function

\[
V(s) = E \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \right],
\]

where \( r_{t+k+1} \) represents the immediate reward from state \( s \) to \( s' \), and the scalar \( \gamma \) \((0 \leq \gamma < 1)\) is a discount factor.

Solving MDP consists in finding a mapping from state to action which is called a policy \( \pi: S \times A \rightarrow [0,1] \). \( \pi(s,a) \) is the probability of choosing action \( a \) in state \( s \). For any finite MDP, there is at least one policy \( \pi^* \) such that

\[
V_{\pi^*}(s) \geq V_{\pi}(s)
\]

for any policy \( \pi \) in state \( s \). Such policy is an optimal policy, and the corresponding value function is compactly denoted as \( V^*(s) \).

Most of the RL algorithms can be placed within the framework of MDP. The famous one is Q-learning. The Q-function can be written as

\[
Q^*(s,a) = r(s,a) + \gamma \sum_{s'} T(s'|s,a)V^*(s'),
\]

where \( V^*(s) = \max_a Q^*(s,a) \). Q-learning allows (3) to be approximated from empirical samples obtained from the actual experiences, and the updating rule is

\[
Q_{k+1}(s,a) = (1 - \alpha_k)Q_k(s,a) + \alpha_k \left[ r_{k+1} + \gamma \max_{a'} Q_k(s',a') \right],
\]

where \( Q_k(s,a) \) is the \( k \)-th estimate of \( Q^*(s,a) \), \( r_{k+1} \) is sampled with mean \( R(s,a) \), and \( \alpha_k \) is a step-size sequence. The sequence \( Q_k \) provably converges to \( Q^\star \) under suitable conditions without the information of the transition and reward functions \cite{10}. Thus, this method is named as model-free RL. The other more efficient learning is model-based RL. It estimates the function of transition \( T \) and the reward function \( R \) through experiences and uses Dynamical Programming (DP) to calculate the Q-function. Due to record the information that the agent experienced, it can learning the optimal policy quickly.

B. Multi-Agent Reinforcement Learning and Markov Games

The prevalent generalization of the MDP to MAS is the Markov games \cite{7}.

**Definition 2**: A Markov game (MG) is a 5-tuple \( \Gamma = (n, S, A, T, R) \), where

1. \( n \) is the number of agents;
2. \( S \) is a finite set of states;
3. \( A = A_1 \times \cdots \times A_n \) is the joint action space;
4. \( T: S \times A \times S \rightarrow [0,1] \) is a transition function that defines transition probabilities between states;
5. \( R^i: S \times A \times S \rightarrow \mathbb{R} \) is the reward for agent \( i \).

In MAS, the transition probabilities and rewards depend on the joint actions of all the agents and each agent has its own reward function, which is the main difference between MG and MDP. Notice that an individual policy is \( \pi^i: S \times A_i \rightarrow [0,1] \) for agent \( i \), and a joint policy is a vector \( \pi = (\pi^1, \cdots, \pi^n) \) of individual policy.

Several multi-agent Q-learning algorithms have been proved to converge in the limit to optimal policy \cite{7}-\cite{9}. Most of the MARL algorithms use the Q-function as

\[
Q_{k+1}(s,a) = (1 - \alpha_k)Q_k(s,a) + \alpha_k \left[ r_{k+1} + \gamma \max_{a'} Q_k(s',a') \right],
\]

where \( s \) and \( a \) represent the joint state vector (this paper does not deal with \( s \), so \( s \) is represented as \( s \) for simplicity) and the joint action vector respectively, and different algorithm has different form of \( V_k(s') \). However, some of these algorithms root in MG, focusing on the convergence only, and suffer from the curse of dimensionality when scaled to large numbers of agents, which may result in huge memory requirement and insufferable slowness in learning speed. Such problems have prevented the MARL algorithms from being scaled up to real-life multi-agent problems, where the state and action spaces are large or even continuous. Until now, these problems have not been solved properly.
III. OTA: OPTIMAL TRACKING AGENT AS A NEW MULTI-AGENT FRAMEWORK

In this section, the formal derivation and realization of the proposed framework are detailed.

Due to the existence of the other agents, the multi-agent environment is filled with dynamics. Thus, neither single-agent RL, which assumes the environment is stationary, nor the conventional MARL or joint-agent RL, which explores joint action space and suffers from curse of dimensionality, is appropriate to be employed.

However, the OTA adopts a new perspective to deal with the other agents and the environment. The OTA treats the other agents and the environment as a whole dynamic system to track and it tries to track the system quickly and accurately, which is much like tracking control in control theory and different from the conventional framework MG.

A. Framework of OTA

In MASs, the reward obtained from system is depended on all the agents’ actions. The OTA’s aim is to obtain the maximum reward from system. The main idea of this framework is that the OTA is to track and adapt to the system to get the maximum individual interest. In cooperative MASs, the individual interest is consistence with collective interest.

The immediate reward for OTA, hereinafter referred to as agent $i$, is represented as $r(s, a^i)$. Thus, like classical RL, agent $i$’s aim is to maximize the reward function

$$V^i(s) = E \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}(s, a^i) \right].$$

(6)

However, since the immediate reward for agent $i$ is depended on all the agents’ actions, $r(s, a^i)$ can not be directly obtained but $r(s, a)$. Thus, the following definition is introduced.

$$R(s, a^i) = E \left[ r(s, \pi^i, \pi^{-i}) \right]$$

$$= \sum_{a^i} \pi(s, a^{-i}) r(s, a)$$

$$= \sum_{a^{-i}} \pi(s, a^{-i}) r(s, a^i, a^{-i}),$$

(7)

where $-i$ indicates the agents except agent $i$, and

$$\pi(s, a^{-i}) = \sum_{a^i} \sum_{a_{-i}} \prod_{i=1, j \neq i} \pi(s, a^j).$$

Thus, $R(s, a^i)$ is an expected immediate reward considering the other agents’ policies.

Hence, for agent $i$, the immediate reward $r(s, a^i)$ is replaced with the expected immediate reward $R(s, a^i)$. Then, (6) is rewritten as follows:

$$V^i(s) = E \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}(s, a^i) \right]$$

$$= E \left[ R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2}(s, a^i) \right]$$

$$= \sum_{a^i} \pi(s, a^i) \sum_{s'} T(s'|s, a^i)$$

$$\times \left[ R_{t+1} + \gamma E \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+2}(s, a^i) \right] \right]$$

$$= \sum_{a^i} \pi(s, a^i) \sum_{s'} T(s'|s, a^i) \left[ R_{t+1} + \gamma V^i(s') \right],$$

(8)

where $T(s'|s, a^i) = \sum_{a^{-i}} \pi(s, a^{-i}) T(s'|s, a^i, a^{-i})$.

Then, to show our new $Q$-function, replacing $R_{t+1}$ with $R(s, a^i)$ for clearly, the optimal $V^i(s)$, which means the value of $s$ assuming the best action is adopted, is represented as $V^{*i}$ and it satisfies the following equation:

$$V^{*i}(s) = \max_{a^i} \sum_{s'} T(s'|s, a^i) \left[ R(s, a^i) + \gamma V^{*i}(s') \right].$$

(9)

Thus, like the conventional RL, the optimal equation for $Q^i(s, a^i)$, which indicates the expected return of taking $a^i$ in joint state $s$ for agent $i$, is defined as

$$Q(s, a^i) = \sum_{s'} T(s'|s, a^i) \left[ R(s, a^i) + \gamma V^{*i}(s') \right]$$

$$= R(s, a^i) + \gamma \sum_{s'} T(s'|s, a^i) V^{*i}(s')$$

$$= \sum_{a^{-i}} \pi(s, a^{-i}) r(s, a)$$

$$+ \gamma \sum_{s'} \sum_{a^i} T(s'|s, a^i) V^{*i}(s'),$$

(10)

where $V^{*i}(s') = \max_{a^i} Q^i(s', a^i)$. Thus, the optimal policy for agent $i$ is $\pi^*(s) = \arg \max_{a^i} Q^i(s, a^i)$.

Assume that all the agents’ action spaces are the same, which indicates that $|A^1| = |A^2| = \cdots = |A^n| = |A|$. In (10), except the immediate reward $r(s, a)$, the dimensions of the other terms are just $|A|$ (without considering the state space in this paper), much smaller than the conventional MARL algorithms, whose learning space is $|A|^n$, which is exponential with the number of agents $n$ and gives rise to the curse of dimensionality. Since the immediate reward $r(s, a)$ in RL is sparse, there always exist methods to deal with it, such as compress in image processing. Consequently, this new framework does not have severe curse of dimensionality.

Bowling and Veloso [16] proposed two specific requirements for effective learning in MASs with their two criteria of rationality and convergence:
**Rationality:** If the other agents’ policies converge to stationary policies, then the learning algorithm will converge to a stationary policy that is a best-response (in the stage game) to the other agents’ policies.

**Convergence:** The learner will necessarily converge to a stationary policy against the other agents using an algorithm from some class of learning algorithms.

According to (10), since
\[
T'(s|s,a) = \sum a' \pi(s,a^{-i})T(s'|s,a',a^{-i})
\]
and
\[
T(s|s,a,a')
\]
is a stationary process, \(T(s'|s,a')\) will become stationary when the other agents’ policies converge using a learning algorithm. Thus, according to Bellman’s principle of optimality, the proposed algorithm satisfies the two requirements. More generally, when the other agents are irrational whether using learning algorithm or not, as long as the other agents’ policies are observable and predictable, the optimal tracking agent always converges to the optimal policy relatively. Therefore, the OTA can be treated as a more general framework for multi-agent learning problems. For OTA, the key point is the estimator \(\pi(s,a')\) that estimates the other agents’ policies.

**B. Implementation of OTA**

Based on the framework of OTA in the previous section, the estimator is very important for learning the optimal policy. Since the environment is dynamic from agent i’s perspective, classical Q-learning, which assumes the environment is stationary and then employs model-free form, is not appropriate. Also, existing model-based RL methods are not suitable either, since they hold the same assumption as model-free ones. However, since the model-based methods capture transitions of environment though stationary, it is possible to use a new estimator to track the dynamic transitions.

Along with this idea, if the dynamics of the system can be estimated and predicted accurately and timely, then the model is accurate and stable to some extent. Thus, the environment can be viewed as stationary and the optimal policy can be calculated through DP.

Since the dynamics are caused by the changes of other agents’ policies, the key point is to estimate and predict their policies, which is very like the tracking control in control theory.

From (10), there are two parameters to be estimated, which are \(\pi(s,a^{-i})\) and \(T(s'|s,a')\). These two terms hold an implication relationship, that is
\[
T'(s|s,a') = \sum a' \pi(s,a^{-i})T(s'|s,a',a^{-i})
\]
Since \(T(s|s,a',a^{-i})\) is a stationary parameter representing the stationary environment without agents, \(T(s'|s,a')\) is just a variant of \(\pi(s,a^{-i})\) in nature. Thus, these two parameters can be estimated in the same way.

Since the policy function \(\pi(s,a')\) of Markov Chain is non-stationary, an estimator equipped with the learning ability is employed. That is Robins-Monro Estimator (RME) [17]. It can be viewed as a stochastic learning automaton, which can approximate stochastic process as soon as possible.

For agent \(i\), it maintains the estimated policy function \(P(s,a')\) for agent \(j\), where \(j = 1, 2, \cdots, n, j \neq i\). It estimates and predicts the others’ agents’ policies based on the past observations of the others’ actions.

When the agent \(i\) observes that the agent \(j\) adopts an action \(a_j^k\), where \(a_j^k \in A_j\). The function \(P(s,a')\) is updated for every available action in \(A_j\) for agent \(j\) in the state \(s\) as follows

\[
P(s,a_j^k) = (1 - \mu)P(s,a_j^k) + \begin{cases} 
\mu, \text{if } a_j = a_j^k \\
0, \text{else}
\end{cases}, \quad (11)
\]

where \(\mu(0 \leq \mu \leq 1)\) controls the executed action’s influence on function \(P(s,a')\). If the other agents’ policies are stationary and the parameter \(\mu\) decreases as an inverse of the observation times, then, \(P(s,a_j^k)\) converges to the empirical probability that the agent executes action \(a_j^k\) in \(s\) [17]. When the other agents’ policies change with time due to learning or designed in advance, the learning function \(P(s,a')\) puts emphasis on recent observations through using a constant as \(\mu\).

**C. Flow of Algorithm**

To sum up, in this section, the flow of OTA is given. Since the action selection mechanism does not affect the convergence of MARL algorithms, and most algorithms in cooperative MASs adopt the \(\epsilon\)-greedy exploration method, this paper employs \(\epsilon\)-greedy. It is applied to MASs in the following way: the agent chooses an exploratory private action with probability \(\epsilon \in [0,1]\), and with \(1 - \epsilon\) picks its part of an optimal (greedy) action with respect to the current Q-value. \(\epsilon\) is asymptotically decreased to zero over time.

For agent \(i\), the learning process can be described as follows.

(1) Initialization. Given the assumption that all the agents share the same state space \(S\) and action space \(A\), let \(Q(s,a') = V(s) = \frac{r_p}{1 - \gamma}\), where \(r_p\) represents penalty. The estimate function of state transition is \(T(s,a',s') = \frac{1}{N_{suc}}\), where \(N_{suc}\) indicates the maximum number of possible successive states. The estimation of policy for agent \(j\) is

\[
P(s,a_j^k) = \frac{1}{\lambda_j}.
\]

(2) Repeat the following procedures:

a). Observe the current state \(s\).

b). Choose the action according to \(\epsilon\)-greedy policy \(\epsilon \leftarrow \epsilon_0 \frac{\text{Count}(s)}{\text{Count}(s)}\), where \(\text{Count}(s)\) indicates the times of \(s\) visited and \(\Omega(s)\) represents the action space to learn in Q-function.

c). Observe the new state \(s'\), the action of its own \(a_j^k\), the others’ actions \(a^{-i}\) and the immediate reward \(r\). Update the \(P(s,a')\) according to (11), and \(T(s,a',s')\) in the same way like (11).

d). If the goal state is achieved, go to e), else go to a).
e). Calculate the Q-function according to (10) by Prioritized Sweeping [18].

Until the terminal condition is satisfied.

IV. SIMULATION AND RESULTS

In order to evaluate the feasibility of our new algorithm OTA, we prepare a formation control task, which is a typical testbed for multi-agent cooperative algorithms.

A. Experiment Description

This simulation is devised to emphasize the learning ability and the adaptability of proposed algorithm in MASs. It can be extended to an \( n \)-agent MAS easily.

In a \( 7 \times 7 \) bounded grid world, there are three agents denoted by triangle and round and each of them has four actions, which are going up, down, left and right as shown in Fig. 1. Thus, for OTA represented as triangle, the available space \( S \) is 49 and the available action space \( A \) is 4. For simplicity without loss of generality, the two round agents follow the two fixed policy as the blue and red dashed lines in Fig. 1. Assume that the two round agents follow the blue trajectory as the policy at the beginning. After a while, these two agents change their policies at the switching state as shown in Fig. 1 and follow the red trajectory. Thus, the policies of these two agents are changing between two policies at the switching state, which are going up and gong right, and fixed in the other states.

The policies including the changing time of the policies at the switching state are unknown to OTA. The OTA’s task is to learn to follow these two agents and keep the formation as starting state.

At the beginning of each experiment, three agents are set at the starting state as shown in Fig. 1. When all the three agents reach the goal state, this experiment is over and all the agents are reset to the starting state to start a new experiment once again. If the two round agents arrive before the OTA, they will wait there for the OTA at the goal state.

B. Results and Analysis

To show the validity and advantage of the proposed algorithm OTA, it is compared with the Friend-Q, which seems to offer the superior performance among all the other classical MARL algorithms for cooperative MASs [20].

The OTA belongs to model-based RL and model-based RL usually learns faster than model-free one [18]. The Friend-Q can be modified in the framework of model-based RL, because it is just a super agent RL. Consequently, the two algorithms are both in the same framework of model-based RL in [18].

The parameters of the two algorithms, which hold the same meaning, are set the same. In particular, when the OTA chooses an action and then keeps the formation as the starting state, it receives reward 20, otherwise -1 as a penalty. The discount factor \( \gamma = 0.8 \), and the initial value of \( \epsilon - \text{greedy} \) policy \( \epsilon_0 = 0.98 \). The parameter \( \mu \) of RME is 0.8. The switching time of policy is set as 800, which means that when the two switching state is visited 800 times respectively by the two round agent, they both change their policies, from going up along with the blue dashed line to right along with the red one. The comparison are repeated 50 times in total.

To guarantee the convergence, most of conventional MARL algorithms including Friend-Q adopt the joint state and joint action. Thus, the learning and storing space of Friend-Q are both \( |S|^n \times |A|^m \). Since OTA incorporates the other agents’ policies into its own decision process in the form of (10) explicitly, the learning space of OTA is \( |S|^n \times |A| \). As in this experiment, the learning and storing space of two algorithms are shown in TABLE I. It is obvious that the space of OTA is extremely smaller than Friend-Q.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Space requirement</th>
<th>Task completion rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friend-Q</td>
<td>7529536</td>
<td>5109</td>
</tr>
<tr>
<td>OTA</td>
<td>470596</td>
<td>11906</td>
</tr>
</tbody>
</table>

![Fig. 1 The description of formation control](image1)

![Fig. 2 Average steps per round to finish the task](image2)
To compare the efficiency of the learning algorithms, three aspects are obtained. The task completion rate means the average times of reaching the goal state. As shown in TABLE I, it is clearly that the OTA can finish the task with the highest rate in the same steps.

The other two aspects are average steps used and average reward obtained during the learning process. From Figs. 2 and 4, it is apparently that the OTA learns faster than Friend-Q and achieves the optimal policy, which is reaching the goal state in 7 steps as shown in Fig. 3 (not including the last step to goal state).

As shown in the Figs. 2 and 4, we can notice that there exists a catastrophe in OTA at round 16000 nearby but dose not obvious in Friend-Q. This catastrophe is caused by the change of the other two agents at the switching state, which results in the change of $T(s', s, a')$. However, since the state transition $T(s', s, a)$ of Friend-Q employs the joint form and is not depended on the other agents’ actions but only on the environment, therefore the others’ switching policy will not lead to distinct change in decision process but only increase the learning space for Friend-Q. Since the RME can learn and estimate the change of policy, the OTA adjusts its policy based on RME after the other two round agents changing their policies and responds to the change optimally and quickly as shown in Fig. 5. Thus, the catastrophe vanishes rapidly.

The experiment results above are reasonable. Friend-Q tries to acquire all the information of the environment, so it records all the agents’ states and actions to obtain the optimal performance by exploring such joint state-action space. Though, it seems powerful but clumsy and does not consider the realizability when the number of agents increases which will incur the curse of dimensionality. The proposed method OTA does not have so severe curse of dimensionality as Friend-Q and achieves better performance using less learning and storing space. Though there is a catastrophe, the RME learns and estimates this change more and more accurately with the learning process. Based on the estimation of others and the reward from system, the OTA learns to make a response to such change of policies more and more optimally. Thus, it is not difficult to infer that when the two round agents change their policies at the other states except the assigned switching state in this experiment, the RME also can learn and track this change and the OTA makes a decision as good as possible based on the RME.

V. CONCLUSION

In this article, a new framework for MARL, named as OTA, is proposed to relieve the curse of dimensionality, which is ubiquitous in current MARL algorithms and hinders the application of these algorithms. The OTA treats the other agents as a part of the environment, and uses the estimator to track the dynamics of the environment. Thus, based on the estimator, the OTA uses model-based RL to make the optimal decision, which is also the best response to the dynamic system.

Though the OTA is start from cooperative MASs, it is interesting to extend this algorithm to competitive multi-agent systems. Meanwhile, developing a more effective estimator is needed to realize quick and accurate response to the dynamics of system.
ACKNOWLEDGMENT

This work was supported by National Nature Science Foundation (Nos. 60874042 and 61074058), Chinese Postdoctoral Science Foundation (No. 200902483), Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20090162120068) and Central South University Innovation Project (No. 2011ssxt221).

REFERENCES


