Efficient Behavior Learning by Utilizing Estimated State Value of Self and Teammates

Kouki Shimada[1], Yasutake Takahashi[1], and Minoru Asada[1,2]

[1] Dept. of Adaptive Machine Systems, Graduate School of Engineering
Osaka University
[2] JST ERATO Asada Synergistic Intelligence Project
Yamadaoka 2-1, Suita, Osaka, 565-0871, Japan
{kouki.shimada,yasutake,asada}@ams.eng.osaka-u.ac.jp
http://www.er.ams.eng.osaka-u.ac.jp

Abstract. Reinforcement learning applications to real robots in multi-agent dynamic environments are limited because of huge exploration space and enormously long learning time. One of the typical examples is a case of RoboCup competitions since other agents and their behavior easily cause state and action space explosion.

This paper presents a method that utilizes state value functions of macro actions to explore appropriate behavior efficiently in a multi-agent environment by which the learning agent can acquire cooperative behavior with its teammates and competitive ones against its opponents.

The key ideas are as follows. First, the agent learns a few macro actions and the state value functions based on reinforcement learning beforehand. Second, an appropriate initial controller for learning cooperative behavior is generated based on the state value functions. The initial controller utilizes the state values of the macro actions so that the learner tends to select a good macro action and not select useless ones. By combination of the ideas and a two-layer hierarchical system, the proposed method shows better performance during the learning than conventional methods.

This paper shows a case study of 4 (defense team) on 5 (offense team) game task, and the learning agent (a passer of the offense team) successfully acquired the teamwork plays (pass and shoot) within shorter learning time.

1 Introduction

There have been studies on cooperative/competitive behavior acquisition in a multiagent environment by using reinforcement learning methods, especially in the RoboCup domain. In such a dynamic multi-agent environment, the state and action spaces for the learning can be easily exploded since not only objects but also other agents should be involved in the state and action spaces, and therefore the sensor and actuator level descriptions may cause information explosion that disables the learning methods to be applied within practical learning time. Kalyanakrishnan et al. [4] showed that the learning can be accelerated by sharing

J. Baltes et al. (Eds.): RoboCup 2009, LNAI 5949, pp. 355–365, 2010.

the learned information in the 5 on 4 game task. However, they need still long learning time since they directly use the sensory information as state variables to decide the situation. Noma et al. [6] achieved the cooperative behavior acquisition in the same 5 on 4 game domain within much shorter time by introducing the macro actions and abstracted state variables based on the macro actions and reducing the size of the state-action space. However, the learning time is still too long to realize real robot learning.

Noma et al. [6] presented a method of hierarchical modular learning in a multiagent environment in order to reduce the exploration space, that is, the state space. Learning modules at the lower layer acquire basic skills for soccer play, for example, dribbling and shooting, passing, and receiving behavior, based on reinforcement learning. The module of the top layer takes the state values of the action modules of the lower layer as state variables to construct the state space for learning the cooperative/competitive behavior. The key idea of their work is to utilize the state values of action modules as abstracted state variables instead of using sensory information directly in order to reduce the size of state space. However, the state value can be utilized in more efficient way to reduce the time learning behavior for more complicated cooperative task.

On the other hand, there are case studies in which evaluation of the player situation is designed by hand and the players behave cooperatively based on the evaluation. Isik et al. [3] proposed a multi-robot control system by sharing utility of certain behavior among the players. Mcmillen et al.[5] shows cooperative behavior with AIBOs by sharing the information of the ball on the field among the teammates. Fujii et al. [2] proposed to share the utility for role assignment. Those methods are useful for realizing cooperative behavior among a number of robots, however, there is no room to improve their performance through trial and error as machine learning, especially reinforcement learning, does.

This paper presents more advanced method to learn cooperative behavior in multi-robot environment efficiently. An appropriate initial controller for learning cooperative behavior is generated based on the state value functions of the action modules at the lower layer. The initial controller utilizes the state values of the macro actions so that the learner tends to select a good macro action and not select useless ones. By combination of the ideas and a two-layer hierarchical system, the proposed method shows better learning performance than conventional methods. This paper shows a case study of 4 (defense team) on 5 (offense team) game task, and the learning agent (a passer of the offense team) successfully acquired the teamwork plays (pass and shoot) within shorter learning time.

2 Task and Assumptions

The game consists of the offence team (five players and one of them can be the passer) and the defence team (four players attempt to intercept the ball). The offence player nearest to the ball becomes a passer who passes the ball to one of its teammates (receivers) or shoot the ball to the goal if possible while the opposing team tries to intercept it (see Fig. 1).

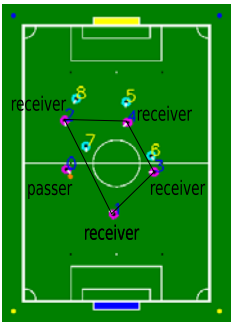


Fig. 1. A passer and the defence formation

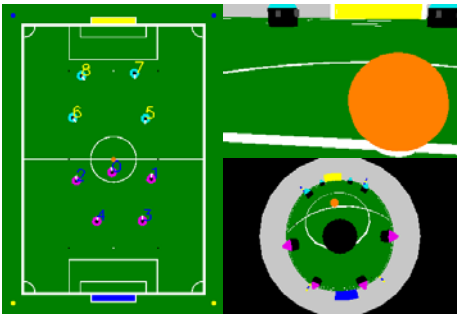


Fig. 2. Viewer of simulator

Only the passer learns its behavior while the receivers and the defence team members take the fixed control policies. The receiver becomes the passer after receiving the ball and the passer becomes the receiver after passing the ball. After one episode, the learned information is circulated among team members through communication channel but no communication during one episode. The action and estimation modules are given a priori.

The offence (defence) team color is magenta (cyan), and the goal color is blue (yellow) in the following figures. The game restarts again if the offense team successfully scores a goal, kicks the ball outside of the field, or the defense team intercepts the ball from the opponent.

2.1 Offence Team

The passer who is the nearest to the ball learns the team player behavior by passing the ball to one of four receivers or dribbling and shooting the ball to the goal by itself. After its passing, the passer shows a pass-and-go behavior that is a motion to the goal during the fixed period of time automatically. The receivers face to the ball and move to the positions so that they can form a rectangle by taking the distance to the nearest teammates (the passer or other receivers) (see Fig. 1). The initial positions of the team members are randomly arranged inside their territory.

2.2 Defence Team

The defence team member who is nearest to the passer attempts to intercept the ball, and each of other members attempts to "block" the nearest receiver. "Block" means to move to the position near the offence team member and between the offence and its own goal (see Fig. 1). The offence team member attempts to catch the ball if it is approaching. In order to avoid the disadvantage of the offence team, the defence team members are not allowed inside the penalty

area during the fixed period of time. The initial positions of the team members are randomly arranged inside their territory but outside the center circle.

2.3 Robots and the Environment

Robots participating in RoboCup Middle Size League are supposed in this paper. Fig. 2 shows the viewer of the simulator for our robots and the environment. The robot has an omni-directional camera system. A simple color image processing is applied to detect the ball, the interceptor, and the receivers on the image in real-time. The left of Fig. 2 shows a situation the robot can encounter while the right images show the simulated ones of the normal and omni vision systems. The mobile platform is an omni-directional vehicle (any translation and rotation on the plane.)

We suppose that the omni directional vision system provides the robot with 3-D construction of the scene. This assumption is needed for the estimation of the state value of the teammates since it is needed to estimate the sensory information observed by other robots.

3 Multi Module Learning System with Other's State Value Estimation Modules

In this section, we briefly review the work of Noma et al. [6]. Fig.3 shows a basic architecture of the two-layered multi-module reinforcement learning system. The bottom layer (left side of this figure) consists of two kinds of modules: action modules and estimation ones that infer the state value of the teammates. The top layer (right side of the figure) consists of a single gate module that learns which action module should be selected according to the current state that consists of state values sent from the modules at the bottom layer. The selected module then sends action commands based on its policy.

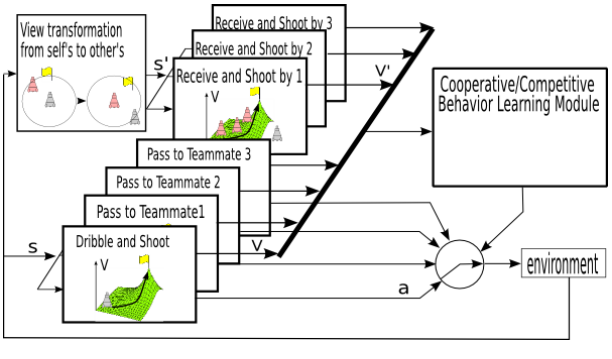


Fig. 3. A multi-module learning system

An action module of the lower layer has a reinforcement learning module which estimates state values for the action. An agent can discriminate a set S of distinct world states. The world is modeled as a Markov process, making stochastic transitions based on its current state and the action taken by the agent based on a policy π. The agent receives reward r_t at each step t. State value V^π, the discounted sum of the reward received over time under execution of policy π, will be calculated as follows:

$$V^\pi = \sum_{t=0}^{\infty} \gamma^t r_t \quad . \tag{1}$$

In case that the agent receives a positive reward if it reaches a specified goal and zero else, then, the state value increases if the agent follows a good policy π. The agent updates its policy through the interaction with the environment in order to receive higher positive rewards in future. For further details, please refer to the textbook of Sutton and Barto[7] or a survey of robot learning[1]. Here, we suppose that the state values in each action module have been already acquired before the learning of the gate module.

As shown in Figure 3, the gate module receives state values of lower modules, that is, the action modules and the other's state value estimation ones, and constructs a state space with them. The state space of the gate module is constructed as direct product of the variables of the state values. In order to adopt a discrete state transition model described above, the state space is quantized appropriately. The action set of the gate module is constructed with all action modules of the lower layer as macro actions. For further details, please refer to [6]. Here, three kinds of action modules are prepared as follows.

- Dribble and Shoot
- Pass to a teammate
- Receive and Shoot

There are 4 "Pass to a teammate" and "Receive and Shoot" modules because there are 4 teammates (receivers) besides the passer. "Pass to teammate 1" module returns the state value when the passer tries to pass the teammate 1. "Receive and Shoot 1" module returns estimated state value of "dribble and shoot" behavior of the teammate 1 if the ball is pass to the teammate 1. Details of those modules are described later.

4 Evaluation of Team Situation

The objective of the team playing soccer is scoring a goal. It is hard to evaluate the situation of the team to score the goal only from positions of teammates, opponents, and a ball. On the other hand, one player situation, how close the player score a goal, can be evaluated based on the state value of the "dribble and shoot" behavior. This behavior can be learned beforehand. If the player can score the goal, then, it means that the team situation is good. Even if the player

with the ball cannot score a goal directly because it is far from the opponent goal or the opponent players are close to it, it can pass the ball to one of the teammates that is close to scoring a goal. If the receiver finds that scoring a goal after it receives the ball, then, it is going to find another teammate that is near to scoring a goal. This idea can be applied recursively. The evaluation of team situation from the view point of player possessing a ball can be approximated as follows:

$$\begin{aligned} E_i^{team} = \max\{ & V_i^{dribble\&shoot}, \\ & \max_j [V_{ij}^{pass} + \beta V_j^{receive\&shoot}], \\ & \max_{jk} [V_{ij}^{pass} + + \beta V_{jk}^{pass} + + \beta^2 V_k^{receive\&shoot})], \\ & \cdots\} \end{aligned} \tag{2}$$

where $V_i^{dribble\&shoot}$, V_{ij}^{pass}, and $V_i^{receive\&shoot}$ indicate state values of player i's behavior "dribble and shoot", "pass" to player j, and "receive and shoot", respectively.

5 Initial Controller Design Based on Team Situation Evaluation

An appropriate initial controller for learning cooperative behavior is generated based on the team situation evaluation. The initial controller utilizes the state values of the macro actions so that the learner tends to select a good macro action and not select useless ones. Based on the approximated team situation evaluation, the initial controller selects one of the macro actions ma as below:

$$ma = \begin{cases} ma^{dribble\&shoot} & \text{if } E_i^{team} = V_i^{dribble\&shoot} \\ ma_j^{pass} & \text{if } E_i^{team} = V_{ij}^{pass} + \beta V_j^{receive\&shoot} \\ ma_j^{pass} & \text{if } E_i^{team} = V_{ij}^{pass} + \beta V_{jk}^{pass} + \beta^2 V_k^{receive\&shoot} \\ \cdots & \end{cases} \tag{3}$$

It is not possible to calculate all possible options within a limited time. Therefore, the set of the options is limited as only the case of just "dribble&shoot" macro action and a combination of "pass" and "dribble&shoot" ones, in this paper. A concrete pseud algorithm is given at Algorithm 1.

6 Structure of the State and Action Spaces

6.1 Gate Module

The passer is only one learner, and the state and action spaces for the lower modules and the gate one are constructed as follows. The action modules are four passing ones for four individual receivers, and one dribble-shoot module. The other's state value estimation modules are the ones to estimate the degree

Algorithm 1. Initial Controller for Passer

```
MaxEvaluation = Value_{Dribble&Shoot}
MaxRobotID = 0
N = Number of Receiver
for j = 1 in N do
    Evaluation(j) = Value_{Pass(j)} + βValue_{Receive&Shoot(j)}
    if Evaluation(j) ≥ MaxEvaluation then
        MaxEvaluation = Evaluation(j)
        MaxRobotID = j
    end if
end for
if MaxRobotID = 0 then
    return DribbleShoot
else if MaxRobotID = 1 then
    return Pass(1)
else if MaxRobotID = 2 then
    return Pass(2)
        ⋮
else if MaxRobotID = N then
    return Pass(N)
end if
```

of achievement of ball receiving for four individual receivers, that is how easily the receiver can receive the ball from the passer. These modules are given in advance before the learning of the gate module.

The action spaces of the lower modules adopt the macro actions that the designer specifies in advance to reduce the size of the exploration space without searching at the physical motor level.

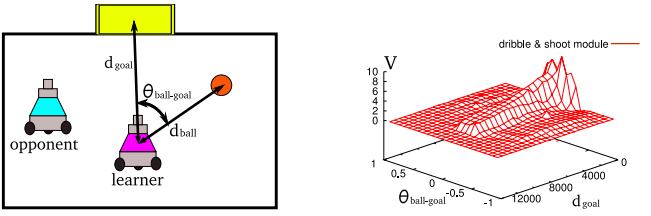
The state space S for the gate module consists of the following state values from the lower modules:

- one state value of dribble-shoot action module,
- four state values of passing action modules corresponding to four receivers, and
- four state values of receiver's state value estimation modules corresponding to four receivers.

The reward 1 is given only when the ball is shot into the goal and reward 0 else. When the ball is out of the field or the pre-specified time period elapsed, the game is called "draw" and one episode is over.

6.2 "Dribble&Shoot" Module

In order to reduce learning time for macro actions, one macro action is decomposed into 2 simple behavior. For example, the "Dribble&Shoot" macro action consists of "single Dribble&Shoot" module and "success estimation" module.



(a) State variables for Dribble&Shoot module

(b) State value function of Dribble&Shoot module

Fig. 4. State variables (a) and learned state value function (b) for the dribble and shoot module

The "single Dribble&Shoot" module learns the state value function of the "dribble and shoot" behavior under the environment where a single player shoot a ball without any teammate or opponent. The "success estimation" module estimates success rate of the "single Dribble&Shoot" behavior in case of existence of an opponent. The "Dribble&Shoot" macro action module combines the two basic modules and estimates state value of the behavior accordingly.

The state space of the "single Dribble&Shoot" module S is defined as follows:

- the angle between the opponent goal and the ball
- the distance to the opponent goal, and
- the distance to the ball

Each of these state values is quantized into 31. A CMAC system is adapted with 8 tilings for the approximation of state value.

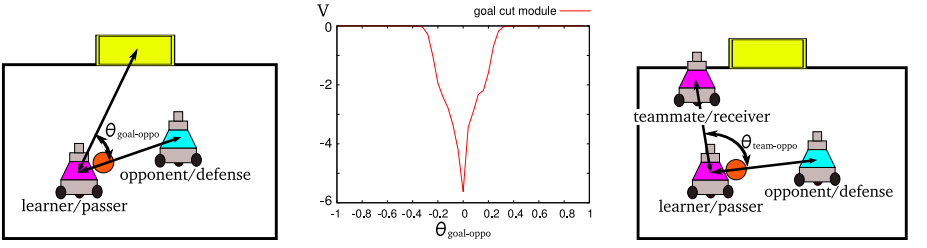
The state space of the "success estimation" module consists of only the angle between the goal and the opponent. The module learns the state value while the player taking the behavior of the "single Dribble&Shoot" module. Negative reward -1 is given when the opponent takes the ball and zero else. Finally, the "Dribble&Shoot" module estimates state value of the behavior by adding the estimated values of two simple modules.

6.3 "Pass" Module

The state space of the passing module consists of the angle between the receiver and the opponent. The state variable is quantized into 31 levels. A CMAC system is adapted again with 8 tilings for the approximation of state value. The state value map is shown in Fig. 5(c) that indicates the smaller the angle between the receiver and the opponent player is, the lower the state value is.

6.4 "Receive&Shoot" Module

The passer infers each receiver's state that indicates how easily the receiver can shoot the passed ball to the goal by reconstructing its TV camera view of the



(a) State variables for success estimation module

(b) State value function of "success estimation" module

(c) State variables for pass module

Fig. 5. State variables and learned state value function of "success estimation" and "pass" modules

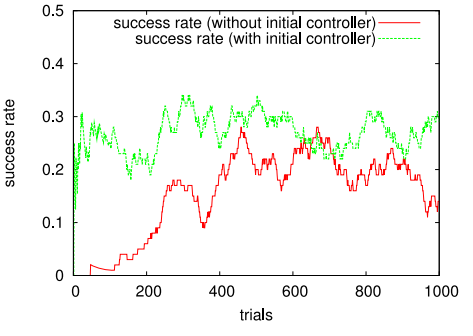


Fig. 6. Curves of success rate

scene from the passer's omnidirectional view. Since we suppose that the passer has already learned the shooting behavior, the passer can estimate the receiver's state value by assigning its own experienced state of the shooting behavior. The "Dribble&Shoot" macro action module is reused for estimation of state value of the "Receive&Shoot" module. This means, the passer estimates the state value of "Dribble&Shoot" behavior on an assumption that the passer successfully pass the ball to the receiver and the receiver controls the ball.

7 Experimental Results

The success rates of case studies with/without the initial controller based on the state value functions of macro action are shown in Fig. 6 where the action selection is 80% greedy and 20% random to cope with new situations. The success rate is moving average during the last 100 trials. The condition of case study without the initial controller is same with the one of Noma et al. [6]. The figure shows the initial controller shows much better performance from the early stage of the learning than the system without the initial controller.

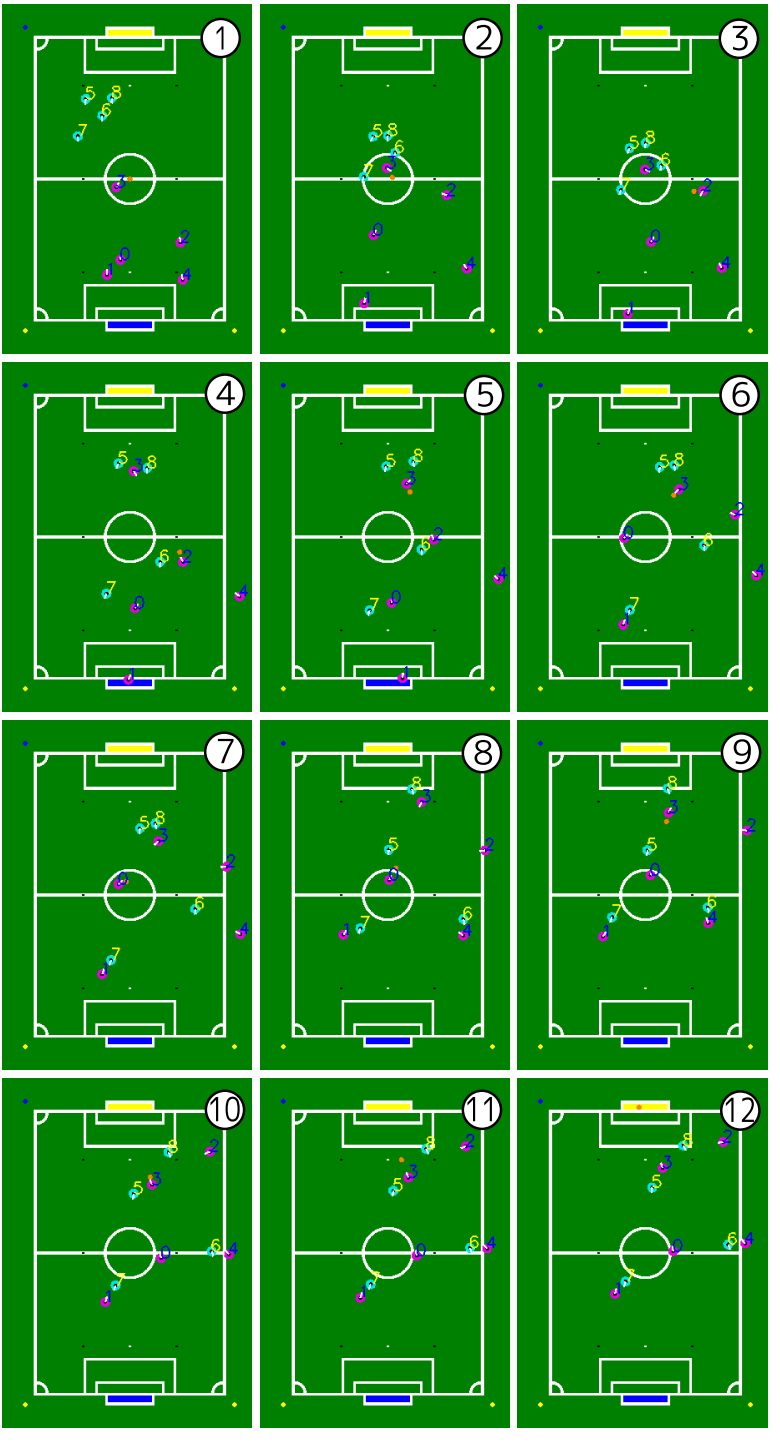


Fig. 7. A sequence of acquired behaviors

An example of acquired behavior is shown in Fig. 7 where a sequence of twelve top views indicates a successful pass and shoot scene.

8 Conclusion

We have utilized the state value functions of macro actions to build a good initial controller for cooperative behavior acquisition instead of learning the behavior from scratch. As a result, we have much improved the performance during the learning compared to the result of the previous method [6].

The initial controller seems to be too good, therefore, the performance of the cooperative behavior during the learning shows little improvement. Further investigation is undergoing for performance improvement of cooperative behavior based on the reinforcement learning.

Real robot experiments are planned in near future because the proposed method reduces actual learning time and it is practical to apply to real robots.

References

1. Connell, J.H., Mahadevan, S.: Robot Learning. Kluwer Academic Publishers, Dordrecht (1993)
2. Fujii, H., Kato, M., Yoshida, K.: Cooperative action control based on evaluating objective achievements. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 208–218. Springer, Heidelberg (2006)
3. Isik, M., Stulp, F., Mayer, G., Utz, H.: Coordination without negotiation in teams of heterogeneous robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 355–362. Springer, Heidelberg (2007)
4. Kalyanakrishnan, S., Liu, Y., Stone, P.: Half field offense in robocup soccer: A multiagent reinforcement learning case study. In: Lakemeyer, G., Sklar, E., Sorrenti, D., Takahashi, T. (eds.) RoboCup 2006 Symposium papers and team description papers, CD–ROM, Bremen, Germany (June 2006)
5. Mcmillen, C., Veloso, M.: Distributed, play-based coordination for robot teams in dynamic environments. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 483–490. Springer, Heidelberg (2007)
6. Noma, K., Takahashi, Y., Asada, M.: Cooperative/competitive behavior acquisition based on state value estimation of others. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007. LNCS (LNAI), vol. 5001, pp. 101–112. Springer, Heidelberg (2008)
7. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)