Cooperative Reinforcement Learning Based on Zero-Sum Games

Kao-Shing Hwang[1], Wei-Cheng Jiang[1],
Hung-Hsiu Yu[2] and Shin-Yi Lin[2]
National Chung-Cheng University / Industrial Technology Research Institute
Taiwan

1. Introduction

Multi-Agent systems (MAS), developed from the artificial intelligent field, include many independent agents, each of which has its own behavior and can achieve a certain goal. The system can be used to solve a complex problem by cooperation and coordination between agents, because most complex problems can be divided into many sub-problems. Therefore, cooperation is central to the topics and operations of MAS (Vassiliades et al., 2011)(Marden et al.,2009).
The use of MAS in teams to play a competitive game such as soccer is major challenge, where agents have to accomplish the goal of scoring or prevent scoring by opponents in real time in a highly varying environment. Being good players, they must be able to adapt the strategy on which they are behaving responding to actions of the opponents. In other words, agents of MAS need to operate appropriately in real time to varying states of the environment. Therefore, it is difficult to establish a well-defined state-action mapping for the agents unless the whole state space has been visited or searched exhaustedly. Instead, a learning mechanism adapting to a highly dynamic environment is more appropriate for the situation (Busoniu et al., 2008)(Xiao et al., 2007).
Reinforcement learning is probably the best-known method of learning due to the similarity of empirical learning in mammals (Gosavi, 2009)(Kaelbling et al., 1996). The learning takes place by trial and error, following by performance evaluation but always provided after a sequence of actions. Besides, the correct solution or behavior for a given trial is not provided. The goal of learning agents is to establish a strategy optimizing a scalar cost or evaluation function in long term. The strategy, also known as action policy, can choose an appropriate behavior or action based on the stated perceived by robots (Barto et al., 1983)(Bingulac et al., 1994)(Tan, 1993). Furthermore, reinforcement learning on Markov games have also been proposed for multi-agent reinforcement learning to solve cooperation problems in social dilemmas under the assumption of Nash equilibrium (Yanco et al., 2002) (Flache et al., 2002) (Littman, 1994). These general-sum stochastic games has been adopted as a framework for multi-agent reinforcement learning and proved to converge to a Nash equilibrium under specified conditions (Hu et al., 1998). That method is adoptable to find an optimal strategy only when there exists a unique Nash equilibrium in a game. However, in

some problems, such as task assignment in robot soccer games, it is not enough to consider only attributes of the agent itself because the whole state space is composed of the ones of partners and components.
Eventually, task assignment is always a major research area in MAS where a number of autonomous agents, either in heterogeneous or homogeneous fashion, work together with individual assignment. Although centralized optimization provides opportunity for efficient optimization, the coordination and the transfer of information among agents are costly and often infeasible. Hence it is important to develop decentralized optimization schemes which permit the individual agents take control of the actions that contribute towards the optimization of a global performance criteria. The motivation for using game theory for assigning tasks are driven by the fact that decentralized optimization in game theory is achieved by the agents (players of the game) acting selfishly. We utilize techniques from game theory to produce an algorithm for matching player and roles (tasks) based on situations on the field. As a result, we have created a deterministic algorithm for task assignment with built-in feedback mechanisms for reinforcement learning.
This chapter presents a cooperation strategy system with self-improving abilities for task assignment for multi-robot systems. The strategy system is eventually a formation mechanism based on reinforcement learning and the game theory through task assignment to individual robot. The mechanism coordinates robots, in they which learn to behave primitively, to work as a team by means of perceptual information with their respective roles such as, attacker, defender, or goalie, to produce appropriate behaviours. The robots with homogeneous behavioral architecture accomplish cooperatively global goals on their local sensory information cooperatively. These robots carrying out tasks of a helper or defender improve action policy at play based on Q-learning inspired by the game theory. Results of experiments demonstrate the effective performance of the proposed method in comparison with renown methods.

2. Task assignment mechanism

Eventually, the assignment of specific teaks to individual robot on a team is a major research in MAS. No matter whether the teams are homogeneous or heterogeneous, different robots will generally be required to perform different task. In our proposed system, individual robots are able to solve certain problem, such as ball shooting, obstacle avoidance, and positioning through reinforcement learning before playing a game. The system makes use of a task assignment mechanism to robots with learned capabilities. The task assignment mechanism further divided into two sub-mechanisms: a task assignment system to decide each robot's temporal task corresponding to perceived geometrical state and a reinforcement learning system to command robots to the most appropriate position for the task assigned. Each robot with the same architecture behaviors on local information accomplishes global goals; that is, each robot chooses and executes it own actions according to its task and the environmental states. Meanwhile, the choosing action strategy associated to a task is modified by means of reinforcement learning.

2.1 Dynamic task assignment for players

The assignment system assigns dynamically four tasks, an attacker, goalie, helper, and defender, respectively, to a robot one at a time.

2.1.1 Tasks of an Attacker

The task of the Attacker is aimed to kick a ball into the opponent's field to score. It is a very important task in a soccer game because the goal of a soccer game is to gain more scores. Thus, the first assigned task in the assignment mechanism is finding an attacker. The position of a robot is a major factor in considering which robot should take this assignment because a fast and precise attack is key to score a point. Therefore, a robot who can spend the shortest time to kick the ball in the correct direction should be an attacker. The procedure to identify which robots are at an advantageous position is defined as follows. With respect to a coordinate frame where origin locates at the center of the ball, and x-axis is the line from the opponent's goal center to the ball as shown in Fig. 1. The good side is defined on the side opposite to opponent's goal of y-axis. In other words, robots on the region are favorable to be an attacker. In turn, the robot on the good side and closest to the ball is assigned as an attacker. Unfortunately, if there is no robot on the good side, any robot is selected as an attacker as long as it is closest to the ball.

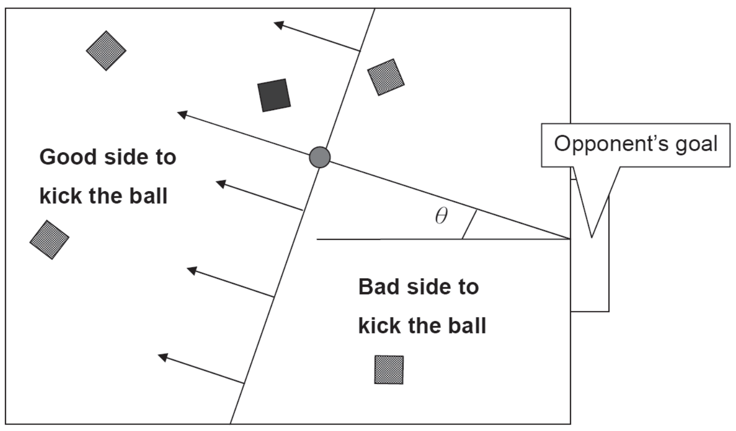


Fig. 1. The rule to select the attacker.

2.1.2 Tasks of a goalie

A goalie is required to move along the direction in parallel to the y-axis of the frame with respect to the ball in the front of its goal. This task is assigned to a robot after the attacker is selected because stopping the opponent's attack in front of the goal is a passive action in a soccer game. A robot is assigned to play this task when it is nearest to its team's goal and does not need to be assigned as an Attacker.

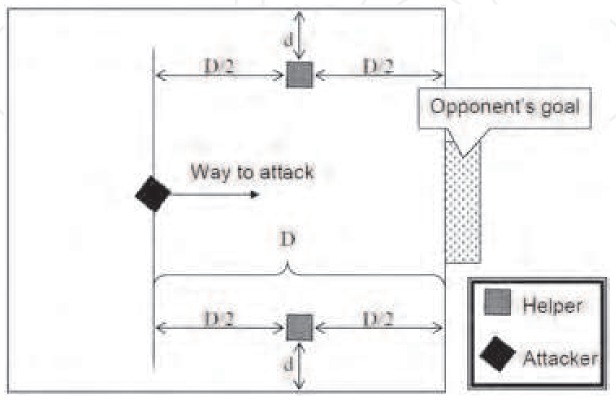
2.1.3 Tasks of a helper

A helper assists the attacker or the defender to accomplish the imminent efforts. This task is assigned in turn thirdly, and two free robots are assigned as helpers. The helpers are capable of learning in the game. Depending on the various situations occurring in the game field, the helpers move to suggested positions according to the formation being taken to ensure the attacker or the defender can get good support. There are three formations, attack formation, normal formation, and defense formation to deal with the highly dynamic situations on the field. Helpers learn empirically to ensure that a most appropriate formation is always taken. The details of the learning mechanism are depicted in the next section.

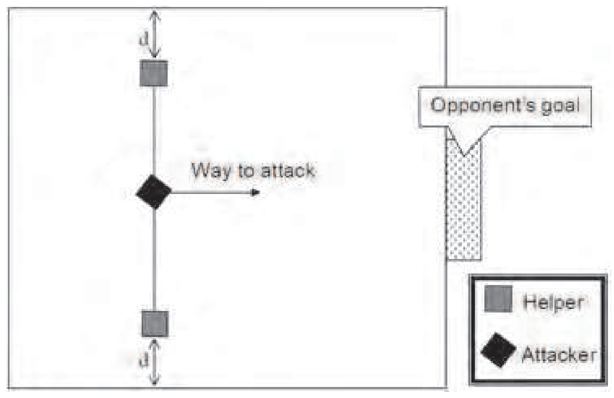
a. In the attack formation, the helpers are dispatched to both side of the attacker on the half way to the opponent's bottom line, as shown in Fig 2(a).

b. In the case of the normal formation, the helpers are aligned with the attacker as shown in Fig 2(b).

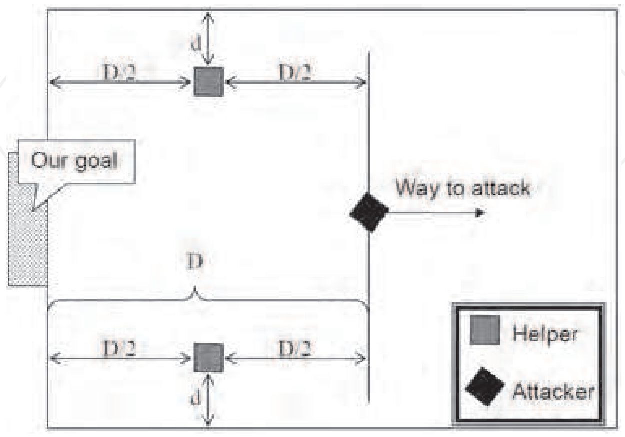
c. If the defense formation is taken, helpers are assigned to two sides of the attacker but at the midpoint between their own goal and the attacker as shown in Fig 2(c). Meanwhile, a robot is assigned as a helper when it is not assigned as either an attacker or a goalie and is nearest to one of suggested positions of the adopted formation.



(a)



(b)



(c)

Fig. 2. Rules to select helpers: (a) attack formation, (b) normal formation, (c) defense formation.

2.1.4 Tasks of a defender

The major function of a defender is to block and prevent opponents from scoring. Hence, it needs to move to appropriate positions where it can be most helpful to the goalie. Similar to the task for helpers, there are three optional positions for the defender as shown in Fig 3. The defender has learning ability, and details of the learning system will be described later.

2.2 Learning to position

As aforementioned, each helper and defender has three prospective formations or positions to choose. A reinforcement learning system is applied to determine which formation the helper should take with respect to the position of the attacker, the defender, and the position of the soccer ball.

The reinforcement learning system associated with an individual is implemented by the Q(λ) algorithm. For computation efficiency, the state space ought to be partitioned to a reasonable amount in the learning state-action mapping function. Therefore, a linear tile-coding function approximation is used for state reception (Hu et al., 1998)(George A. Bekey, 2005) where each instance of decoded state vectors includes the information of position, speed, and direction of the soccer. Furthermore, a ball position is presented by the coordinate on the field which is divided into $6 \times 9 = 54$ grids; ball speed is mapped to a bipolar value; the direction of ball is also presented by eight different symbols. The output of the reinforcement system decides which formation should be taken.

In addition, in order to respond and adapt timely to various situations on the field, the learning system should take the opponents' current actions into consideration. Unfortunately, it is impossible, in practice, to know which action is being and going to be taken by the opponent. Thus, observing the sequence of opponents' behaviors to predict the situations is a good way to get related information. Eventually the action set of opponents cannot be obtained but temporal actions executed by the opponents are observable. Instead of obtaining the action chosen directly, positions where each opponent is standing at are followed up, because the action being executed is regarded as the output of the function of current robot formation. The quantization of the opponent's action is done by a Cerebellar Model Articulation Controller (CMAC) decoder (K. S. Hwang et al., 1998). Since the combinational state space of opponents in the opposing team will be extremely large if all the information for each opponent is taken into account. Thus, only the state of the attacker in opposing the team is considered, and that attacker is an opponent closest to the ball than the other players.

For discretizing the receptive space into a state space, a filter partitions the region around the ball into 16 cells, as shown in Fig. 4. The center of the circle is at the position of a ball, and the radii of the inner and outer circles are 3 and 6 units, respectively. Instead of Euclidian distance presentation, the distance between the opponent and ball is a scale value assigned by this filter.

The reward to each individual learning system is different. The sparse rewards of the helpers' learning system are as follows:

- Get Point: get reward +1;
- Lose Point: get reward -1;
- Become the Attacker: get reward +0.5.

The rewards for the defender are:

- Lose Point: get reward -1

- Become the Attacker: get reward +1

When the period of finding a new action is too short, the previous action will have not enough time to execute well because executing it needs enough time. This will provide wrong results when the system is just beginning to learn. To ensure sufficient time to execute each action command, a new action is triggered only after the task of an attacker is shifted to the teammates by the assignment system. According to the rule of task selection, the attacker is changed under certain situations. Therefore, the previous action can have enough time to execute.

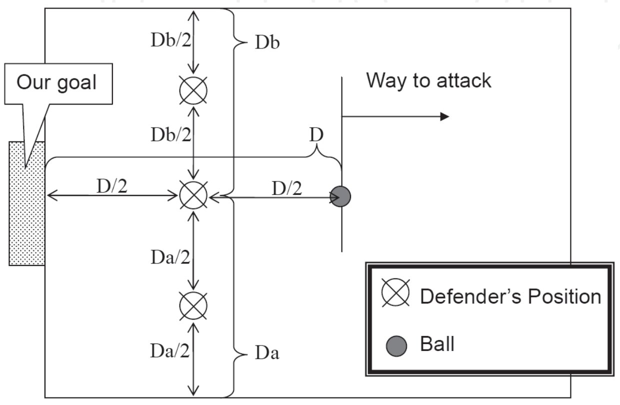


Fig. 3. Three positions of defender.

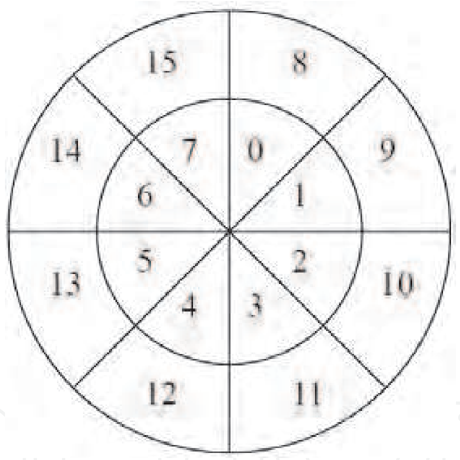


Fig. 4. Decompositions of an opponent's position.

3. Q-learning for MAS

Spontaneous behaviors of the helpers or defenders for positioning are activated by a local learning system. A Q-learning for MAS derived from the game theory is proposed to obtain a better policy in a team. In the two-player zero-sum game, the action is taken according to the opponent's behavior. If the opponent makes an action to get their best benefit, the action may result in the worst situation to our team. On the other hand, if we want to win the game in this situation, best actions to deal with the action being taken by the opponent must be always taken. Therefore, the Q-learning is modified to accommodate the principle of game theory. The original Q-learning rule is as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)], \tag{1}$$

where ***s*** is the state; ***a*** is the action; ***A*** is an action set; ***a*** is the learning rate; ***γ*** is the discount factor; ***r*** is the reward. This equation tries to maximize the Q-value for the future and only consider the action that we take. So, it uses $\max Q(s',a'), a' \in A$ to update its $Q(s,a)$. In game theory, because we need to consider the opponent's action ***o***, the notation, ***o***, denotes the opponent's action set. We redefine $Q(s,a)$ to $Q(s,a,o)$ and redefine the $\max Q(s',a'), a' \in A$ to $\max\min Q(s',a',o'), a' \in A, o' \in O$. Therefore, we get the following equation:

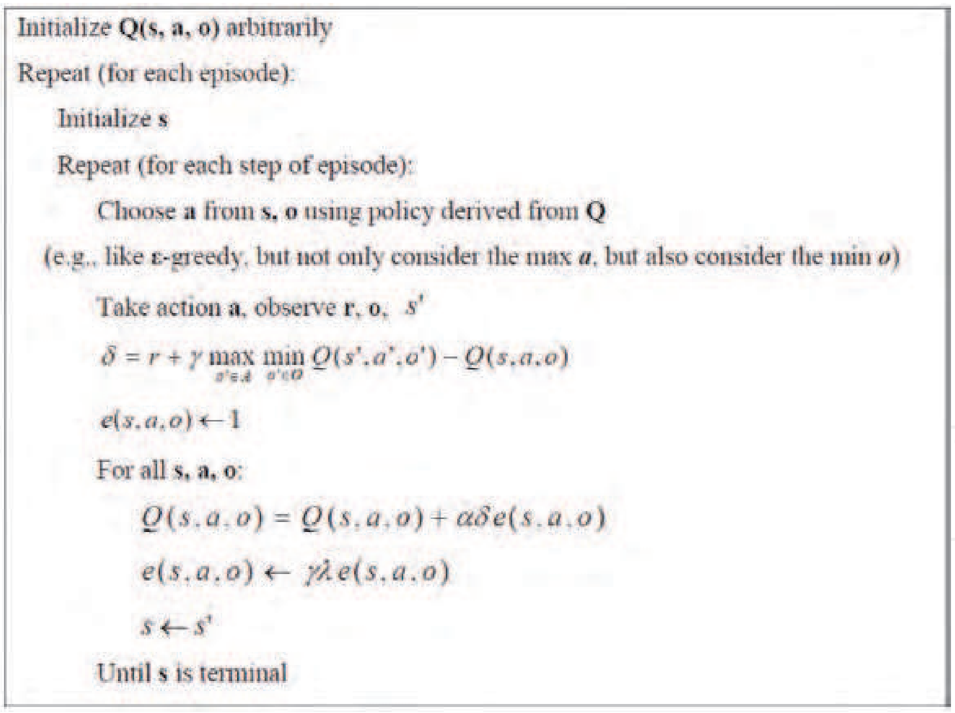
$$Q(s,a,o) \leftarrow Q(s,a,o) + \alpha[r + \gamma \max_{a' \in A} \min_{o' \in O} Q(s',a',o') - Q(s,a,o)]. \tag{2}$$

Eligibility traces are also taken into account. The algorithm is further modified by the *replace-trace* method:

$$Q(s,a,o) \leftarrow Q(s,a,o) + \alpha[r + \gamma \max_{a' \in A} \min_{o' \in O} Q(s',a',o') - Q(s,a,o)]. \tag{3}$$

where $\delta = r + \gamma \max_{a' \in A} \min_{o' \in O} Q(s',a',o') - Q(s,a,o)$

Actually, this algorithm is derived from the classic Q-learning algorithm but inspired the game theory and eligibility traces. Fig.5 shows the complete algorithm in pseudo code.



```
Initialize Q(s, a, o) arbitrarily
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      Choose a from s, o using policy derived from Q
   (e.g., like ε-greedy, but not only consider the max a, but also consider the min o)
      Take action a, observe r, o, $s'$
      $\delta = r + \gamma \max_{a' \in A} \min_{o' \in O} Q(s',a',o') - Q(s,a,o)$
      $e(s,a,o) \leftarrow 1$
      For all s, a, o:
         $Q(s,a,o) = Q(s,a,o) + \alpha\delta e(s,a,o)$
         $e(s,a,o) \leftarrow \gamma\lambda e(s,a,o)$
         $s \leftarrow s'$
      Until s is terminal
```

Fig. 5. The proposed Q-Learning.

4. Soccer robot behaviors

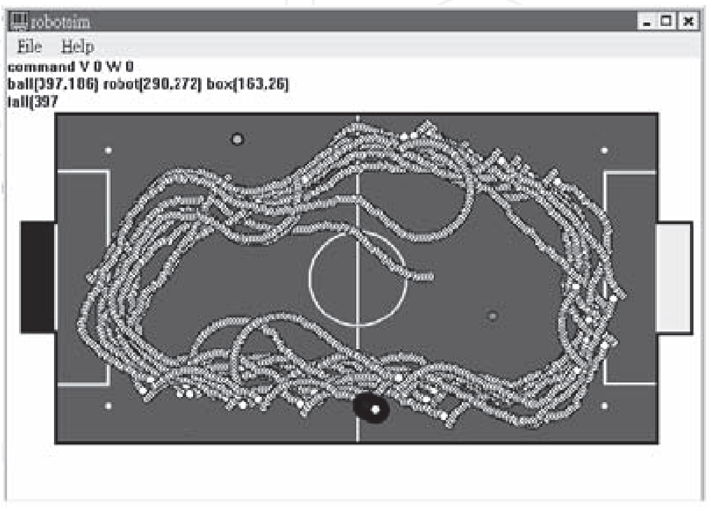
A robot has been required to learn three basic types of behaviors, ball shooting, obstacle avoidance, and target reaching. These primitive behaviors can be combined with more complex behaviors, such as defending, or attacking. Ball shooting is a simple but essential behavior. Through this behavior, before a robot comes to the ball position, it follows an imaginary shooting line, and switches the kicking motor on. The imaginary shooting line is a line passing through the ball and the target door. Obstacle avoidance is necessary because during the game, a robot is not allowed to collide with other robots. The temporal information about the positions and moving direction of other robots is provided by a vision system. The robot moves in a direction to avoid collision according to the distance between other robots and the current position by means of a reinforcement learning mechanism (K. S. Hwang et al., 1998). The outputs of all available actions of the learning mechanism are regarded as reference signals for driving motorizing wheels. The behavior of target reaching drives a robot to the desired position and orientation.

5. Simulations

There are two simulations, one is for the basic behaviors, and the other is for the strategy algorithm.

5.1 Behavior demonstration

There are three different basic behaviors that to be learnt: collision avoidance, target moving, and ball shooting. The simulation environment to emulate the soccer field is defined by the Federation of International Robot-soccer Association (FIRA) (FIRA, 2009). A robot is initially placed at the center but slightly to the right. The robot motion track is shown by a series of white dots. Fig. 6 shows the behavior of obstacle avoidance learned by a robot. The robot walks randomly on the field without going over the boundary, which is regarded as an obstacle. Fig. 7 shows a robot’s capability of moving to a target. In this figure, the robot moves as closely as possible to the assignment posture (position with a certain orientation) with a slight tolerance represented by a hidden region centered by the assigned position. Fig. 8 shows the learnt ball shooting behavior. The robot moves to an aiming position where the robot aligns the line between the center of the opponent’s gate and ball, but with a certain distance from the ball in preparation for kicking.



```
command V 0 W 0
ball(397,186) robot(290,272) box(163,26)
lall(397
```

Fig. 6. Obstacle avoidance behavior.

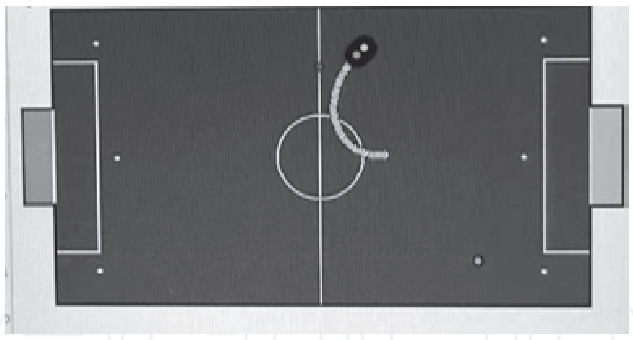


Fig. 7. Target moving behavior.

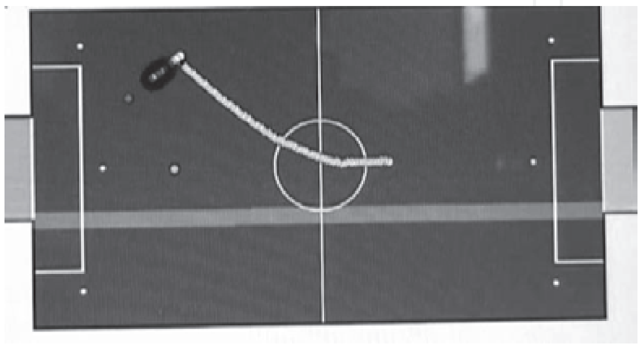


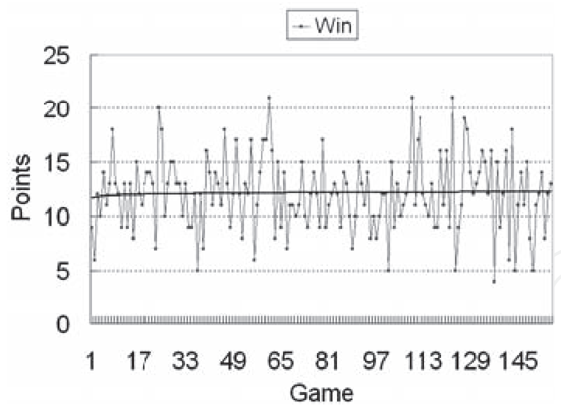
Fig. 8. Ball shooting behavior.

5.2 Strategy demonstration

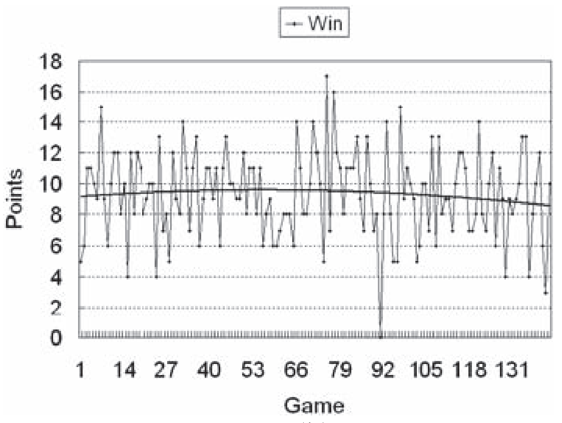
The Proposed Q-learning algorithm is compared with the renowned Q(λ)-learning and minimax-Q under a 5-vs-5 simulation environment provided by FIRA (FIRA, 2009). The Q(λ)-learning is a modified Q-learning methods with the eligibility skill, and the minimax-Q further combines Q-learning with a simple game theory. These three algorithms are arranged, respectively, to play against a test-bed algorithm. In the first game, the team with the Q(λ)-learning algorithm plays against the team using the default algorithm, Lingo. Lingo is a well-established landmark algorithm along with the FIRA robot soccer simulator. Meanwhile, the minimax-Q algorithm is used to play against the team using Lingo in the other game. In another game, the proposed algorithm, Proposed Q, competes with Lingo. The scores gained by the competitors are recorded in Fig. 9. The horizontal axis of the chart expresses the number of rounds each of which lasts 10 minutes, and the vertical axis expresses the total scores the comparing team gets. The solid line is a regression of the saw-toothed line to represent the trend of scores gained along the number of rounds.

As shown in Fig. 10, where the regression lines in Fig. 9 are redrawn, the proposed algorithm beats the opponent by about 16 to 18 points. This result is better than the results of Q(λ)-learning algorithm and the minimax-Q algorithm.

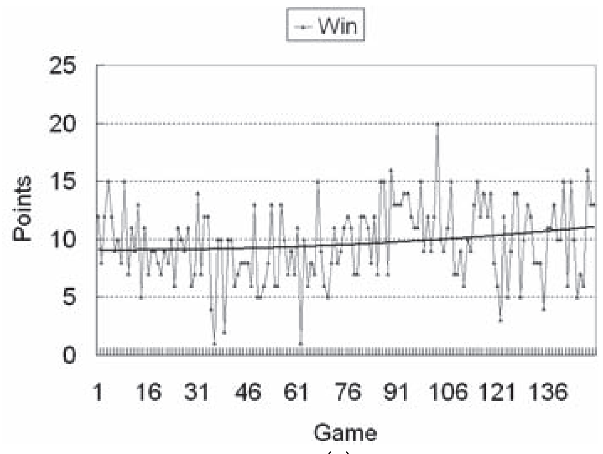
In the second experiment, which is similar to the first experiment, Lingo is replaced by a rule-based strategy introduced in (K. S. Hwang et al., 2007). This rule-based strategy opponent has beaten Lingo before. The comparison between Q(λ)-learning algorithm, minimax-Q algorithm, and the proposed algorithm is indicated on Fig. 11. Intriguing enough, the opponent’s score decreases more obviously when the challenger uses the proposed algorithm.



(a)



(b)



(c)

Fig. 9. (a) Q(λ)-learning vs. Lingo: Q(λ)-learning win scores log and trend-line, (b) minimax-Q vs. Lingo: Minimus-Q win scores log and trend-line, (C)Proposed Q vs. Lingo:Zero-Sum-Q(λ) win scores log and trend-line.

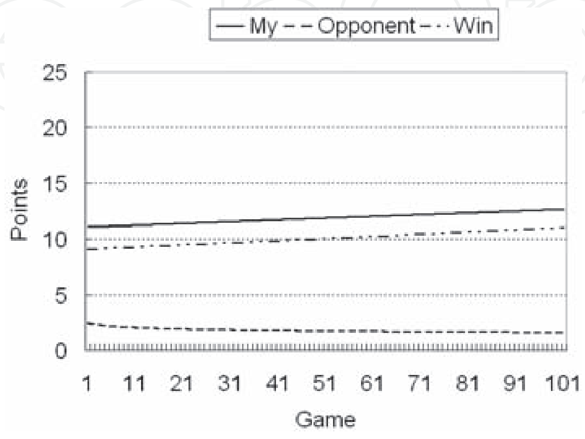
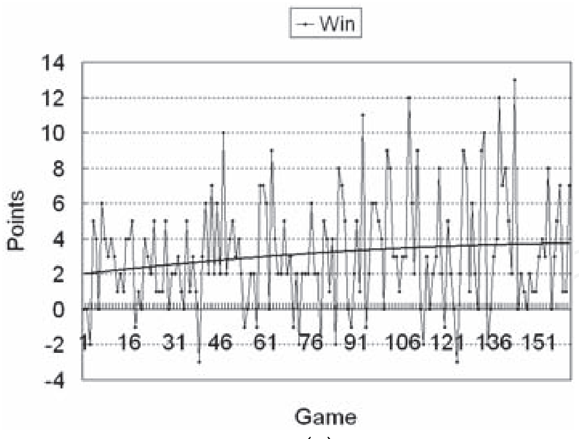
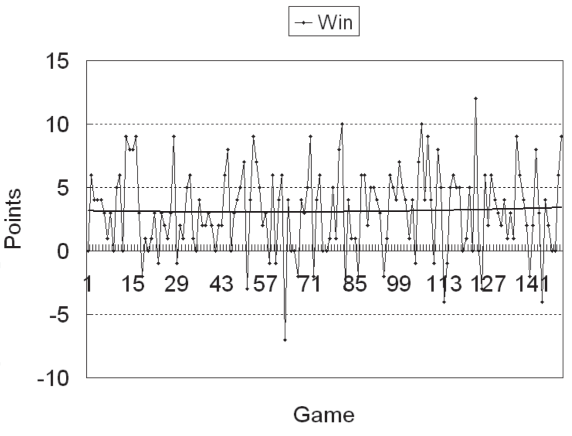


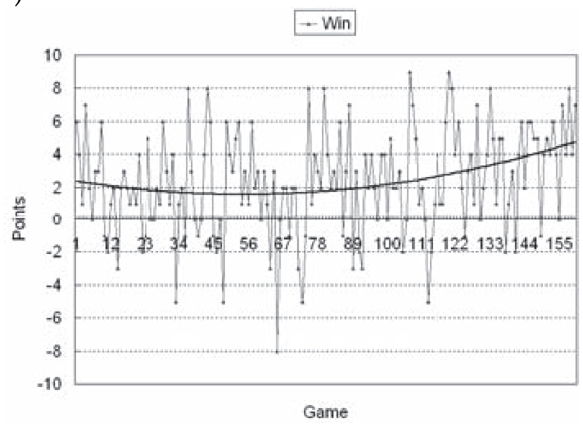
Fig. 10. Proposed Q vs. Lingo: Three trend-lines in the last 100 games.



(a)



(b)



(c)

Fig. 11. (a) Q(λ)-learning vs. hand code C: My scores, (b) minimax-Q vs. hand code C: My scores, (c) Proposed Q vs. hand code C: My scores.

5.3 Communication protocols

The proposed method has been further implemented into an actual soccer robot system to demonstrate the performance in a real environment. The robot soccer system is composed of a control center, vision system, and three soccer robots. The vision system recognized the ball, the location of our team's three robots and the opponent robots and their movement. The control system consists of Bluetooth dongles and a software system. A personal computer (PC) system is used here because of its low cost and high performance. The software system includes a control subsystem, a strategy subsystem, and a communication subsystem, called the subsystem of robot protocol (SRP). A soccer robot is based on an embedded system, where the system controls the rotating speed of the wheels according to the commands from the control center. Thus, the embedded system is connected to a motor driving system that drives wheel motors and a communication system that communicates with the control center. An SRP is a self-defined protocol for communication between the PC and soccer robots through Bluetooth. The protocol regulates how the control system sends commands to robots. The protocol is very simple because the communication is only one-way from the center to robots. Each robot command is sent in turn and repeatedly as shown in Fig. 12.

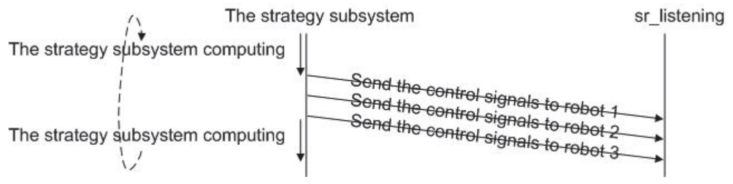

The strategy subsystem
sr_listening
The strategy subsystem computing
Send the control signals to robot 1
Send the control signals to robot 2
Send the control signals to robot 3
The strategy subsystem computing

Fig. 12. The SRP protocol.

The strategy subsystem, learnt empirically in the simulator, decides how the robot moves by receiving environmental situations from the vision system and makes decisions through the proposed mechanism.
The control subsystem is initially in charge of the robots (before starting the soccer robot game), and also in charge of beginning and maintaining the connection between control system and robots through Bluetooth. In addition, it also has the responsibility of controlling how robots act when the strategy subsystem does not control the robots in order to handle exceptions. The connection beginning procedure is described in the flowchart shown in Fig 13. In the figure, the service program, sr_listening, is a program running in each robot's embedded system to receive commands from the control center.

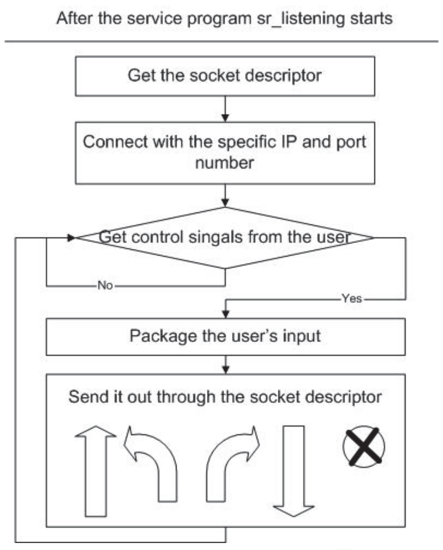

After the service program sr_listening starts
Get the socket descriptor
Connect with the specific IP and port number
Get control singals from the user
No
Yes
Package the user's input
Send it out through the socket descriptor

Fig. 13. The flowchart of making and maintaining the connection between the control center and robots.

The embedded system is the core of soccer robots. The duty of this system is to both receive the commands from the control center, and to control locomotion. The overall system architecture of soccer robots is shown in Fig. 14. A program called sr_listening implements the receiving command function. It is a non-stop procedure and begins to run when the robot power is switched on. The flowchart of the procedure is shown in Fig. 15. A sr_listening process invokes a threat, as the flowchart shows. The threat of each robot has its own port number and service number. According to these numbers the procedure can determine whether or not the message sent needs to be processed. The Motorola 68K core series MC68VZ328 processor has been used. Since it has low power consumption and high performance, it is appropriate for use in a soccer robot. The driving subsystem receives reference signals from the embedded system. Thus, the control system of the wheel is in the embedded system and is a software implementation. The advantage is that the control rule can be modified without changing the hardware, and so is more flexible. The control rule here is a simple open loop control, which means that there are no sensors on the wheels to feedback speed or other information.

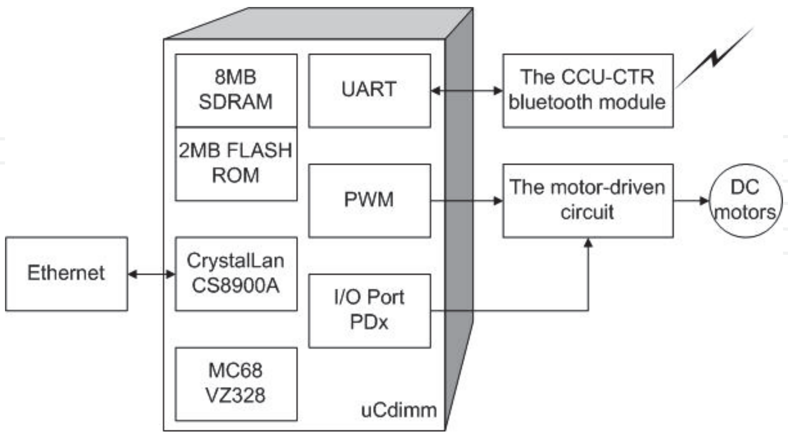


Fig. 14. The architecture of a soccer robot system

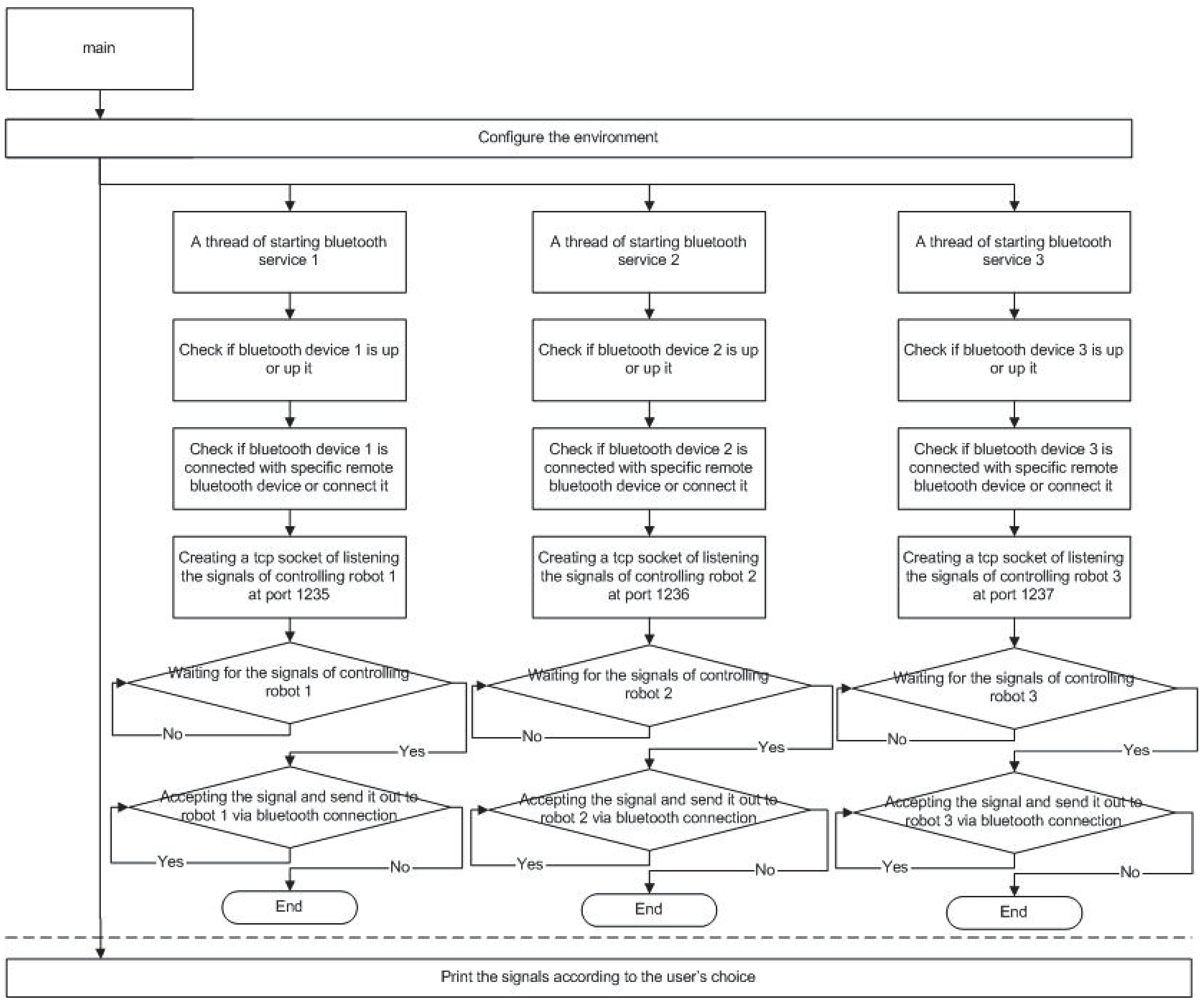

main
Configure the environment
A thread of starting bluetooth service 1
Check if bluetooth device 1 is up or up it
Check if bluetooth device 1 is connected with specific remote bluetooth device or connect it
Creating a tcp socket of listening the signals of controlling robot 1 at port 1235
Waiting for the signals of controlling robot 1
No
Yes
Accepting the signal and send it out to robot 1 via bluetooth connection
Yes
No
End
A thread of starting bluetooth service 2
Check if bluetooth device 2 is up or up it
Check if bluetooth device 2 is connected with specific remote bluetooth device or connect it
Creating a tcp socket of listening the signals of controlling robot 2 at port 1236
Waiting for the signals of controlling robot 2
No
Yes
Accepting the signal and send it out to robot 2 via bluetooth connection
Yes
No
End
A thread of starting bluetooth service 3
Check if bluetooth device 3 is up or up it
Check if bluetooth device 3 is connected with specific remote bluetooth device or connect it
Creating a tcp socket of listening the signals of controlling robot 3 at port 1237
Waiting for the signals of controlling robot 3
No
Yes
Accepting the signal and send it out to robot 3 via bluetooth connection
Yes
No
End
Print the signals according to the user's choice

Fig. 15. The flowchart of the procedure sr_listening

5.4 Experiments

The experiments are executed on a standard robot soccer game field. As Fig.16, there are two teams in these experiments. One team in right side has been trained by a simulator, but the other has had no training at all. The names, Green, Pink and Purple will be referred to the robot which is with the green, pink and purple mask, respectively. The opponents just stand still on the field to emulate a fair competition situation for comparison. The experiments are designed to compare the results of two teams, each of which is playing against the same opponent for ten rounds with around five minutes time elapsed such that the performance of the proposed algorithm is evaluated by comparing the performance of these two teams. Table 1 shows results of algorithms versus Lingo. The results of each strategy do not improve very much. The Q(λ)-learning algorithm gets the highest scores and a little improvement after learning, because it is designed to find the maximum value and it uses eligibility traces. The minimax-Q algorithm is not work very well in this situation, because the opponent is too simple. The zero-sum game theory does not work very well when the opponent is not smart. The scores of Zero-Sum-Q(λ) algorithm can get a little improvement after learning, because it uses eligibility traces.

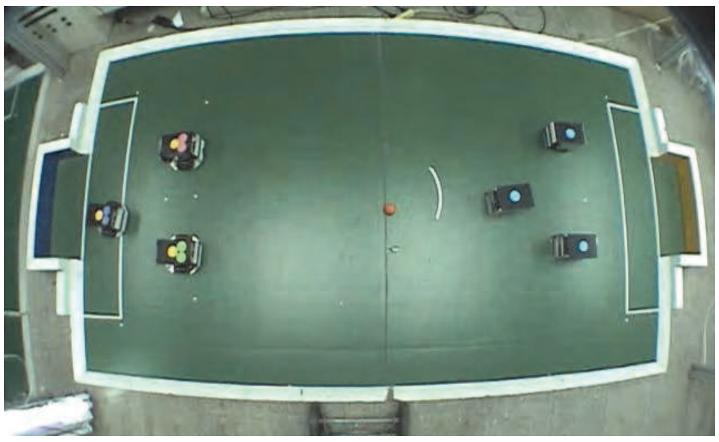


Fig. 16. The experiment platform

Scores	Q(λ)-learning Vs. Lingo	Minimax-Q Vs. Lingo	Zero-Sum-Q(λ) Vs. Lingo
My	13~14	11	11~13
Opponent	2	1.5~2	2~1.5
Win	11-12	9	9~11

Table 1. Results of algorithms vs. Lingo

The Fig. 17 and Fig. 18 illustrate the results of experiment that compete Zero-Sum-Q(λ) team(ZSQ(λ)) with Lingo team. In Fig. 17, the Zero-Sum-Q(λ) team does not have any experiences, the team trains policy in the game process. The difference of score is not obvious. In Fig. 18, the Zero-Sum-Q(λ) team that has trained can show the ability of offenseand defence. The complete score is recorded in Table 2, which indicates that the non-trained team scored fewer points (22) than the trained team (25). In addition, the non-trained team lost more scores (21) than the trained team (5). This shows that the learning mechanism has influenced the behaviours of soccer robots and the hardware has worked effectively.

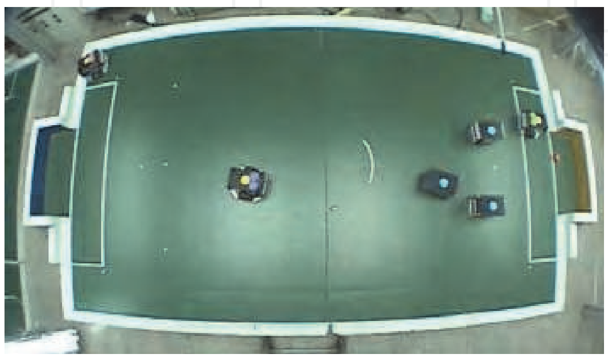


Fig. 17. (a) Game 1: ZSQ(λ) scored at the time 01:10 when Green dribbled the ball into the goal straight.

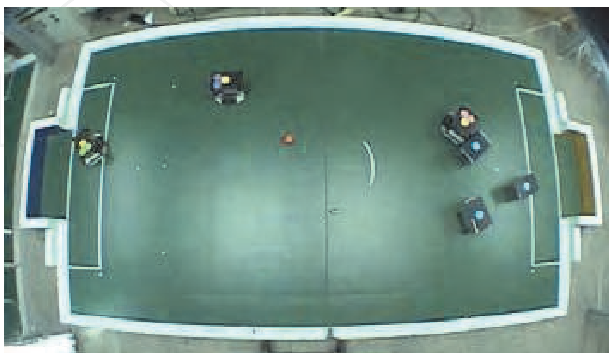


Fig. 17. (b) Game 2: In this game, ZSQ(λ) didn’t score any point.

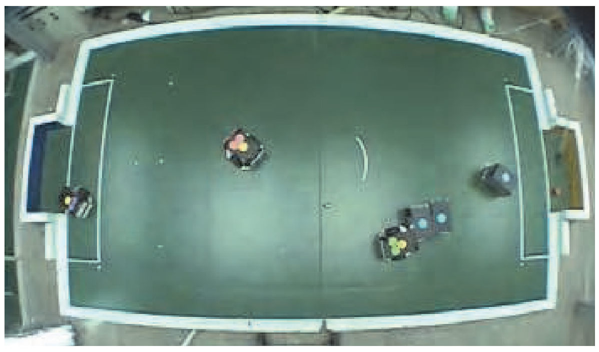


Fig. 17. (c) Game 3: ZSQ(λ) scored at the time 00:15 and 03:07. The ball rebounded toward the goal by Green.

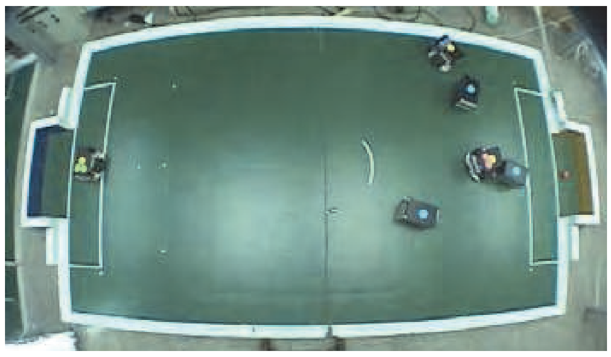


Fig. 17. (d) Game 4: ZSQ(λ) scored at the time 02:48, 02:57 and 03:48. The Pink dribbled the ball into the goal.

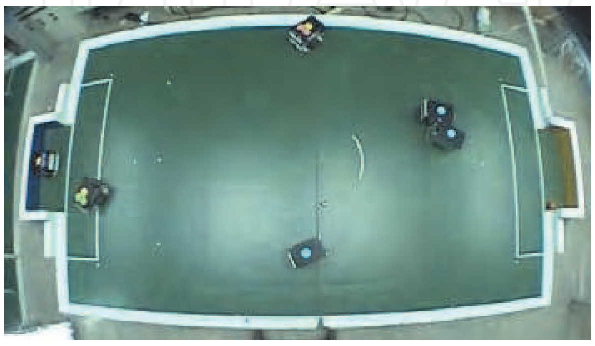


Fig. 17. (e) Game 5: ZSQ(λ) scored at the time 00:10, 01:52, 02:26 and 03:00. The ball rebounded toward the goal by Pink.

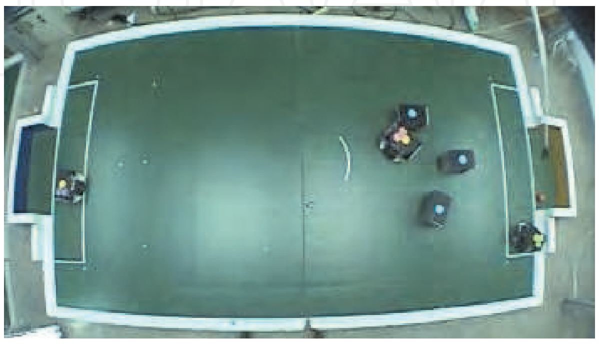


Fig. 17. (f) Game 6: ZSQ(λ) scored at the time 01:05, 02:48 and 04:00. The Green kicked the ball into the goal.

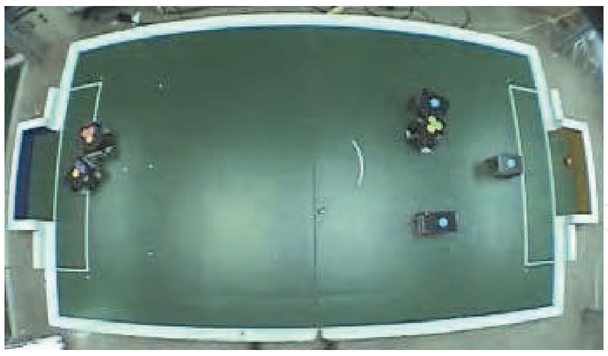


Fig. 17. (g) Game 7: ZSQ(λ) scored at the time 00:09 and 01:38. The Green kicked the ball straight toward the goal.

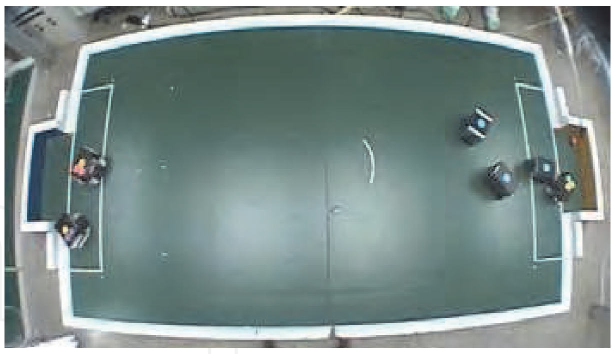


Fig. 17. (h) Game 8: ZSQ(λ) scored at the time 00:31, 01:28, 02:38 and 04:18. The Green got the fourth point.

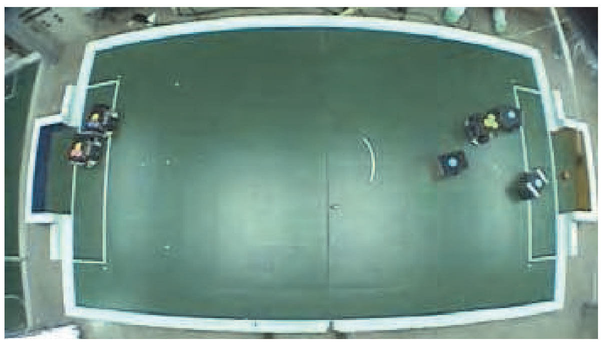


Fig. 17. (i) Game 9: ZSQ(λ) scored at the time 00:26 and 00:28. The ball was intercepted into the goal by Green.

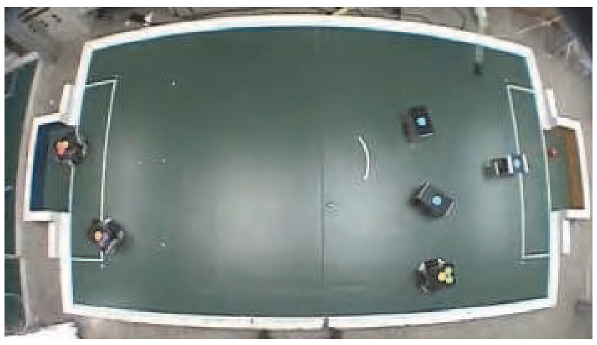


Fig. 17. (j) Game 10: ZSQ(λ) scored at the time 02:02. The ball rebounded toward the goal by Green.

Fig. 17. The snapshots of the competitions between the ZSQ(λ) team that had not trained and the Lingo team.

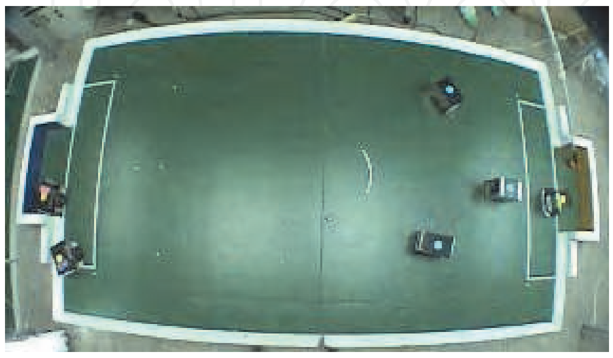


Fig. 18. (a) Game 1: ZSQ(λ) scored at the time 01:42. The Green dribbled the ball toward the goal.

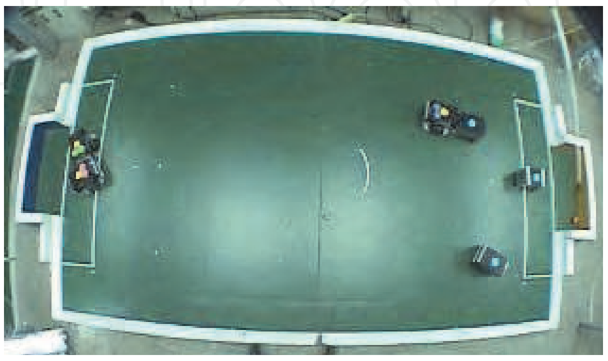


Fig. 18. (b) Game 2: ZSQ(λ) scored at the time 00:08, 00:15, 00:42, 00:53 and 01:49. The Purple kicked the ball toward the goal.

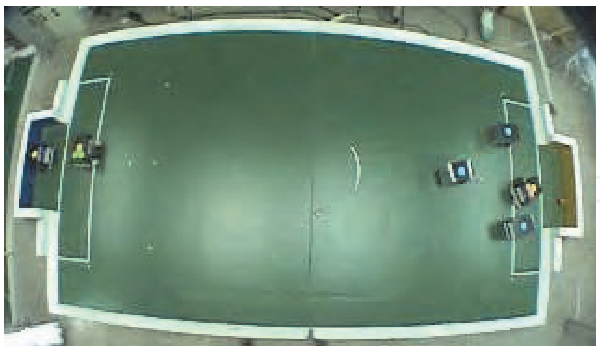


Fig. 18. (c) Game 3: ZSQ(λ) scored at the time 00:36 and 02:42. The Pink dribbled the ball toward the goal.

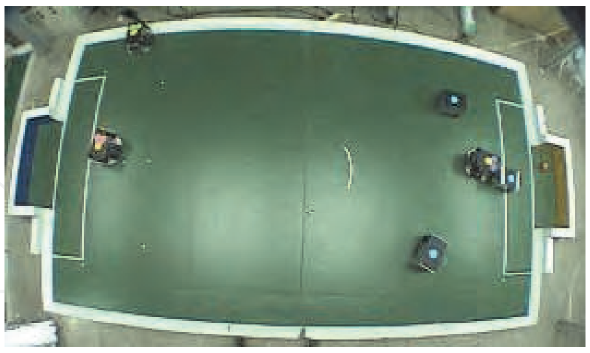


Fig. 18. (d) Game 4: ZSQ(λ) scored at the time 02:03, 02:15 and 03:42. The Purple dribbled the ball toward the goal.

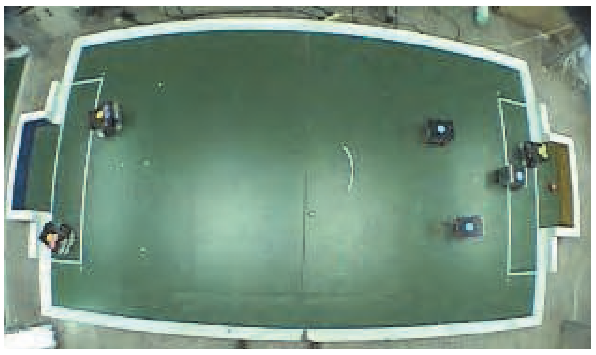


Fig. 18. (e) Game 5: ZSQ(λ) scored at the time 01:26, 03:20 and 04:38. The Green got the third point.

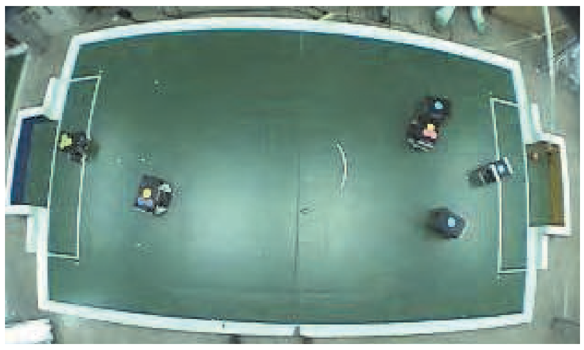


Fig. 18. (f) Game 6: ZSQ(λ) scored at the time 02:00 and 02:54. The Pink kicked the ball toward the goal.

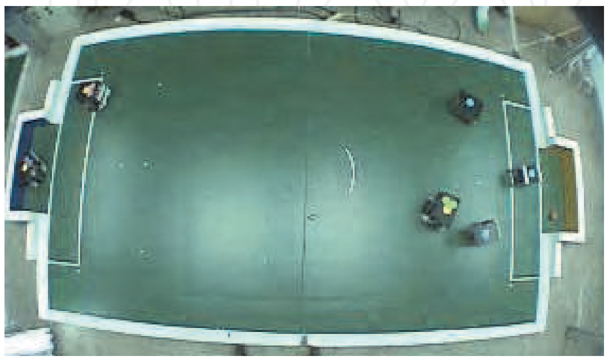


Fig. 18. (g) Game 7: ZSQ(λ) scored at the time 00:16, 02:18 and 03:07. The Green dribbled the ball toward the goal.

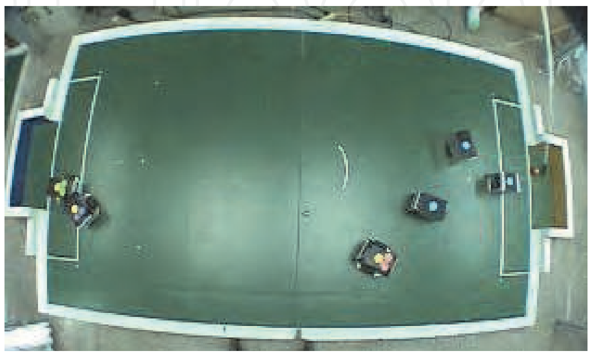


Fig. 18. (h) Game 8: ZSQ(λ) scored at the time 03:14. The Pink kicked the ball toward the goal.

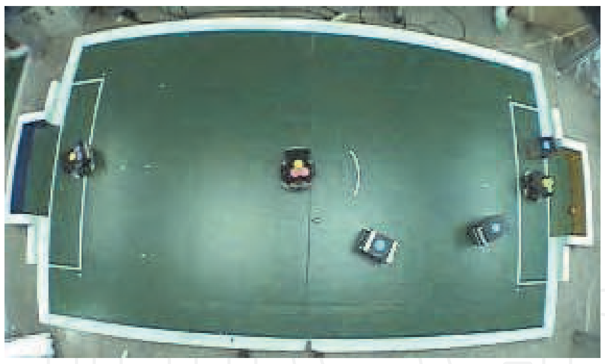


Fig. 18. (i) Game 9: ZSQ(λ) scored at the time 00:45, 01:37 and 03:39. The Green dribbled the ball toward the goal.

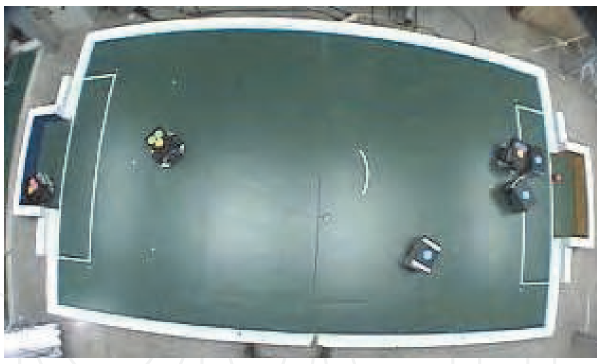


Fig. 18. (j) Game 10: ZSQ(λ) scored Expert scored at the time 03:43 and 04:23. The ball rebounded toward the goal by Green.

Fig. 18. The snapshots of the competitions between the ZSQ(λ) team that had trained and the Lingo team.

	Non-Trained Team		Trained Team	
Times	Scores	Time	Scores	Time
1	1:4	6′09″	1:1	4′50″
2	0:3	5′21″	5:0	4′38″
3	2:0	4′50″	2:1	3′28″
4	3:3	4′40″	3:0	3′51″
5	4:1	5″04″	3:0	5′51″
6	3:2	4′27″	2:0	5′08″
7	2:1	4′46″	3:2	4′12″
8	4:2	5′07″	1:1	3′41″
9	2:4	4′07″	3:0	3′50″
10	1:1	4′17″	2:0	4′29″
Total	22:21	48′54″	25:5	44′02″

Table 2. Results of experiments

6. Conclusions

Multi-robot at a game is an important application of MAS, which can be used to demonstrate the theories and mechanisms of MAS. In addition, because cooperation is central to the operations of MAS, we develop a mechanism to achieve cooperation and demonstrate it in a soccer game. According to the experimental results, using the task assignment mechanism with self-improving ability is a feasible and successful method for cooperation. Comparison with other methods also shows that the proposed method is better because the reinforcement learning in the mechanism gives it a good learning ability to adapt to varying environments. Moreover, the principle in game theories also successfully improves the action-selection ability of the reinforcement learning by considering opponent behaviors. Solving the cooperation problem of the MAS not only can be applied to soccer robot systems, but also can solve strategy determining problems of cooperation. Of course, this study is not the first to use the game theory to improve the reinforcement learning, but the idea of using the task-assignment mechanism with the reinforcement learning is novel. The tasks in the mechanism should be determined in advance, which means that determining appropriate tasks depends on questions, and is a problem for future research. However, the proposed method is still a new approach to cooperation problems, despite the fact that the task self-determining ability merely gives the mechanism more functions and reduces the work of preparation.

7. References

A Flache & M.W. Macy. (2002). The Power Law of Learning, *Journal of Conflict Resolution*, Vol. 46, no. 5, pp. 629-653.

A Gosavi. (2009). Reinforcement Learning: A Tutorial Survey and Recent Advances. *INFORMS Jounral on Computing*. Vol 21(2), pp. 178-192.

A. G. Barto, R. S. Sutton, & C. W. Anderson. (1983). Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems, *IEEE Transactions on System, Man, and Cybernetics*, vol. 13, no. 5, pp. 834-846.

D. Xiao & A. H. Tan. (2007). Self-Organizing Neural Architectures and Cooperative Learning in a Multiagent Environment. *IEEE Transactions on systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 37, no. 6, pp. 1567-1580.

Federation of International Robot-soccer Association (FIRA) ,2009. http://www.fira.com.

George A. Bekey. (2005). Autonomous Robots, *The MIT Press*.

H. Yanco & L.A. Stein. (1993). An adaptive communication protocol for cooperating mobile robots, *in J. A. Meyer, H. L. Roitblat, and S.W. Wilson, editors, Animals to Animats: Proceedings of the Second International Conference on the Simulation of Adaptive Behavior*, pages 478--485. MIT Press, Cambridge.

J. Hu & M. P. Wellman. (1998). Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm, *Proceedings of 15th International Conference on Machine Learning*, Morgan Kaufmann, pp. 242-250.

J. R. Marden, G. Arslan, & J. S. Shamma. (2009). Cooperative Control and Potential Games. *IEEE transaction on Systems, man, and Cybernetics-Part B*, vol. 39, no. 6, pp. 1393-1407.

K. S. Hwang & J. S. Lin. (1998). Smoothing Trajectory Tracking of Three-Link Robot: A Self-Organizing CMAC Approach, *IEEE Transactions on System, Man, and Cybernetics-part B*, Vol. 28, no. 5, pp. 680-692, Oct. 1998.

K. S. Hwang. & Y. Z., Chen. (2007). Reinforcement Learning on Strategy Selection for Cooperative Robot System. *IEEE transaction on Systems, man, and Cybernetics-Part A*. Vol. 37, No. 6, pp. 1151-1157.

L. Busoniu, R. Babuska, & B. D. Schutter. (2008). A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 38, no. 2, pp. 156-172.

L. P. Kaelbling, M. L. Littman, & A. W. Moore. (1996). Reinforcement Learning: A Survey, *Journal of Artificial Intelligence Research* 4, pp. 237-285.

Littman, M. L. (1994). Markov Games as a Framework for Multi-agent Reinforcement Learning, *in W. W. Cohen & H. Hirsh, eds., `Proceedings of the Eleventh International*

Conference on Machine Learning (ML-94)', Morgan Kauffman Publishers, Inc., New Brunswick, NJ, pp. 157-163.

M. Tan. (1993). Multi-agent reinforcement learning: independent vs. cooperative agents," Proceedings of the Tenth International Conference on Machine Learning, Amherst, Morgan Kaufmann, pp. 330-337.

S. P. Bingulac. (1994). On the compatibility of adaptive controllers, *in Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory*, New York, pp. 8–16.

V. Vassiliades, A. Cleanthous, & C. Christodoulou. (2011). Multiagent Reinforcement Learning:Spiking and Nonspiking Agents in the Iterated Prisoner's Dilemma. *IEEE Transactions on Neural Network,* vol. 22, no. 4, pp. 639-653.