HELIOS2013 Team Description Paper

Hidehisa Akiyama[1], Tomoharu Nakashima[2], and Katsuhiro Yamashita[2]

Faculty of Engineering, Fukuoka University, Japan[1]
`akym@fukuoka-u.ac.jp`
Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Japan[2]
`tomoharu.nakashima@kis.osakafu-u.ac.jp`,
`katsuhiro.yamashita@cs.osakafu-u.ac.jp`

Abstract. HELIOS2013 is a 2D soccer simulation team which has been participating in the RoboCup competition since 2000. We recently focus on an online multiagent planning using tree search methodology. This paper describes the overview of our search framework and opponent modeling approach to improve the search result.

1 Introduction

HELIOS2013 is a simulated soccer team for the RoboCup soccer 2D simulation league. The team has been participating in the RoboCup competition since 2000, and won 2 championships and 2 runner-ups in the past 4 years of RoboCup competitions. Moreover, the team have released a part of their source codes in order to help new teams to participate in the competitions and to start the research of multiagent systems.

We recently focus on an online multiagent planning using a tree search methodology. In this paper, we briefly introduce our search framework and an approach for predicting opponent positions in order to improve the accuracy of search results.

2 Related Works

We have released several open source software packages that help us to develop a simulated soccer team[1]. Now, we are mainly maintaining following software packages:

- librcsc: a base library for a simulated soccer team.
- agent2d: a sample team program using librcsc. Newbies can use agent2d as a start point for developing their own team.
- soccerwindow2: a high functional viewer, which can be used as a monitor client, a log player and a visual debugger.

[1] Available at: http://sourceforge.jp/projects/rctools/

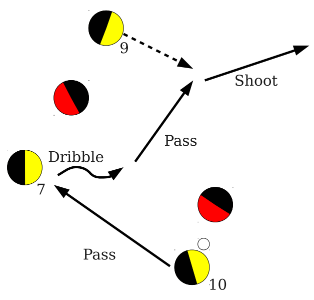

Fig. 1. An example of action sequence. This image shows the chain of four actions: 1) pass from Player 10 to Player 7, 2) dribbling by Player 7, 3) pass from Player 7 to Player 9, and 4) Player 9 shoots to the goal.

– fedit2: a formation editor for agent2d. fedit2 enables us to design a team formation using human's intuitive operations.

They are implemented from scratch without any source codes of other simulated soccer teams. However, several idea were inspired from other released codes, such as CMUnited[1], FC Portugal[2], YowAI[3], TsinguAeolus[4], UvA Trilearn[5, 6] and Brainstormers[7, 8]. We would like to thank those teams for their effort and achievements.

In previous years, we proposed a team formation model that uses Delaunay triangulation [9] and a multiagent planning framework [10] described in the next section. They have been already available in the released software.

3 Online Multiagent Planning using Tree Search

We developed a framework to search for the suitable action sequence in a continuous state-action space using a game tree search methodology. The framework enables an agent to plan cooperative behavior which involves other agents. The first version of search framework was developed in 2009, and we have been continuously improving it in various way. For more details of this framework, please refer [10, 11].

3.1 Framework for Searching for Action Sequences

In order to simplify the problem, we consider only ball kicking actions in offensive situations. This means that a cooperative behavior can be represented as a sequence of kick actions that are taken by multiple agents. Under this assumption, a cooperative behavior can be generated by tree search algorithms.

The framework generates and evaluates a number of action sequences performed by multiple agents. Generated actions are stored as a node of a search tree. A path from the root node to a leaf node represents an action sequence that

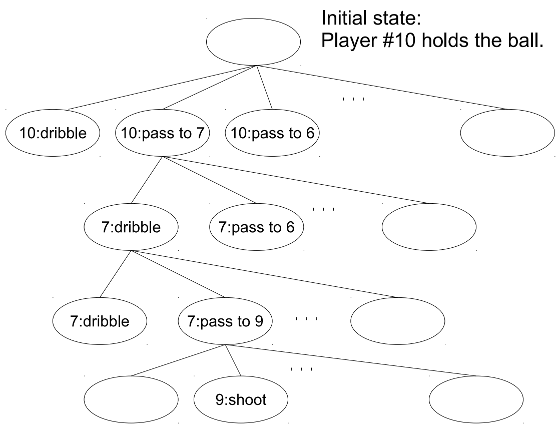

Fig. 2. Example of search tree. A multi-branch tree search is performed. Each node corresponds to an action-state pair instance. The predicted state maintained in the preceding node is used to generate new nodes.

defines an offense plan taken by multiple agents. Figure 1 shows an example of an action sequence.

The concept of the tree search used in this framework is shown in figure 2. As shown in this figure, a multi-branch tree is used in the tree search. Each node in the tree corresponds to an action-state pair. The result action sequence is obtained by traversing a path from the root node to a leaf node.

In the current implementation, we employed the best first search algorithm [12] as a tree search algorithm. Each node has a value calculated by an evaluation function based on the corresponding action-state pair instance. Since we use the best first search algorithm, the convergence of search is not guaranteed.

3.2 Current Problems

Some experiments show that the maximum tree depth has no correlation to the team performance. We guess that the oscillation in the action sequence generation caused by the poor accuracy of predicted state produced these results. This means that it is necessary to establish the method to predict future state more accurately. We are now trying to construct the model for predicting the opponent positions in order to solve this problem.

4 Predicting Opponent Positions for Generating Action Sequences

4.1 Overview

As stated in the last section, One of the characteristic features of HELIOS is the use of action sequences in determining the next action. The process of generating action sequences involves the prediction of all players from both teams. In the last year's implementation, the prediction of opponent players was done in a simple way: The opponent players are assumed not to move from their current positions during the generation process of action sequences. In this paper, we introduce the prediction of opponent positions that is achieved by using neural networks.

4.2 Neural Networks for Learning Opponent Positions

Three-layered neural networks are used to learn the opponent positions. One neural network corresponds to the mapping from a field status to the position of an opponent player. Thus, there are 11 neural networks in total for learning the mapping of 11 opponent players. Each neural network has five input units, ten hidden units, and two output units. The field status of a particular time step is represented by a five-dimensional vector that includes the current position of the ball, and the smallest time step to reach the ball among all players, and the estimated ball position after the smallest time step. The output of the neural network is the predicted position of an opponent player for the corresponding field status.

4.3 Experiments

In order to evaluate the effectiveness of the neural network for predicting the opponent positions, a series of experiments are performed. The coach agent is used to train the neural networks as it can access the perfect information on the soccer field during the match. The coach agent collects the training patterns during the play_on status of the first half (i.e., the first 3000 time steps) and records the input-output information that are used in the back-propagation learning of 11 neural networks. During the half-time break, the coach agent performs the training of 11 neural networks and sends the weights of trained neural networks to all teammate players. The performance of trained neural networks are evaluated by measuring the distance (i.e., prediction error) between the predicted opponent positions and their actual ones for the training patterns. This experimental setting is applied for the performance evaluation of the neural networks against six teams from RoboCup2012 and agent2d.

The experimental results are shown in Table 1. From this table, it can be seen that the opponent positions are predicted with the error of around 10m. It should be noted that the prediction performance of neural networks is almost the same for any opponent teams.

Next, the performance of neural networks is evaluated for unseen patterns. That is, each field player uses the trained neural networks that are sent from the coach agent in the second half. In the second half, neural networks remains unchanged and not trained any further. The neural networks are used to predict opponent positions by the kicker player who possess the ball. The prediction performance of the trained neural networks for the second half is shown in Table 2. Table 2 also shows the prediction performance of the conventional HELIOS2012

Table 1. Performance evaluation of trained neural networks for the first half.

Opponent	Average prediction error (m)	Standard deviation
agent2d	10.53	7.89
WrightEagle	11.88	8.04
Marlik	11.07	9.01
Ri-one	11.26	6.84
Oxsy	13.62	9.21
YuShan	13.57	9.30
NADCO-2D	10.85	8.00

that only assumes the position of opponent players are fixed during the generation of action sequences. From this table, it is found that the neural networks do not work much better than the conventional prediction method in terms of average errors. However, the low values of the standard deviation show the stability in the prediction by the trained neural networks.

Table 2. Performance evaluation of trained neural networks for the second half.

Opponent	HELIOS2012	HELIOS2012+Neural networks
agent2d	15.00 (±78.2)	17.80 (±27.14)
WrightEagle	15.06 (±101.76)	17.56 (±25.30)
Marlik	27.55 (±100.17)	24.97 (±31.25)
Ri-one	12.92 (±70.30)	16.89 (±16.10)
Oxsy	18.27 (±78.86)	21.40 (±16.82)
YuShan	10.51 (±44.19)	16.65 (±18.20)
NADCO-2D	19.55 (±84.31)	20.06 (±28.61)

5 Conclusion and Future Works

This paper described the research focus and the current effort of HELIOS2013. We have been applying a game tree search methodology for online multiagent planning. Now, we are trying to construct the model for predicting opponent positions in order to improve the accuracy of search results. The prediction model still needs to be improved, but the experiments showed promising results.

As future works, we have to establish more effective search algorithm. We are now trying to introduce various game tree search algorithms such as Monte Carlo Tree Search [13].

References

1. Stone, P., Riley, P., Veloso, M.: The CMUnited-99 champion simulator team. In Veloso, M., Pagello, E., Kitano, H., eds.: RoboCup-99: Robot Soccer World Cup

III. Volume 1856 of Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin (2000) 35–48

2. Reis, L.P., Lau, N., Olivéira, E.: Situation based strategic positioning for coordinating a simulated robosoccer team. Balancing Reactivity and Social Deliberation in MAS (2001) 175–197
3. Nakayama, K., Ako, T., Suzuki, T., Takeuchi, I.: Team YowAI-2002. RoboCup 2002: Robot Soccer World Cup VI (2002)
4. Yao, J., Chen, J., Cai, Y., Li, S.: Architecture of tsinghuaeolus. In: RoboCup 2001: Robot Soccer World Cup V, Springer-Verlag (2002) 491–494
5. Kok, J.R., Vlassis, N., Groen, F.: UvA Trilearn 2003 team description. In Polani, D., Browning, B., Bonarini, A., Yoshida, K., eds.: Proceedings CD RoboCup 2003, Padua, Italy, Springer-Verlag (July 2003)
6. UvA Trilearn 2003. http://www.science.uva.nl/˜jellekok/robocup/2003/
7. Riedmiller, M., Gabel, T., Knabe, J., , Strasdat, H.: Brainstormers 2d - team description 2005. In Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y., eds.: Proceedings CD RoboCup 2005, Springer-Verlag (2005)
8. Brainstormers 2d: Brainstormers Public Source Code Release. http://sourceforge.net/projects/bsrelease/
9. Akiyama, H., Noda, I.: Multi-agent positioning mechanism in the dynamic environment. In: RoboCup 2007: Robot Soccer World Cup XI. (2008)
10. Akiyama, H., Nakashima, T., Aramaki, S.: Online cooperative behavior planning using a tree search method in the robocup soccer simulation. In: Proceedings of 4th IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS-2012). (2012)
11. Akiyama, H., Nakashima, T.: HELIOS2012: RoboCup 2012 Soccer Simulation 2D League Champion. RoboCup-2012: Robot Soccer World Cup XVI (2013)
12. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall (2009)
13. Gelly, S., Wang, Y., Munos, R., Teytaud, O.: Modification of UCT with patterns in monte-carlo go. Technical report, INRIA RR-6062 (2006)