HELIOS2012: RoboCup 2012 Soccer Simulation 2D League Champion

Hidehisa Akiyama[1] and Tomoharu Nakashima[2]

[1] Faculty of Engineering, Fukuoka University, Japan
akym@fukuoka-u.ac.jp
[2] Graduate School of Engineering, Osaka Prefecture University, Japan
tomoharu.nakashima@kis.osakafu-u.ac.jp

Abstract. The Soccer Simulation 2D League is one of the oldest competitions among the RoboCup leagues. In the simulation 2D league, the simulator enables two teams of 11 simulated autonomous agents to play a game of soccer with highly realistic rules and game play. This paper introduces the RoboCup 2012 Soccer Simulation 2D League champion team, HELIOS2012, a joint team of Fukuoka University and Osaka Prefecture University.

1 Introduction

The RoboCup Soccer Simulation 2D League is one of the oldest competitions among the RoboCup leagues. It is based on the RoboCup Soccer 2D Simulator [1] that enables two teams of 11 autonomous player agents and an autonomous coach agent to play a game of soccer with highly realistic rules and game play. Due to its stability, the 2D soccer simulator is a very good research and educational tool for multiagent systems, artificial intelligence, and machine learning.

The 2D soccer simulator models only the (x, y) positions of objects. The players and the ball are modeled as circles. In addition to its (x, y) location, each player has a direction that its body is facing, which specifies the direction it can move, and a separate direction in which it is looking, which determines the vision area that the agent covers. Actions are abstract commands such as turning the body or neck by a specified angle, dashing to one of eight directions with a specified power, kicking at a specified angle with a specified power (when the ball is near), or slide tackling in a given direction. A team consists of 11 players including a goalie that has special capabilities of catching the ball when it is near the goalie. The 2D soccer simulator does not model the physical motion of any particular robot, but does capture realistic team level strategic interactions.

In 2012, up to 24 teams were allowed to participate in the 2D competitions. Since the teams competing in the 2D League are already highly competitive, the qualification was done based on a measurement of the quality of the team's scientific work expressed in the submitted team description paper and also the log files with appropriate annotations. Finally, the 2D competitions included 19 teams from 9 countries.

X. Chen et al. (Eds.): RoboCup 2012, LNAI 7500, pp. 13–19, 2013.

This paper introduces the RoboCup 2012 Soccer Simulation 2D League champion team, HELIOS2012, a joint team of Fukuoka University and Osaka Prefecture University. Especially, we explain the planning framework implemented in team HELIOS2012. There are two characteristic features in HELIOS2012. One is online multiagent planning using tree search, and the other is the decrease in oscillations in decision making. The online multiagent planning using tree search was first implemented in 2010, when HELIOS won the first championship in RoboCup 2010. This framework plays an important role again for more flexible and appropriate action selection in RoboCup 2012. The technique for decreasing oscillations in decision making gives the stability of agent's decision making so that a particular action sequence is likely to be fully completed before the agent changes its mind.

The remainder of this paper is organized as follows. Section 2 introduces the tree search framework implemented in HELIOS2012. Section 3 introduces the modified evaluation function model to decrease oscillations during planning iterations with the tree search framework. Section 4 shows the result of RoboCup 2012 competitions. Section 5 concludes.

2 Online Multiagent Planning Using Tree Search

This section presents the tree search framework for online multiagent planning. This framework is implemented in HELIOS2012. It enables an agent to plan cooperative behavior which involves other agents. For more details of this framework, please refer [2].

2.1 Framework for Searching Action Sequence

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 in a continuous state-action space. 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 defines an offense plan taken by multiple agents. Figure 1 shows an example of an action sequence.

This framework generates action sequences and evaluates their values using the following modules:

– ActionGenerator: This module generates candidate action instances for a node in the search tree. An action instance is generated if it is likely to be performed successfully. The action instance and the predicted state are combined to form an action-state pair instance. The action-state pair instance is added as a new node in the search tree.

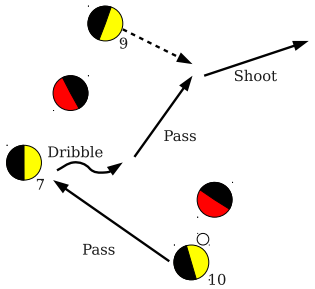


Fig. 1. An example of an action sequence. The chain of four actions is shown: 1) pass from Player 10 to Player 7, 2) dribbling by Player 7, 3) pass from Player 7 to Player 9, and 4) Player 9 shoots to the goal.

– FieldEvaluator: This module evaluates the value of the action-state pair instances that are generated by ActionGenerator. We introduced various state variables and hand-coded rules into the implementd Evaluator instance. The rules evaluate each state variable and the sum of them is returned as the value of action sequence.

In the current implementation, we employed the best first search algorithm [3] as a tree search algorithm. Each node has a value calculated by FieldEvaluator based on the corresponding action-state pair instance.

2.2 Experiments

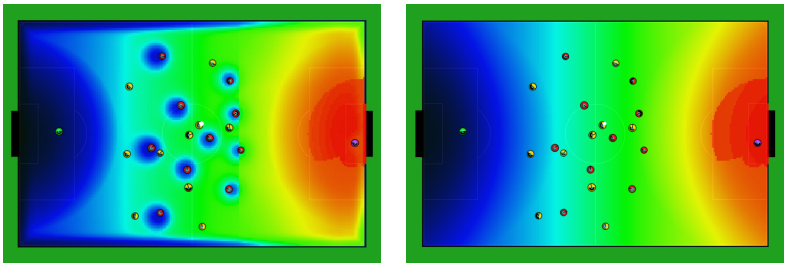
In order to analyze the performance of our framework, we performed computational experiments with several parameter specifications. We used the following parameters:

– Maximum tree depth : { 1(no tree search), 2, 3, 4, 5 }
– Maximum number of traversed node : { 10, 100, 1000, 10000, 100000 }
– ActionGenerator : { Normal, Reduced }
– FieldEvaluator : { Complex, Simple }

Type Normal for ActionGenerator is the same as the one used by HELIOS2011. The number of actions that Type Reduced for ActionGenerator is allowed to generate is about a half of that for Type Normal.

Type Complex for FieldEvaluator is the same as the one used by HELIOS2011, which uses the hand-coded rules with various state variables. Type Simple for FieldEvaluator also uses the hand-coded rules, however they are much simpler than Type Complex. Figure 2 shows an example value mapping on the 2D simulation soccer field.

We used an average goal difference as a team performance indicator. Figure 3 shows the results for each parameter specification. All values are the average of 100 games. In all cases, we can find that the team performance becomes worse



(a) Example value mapping evaluated by the Complex type FieldEvaluator.

(b) Example value mapping evaluated by the Simple type FieldEvaluator.

Fig. 2. Example value mapping evaluated by FieldEvaluator used in the experiments. The red color means the highest value and the black means the lowest value.

when the maximum number of traversed node is ten. This is because agents easily fell into the local minimum. On the other hand, it seems that the team performance is stable if the maximum number of traversed node is more than or equals to 100. This result means that the valuable action sequences can be found within nearly 100 node traversals.

Figure 4 shows the results for each pair of ActionGenerator and FieldEvaluator. The results show that various state variables and rules should be considered in FieldEvaluator. Furthermore, we can find that the number of action patterns generated by ActionGenerator has some impact on the team performance. We could not find the clear reason why the maximum tree depth has no correlation to the team performance. We guess that the oscillation of decision making caused by the poor accuracy of predicted state produced these results.

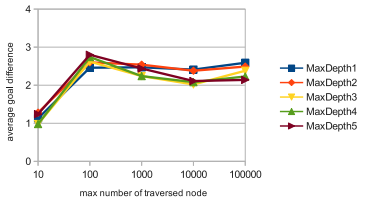
As future works, we have to establish the method to predict future state more accurately and have to establish more effective search algorithm. We are now trying to introduce various game tree search algorithms such as Monte Carlo Tree Search [4].

3 Decreasing Oscillations in Multiagent Planning

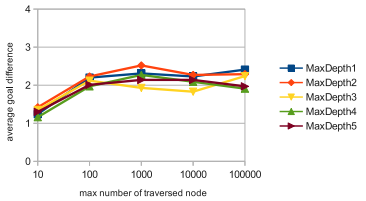
The oscillation of decision making in this paper is defined as follows: When the ball owner agent holds the ball more than one cycle,

- the action type is changed,
- the target player is changed, or
- the error of target position is over the pre-specified threshold.

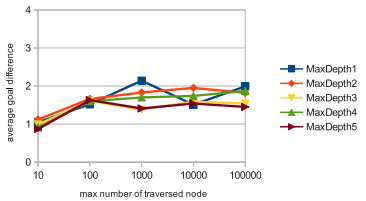
It is important to decrease the oscillations of decision making in order to stabilize the agent's behavior. In this section, we introduce a modified evaluation function model to decrease the oscillations of decision making.



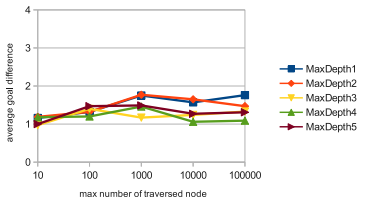
(a) ActionGenerator: Normal type. FieldEvaluator: Complex type.



(b) ActionGenerator: Reduced type. FieldEvaluator: Complex type.



(c) ActionGenerator: Normal type. FieldEvaluator: Simple type.



(d) ActionGenerator: Reduced type. FieldEvaluator: Simple type.

Fig. 3. The results of average goal difference for each setting

3.1 Modified Evaluation Function Model

We propose a modified evaluation function model that adjusts the values evaluated by FieldEvaluator. The evaluation value e is modified by the following equation:

$$e' = e \times \exp(-k\frac{||\boldsymbol{p}_{t_n} - \boldsymbol{p}_{t_m}||}{(1 + (t_n - t_m))}), \tag{1}$$

where e' is the modified evaluation value, t_n and t_m are the current time and the time at the previous decision making respectively, p_{t_n} and p_{t_m} are the current target position and the target position at the the same tree depth of the previous decision making, and k is a non-negative real value parameter to change the effect of the time and the distance.

3.2 Experiments

Table 1 shows that the proposed model decreases the oscillations of decision making. It seems that the suitable value of parameter k is between 1.0 and 5.0.

Table 2 and 3 shows the performance evaluation against the opponent teams that participated in the RoboCup2011. We performed 20 games for each team and analyzed the average ball possession ratio and the average goal difference. The result shows the ball possession becomes better for some teams. On the other hand, the goal difference becomes worse for most teams. It is necessary to analyze the games in more detail to evaluate the team performance.

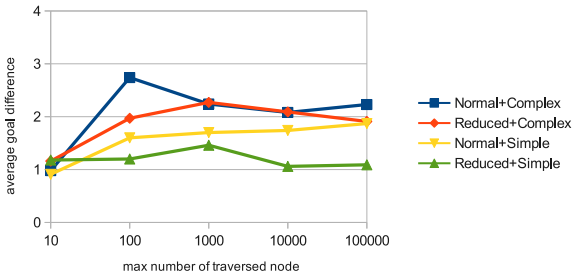


Fig. 4. The average goal difference for each pair of ActionGenerator and FieldEvaluator. The maximum tree depth is fixed to four. Each line corresponds to the pair of ActionGenerator and FieldEvaluator.

Table 1. The number of oscillations and their ratio. The results are the average values of 20 games against agent2d.

k	# of decision making	# of oscillations	ratio
0(No effect)	1926	1389	0.7212
0.1	2677	791	0.2955
0.5	3130	709	0.2265
1.0	3634	594	0.1635
3.0	4047	619	0.1530
5.0	4085	643	0.1574
10.0	4414	966	0.2188
50.0	5264	1012	0.1922
100.0	4676	963	0.2059

Table 2. Ball possession rate for each opponent team

	Without model	With model
agent2d	0.6578	**0.6965**
Edin	0.6429	**0.6840**
Hfut	0.5909	0.6014
Photon	0.5454	**0.6037**
RMAS	0.7720	0.7761
Wright	0.4381	0.4272

Table 3. Average goal difference for each opponent team

	Without model	With model
agent2d	2.5	2.8
Edin	**19.85**	17.15
Hfut	**19.2**	15.8
Photon	18.65	18.05
RMAS	**19.7**	16.7
Wright	4.0	3.0

4 RoboCup 2012 Soccer Simulation 2D League Results

In RoboCup 2012, team HELIOS2012 won the championship by winning all 22 games during the competition, scoring 118 goals and conceding 3 goals.[1] Team WrightEagle from University of Science and Technology of China won the second place, and team MarliK from University of Guilan of Iran won the third place.

5 Conclusion

This paper introduced the champion of RoboCup 2012 Soccer Simulation 2D league. First, we described the tree search approach for multiagent planning implemented in HELIOS2012. Second, we described the modified evaluation funciton model to decrease oscillations in planning iterations. The HELIOS2012 team won 2 championships and 2 runner-ups in the past 4 years of RoboCup competitions. Moreover, team HELIOS 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[2].

Acknowledgment. The authors would like to thank the additional contributing members of HELIOS2012 (Yosuke Narimoto and Katsuhiro Yamashita)

References

1. Noda, I., Matsubara, H.: Soccer server and researches on multi-agent systems. In: Kitano, H. (ed.) Proceedings of IROS 1996 Workshop on RoboCup, pp. 1–7 (November 1996)
2. Akiyama, H., Nakashima, T., Aramaki, S.: Online cooperative behavior planning using a tree search method in the robocup soccer simulation. In: Proceedings of the 4th International Conference on Intelligent Networking and Collaborative Systems (2012)
3. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall (2009)
4. Gelly, S., Wang, Y., Munos, R., Teytaud, O.: Modification of UCT with patterns in monte-carlo go. Technical report, INRIA RR-6062 (2006)

[1] The detailed competition results and all game log files can be found at: `http://www.socsim.robocup.org/files/2D/log/RoboCup2012/`

[2] More information about the HELIOS2012 team can be found at the team's website: `http://sourceforge.jp/projects/rctools/`