The Decision-Making Framework of WrightEagle, the RoboCup 2013 Soccer Simulation 2D League Champion Team

Haochong Zhang and Xiaoping Chen

Department of Computer Science, University of Science and Technology of China
xpchen@ustc.edu.cn

Abstract. This paper presents the latest progress of WrightEagle, the champion of RoboCup 2D simulation league. We introduce a decision-making framework, an extension of MAXQ-OP framework using multiple heuristic functions and a reachable state checking method. The experimental results show that our approach improves the quality of solutions in complex situations.

1 Introduction

The most important characteristic of Robocup 2D simulation league is that the soccer matches only run in computers, where there is only software without any physical robots or other hardwares. Without physical layers, e.g., motion control and image processing, Robocup 2D simulation teams can avoid hardware limitations and focus on research topics such as high level decision-making, learning, and multi-agent collaboration. Therefore, the Robocup 2D simulation competition becomes an important test bed of large-scale decision-making, machine learning and multiagent system research. Each participating team will compete with many other teams, which may be developed in different approaches. For instance, at Robocup 2013, there were a total of 20 teams from 9 countries entering the 2D simulation competition[1].

Taking into account the scale of the 2D simulation competition, making good decisions against so many different opponents is a very challenging task and dealing with complex and varied situations is very difficult[3].

This paper introduces the RoboCup 2013 Soccer Simulation 2D League champion team, WrightEagle, that has won 4 champions and 5 runners-up in the past 9 years. Particularly, we introduce some latest innovations in WrightEagle2D Simulation Team. These innovations expand and implement MAXQ-OP framework and focus on how to extend the search depth in the decision-making procedure to deal with complex and diverse situations and improve the quality of solutions.

The remainder of this paper is organized as follows. Section 2 introduces some background knowledge and motivations. Section 3 presents the decision-making

[1] The detailed competition results can be found at:
`http://www.oliverobst.eu/research/gliders2013-simulation-league-robocup-team-overview/robocup-2013`

framework we developed. Section 4 and Section 5 describe more details about the decision-making framework including the reachable states checking method and the evaluation system. Section 6 presents the experimental results. We discuss some related work briefly in Section 7 and conclude the paper in Section 8.

2 Background

2.1 Robocup Soccer Simulation League 2D

In 2D simulation competition, each team is constructed of 12 agents, including 11 players and an online coach. Then, all the agents separately connect to the competition server which simulates the world including the dynamics and kinematics of the players and ball. The information sent to each player in every decision cycle by the server is highly uncertain and incomplete, which depends on the field's and the agent's own situation. Then, after receiving information, each agent has to send the actions back to server for execution in the simulated world. The whole process takes no more than 100 milliseconds (ms) which is one of the most difficult forces, so the agents must respond in a rapid way. In 2D simulator, actions are abstract commands such as turning the body and neck by a specified angle, dashing to a specified angle with a specified power, kicking to a specified angle with a specified power or tackling in a specified angle. The 2D simulator does not model the motions of any real robot but an abstracted player.

In Robocup 2D simulation league, we face up to two challenges. Firstly, the search space is extremely large. The simulator simulates the whole continuous 2D space on the standard $65m \times 105m$ soccer field. Moreover, the number of agents greatly influences the decision-making efficiency. Since each player has 10 teammates, 11 opponents and a coach on both sides, to build the teammate model and the opponent model is an extremely demanding job. All of the above factors increase the difficulty of designing an effective search algorithm for this problem. Secondly, the other challenge is the limited computation time in each decision cycle. In every cycle, which is 100 ms, considering the network delay and other necessary time cost, the agent has to make a decision within 70 ms. Due to the above restrictions, it is a great challenge to extend the search depth to a satisfactory level.

2.2 MAXQ Hierarchical Decomposition

The WrightEagle team is developed base on the Markov Decision Processes (MDPs) framework [1] with the MAXQ hierarchical structure [2] and heuristic approximate online planning techniques in the past years[3][4].

Generally, the MAXQ technique decomposes a given MDP into a set of sub-MDPs arranged over a hierarchical structure. Each sub-MDP is treated as a distinct subtask.

Given the hierarchical structure, a *hierarchical policy* π is defined as a set of policies for each subtask $\pi = \{\pi_0, \pi_1, \cdots, \pi_n\}$, where π_i is a mapping from active states to actions $\pi_i : S_i \rightarrow A_i$. The *projected value function* of policy π for a subtask M_i in state s, $V^\pi(i,s)$, is defined as the expected value after following policy π in a state s until the subtask M_i terminates in one of its terminal states in G_i. Similarly, $Q^\pi(i,s,a)$ is the expected value by firstly performing action M_a in state s, and then following policy π until the termination of M_i. It is worth nothing that $V^\pi(a,s) = R(s,a)$ if M_a is a primitive action ($a \in A$).

Dietterich [2] has shown that the value function of policy π can be expressed recursively as:

$$Q^\pi(i,s,a) = V^\pi(a,s) + C^\pi(i,s,a) \tag{1}$$

where

$$V^\pi(i,s) = \begin{cases} R(s,i) & \text{if } M_i \text{ is primitive} \\ Q^\pi(i,s,\pi(s)) & \text{otherwise} \end{cases} \tag{2}$$

$C^\pi(i,s,a)$ is the *completion function* that estimates the cumulative reward received with the execution of action M_a before completing the subtask M_i, as defined below:

$$C^\pi(i,s,a) = \sum_{s',N} \gamma^N P(s',N|s,a) V^\pi(i,s'), \tag{3}$$

where $P(s',N|s,a)$ is the probability that subtask M_a in s terminates in s' after N steps.

3 The Decision-Making Framework

This section presents a decision-making framework that has been implemented in WrightEagle 2D simulation team for for the macro action attack. This framework implements and extends the MAXQ-OP framework. We introduce more complete ideas of heuristic search and other techniques for transforming MAXQ-OP framework to adapt to the needs of the 2D simulation league. In past years, we have used MAXQ-OP framework to define a series of subtasks at different levels[4], the decision-making framework is operating at the level of shoot, dribble, and pass etc.

In order to maximally approximate the optimal $Q^\pi(i,s,a)$ values, we use a heuristic method to choose the direction of extension of the $Q^\pi(i,s,a)$.

Overall decision-making framework presented in Algorithm 1 is similar to the best-first search. The search process forms a search tree starting from the initial state and always extends the state which has the best evaluation. The framework generates next level states from a certain state and pushes these states into an ordered list sorted by evaluation. And we introduce the idea of anytime algorithm into the decision-making framework and limited search depth to an acceptable range.

In order to meet the time cost restrictions of Robocup 2D Simulation League meanwhile making good decision results, the framework has following features and components:

State and Action. The state in the framework records most of the information from the field, such as the positions of all players and ball. Except field information, we add the transition probability from an initial state to the current state. We will introduce more information of the transition probability in Section 4. We use macro actions, defined in MAXQ-OP framework[4], to get the successor states.

Evaluation System. In order to evaluate and choose an action, the MAXQ-OP framework compute the optimal *projected value function.* And to meet the needs of the Robocup 2D simulation league, we have to extend the MAXQ-OP framework. Then, we develop the evaluation system to evaluate all the actions and states generated by decision-making framework. More details are introduced in Section 4.

Reachable States Checking. To achieve better search results in limited time, we have to skip those unreachable states during the search process. We developed a reachable states checking method. We will presents more details in Section 5.

Algorithm 1. Decision(s)

```
Input: A state, s.
Output: Evaluation of the state.
 1: $MaxEva \leftarrow -\infty$
 2: $SearchNode \leftarrow 0$
 3: Insert(SearchList,InitState)
 4: while Empty(SearchList) = False and $SearchNode < MaxSearchNode$ do
 5:     $State \leftarrow Top(SearchList)$
 6:     Pop(SearchList);
 7:     $SearchNode \leftarrow SearchNode + 1$
 8:     if $GetDepth(State) \geq MaxSearchStep$ then
 9:         continue
10:     else
11:         for all Teammate do
12:             $BehaviorList \leftarrow CalcAction(Teammate, State)$
13:             for all Behavior in BehaviorList do
14:                 $NextState \leftarrow GenerateNextState(Behavior)$
15:                 $NextEva \leftarrow Evaluate(NextState)$
16:                 $MaxEva \leftarrow Max(MaxEva, NextEva)$
17:                 Insert(SearchList,NextState)
18:                 SortByEvaluation(SearchList);
19:             end for
20:         end for
21:     end if
22: end while
23: return MaxEva
```

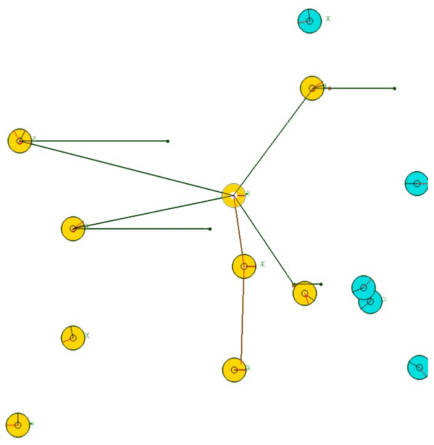

Fig. 1. Example of search process

4 The Evalution System

As we mentioned in Section 3, we hope that algorithm approximate the optimal $Q^{\pi}(i,s,a)$ value as soon as possible. We use heuristic method to choose the direction of extension of $Q^{\pi}(i,s,a)$. This heuristic method is natural to combine with the sparse property of the reward function of Robocup 2D simulation league.

From the perspective of soccer, when scoring, the soccer team get a high reward. Otherwise the reward is 0. The reward function has a sparse property which has the effect that the forward search process often terminates without any reward obtained. So we have to find an approach to help us to distinguish the states with a high possibility of scoring from all states to accelerate the search process and seize opportunities on the competition. An intuitive idea is using a heuristic function as the reward function.

We developed a heuristic evaluation system for long-term search algorithm. The input is a state and the output is the estimates of the possibility to score from this state. The evaluation system is composed of simple and intuitive heuristic functions such as the distance of the ball to the opponent's goal.

After combining MAXQ hierarchical decomposition and heuristic reward, we get a complete evaluation system as:

$$Q(i,s,a) = V(a,s) + C(i,s,a) \tag{4}$$

$$V(i,s) = \begin{cases} max_a Q(i,s,a) & \text{if i is composite} \\ R(s,i) & \text{otherwise} \end{cases} \tag{5}$$

and

$$C(i,s,a) = \sum_{s',N} \gamma^N T(s',N|s,a) V(i,s') \tag{6}$$

$$R(s,i) = \begin{cases} H(s,i) & \text{if s is a terminal state} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Figure 2 and Figure 3 are examples of this Evaluation System.

$$\begin{tikzcd}
\text{start} \arrow[r] & S_0 \quad R=0 \arrow[d, "{a_1,P_1,C_1}"] \\
& S_1 \quad R=0 \arrow[d, "{a_2,P_2,C_2}"] \\
& S_2 \quad R=0 \arrow[d, "{a_3,P_3,C_3}"] \\
& S_3
\end{tikzcd}$$

$$HeuristicReward = H(S_3)$$

$$E_3 = P_1 * P_2 * P_3 * R(S_3) * \gamma^{(C_1+C_2+C_3)}$$
$$E(S_0, a_1) = max(E_1, E_2, E_3)$$

Fig. 2. An example of evaluation system. States S_0, S_1, S_2 are intermediate states and state S_3 is the terminal state.a_1, a_2, a_3 are actions. P_1, P_2, P_3 are transition probabilities. C is the number of steps of corresponding macro actions.

When the search precess begins, starting from the initial state, the evaluation system evaluates the initial state to get a heuristic reward, $H(s)$. Then the algorithm will extend this state to get the successor states. When a new state is found, the algorithm replaces the predecessor state's heuristic reward with the actual reward, which equals to 0 most of the time, and the evaluation system gives the new state a heuristic reward, $H(s')$.

In the search process, the terminal state will continue to be extended to become an intermediate state. This evaluation method not only depends on the current state but also the sequence of states that preceded it. However, if add a variable, the transition probability from the initial state to the state, into the state, then the evaluation system would not depend on the specific search history.

5 The Reachable States Checking

Through observation of the MAXQ-OP framework equation 8 and 2D simulation league, most of the transition probability, $P(s', N|s, a)$, between states is 0. So we only extend a small number of successor states in the decision-making process.

$$\begin{tikzcd}
\text{start} \arrow[r] & S_0 \arrow[d, "{a_1,P_1,C_1}"] \\
& S_1 \arrow[d, "{a_2,P_2,C_2}"] \\
& S_2 \arrow[d, "{a_3,P_3,C_3}"] \\
& S_3 \arrow[d, "{a_4,P_4,C_4}"] \\
& S_4
\end{tikzcd}$$

$$E_4 = P_1 * P_2 * P_3 * P_4 * R(S_4) * \gamma^{(C_1+C_2+C_3+C_4)}$$
$$E(S_0, a_1) = max(E_1, E_2, E_3, E_4)$$

Fig. 3. After extented state S_3, the evaluation system replace the heuristic reward of S_3 with 0 and give the new state S_4 a heuristic reward

And if we can greatly speed up the unreachable states detection speed, we can deepen the search depth.

$$C^{\pi}(i, s, a) = \sum_{s',N} \gamma^N P(s', N|s, a)V^{\pi}(i, s'), \tag{8}$$

According to the transition function as given in [4], we simplify the state reachable problem as from an initial position after certain cycles whether a player is able to reach a particular position. So we developed a teammates control area marking system to check whether the state is reachable to speed up our search process.

In order to ensure the checking speed, we gave up the pursuit of absolute accuracy:

- We simplify player motion model, in the simplified motion model and player state, consider only dash, do not consider turn or other actions.
- We simplified player state, consider only distance, do not consider body angle or other information.
- We divide the continuous soccer field into the discrete space.

In fact, if we compute the minimum number of cycles that players from an initial state to reach another state, the result to use simplified state and motion model is always the lower bound of the result to use full state and motion model. It means if a state is unreachable in simplified state and motion model it must be unreachable in the normal model. But the other way around is non truth, if a state is reachable in simplified state and motion model, it may be reachable or unreachable in the normal model, such cases require a more precise computation.

As in the figure, Figure 4, we divided the field into a number of small areas. Each area has a controller, and in the example, different colors represent different controller. The longer idled player has a bigger control area. If a area is not being controlled by any player, it is an unreachable state.

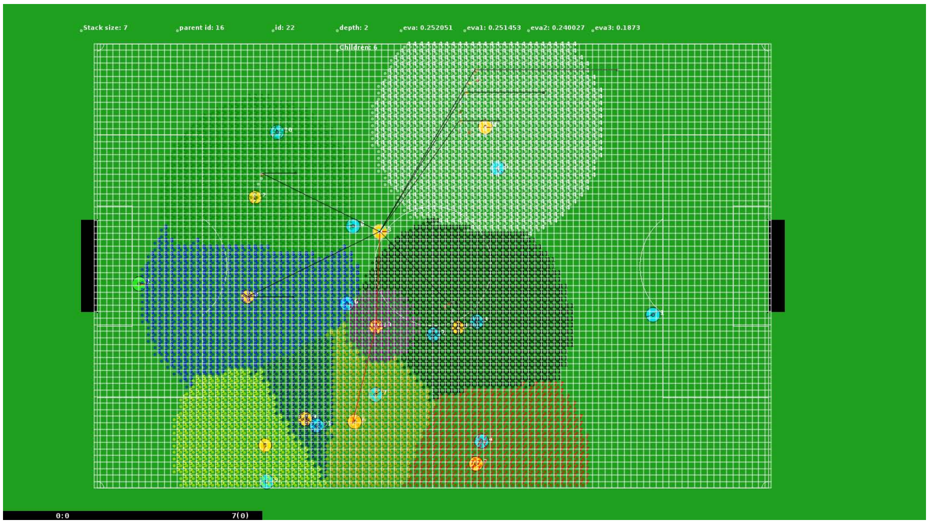

Fig. 4. Red lines mean the passes in history, black lines mean the considering passes. Different colors represent different teammates' control areas.

6 Experiments

In order to evaluate our framework, we use both time consumption and win rate to analyse the performance.

In the aspect of time consumption, worst case is most important, because time limitation is fixed. All we need to do is make sure the decision-making process in the worst case can be finished in time limitation.

To ensure the increase of time consumption is caused by the increase of search depth. We use tools[2], included in WrightEagleBase, DynamicDebug and TimeTest to finish this job. DynamicDebug allows us to reproduce the game scene and TimeTest allows us to analyse time consumption of specific function.

Table 1 shows the time consumption of different search depth with same DynamicDebug log files, which mean the same game and player role. Because of the search speed up methods, for example the reachable states checking, time

[2] Downloadable from `http://ai.ustc.edu.cn/en/robocup/2D/releases/WrightEagleBASE-4.0.0.tar.gz`

consumption increases linearly. However, in practice, after taking into account other factors such as network communication delay and other behaviors, the max acceptable search depth is 3. When search depth reaches 4, the players begin to appear to miss the decision cycle.

Table 1. Time consumption

Search Depth	1	2	3	4
Sum Consumption (ms)	77.7	106.4	150.7	192.7
Max Consumption (ms)	14.3	26.2	39.7	54.9
Min Consumption (ms)	0.1	0.1	0.2	0.2

For each version, we tested the team in full games with Helios2012, the champion of Robocup 2012. We independently ran each version against the binary code of Helios2012 officially released by Robocup 2012 for 150 games on the same hardware and software environment.

Table 2. Winning Rate

Search Depth	Winning Rate
1	58.0%
2	62.0%
3	68.7%
4	59.2%

Table 2 shows the winning rates of different search depth of 150 games. The result shows the team performance is increased when search depth increases. However, in the case of search depth 4, the impact of computation timeout caused the decrease of winning rate.

The search depth 1 performs a greedy strategy, the greedy strategy is not bad in most of time. Players continue to play forward, and as long as we are lucky enough, the ball will be eventually scored the goal. Of course, we cannot be always lucky enough. In this situation, the players often ignore long-term rewards. The player only holds the ball and waits for opponents to scramble.

Then, in the search depth of 2 and 3, this problem is more and more improved. The players will avoid some of the obvious short-sighted behavior. For example, if the direction of attack stuck on, and the whole team will make several passes, thereby changing the direction of the attack. Otherwise in greedy strategy, the player will hold the ball and the match will fall into deadlock.

7 Related Work

Some teams in the 2D league using search techniques make decisions. The Helios team [5,6] have developed a framework for online multiagent planning. The framework includes ActionGenerator and FieldEvaluator. The ActionGenerator generates candidate action instances for a node in the search tree, a node from the root node to leaf node is an action sequence.The FieldEvaluator evaluates the value of the action-state pair instances that are generated by ActionGenerator. The value of an action sequence is the sum of the each state variables. The WrightEagle team used an approach based on MAXQ-OP to automate planning. It combines the benefits of a general hierarchical structure based on MAXQ value function decomposition with the power of heuristic and approximate techniques. In this paper, we further develop this work. We created a new decision-making framework based on the MAXQ-OP framework with heuristic functions and other methods to accelerate the search process.

More teams in the 2D league[7,8] tried to improve performance with machine learning (ML) techniques. The most preferred ML methods is Reinforcement learning (RL), based on the general idea that an autonomously acting agent obtains its behavior through repeated interaction with its environment on a trial and error basis, by processing and integrating the experience the agent has gathered into a value function that tells how much it may be worth entering some state by taking some specific action. For example, team AT Humboldt [7], focusing on the task of learning to dribble, used a specialized version of Watkins Q-learning algorithm and integrates it into a framework where a proper representation of the state and action spaces was learned using an evolutionary approach. The opposite problem, i.e. the task of attacking and disturbing a dribbling opponent player in order to scotch the opponent teams attack at an early stage and to gain ball possession, has been targeted by the Brainstormers team. Using a partial model of the soccer environment and utilizing a temporal difference RL approach in conjunction with neural net-based value function approximation, they learnt a behavior for the mentioned task which significantly boosted team performance [8].

8 Conclusion

This paper introduces the champion of the RoboCup 2013 2D simulation league, WrightEagle. We describe some recent efforts to deal with complex and diverse situations and improve the quality of the solutions.

First, we developed a decision-making framework which transformed and extended the MAXQ-OP framework into Robocup simulation league 2D. The decision-making framework combined the heuristic evaluation function and best-first search algorithm. Second, we used the heuristic function to extend the definition of reward function of MAXQ-OP framework, and discussed the properties and features of this method. Third, we introduced a reachable state checking method to speed up the search process. And we stated if the checking result of a state is unreachable, then it promised to be unreachable. Otherwise, it may or may not be reachable.

Combined above methods, we successfully extended the search depth and got better search results in limited decision-making time.

Acknowledgments. The authors would like to thank the other members of WrightEagle, Aijun Bai, Miao Jiang, Haibo Dai, Guanghui Lu, Xiao Li, Rongya Chen, for their contributions to WrightEagle2D simulation team.

This work is supported by the National Hi-Tech Project of China under grant 2008AA01Z150 and the Natural Science Foundation of China under grants 60745002 and 61175057, as well as the USTC 985 Project.

References

1. Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming, vol. 414. Wiley. com (2009)
2. Dietterich, T.G.: The maxq method for hierarchical reinforcement learning. In: ICML, pp. 118–126. Citeseer (1998)
3. Bai, A., Wu, F., Chen, X.: Online planning for large mdps with maxq decomposition. In: Proceedings of the Autonomous Robots and Multirobot Systems Workshop, at AAMAS 2012 (June 2012)
4. Bai, A., Wu, F., Chen, X.: Towards a principled solution to simulated robot soccer. In: Chen, X., Stone, P., Sucar, L.E., van der Zant, T. (eds.) RoboCup 2012. LNCS (LNAI), vol. 7500, pp. 141–153. Springer, Heidelberg (2013)
5. Akiyama, H., Nakashima, T.: HELIOS2012: RoboCup 2012 Soccer Simulation 2D League Champion. In: Chen, X., Stone, P., Sucar, L.E., van der Zant, T. (eds.) RoboCup 2012. LNCS (LNAI), vol. 7500, pp. 13–19. Springer, Heidelberg (2013)
6. Akiyama, H., Nakashima, T., Yamashita, K.: HELIOS2013 Team Description Paper. In: RoboCup 2012: Robot Soccer World Cup XVI, pp. 1–6 (2012)
7. Burkhard, H.-D., Hannebauer, M., Wendler, J.: At humboldt development, practice and theory. In: Kitano, H. (ed.) RoboCup 1997. LNCS, vol. 1395, pp. 357–372. Springer, Heidelberg (1998)
8. Gabel, T., Riedmiller, M., Trost, F.: A case study on improving defense behavior in soccer simulation 2d: The neurohassle approach. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS (LNAI), vol. 5399, pp. 61–72. Springer, Heidelberg (2009)
9. Stone, P.: Layered learning in multiagent systems: A winning approach to robotic soccer. The MIT Press (2000)
10. Stone, P., Sutton, R.S., Kuhlmann, G.: Reinforcement learning for robocup soccer keepaway. Adaptive Behavior 13(3), 165–188 (2005)