WrightEagle 2D Soccer Simulation Team Description 2014

Haochong Zhang, Guanghui Lu, Rongya Chen, Xiao Li and Xiaoping Chen

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

Abstract. WrightEagle 2D soccer simulation team has been participating in annual RoboCup competitions since 1999 and won 4 champions and 5 runners-up at RoboCup world championships in the past 9 years. In this paper, we briefly present our current research efforts and some newly introduced techniques since the last year competition.

1 Introduction

WrightEagle 2D soccer simulation team, which was established in 1998 as the first branch of WrightEagle RoboCup team developed by Multi-agent Systems Lab. of USTC, has been participating in annual competitions of RoboCup since 1999. In recent years, we have won the champions of RoboCup 2013, 2011, 2009 and 2006, the runners-up of RoboCup 2012, 2010, 2008, 2007 and 2005.

We take RoboCup soccer simulation 2D as a typical problem of multi-agent systems, and we concentrate on multi-agent planning and other challenging problems from point of view of Robotics and artificial intelligence [1]. This year, we implemented a multi-step planning framework in WrightEagle2014, and proposed some defensive methods, based on our research efforts[2,3,4,5,6,7,8,9,10,11,12]. In this paper, we present a brief description of this progress mentioned above.

In Jan 2013, we released the latest version (4.0.0) of our team's base code WrightEagleBASE to the public as an open-source software which can be freely accessed from our team's website.[1] After this update, we gave WrightEagleBASE a new module called "Trainer". We hope that the released software will be helpful for exchange between teams and participants of this league and for anyone who wants to start to participate in the RoboCup event and/or research of multi-agent systems.

The reminder of this paper is organized as follows. Section 2 introduces the multi-step planning framework implemented in WrightEagle. Section 3 presents the method we used to improve defense behavior. Finally, the paper is concluded in Section 4.

[1] `http://www.wrighteagle.org/2d/`

2 Long-term Decision Making

2.1 MAXQ hierarchical decomposition

The WrightEagle team is developed based on the Markov Decision Processes (MDPs) framework [13] with the MAXQ hierarchical structure [14] and heuristic approximate online planning techniques introduced in the past years[12][11].

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 $\pi_i : S_i \to A_i$. 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 [14] 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.

2.2 Decision-making Framework

We introduce a decision-making framework, an extension of MAXQ-OP framework with multiple heuristic functions and a reachable state checking method. The experimental results show that our approach improves the quality of solutions in complex situations.

This section presents a decision-making framework that has been implemented in WrightEagle 2D simulation team for the macro action *attack*, which 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[11], 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 use macro actions, defined in MAXQ-OP framework[11], 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.

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.

3 Defensive Net

3.1 Defensive Net

In previous WrightEagle, every player is responsible for the defense of an opponent. Although this method is simple and efficient, there are some problems with it. To solve these problems, we have implemented a new method — Defensive Net. Defensive Net is a method to determine defense area. Under defensive net, every player is responsible for the defense of an area. The area is specified like this. Firstly, for each opponent, the weight distribution of its position is calculated. Secondly, every player choose several opponents to consider. The weight of those opponents are summed up and normalized. Thirdly, the area with greater weight is chosen to be the defense area.

For example, player 9 considers opponent 8, 9, 10 and 11. The area within each black circle is the corresponding defense area of every single opponent. And the area within the blue circle is the final defense area.

Our future work will focus on how to manage defenders, such as opponent team modeling in defense.

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
```

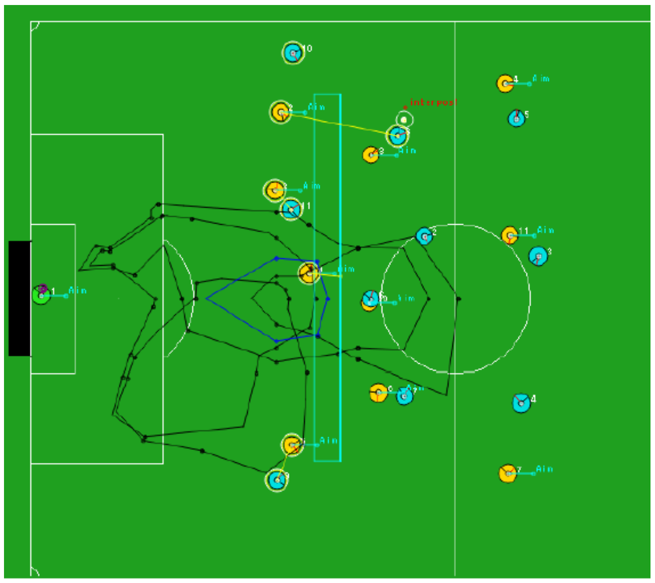

Fig. 1. Under defensive net, every player is responsible for the defense of an area.

3.2 Goalie's Positioning Mechanism in Defensive Environment

In previous WrightEagle, goalie's position is usually fixed to a certain area. As it's not suitable for defending opponent's side attack, we adopt a modified evaluation function model for goalie's position. The evaluation value is influenced by the state of both ball and key players. When the opponent ball owner agent is breakaway attacking, all opponent agents who will perform shoot action are chosen as key players. Each key player has different impact factor to goalie's positioning in the modified evaluation function.

We have accomplished a simple version of the improvement. In the future we will continue with how to improve it to fit for more scenarios.

4 Conclusions

This paper describes some of our current research efforts and newly introduced techniques since last competition. The multi-step planning framework allows agents to search for series of offence behaviors using two more macro-actions: dribble and shoot. We will continue working on this framework.

References

1. Chen, X.: Challenges in research on autonomous robots. Communications of China Computer Federation **3**(12) (2007)
2. Fan, C., Chen, X.: Bounded incremental real-time dynamic programming. In: Frontiers in the Convergence of Bioscience and Information Technologies, 2007. FBIT 2007, IEEE (2007) 637–644
3. Wu, F., Chen, X.: Solving large-scale and sparse-reward dec-pomdps with correlation-mdps. RoboCup 2007: Robot Soccer World Cup XI (2008) 208–219
4. Wu, F., Zilberstein, S., Chen, X.: Multi-agent online planning with communication. In: Proc. of the 19th Int. Conf. on Automated Planning and Scheduling. (2009) 321–328
5. Shi, K., Chen, X.: Action-driven markov decision process and the application in robocup. Journal of Chinese Computer Systems (2009)
6. Wu, F., Zilberstein, S., Chen, X.: Point-based policy generation for decentralized pomdps. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems (2010) 1307–1314
7. Zhang, Z., Chen, X.: Accelerating point-based pomdp algorithms via greedy strategies. Simulation, Modeling, and Programming for Autonomous Robots (2010) 545–556
8. Wu, F., Zilberstein, S., Chen, X.: Online planning for multi-agent systems with bounded communication. Artificial Intelligence **175**(2) (2011) 487–511
9. Bai, A., Wu, F., Chen, X.: Online planning for large mdps with maxq decomposition (extended abstract). In: Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems, Valencia, Spain (June 2012)
10. Wu, F., Zilberstein, S., Chen, X.: Online planning for ad hoc autonomous agent teams. In: Twenty-Second International Joint Conference on Artificial Intelligence. (2011)

11. Bai, A., Wu, F., Chen, X.: Towards a principled solution to simulated robot soccer. In Chen, X., Stone, P., Sucar, L.E., der Zant, T.V., eds.: RoboCup-2012: Robot Soccer World Cup XVI. Volume 7500 of Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin (2013)
12. 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-12). (Jun 2012)
13. Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming. Volume 414. Wiley. com (2009)
14. Dietterich, T.G.: The maxq method for hierarchical reinforcement learning. In: ICML. (1998) 118–126