Everest 2002 Team Description

Gu Yang, Liu Junfeng, Cui Lihui and Pan Feng

Department of Automatic Control,
Beijing Institute of Technology, 100081, P. R. China
{guyang_everest, liujf_everest, cuilihui_everest, panfeng_everest}@x263.net

Abstract. This paper presents an overview of the architecture and scientific approach of the *Everest 2002* simulation soccer team played in RoboCup environment. Our research is focused on how to apply fuzzy logic theory to multi-agent system, and now we have successfully realized basic skills like dribble, kick and high-level decision-making with fuzzy logic. Our setting goal is to take advantage of fuzzy logic in offline agent learning and combine that with online dynamic programming.

1 Introduction

Everest was initiated in June 2001 by 4 master students majored in Pattern Recognition and Intellectual System. Our original goal was to put what we learned in AI to special applications. In ChinaRobocup2001 held in August 2001, Kunming, China, *Everest 2001* won 4th place and "Best Performance Award" among the 12 teams coming from all over the country. Now we are devoting ourselves to the research under multi-agent system including multi-agent learning, multi-agent collaboration and opponent modeling. Our emphasis is put on the offline agent learning and online dynamic programming with the combination of fuzzy logic and other AI methods.

2 Fuzzy-Based Basic Skills

Basic skills like kick and dribble are critical to a team because they belong to the middle layer, which tightly connects the lower-level and higher-level layer together. We have explored in this field with the great help of fuzzy logic. Some approaches are as follows.

2.1 Fuzzy-Based Kick

Kick is one of the basic skills of soccer simulation. In most cases, agent cannot achieve the desired ball movement and desired angle with only one kick command. In a given situation, agent has to determine how many kick commands he needs and how to select several intermediate kick commands. The best intermediate kick sequence must help agent know exact angle and velocity, minimum times and avoid nearby enemies.

Taking only ball's relative position and velocity into consideration, we find that the problem can be simplified to how to adjust the ball within minimum time [1]. However, there are a huge number of possible intermediate kicks. With the start and terminal positions of the ball in a cycle, we can cipher out the kick command. So we introduce the concept of discrete position in the kickable area. In addition, the density of the points is high enough to guarantee the quality of the resulting kick plan.

Then fuzzy logic is used to evaluate all possible kick costs between two discrete positions. If ball's relative velocity vector is too great or relative position is too far, it may be out of the kickable margin of the agent in next cycle. So the current ball relative speed V and the current ball relative position R are used as inputs. Accordingly, we get the kick cost table, whose surface plot is shown as figure 1.

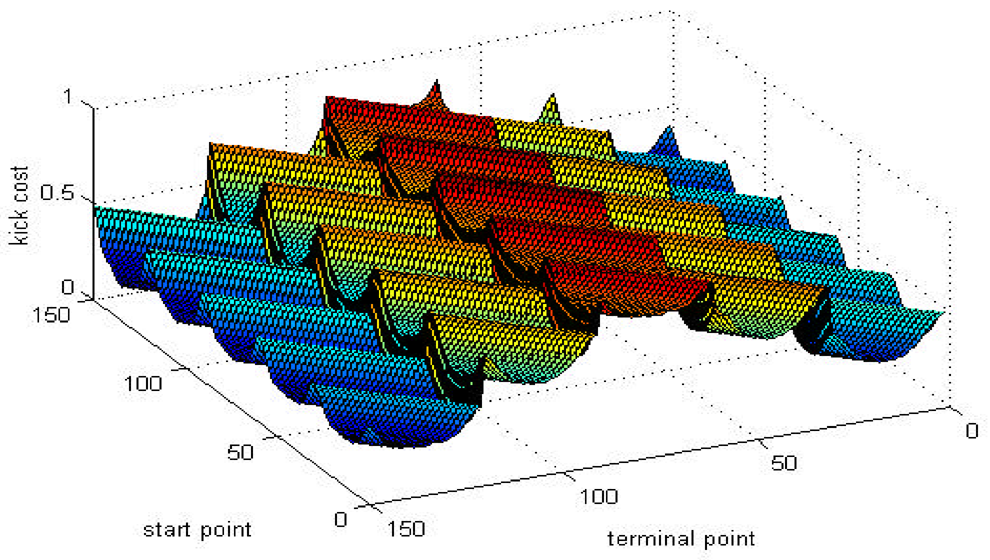
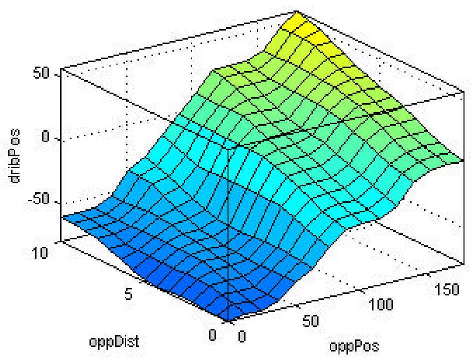


Figure 1. Kick cost evaluated by Fuzzy Logic

The table is finally taken by the agent as his knowledge. With its help, we use a searching algorithm (A*) to find a possible route to avoid the nearby opponents and reach the expected target velocity [2]. This approach is proved to be successful.

2.2 Fuzzy-Based Dribble

When implementing this skill in every cycle of simulation, 2 important output parameters are needed. One is the dribble angle, which is the relative angle between the ball and the dribbler; the other is the destination or target position of dribble action. Under such a condition, we simplify the decision-making of Dribble Target Position (DTP) to be based on the inputs of relative distance (R_d) and the relative angle (R_a) between the opponent and the agent and the output of DTP as well. Here according to the symmetry of direction, only half of the variables are taken into consideration. Figure 2 shows part of the rules we use and the surface of output variable DTP.



Rule:

If ***Rd*** **is** ***near*** **and** ***Ra*** **is** ***west***
then ***DTF = northeast,***
If ***Rd*** **is** ***normal*** **and** ***Ra*** **is** ***west***
then ***DTF = north,***
If ***Rd*** **is** ***far*** **and** ***Ra*** **is** ***west***
then ***DTF = northwest,***
…

Figure 2. Dynamic DTP evaluated by Fuzzy Logic

3 Fuzzy-Based High Level Decision-Making

When lower level possible actions are provided, agent has to choose the best one according to his role, play mode and game situation. Furthermore, he has to achieve multiple objects at the same time. This is a typical multi-object decision-making problem, which provides an important object set {O} and needs the decision-maker to choose an action from the action set {A}. As far as different players are concerned, they are expected to achieve a role-based object. For example, a defender may consider keeping formation as his most important task, and a forward may need to maximize the probability of goal. Here we epitomize the problem as the following one,

$A = \{shoot, dribble, pass, holdball\} = \{a_1, a_2, a_3, a_4\}$,
$O = \{VisualReq^1, LossBallProb^2, FmKeep^3, GoalProb^4\} = \{O_1, O_2, O_3, O_4\}$,
$P_i = \{b_{i1}, b_{i2}, b_{i3}, b_{i4}\}\ (i = 1, 2, \dots, 11)$.

We predefine O and P_i for every agent based on his role and game situation, and then this multi-object decision-making problem is solved with fuzzy logic. Detailed implementation may refer to [3].

4 Architecture

To achieve real time performance, we adopt a modular approach, which is shown in figure 3. In such a design, each action is implemented as a module and every module has a different priority.

Agent starts with initialization, such as reading configuration files and initializing some variables. And then agent invokes two main loops. One is used for information perception and the other is the action-generating loop. The information perception loop receives environment information and builds a world model. Action-generating

1 To satisfy as many Visual Requests as possible
2 To minimize Loss Ball Probability
3 To Keep the Formation as well as possible
4 To maximize Goal Probability

loop creates several schemes according to the world model and his role, makes decision after evaluating those schemes, and finally executes the best one of them.

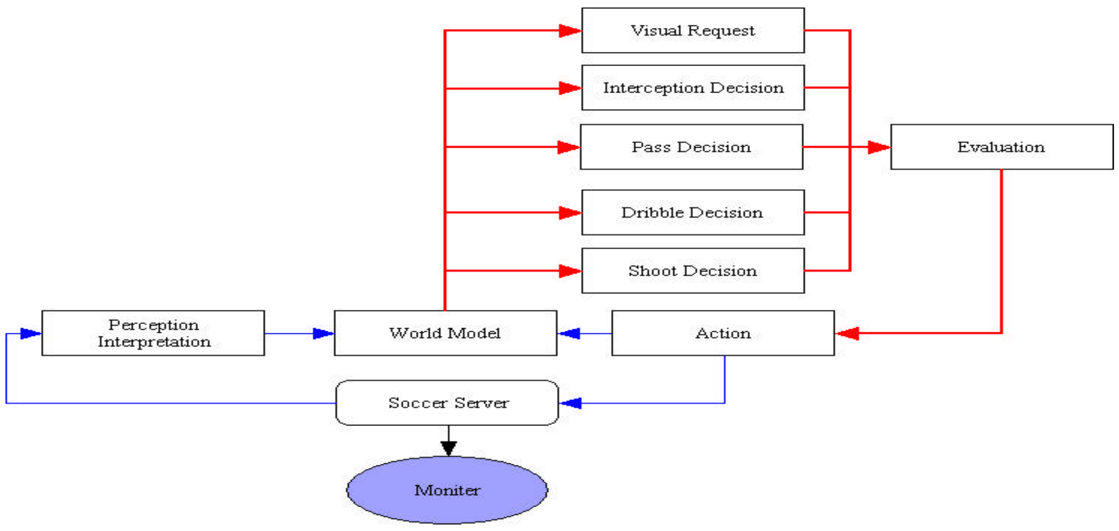


Figure 3. Agent Architecture

5 Conclusion and Future Work

With the development of our team, we become to pay more attention to high-level intelligent decision capabilities and cooperation. In our construction, Member Function selection has become more and more difficult, and now we are trying our best to make use of Neural Network as a tool. Our next goal is to apply fuzzy logic to offline agent learning and combine that with online dynamic programming. And till now we have done little work on online coach and heterogeneous player, which will be considered in the near future.

We believe that our agents can achieve good performance after combining successful methods and new techniques.

Acknowledgements

We would like to own our thanks to RoboCup Committee for providing such a good platform for multi-agent system. We would like to thank CMU, FCPortugal and TsinghuAeolus for their published source code, which saved a mount of time at the beginning of our exploration.

Reference

[1] Yao Jinyi, Chen Jiang. Architecture of TsinghuAeolus. TsinghuAeolus 2001 Team Description Paper. ftp://166.111.68.41/robocup/Docs/TsinghuAeolus2001.pdf

[2] Yao Jinyi, Chen Jiang. Q-Learning and Adversarial Planning with Application in RoboCup. ftp://166.111.68.41/robocup/Docs/TsinghuAeolus_kick.doc

[3] Timothy J. Ross. Fuzzy Logic with Engineering Applications. 1995, McGraw-Hill Companies, Inc.