BlueSpider
Team Description Paper for RoboCup 2011
2D Soccer Simulation League

Yadong Xie, Che Zhang, Zhen Wang, Duanfeng Gao
Shancheng Wang, Chunyang Wu, Dingyi Yang

RoboCup team, Engineering workshop,
Xi'an Jiaotong University, Republic of China
vthinkxie@gmail.com

Abstract. In this description document, we outline the main idea of the team and the strategy used in the RoboCup 2011 2D Soccer Simulation League competition. The major research issue is the cooperation of members in the team.

1 Introduction

BlueSpider 2D soccer simulation team, which was established in 2010 as a project developed by Engineering Workshop of XJTU, has participated in the competition of RoboCup in 2010. The team members are all undergraduate students, and we try our best to ensure the sustainable development of the team.

In the competition of this year we use the team's base source code of Agent 2D[1] instead of the WrightEagle Base, and focus on the problem of team's cooperation.

2 Focus

2.1 Fuzzy inference

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic, which has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. The mapping then provides a basis from which decisions can be made, or patterns discerned.

[1] Agent 2d can be download from http://en.sourceforge.jp/projects/rctools/

Fuzzy inference process comprises of five parts: fuzzification of the input variables, application of the fuzzy operator (AND or OR) in the antecedent, implication from the antecedent to the consequent, aggregation of the consequents across the rules, and defuzzification.

2.1.1 Fuzzification of the inputs and output

Fuzzification defines the mapping of input variables into man-defined truth degrees. In this case, the two inputs are security and offensive, and the output is the priority, which are defined based on the Sugeno method as below.

Security: {lower, low, normal, high, higher}

Offensive: {weaker, weak, normal, strong, stronger}

Priority: {lower, low, normal, high, higher}

In order to reduce the computing, we utilize the triangle as the membership function, as the following figure shows.

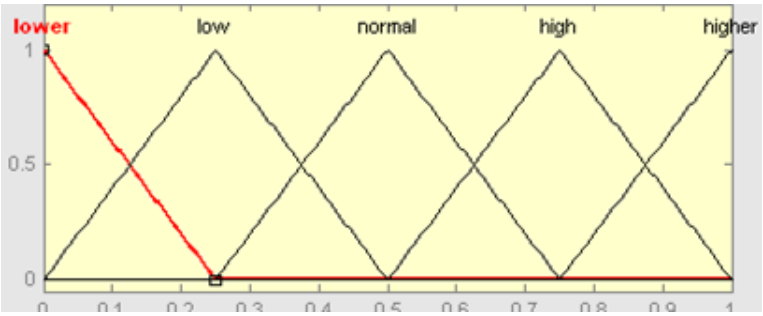


Table 1 Membership Function

Degree	Left Arm		Right Arm	
Lower			y=-4x+1	(0<=x<0.25)
Low	y=4x	(0<=x<0.25)	y=-4x+2	(0.25<=x<0.5)
Normal	y=4x-1	(0.25<=x<0.5)	y=-4x+3	(0.5<=x<0.75)
High	y=4x-2	(0.5<=x<0.75)	y=-4x+4	(0.75<=x<1)
Higher	y=4x-3	(0.75<=x<1)		

2.1.2 Rules

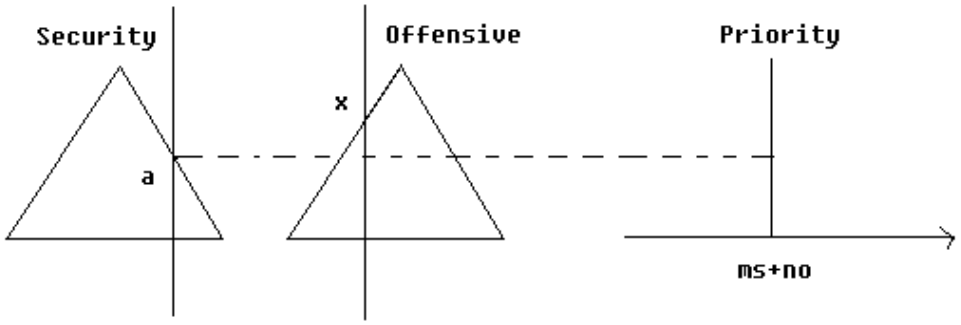
Rules define the mapping of fuzzified inputs to fuzzified outputs.

Table 2 Definition of Rules

Security(s)	Offensive(o)	Coefficient of rules((m,n))	Priority(s*m+o*n)
Lower	(from weaker to stronger)	(1,0)	s
Low	(from weaker to stronger)	(0.8,0.2)	0.8s+0.2o
Normal	(from weaker to stronger)	(0.5,0.5)	0.5s+0.5o
High	(from weaker to stronger)	(0.4,0.6)	0.4s+0.6o
Higher	(from weaker to stronger)	(0.3,0.7)	0.3s+0.7o

2.1.3 Computing the Priority

The computing process is shown as below.



And the Priority represented by the variable P can be concluded by the following equation.

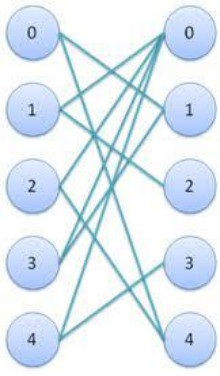
$$P = \frac{\sum_{i=0}^{3} Pi \cdot Pmi}{\sum_{i=0}^{3} Pmi}$$

2.2 KM algorithm

M is an algorithm handling Bipartite graph maximum weight matching.

A bipartite graph is a graph whose nodes can be divided into two groups so that every edge is between group A and group B. We call a set of edges a matching if and only if no edges share the same node. If every edge has a weight, maximum weight matching is a matching with the maximum sum of weight, Here a KM algorithm is to find such a matching.

A bipartite graph for example



KM algorithm assign every node a weight, here we note as A[] and B[]. And KM algorithm build its logic on this theorem:

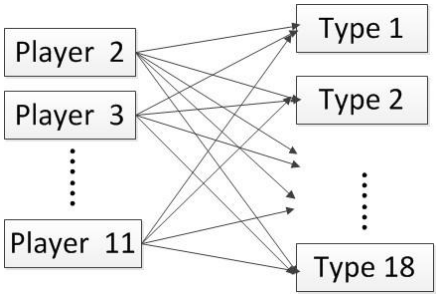
If a sub graph consisting of every edge satisfying A[i] + B[j] = w[i,j] has a perfect matching, then this matching is the maximum weight matching.

We build the graph like this:

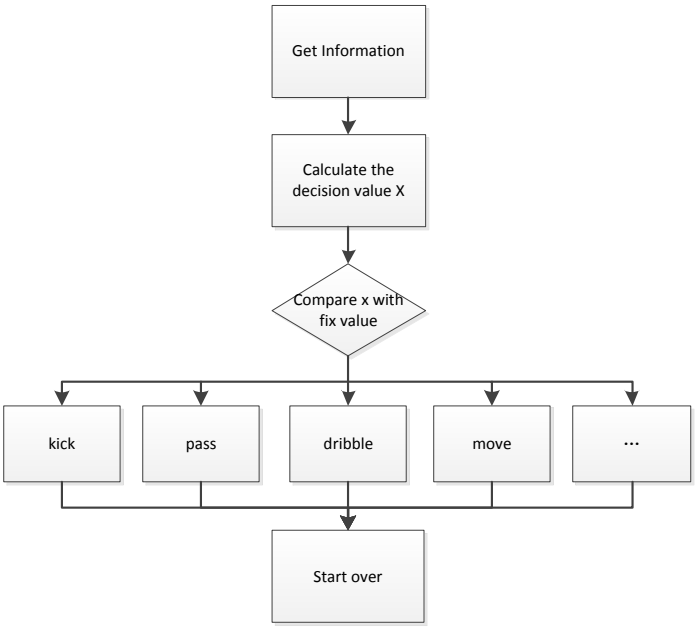
Group A stands for the player type obtained from the server, Group B stands for the real player on the field.

With fuzzy inference, we can get weight between any player and player type. So a matching is wanted to choose player type for players.(The goalie has a fixed player type, so we don't need to choose a player type for goalie.)

Note that the same player type can be chosen at most 3 times, so we can add a restriction that the Type node .



3 Framework



An World Model is introduced to get global information. Sometimes, static information can be obtained by calling Strategy and ServerParam. After getting enough information, decision can be made by choosing the following basic actions in logic : Kick(Shoot), Pass, Dribble, Move and so on. Note these actions are sealed on the basis of basic actions in server.

We got fixed value(called “weight”) by training our team with other teams. And every action has a calculated value standing for its “weight”. The algorithm, fuzzy inference, has been introduced above.

Of course, after this action, a method must be called to judge whether this action has made positive effects. The result is stored in order to make better decisions later.

4 Future Work

At the very start, we try to use the Action Chain Search Framework and Delaunay Triangulation related in the team description paper of Helios 2010, but we gave up in the end because the team based on them could not work very well.

We will try our best to make use of Action Chain Search Framework and Delaunay Triangulation in the future.