NuBot 2005 Team Description

Wenjie Shu[†], Fangyi sun and Zhiqiang Zheng[‡]

College of Electro-Mechanic and Automation
National University of Defense Technology
Changsha,Hunan, 410073 P.R.China
Email:[†]wjshu_0407@hotmail.com and [‡]xyzheng@sohu.com

Abstract. The Robosoccer simulation environment is an excellent simulation of the soccer domain, in which software agent act autonomously, with limited perception, action and communication abilities. In this paper, we describe the main features of our soccer simulation team — NuBot, which include the defense strategy and the Heteroplayer strategy of our team. The simulation match results show that our improvement of defense strategy is obvious. And also the methods proposed to solve the heteroplayer strategy achieve superiors performance.

1 Introduction

The Robosoccer simulation environment is an excellent simulation of the soccer domain, in which software agents act autonomously, having very limited perception, limited action abilities and a single unreliable channel with low bandwidth for communication [3]. A set of eleven agents must work together trying to optimize their defensive and offensive strategy to the best of their abilities in every respect.

In this paper, we propose the hybrid learning method that is based on the combination of the static learning and dynamic learning methods and applies to the defense strategy. The static learning is the most method that used in the Robosoccer simulation and work well in some scenes. But it can not satisfy the dynamic requirement of the Robosoccer simulation environment. The samples of training are not so self-contained and the training exists partial minimum value, especially some samples are not training. The dynamic learning can offsets the shortage of the static learning.

Heteroplayer selection strategy is another challenge of Robosoccer simulation. In this paper, we first transfer this problem to an assignment problem and propose two methods to solve it.

The remainder of this paper is organized as follows. In section 2, the global planning for defense based on the hybrid learning is presented. The Heteroplayer strategy problem is described in section 3. In section 4, we proposed two methods to solve the heteroplayer selection problem. The simulation match results is given in section 5. Finally, we will give some conclusions and describe future research directions in section 6.

2 Hybrid Learning Method and its application in the planing of Defense

The hybrid learning combine the static learning and dynamic learning. The former is mostly used in the simulated Robosoccer terms and can make most use of the expert knowledge. However, the static learning which is based on the Self-Organizing feature mapping often lead to local diminutive value, as the world model is not so exactly. The latter can offsets the shortage of the static learning.

In this section, we will apply the hybrid learning method to the planning of Defense. Defense plays an indispensable role in real soccer game and in Robosoccer simulation. The defense problem in Robosoccer simulation game is obviously a process of distributed planning and dynamic coordination.

2.1 The defensive approach based on the static learning

The static learning based on learning contains the basic formation, the design of defense action and the assignment of arrangements. The basic formation determines a player's position by standard role, ball position and opponent position. The opponent basic formation contains the possible route pass.

In our simulated Robosoccer teams Nubot 2005, four types of defense actions are defined as in [2]:

— **1. Block:** intersecting an opponent possessing the ball in the outer side of our goal, preventing him from pushing forward.

— **2. Mark:** keeping up with an opponent without ball so that his teammates cannot pass the ball to him.

— **3. Point Defend:** staying at the basic formation position, this will benefit when a nearby teammate fail in 1 vs. 1 defense or when the team regains control of the ball.

— **4. Interception:** when the ball is free or not controlled by the opponent.

To simplify the problem, we set the rule that one defender cannot defend against two opponent at the same time, but permit two defenders to defend against the same opponent, which is difference with the muter operator in [2].

We defined one evaluation function for each type of function. Three variables are used as the input of each evaluation function. The three variables are related to the player's current position and basic formation position, the ball's position, the opponents' position.

We simulate the thoughtway of human football and get the sample of training from the expert's experience. Self-Organizing feature mapping has favorable characteristic of self-Organizing, which is close to the human thoughtway. It has strong functional capacity. Here we choose the Self-Organizing Feature Mapping for encoding evaluation functions. To attain the actual value of each function, we set some typical scenes, extract input values from them, and endow output values to the functions. After training, networks are tested to ensure that they fit requirements of our defense strategy of our team, thus evaluation functions are defined. The rest of the planning goes as the TinghuaAelous standard procedure [2] and no further discussion is needed.

2.2 The defensive approach based on the dynamic learning

Although the static learning work well in most scenes, it can not satisfy the dynamic requirement of the Robosoccer simulation environment. The samples of training are not so self-contained and the training of Self-Organizing feature mapping exist partial minimum value, especially some samples are not training.

The dynamic learning offsets the shortage of the static learning learning by noting the opponent's characteristic and the opponent's strategy. When the opponent was failed in aggression, we define that the defensive strategy was successive. So we will use the referenced data as the next defensive bout. We also define three basic offensive characteristic of the opponent:

— **1. Pass:** The team is accomplished in pass, such as the Tinghuaelous team.

— **2. Cincture:** The team is accomplished in taking the ball, such as the STEP team.

— **3. Pass And Cincture:** The team is accomplished in pass and taking the ball.

The next question is how to differentiate the basic offensive characteristic of the opponent. We define one evaluation function for each type of function. Two variables are used as the input of each evaluation function, Control-ball-time, which describe the time of the ball is controlled by the opponent, and the ball's transformation which measures the ball's angle transformation per cycle relative to the goal. The function output increases with the control-ball-time and decreases with the ball's transformation. To attain the actual value of each function, we collect lots of samples which are related to the two input variables from the match vs the STEP team and TinghuAeolus team. After training, networks are tested to ensure that they fit requirements mentioned in the previous section, thus evaluation functions are defined. We also define three basic defensive characteristic of ourselves which are related to the player's basic point:

— **1. Active:** The defensive strategy is active, which puts up the player is inclined to take the action such as the block and mark.

— **2. Passive:** The defensive strategy is passive, which puts up the player is inclined to take the action such as the point defend.

— **3. Eclectic:** The defensive strategy is not so active and passive, which puts up the player is inclined to take the action such as the block and point defended at right time.

We will transfer different database which decide three different defensive characteristic of ourselves to determine our defensive strategy according to different opponent characteristic, which is determined by the function output.

The dynamic learning can take different action according to different opponent characteristic, though this learning mode is limited.

3 Problem Description of Heteroplayer Strategy

Before the problem description, we shall first define a set of index function which reflect the requests for the heteroplayer of our Robosoccer simulation team.

Heteroplayer has different characteristics which are based on certain trade-offs with respect to the player parameter values. The trade-offs of current server are shown in Table 1 [1].

Table 1. Trade-offs between players parameters for heteroplayer

Improvement	Weakness	Trade-off description
player_speed_max	stamina_inc_max	Player is faster, but his stamina recovery is slower.
player_decay	interia_moment	Player's speed decay slower, but he can turn less.
dash_power_rate	player_size	Player can accelerate faster, but he is smaller
kickable_margin	kick_rand	Kickable distance is larger, but noise is added to kick
extra_stamina	effort_min_effort_max	Player has more stamina, but dash is less efficient

Different teams don't always have the same requests for each player. Usually, we hope our players have high speed, acceleration, stamina, and also can turn and intercept the ball quickly. In this paper, we set up a set of function of these indexes called index function which uses the heteroplayer parameters values provided by the server.

Speed_Index f_1 Speed_Index reflects the high speed that the player can be reached, and it is scaled by the basic parameter player_speed_max, i.e.,

$$f_1 = player\_speed\_max \tag{1}$$

Acceleration_Index f_2 We hope our player can reach his max_speed as quickly as possible. The acceleration process can be described as Fig. 1. Acceleration is another important index for the players. For simplification, we assume that a player can reach his max_speed in two cycles and choose the average speed during first two cycles as the index.

Suppose the parameter effort of the stamina model in [1] is constant and equals to effort_max. The initial speed is $v = 0$.

According to the Movement Model, the speed of the first cycle is $v_1 = a$, where $a = power * dash\_power\_rate * effort\_max$. Eventually speed after two cycles is

$$v_2 = a * player\_decay + a \tag{2}$$

Acceleration can be formulated as the average speed during first two cycles:

$$f_2 = \bar{v} = \frac{v_1 + v_2}{2} = \frac{a * (2 + player\_decay)}{2} \tag{3}$$

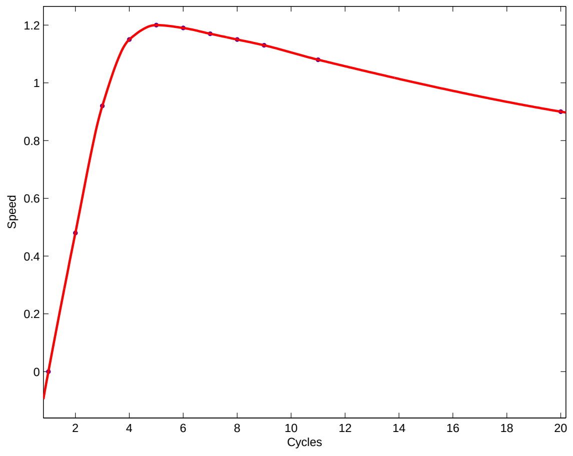

Fig. 1. Speed Vs. Cycles

Turn_Index f_3 For some players, we expect they turn neatly on the field which is described by the turn_index. Here we let it equal to the actual-angle that a player turns after carrying out command turn. It can be calculated as

$$f_3 = moment/(1.0 + interia_moment * player - speed) \tag{4}$$

where moment is a argument of command turn, whose valid value is between *minmoment* and *maxmoment* [1].

If a player is quiescent, his actual angle will equal the value of moment. But when he is moving, it will be more difficult for him to turn because of inertia and the actual angle will be less than moment.

Stamina_Index f_4 The stamina_index function can be described as the cycles that the player keep running at the max_speed ($v = 1.0$) of the default type player before his stamina is lower than the terminate value of the stamina. Suppose the player's initial stamina is S_0 and the terminate value of the stamina is S_t, which is higher than recover decrement threshold (it can be expressed as $S_t \geq recover\_dec\_thr * stmani\_max$). At each cycle, player must use dash command to compensate for the reduction of his speed. According the the movement model and dash model in [1], we can get

$$v * (1 - player\_decay) = power * effort\_max * dash\_power\_rate \tag{5}$$

then

$$power = \frac{v * (1 - player\_decay)}{effort\_max * dash\_power\_rate}. \tag{6}$$

Furthermore, a player's stamina gets restored by stamina_inc_max in each cycle. So a player's actual decrement of stamina in each cycle is

$$\triangle S = power - stamina\_inc\_max, \tag{7}$$

then f_4 can expressed as

$$f_4 = \lfloor \frac{S_t - S_0}{\triangle S} \rfloor. \tag{8}$$

Intercept_Index *f_5* Intercept_Index reflects the player's ability of interception, and it is scaled by the basic parameter kickable_margin.

$$f_5 = kickable\_margin \tag{9}$$

In order to make these indexes function comparable, we should first standardize them. Suppose a particular function f_i alters in the set $[f_{min}, f_{max}]$, it can be standardized as

$$f_i^{'} = \frac{f_i - f_{min}}{f_{max} - f_i}. \tag{10}$$

Now we suppose all these index functions are standardized, then they are comparable.

3.1 Problem Description

In fact, the problem of heteroplayer strategy can be described as an assignment problem: there are altogether twenty-eight players (including three of each type of heteroplayers and ten default ones: $3 * 6 + 10 = 28$). We need choose ten players from all the 28 players to build up our team (excluding the goalie). Let the player type chosen by the role j among all the 28 players is $k(m)$. We set $x_{mj} = 1$ if role j choose the mth player from the 28 players, otherwise $x_{mj} = 0$. we first set the evaluation function of the role j selecting player type $k(m)$ as

$$F_{jk(m)} = \sum_m \sum_i x_{mj} c_{ij} f_{ik(m)} \quad \text{for i = 1, \ldots, 5} \quad \text{m = 1, \ldots, 28} \tag{11}$$

where $f_{ik(m)}$ is the ith index function of heteroplayer type $k(m)$ and c_{ij} is the weight of the index function f_i to role j.

Then the object function of the team's Heteroplayer strategy can be described as following

$$\max G = \sum_j h_j F_j \quad \text{for j = 1, \ldots, 10} \tag{12}$$

s.t.

$$\sum_j x_{mj} = 1, \quad \text{j = 1, 2, \ldots, 10}$$

$$\sum_m x_{mj} = 1, \quad \text{m = 1, 2, \ldots, 28}$$

where h_j is the weight of role j in the whole team.

This is a special kind of 0-1 programming. And the value of c_{ij} and h_j can be get by the Analytical Hierarchy Process (AHP) in the following section.

3.2 Analytical Hierarchy Process (AHP)

Aforesaid five indexes are not of the same importance to role j, i.e., each index function f_i has different weight c_{ij} to role j in the estimation function. Here we use the AHP method to decide the value of the c_{ij}, which can be expressed as following:

Step 1: w_{ij} is assigned to each pair of (f_i, f_j). Comparing the indexes pairwise and express your judgements. By choosing one of the following nine expressions: equal importance, moderate dominance, strong dominance, very strong dominance, extreme dominance, or an intermediate judgement between two consecutive ones of those four, each answer is then automatically converted into a number w_{ij} according to Table 2.

Table 2. Converting "judgements" to "numbers"

Expressions	Associated Numbers
equal	1
equal to moderate	2
moderate	3
moderate to strong	4
strong	5
strong to very strong	6
very strong	7
very strong to extreme	8
extreme	9

Step 2: After the questioning process, a positive reciprocal matrix (one line and one column for each element $f_1, f_2, \ldots, f_5$) is filled in placing at the intersection of the line of f_i with the column of f_j the number.

$$\begin{cases} w_{ij} & \text{if } f_i \text{ dominates } f_j \\ 1/w_{ij} & \text{if } f_j \text{ dominates } f_i \\ 1 & \text{if } f_i \text{ does not dominate } f_j \text{ and also } f_j \text{ does not dominate } f_i \end{cases} \tag{13}$$

$$W = \begin{pmatrix} 1 & w_{12} & \ldots & w_{15} \\ 1/w_{12} & 1 & \ldots & w_{25} \\ \vdots & \vdots & \ddots & \vdots \\ 1/w_{15} & 1/w_{25} & \ldots & 1 \end{pmatrix} \tag{14}$$

Step 3: calculates the maximal eigenvalue of matrix W and determines the respective normalized eigenvector: the components of this vector are the numerical scale $w : (f_1, f_2, \ldots, f_5) \rightarrow \Re$ obtained is a ratio scale.

Step 4: Consistency test prevents the acceptance of the priorities if the inconsistency level is high. A group values of w_{ij} for a center forward and a group values of c_{ij} for a team which values offence under 4-3-3 formation are in Table 3 and Table 4.

Table 3. Values of w_{ij} for a center forward

	Speed	Acceleration	Stamina	Turn	Intercept
Speed	1	0.5	2	6	6
Acceleration	2	1	3	7	7
Stamina	0.5	0.33	1	4	4
Turn	0.167	0.143	0.25	1	1
Intercept	0.167	0.143	0.25	1	1

Table 4. Value of c_{i*}

roles	c_{1*}	c_{2*}	c_{3*}	c_{4*}	c_{5*}
MidForward	0.29103	0.43805	0.047261	0.17639	0.047261
SideForward	0.30301	0.47394	0.036004	0.15105	0.036004
MidBack	0.21405	0.34472	0.037547	0.344472	0.058969
MidField	0.1873	0.28713	0.055918	0.43387	0.035781
BackSide	0.21133	0.34129	0.062811	0.341290	0.043293
MidSide	0.21282	0.3409	0.064995	0.3409	0.040387

h_j can be obtained by the same way.

4 Heteroplayer Selection Strategy

In a team, the importance of every player in different position are not same. For instance, a more aggressive team hope to select an excellent forward while a less aggressive one may pay more attention to defenders. Then how could we deal with the difference of players' significance. One method is the Hungary method, which deals with the difference of players' significance by giving different weight in an estimation function. Another method is to choose the players one by one, that is to choose the most significant player first, then consider the less important ones. In that case, we formed the Step-by-Step algorithm .

4.1 The Hungarian Method

Here we use Hungarian method to solve the 0-1 programming described in the section 3.1. Hungarian method make full use of particularity of an assignment problem and reduce the calculate quantity effectively, which can be described as following. We first define the square benefit matrix B.

$$B = \begin{pmatrix} F_{1k(1)} & F_{1k(1)} & \dots & F_{1k(28)} \\ F_{2k(1)} & F_{2k(1)} & \dots & F_{2k(28)} \\ \vdots & \vdots & \ddots & \vdots \\ F_{10k(1)} & F_{10k(1)} & \dots & F_{10k(28)} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \tag{15}$$

Step 1: Subtract the minimum number in each row of benefit matrix from the entire row.

Step 2: Subtract the minimum number in each column of benefit matrix from the entire column.

Step 3: Cover all zeroes in the benefit matrix with as few lines (horizontal and/or vertical only) as possible. If the number of lines equals the size of the benefit matrix, find the solution. If you have covered all of the zeroes with fewer lines than the size of benefit matrix, find the minimum number that is uncovered. Subtract it from all uncovered values and add it to any values(s) at the intersections of your lines.

Step 4: Repeat **Step 3** until there is a solution.

The Hungarian method can achieve the global optimization, but also it is much complicated.

4.2 The Step-by-Step Algorithm

In order to overcome the shortcomings of the Hungary algorithm, we proposed another algorithm called Step-by-Step algorithm, which in nature is a greedy algorithm. We choose the heteroplayer one by one according to our team's characteristics. At each step, we always require to achieve the maximum of $G = \sum_{j=1}^{10} k_j F_j$, which can be done by means of maximizing $F_j = \sum_{i=1}^{5} c_i f_i$.

Step-by-Step algorithm is a local optimal algorithm. Comparing to the Hungarian method, this method is much more simple, and can satisfy different strategy of different team.

5 Simulation Results

To evaluate our heteroplayer selection strategy, we apply it to 100 matches between our Robosoccer simulation team NuBot 2005 and other teams, such as STEP, UVA, Tsinghua and Appol. NuBot uses 4-3-3 formation and pays more attention to the aggression. Under this circumstance, they have rigorous requests for their forward. The results are summarized in Table 5.

Table 5. Comparing Result of two algorithms

	Hungarian method	Step-by-Step algorithm	Default
$\bar{G}$	0.5942	0.5762	0.2014
$\bar{F}_1$	0.5087	0.5289	0.2160
$\bar{F}_2$	0.6123	0.5155	0.1923
$\bar{F}_3$	0.5087	0.5294	0.2160
$\bar{F}_4$	0.6123	0.5156	0.1923
$\bar{F}_5$	0.5681	0.4864	0.2101
$\bar{F}_6$	0.5415	0.5315	0.2160
$\bar{F}_7$	0.5324	0.5331	0.2160
$\bar{F}_8$	0.6207	0.6777	0.1987
$\bar{F}_9$	0.6569	0.6826	0.1930

The simulation match results show that in the whole the Hungarian method performs better than the Step-by-Step algorithm, and the whole team also achieve the balance in the section of heteroplayer. However, the Step-by-Step algorithm is much more simple and can select the heteroplayer according to the team's characteristic. And also the improvement of our defense strategy is obvious. The loss goals of every match reduces 2-3 against the RoboCup04 champion team STEP.

6 Conclusion and Future Work

In this paper we have improved our team's defense strategy and the simulation match results show this improvement is obvious.

We also proposed two methods to solve the heteroplayer selection strategy. The performance of them is superiors.

In our future work, we will the defense strategy and heteroplayer selection strategy. Also will pay more attention to propose new methods in solving the high-level strategy of the whole team.

Acknowledgment

We first thank to the members of TinghuAelous team very much, for the Nubot 2005 soccer simulation team was built on the TinghuAeolus2002's source code. The released source code save us much time, such that we can pay more attention on the high-level strategy.

References

1. Itsuki Noda et al, Soccer Server Manual, RoboCupFederation. http://www.robocup.org.
2. Yunpeng Cai, Jiang Chen, Jinyi Yao, Shi Li: Global Planning from Local Eyeshot: An Implementation of Observation-Based Plan Coordination in RoboCup Simulation Games. RoboCup 2001: 12-21.
3. Luis Paulo Reis, Jose Nuno Lau, Luis Seabra Lopes : F.C.Portugal Team Description Paper. RoboCup 2001.