Lion-griffin 2010 2D Soccer Simulation

Technical Description Paper

S.Karimzadeh[1]
E.Iravani[2]
M.A.Sadeghi[3]
R.Movahed[4]

Islamic Azad University, Tehran North Branch

[1]samira.karimzadeh@gmail.com
[2]elhamiravani.89@gmail.com
[3]sadeghi.mahdi@gmail.com
[4]reyhane_movahed@yahoo.com

http://www.skz.moonfruit.com

Abstract. Lion-griffins Team has chosen ***Agent2D*** base as the base to work on. This is how we could solve some important problem of the Agent2D base and added more abilities to it by using a simple solution. The most important aim for the team is to create the ability of learning. This method uses some famous algorithm with a little change and some new algorithm that design by our. Using this method had a great influence in the game mode and has lead to great results for the team against opponents who had used better and stronger bases in local and worldwide competitions.

Key words. Reachable graphs, Statistical learning, Ants' colony, Fuzzy logic, Learning ability, Reinforcement learning, Q learning;

1 Introduction

The members of Lion-griffins 2D Soccer Simulation team have started to work under supervision the Tehran Education Organization and the Sharif University of Technology since 2004. These members participated in Incredible and SAMIN teams, that Incredible 2D Soccer Simulation team have achieved the eighth place and Incredible Rescue Simulation team have achieved the seventh place in Robocup 2006 Bremen, German and SAMIN team have achieved the seventh place in Iran Open 2008, the seventh place in Iran Open 2009, the sixteenth place in China Open 2007 and the sixteenth place in China Open 2008. All of these members join to Lion-griffins team in 2009 and All Lion-griffins' rights were left to the Islamic Azad University, Tehran north branch.

This article focuses on explaining the strategies and technical parts of the team, while trying to present solutions for the basic problems of soccer simulation bases, such as the shared soccer agents system of decision making and the disability of decision making based on the strategies of the opponent's team.

2 Agents' abilities

2.1 Pass

Passing algorithms used in the source:

Basically, we use 7 functions for the passing operation, which act in different situations for better performance.

The 'checkSituation' function is the first one to be called. It uses some simple analyzing algorithms to check the position of the agent in the field and the situation of other teammates and opponents and the planned strategies for the game in the present cycle.

First of all it checks for the presence of the teammates and opponents in our field and the strategies planned by other parts of the source. The function returns a situation value which is processed by some algorithms based on the positions of the teammates and the opponents around the agent. The returned value is a basic number for other functions to act and plays and important role in the decision making process of the agent and the accuracy of the passes.

The second parameter checked by this function is the situation of the game in our field. So if we are under attack and most of the players are in our field, but we have got a chance to gain control over the game, the 'offendResponse' function is called. It makes decision on how to pass to a teammate so it can moves to the opponent's target based on the condition's strategy which is planned by other parts of the source. It also checks for the offside line of the opponent's players in case it can pass to the desired

teammate with low offside foul possibility.

The third situation to check is the presence of the players in the opponent's field which means we have controlled the ball and moved it to a goal chance condition. The 'passToBehindDefenders' function is called in which we have to figure out the formation of the opponent's defenders and the offside line created by their positions so we can pass to a teammate which is close enough to the opponent's target and is possible to pass to, too. It also checks for the strategy of the game and uses some graphs to the opponent's target to move to ball toward it.

But most of the time the conditions do not meet the situations above and the ball is positioned in the middle of the field. We have implemented 4 other functions for this situation.

The 'advancePass' is called when the returned value by the first function is a set to more opponents than teammates. The basic way of analyzing the area by the function is to create some sectors around the agent and make routes through the opponents and find pass points with high accuracy and best teammates close to them to pass to through the routes.

The 'moderatePass' is another function designed for the massive decision making process. It uses sectors again but the difference is in number and degrees of the sectors. They are set to check for the presence of any opponent in front of the teammates and also between the agent and the teammate. The best teammate to pass to is the one which has the ability to move forward and pass to another teammate together. The accuracy of this pass method is checked by the presence of any opponent's player on the route.

The 'simplePass' function is the function with a summarized algorithm in comparison with the 'moderatePass' function. It just checks whether it can pass to a teammate with high accuracy and the teammate is able to pass to another one, too. The process is by creating sectors and making routes between a few teammates.

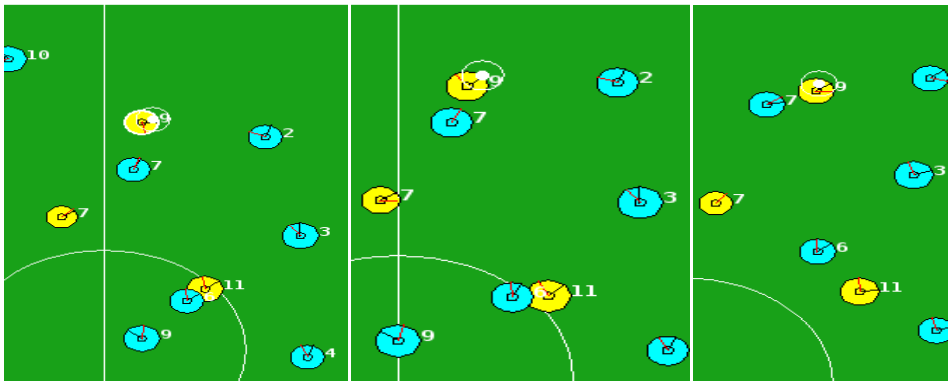
The 'nearPass' function is the final one which is called when none of the functions above were able to pass to a teammate with high accuracy. This function passes to the nearest teammate without checking any kind of opponent's players presence or the ability of the teammate to move forward or pass to another teammate. The point of this function is the speed of its execution which helps the team to move toward the opponent's half in a short time. But it does not have much accuracy and may lose the ball if done not properly.

2.2 Dribble

We utilize tree kind of dribbles depending on players' situations:

2.2.1 Simple dribble

Agents probe in some surfaces depending on players' positions and their situation via opponents. Then they select a suitable angle for dribble in safe surface if any safe surfaces exist. Fig.1 shows these actions.

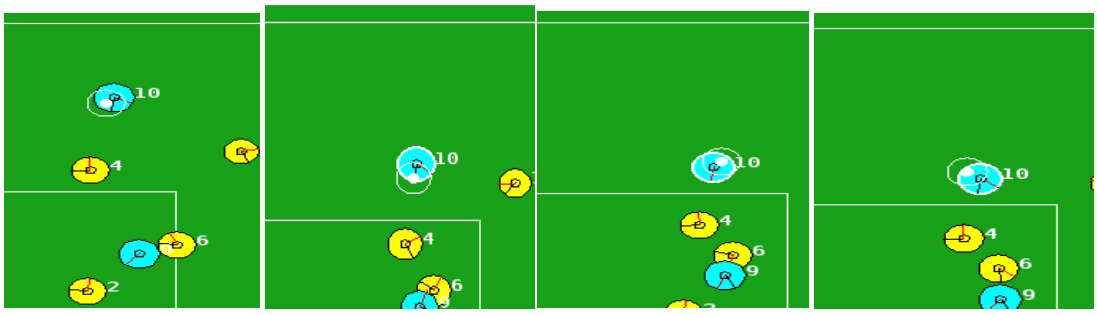


(a) (b) (c)

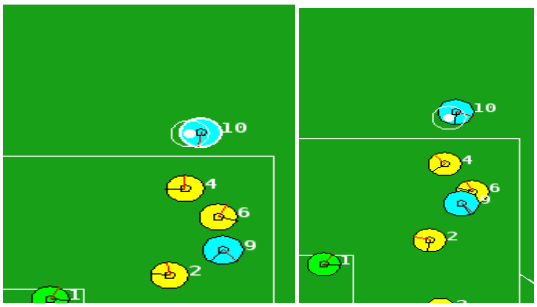
Fig.1. simple dribble

2.2.2 Move and turn with ball

If agents don't succeed to continuing the path with before algorithm, they should use another algorithm. If the nearest opponent will be in certain limited length and there are two of them at least, the agent has to keep the ball from opponents by the way. Fig.2 shows these steps.



(a) (b) (c) (d)

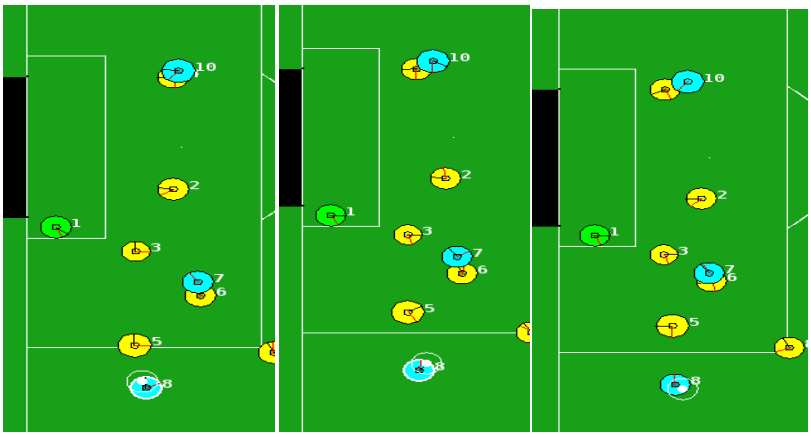


(e) (f)

Fig.2. Move and turn with ball

2.2.3 Turn with ball

If the agents don't find any way to continue, they should use another and last algorithm. If they see an opponent too near themselves they turn with ball. Fig.3 shows this action.

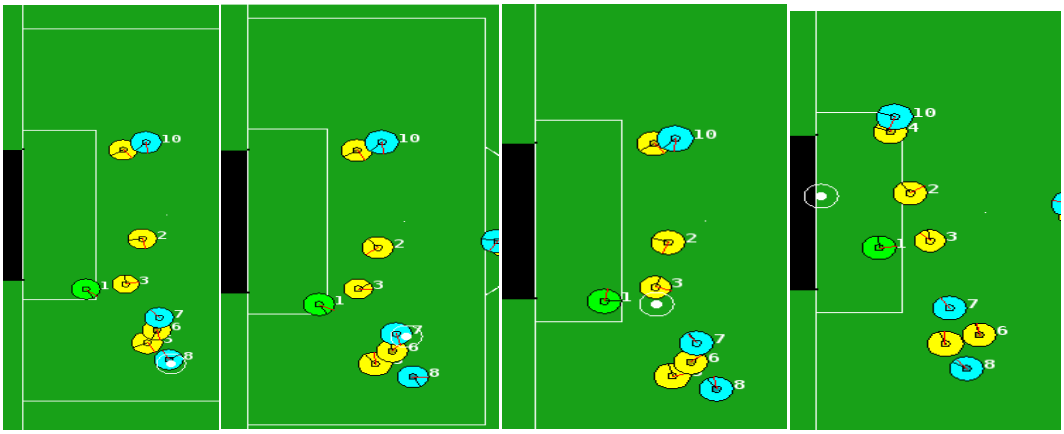


(a) (b) (c)

Fig.3. Turn with ball

2.3 Shoot

The agents can choose a method for shoot. The first one that is good situation is: They check with some lines (paths). If they don't find any position, they act with another method. The second is: The agent looks for good teammate that it has a good situation, like first method, then it pass ball to it. Fig.4 shows this methods.



(a) (b) (c) (d)

Fig.4. shoot

3 Learning Ability

3.1 Reinforcement Learning

The aim of these algorithms is to find the best performance in different environmental situations. Different performances are considered for different situations and the agent that fits the environmental situation best selects an action and performs it. As a result, it gains reward from the environment. In other words, these algorithms try to find the best performance which results in gaining the maximum amount of reward. [2][3]

The reasons causing the variety of reinforcement learning algorithms are as follows:

- Reward function and the next environment forms, exists or they are announced by the coach each time.(online or offline learning mode)
- Specified or non-specified reward function
- Existence or not-existence indirect and delayed rewards
- Specified or non-specified effects of taken performs on the environment (The agent may have not enough information about its effects if non-specified.)

3.2 Multiple Moving Agents' statistical learning

We utilize efficient algorithms and statistical theorems for preparing reachable graphs that can be solved from any initial arrangement through the agents' moves along the graph's edges. These techniques can be used in solving robot motion planning problem, especially in simulation planning. [4][5]

The main purpose is getting some pattern graphs that can present safe regions of

field until deciding become easier.

An initial algorithm presents simple and mid-safe graph in a certain period. The main algorithm is used for normalization, especially for coordinating of graphs. Finally, this algorithm prepares one graph that is not only a reasonable result but also a statistical one. Because each initial graph is checked 20 times and final graph is normalized from 7 initial graphs. So it would be a confident result.

There are two algorithms for finding initial graphs:

1. "Ants' colony" algorithm
2. Finding safe nodes algorithm by using a special matrix (with considering a 17*35 matrix for 2D soccer simulation field)

The purpose of this algorithm is to find safe regions in initial graphs. It is important that we get a path from algorithm-1, but get regions (vertices that restrict each player to a limited area) from algorithm-2.

Also, there are two algorithms for normalizing and coordinating 7 initial graphs:

1. An algorithm which is corresponding to fuzzy logic theorems.
2. An algorithm which intersects vertices and creates possible graphs.

The first algorithm should be used for normalizing "Ants' colony" algorithm, and the second one should be used for the other.

3.2.1Strategies

3.2.1.1 Graphs

We store the results of "Multiple Moving Agents' statistical learning" for making up the good strategies. It needs some good formations for each strategy.

3.2.1.2 Formation

It is good for a good game that the formations are changed in suitable cases. It helps that strategies act very well.

4 Future Plans

4.1 Q Learning Algorithm

Standard reinforcement learning or Q learning, is an online learning algorithm, which can learn a controlled policy for Marcov Decision Processing (MDP) based on the long term delayed reward in the presence of the reward function and specified

perform effect. The output of this algorithm contains the values in table.1. (The table contains learned values by the Q function when 'a' action is taken in 's' situation.)[1]

State	Action	Q(s, a)

Table.1. Table Q

4.1.1 Q Learning Algorithm Steps

- Inserting 0 as a primary value for each 's' and 'a' in the Q table
- Getting the present situation of the environment
- Repeating the loop below until an end condition:

  1- Choosing an action (a) by one of ways below and performing it

- Randomly (discovering): An action is selected without any refer to the Q table. This results in discovering optimum actions that have not been chosen so far and adding them to the table.
- Based on the Q table (exporting): The best action is selected based on the Q table and the actions that are gained so far.

  (It is needed to mention that choosing actions uses more discovering mode in the primary levels and inclines to the exporting mode in the finishing levels.)

  2- Gaining reward from the environment

  3- Caching the new situation of the environment (s')

  4- Changing the value of Q(s, a) according to the formula below:

$$Q(s, a) = r + Y \max. Q(s', a') \qquad (1)$$

  5- Switching present situation with the next one: $s' \rightarrow s$

The algorithm above begins from a primary situation and follows a few performs and gains reward to reach a final situation. (Each of these performs are called

episode.) It is common that in the aim situation, the agent returns to the situation after each action and does not gain reward from the environment. (These forms are called attractors.) [3]

The quite linear divider function will be calculable based on these educational data (as formula 2) and the passage of the ball by the goalkeeper probability is calculated by formula 3.

$$u = (a - 26.1) * 0.043 + (d - 9.0) * 0.09 - 0.2 \qquad (2)$$

$$P(\text{pass\_goalkeeper} \mid u) = \frac{1}{1 + \exp(-9.5\,u)} \qquad (3)$$

Agent2D team has been able to increase %70 in scoring efficiency by paying attention to the two parameters of angle and distance. (This efficiency is gained by assuming %100 of threshold limit.) As a result, it is possible to efficient the skills by adding other opportunities. (Figure 2)[1]

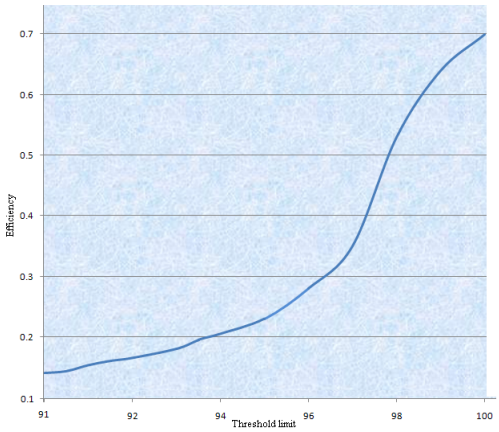


Fig.2. Efficiency of scoring skills in the Agent team

5 Conclusions

6 References

[1] Ghasem Aghayi, Naser & Rabiyi, Azam. "A Scoring Policy for Simulated Soccer Agents Using Reinforcement Learning" *(10th Iran Computer Society Conference, Technical and Engineering Department, University of Isfahan, 2004.)*

[2] Mitchell T.M., Machine Learning, *McGraw-Hill Press, International Edition, 1997.*

[3] Berto A.G & Sutton R.S., Reinforcement Learning: An Introduction, *MIT Press, 1998.*

[4] V.Auletta, D.parente, G.persiano, A New Approach to Optimal Planning of Robot Motion on a Tree with Obstacle, *proceeding 4th European Symposium on Algorithm (ESA), Spain*, 25-27, 1996

[5] Boutillier Craig, Planning Learning and coordinate on Multiagent Decision Processes, *Department of computer science university of British Columbia, Vancouver*