Team Description of Thunder 2002

Maysam Yabandeh, Ehsan Khosrowshahi, Amir Nayyeri
Tehran Institute of Electrical and Computer Engineering
Electronic and Computer Faculty, Tehran, IRAN
robocup@khorshid.ece.ut.ac.ir
ehsan_ka@yahoo.com

Abstract

This paper explains how and why we have implemented Genetic Algorithms finding the best parameters for our agents decision brain for Passing and then there is an overall explanation of our positioning strategies while defending and attacking. On the other hand we have not implement solid and efficient Low-level skills yet but the team plays a good solid game and scores well. Implementing GA for our team's Low-Level behavior will be our future work.

Introduction

The problem is what a good soccer team should do in order to win, or simply how a good soccer team plays. As we know a soccer team consists of eleven players so a simple answer to the question can be this, "All of the members of a perfect team should play perfect, individually and collectively".

So, what is a perfect player? This is the question we tried to answer with our code.

What does a good player do in different conditions? Which conditions can player be in? These are the two situations which we will discuss about them in this parer:

- Playing with ball (The ball is under the control of the player, here just about passing strategies)
- Positioning Strategies (The ball is not under the control of the player)

We discuss each case in one of the following sections.

Playing with ball

First we try to find the player in the field, having the best situation and the possibility of receiving the pass. If the guy isn't a player himself the player passes the ball to him, otherwise he himself continues with the ball by either shooting or dribbling. Now the problem is how to find the best teammate for receiving the ball.

We solve this problem by grading, giving a score for every teammate, then simply the with maximum score is the best one for receiving the ball. In order to give a score to every player we consider different parameters (for example the situation of him for shooting toward the goal, the number opponent players near him, etc).

First we decided to simply multiply the parameters but there were some problems and players couldn't find the best way as we wanted. The problem could be originated from different things. We thought that the parameters are not equally important. Thus we decided to replace every parameter with a linear combination of it (x[i] with a[i].x[i]+b[i]). First, It seemed easy, but a problem raised when we tried to implement it, What are a[i] and b[i] for each of them? Therefore we used the ideas of *Genetic Algorithm* to solve this problem. First we should configure the initial population so we form different a[i] and b[i] in the range of (0,1). We considered different situations, put each player in every situation and gave him a score from a hundred. That score showed how many copies of this coefficient system should participate in the initial population.

As you know we will have correct answer after adequate times of cross over in GA.

Cross Over:

Imagine that we have two arrays of coefficients and we want to mix them in order to produce new children.

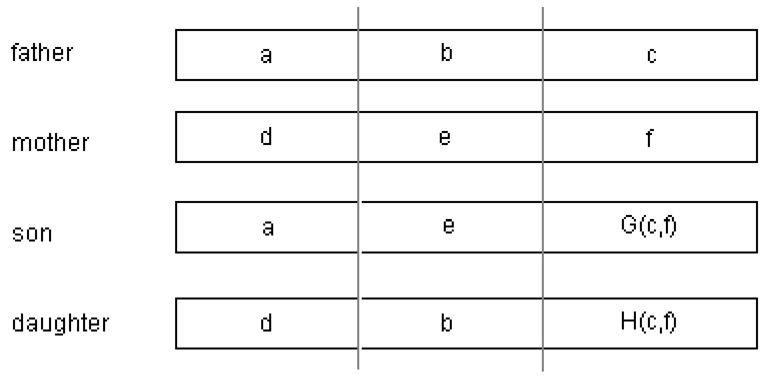

Figure I
Note that each letter shows a series of parameters

As is clear in the **figure I** a son or daughter inherited some parameters exactly from his/her father, some parameters exactly from his/her mother and some parameters are a mixture of both of them, which appeared in the form of functions G(c,f) and H(c,f). (These functions are average in our code, may be improved someday)

Our playing became much better after that, but there were something buttering us. We could find new situations in each game, put them in the GA pool after the game and improve, but players didn't improve during the game. **Figure II** was the answer of this problem. The Idea was gotten from the digital logic circuit course. We put a feedback analyzer. We gave it the possible data for k seconds after the player made the decision and we give it some limited opportunities to change the coefficients.

For example if we throw a pass, and the opponent agents cut the pass, we should increase the security coefficient of the pass for that player, but these variations should be limited.

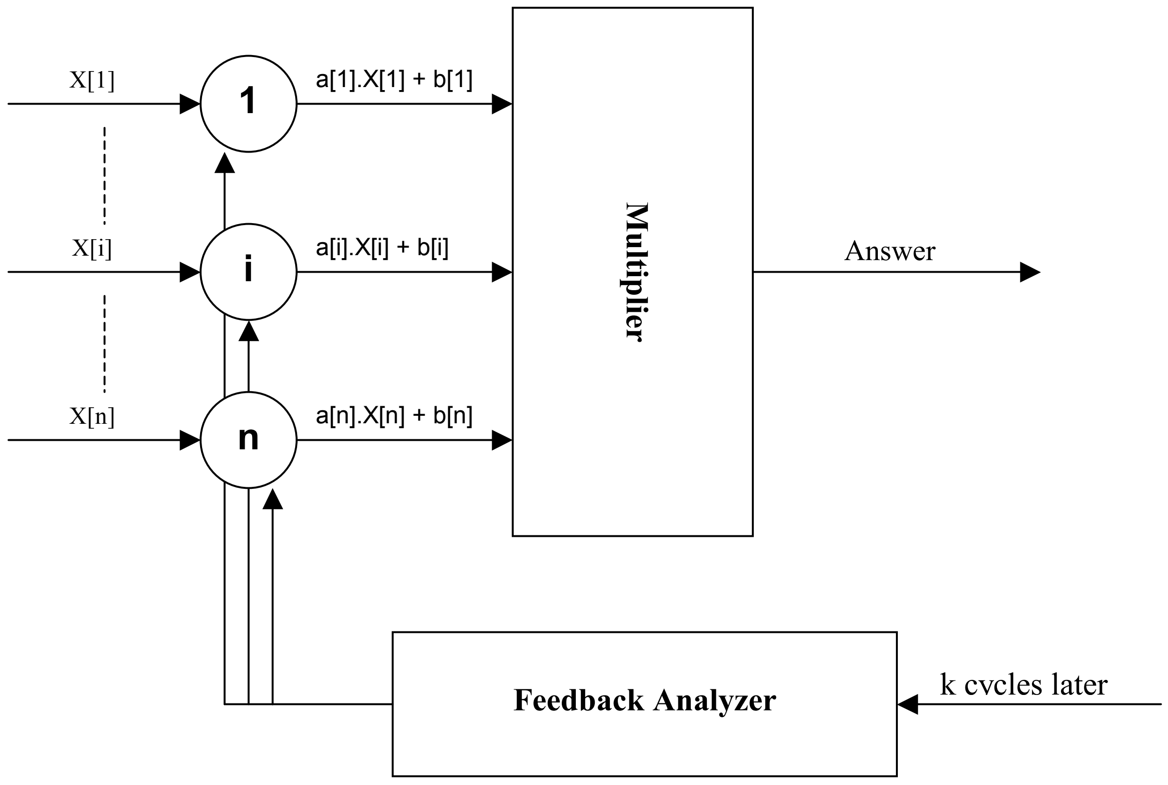

Figure II

This is the way we find the best player for reception.

Positioning Strategies

For positioning we have implemented basic and classical strategies. Putting players in lines and curves with considering the position of ball and opponent players and the amount of their attraction and weights in our conf files. Gaining some of them by our own experiences and the others by some simple Neural Network algorithms. Any way we have not implement GA for our positioning strategies, and we will not still work on that because the process time of GA algorithms is not efficient for many categories. As you know positioning requires lots of parameters and categories for these algorithms. So we are working on less complex algorithms for this stuff and using curved geometries instead of linear ones for our agent positioning. This makes us more available parameters (in our conf files) thus making more flexibilities and options in our positioning strategies.

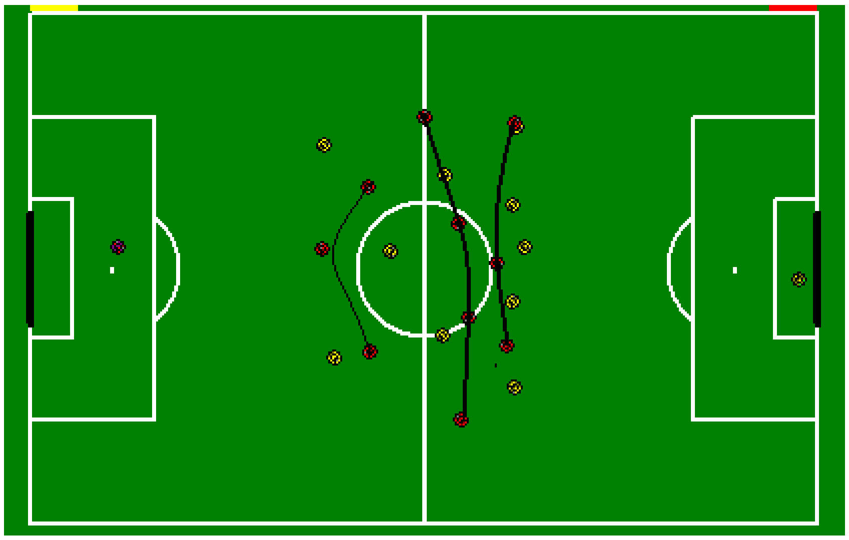

Figure III
Positioning Curves

Conclusion and Future Work

In this paper, we have shown how and why we implement Genetic Algorithms for our passing decisions. We are still working on our parameter categories making the GA more efficient. GA helped us to solve the problem of finding the best answer in a huge pool of possible conditions. We used a simple model to improve players during the game. As a matter of fact we implement learning with trying to find the best coefficients. Different conditions need different coefficients and we tried to find the best coefficients for each condition.
We want to work more on low-level behaviors, find the most adoptive to our works. We should implement a good goalie. We also will improve our passing, shooting, and positioning to make them work more accurately.

References

1. Fogel, David B. Evolutionary computation: toward a new philosophy of machine intelligence / David B. Fogel. – 2nd ed.
2. Holland, J. H. (1992). “Genetic Algorithms,“ Scientific American, pp. 66-72
3. Atmar, W.(1994) “Notes on the Simulation of Evolution,” *IEEE Trans. Neural Networks,* Vol. 5:1, pp. 130-148
4. IEEE potentials (2001) “Evolution and the Genetic Algorithm” *IEEE potentials, the magazine for up-and-coming engineers*, APRIL-MAY 2001, pp. 17-24