Mersad 2004 Team Description

Ahmad Boorghany Farahany, Mostafa Rokooey, Mohammad Salehe, and Meisam Vosoughpour

Allameh Helli High School,
National Organization for Development of Exceptional Talents
http://www.nodet.net

Abstract. *Mersad* is a Soccer Simulation Team with a multi-layer architecture and rich set of advanced actions. In addition, a powerfull system for designing strategies integrates the individual skills of players to achieve a cooperating team. In the paper we address the main features of our team and try to demonstrate the suitability of our approach.

1 Introduction

Our team is consisted of 4 students of Allameh Helli High School. Several teams from our high school have participated previous RoboCup competitions and proved to be powerful teams in the competitions. Our previous experiences of working with these teams and also having an online soccer coach team led us to design and implementation of a team based on the idea of advisable players, scenario-based teamwork and evaluation of strategies.

In this paper we present and specify the architecture of our agents and then introduce a powerful system for declaring and assigning stratefies to soccer agents. At last we will describe our dribble system and characterize the features of each kind of dribbling in detail.

2 Architecture

Mersad team has four executive layers, and one data layer. Also, there is one separate Coach that stands above all of them.

The Information Center layer contains all the agent's information. All sensory information is kept and processed in this layer. Authorized sections of the agent will later use this information for various needs.

The lowest layer is the "Connection and Synchronization" layer which sends and receives data from and to the Simulation Server. This layer is also responsible for timing procedures of data flow. This layer receives the commands it sends to the server from the "Basic Actions" layer, and puts the data it receives from the server in the "Information Center".

The next layer is "Basic Actions". This layer (as its name suggests) does the simple tasks of the Agent, all of which can be performed in just a couple of cycles,

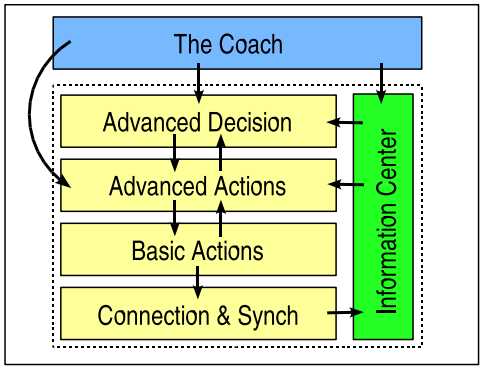

Fig. 1. Mersad Agent and Coach Architecture

e.g., kicks based on the number of cycles, or speed to other points. The most important and complicated function that this layer provides is InterceptBall.

The next layer is "Advanced Actions". This layer provides more advanced functions, and has 5 sections: Defense, Offense, Dribble, Shoot and Pass. Decisions made in this section will normally last for several ten cycles. The above layer decides which of these functions are to be called.

The highest layer is called "Advanced Decision". This layer selects one of the five functions provided by "Advanced Actions". The decision system is weight based; Each function gets a weight based on the situation. The function having the maximum weight is executed.

The Coaching section will affect the "Advanced Decision" and "Advanced Actions" sections of the players. It may also change some parts of "Information Center" such as Plan Sources.

3 Plan System

3.1 What is Plan?

The "plan" system is an advanced system for making decisions, which defines an overall strategy for the game. The system explained below is only used for offense, but it can also be adapted to be used for defense.

The first section of the plan is a major path, which is suggested by the plan to the players. The player tries to keep the ball as close as possible to this path.

The next system defines the pass sequence (i.e. Who passes to whom, then who is next). This system also has a time dimension, which tells the player when to be around what point. This section also forecast the ball location, if the plan is executed perfectly.

3.2 Plan Effects

The plan Affects offense, pass and dribbling.

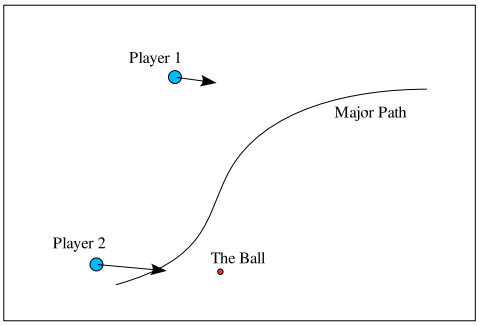

Fig. 2. Effect of Plan on Offense and Pass

Plan's Effect on Offense For example suppose a plan has defined the path shown in 2. First of all, the plan recognizes what stage of passing we are in, and determines the next pass. Suppose the next pass is going to be from player 2 to player 1, and the ball is headed for player 2. In such a situation, plan will affect the player 1's offense system. The plan tells player 1 to go to a point on the major path (Considering the path's direction) within the next "n" cycles, so that it can receive the pass coming from player 2.

Plan's Effect on Pass In the last example the situation is different for player 2. Suppose player 2 has reached the ball. There is a relatively large weight in the pass decision section which imposes plan's idea on the players by telling them where to pass and to whom(close to the major path).

Plan's Effect on Dribbling Dribbling will closely follow the path that plan defines. This method is implemented by giving this factor a high weight.

3.3 Dynamic and Indefinite Plans

The plan system tries to predict the game as best as it can, but sometimes the game will not follow the plan's prediction. We have also foresighted this, and have handled this in two different solutions.

Bridges Suppose that an opponent player has blocked a part of the path, such that we can not bypass it by a direct pass or dribble. In such a situation, the plan system will select a player to act like a bridge, by receiving the ball outside the major path, and resending the ball to the path by a very high priority pass.

Indefinite Situations When we reach close to the opponent's goal lime, it is practically impossible to define a precise path to follow. In order to solve this

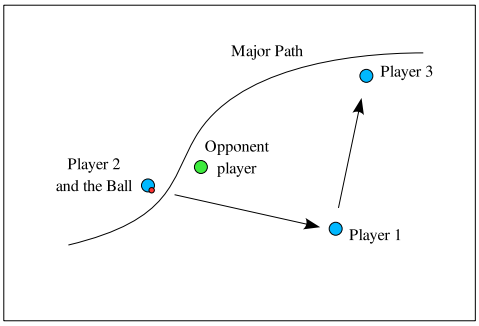

Fig. 3. A situation where the plan offers a bridge

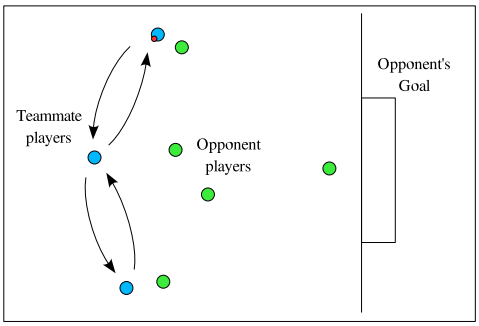

Fig. 4. A situation where the plan is in Indefinite Mode. The arrows show paths offered to Pass and Offense by the Plan.

problem, we introduce a passing loop. The players are to pass within these loops, until they find a way to the goal. These loops only change the weights of passing, and have no effect on dribbling.

3.4 Plan Change

Each player may have several plans in it's memory. If for any reason, it is decided to change the plan of the entire team, all players must change the plan to one particular plan, synchronizing via a method like "Say".

3.5 Learning the Plan

The last issue about plan is how it is built. The plan may be built by the programmer, and be given to the player, before the game starts. With an intelligent coach, it is possible to make a new plan in the middle of the game. The most important part of this idea is how the coach will adapt the plan to the playing style of the opponent.

4 Dribble System

4.1 What is Dribble?

Dribbling is one of the most important skills that distinguishes a Robocup Soccer Simulation agent by means of personal skills. Also, dribbling is important to the general procedure of the game. In our team, there exist three types of dribbling.

RunWithBall. One of the types is the dribble that is similar to running with the ball which is performed where a reasonable amount of open space is available to the player; In other words no opponent interferes with the player's path or at least there are no opponents near the player. In this dribbling style there is no limit on the maximum distance of the ball from the players. This dribble is used to obtain high speeds and to penetrate the opponent's defense line and works when there is no interfering opponent. We call this type of dribbling RunWithBall.

DribbleInKickable RunWithBall is not useful when opponents are close. In such situations, the player must be able to control the ball in any cycle he feels that the probability of losing the ball is high. He must also be able to move the ball into the least dangerous position. This means that the ball must be highly protected and always be in the kickable area, which is called DribbleInKickable. When an opponent attacks the ball owner, and covers a part of the teammate's kickable area with that of his own, or when the player decides to go through a place when opponents are present, the teammate must use DribbleInKickable and depending on the relative position of the obstacle to the body, the player will select a point in its kickable area, such that the density of opponents should be minimum in that point, relative to all other points in the kickable area. The player must also be able to move the ball with DribbleInKickable, as long as the following conditions are satisfied:

1. The ball is kept in kickable area.
2. The relative coordinates of the ball to the body, remains constant. In this situation it is possible for a player to be forced to use dash 40, or even less to control the ball. It is even possible for a player not to move and to protect the ball to obtain more caution. Due to precaution, and physical situations of the player, this dribble will not obtain a very high speed, therefore having a high speed in this dribble is an important advantage, which we have worked on it.

SRPDribble There also exists another dribble type called SRPDribble which is used to leave the direct opponent behind. It works by taking advantage of the opponents physical weak point, and the improper positioning of the opponent's body angle, to take the ball through. By sending the ball behind the opponent, while at the same time trying to reach it, we leave the opponent behind (similar to real soccer techniques).

4.2 Dribble Path Finding

Dribbles explained in previous section, are executed when the major path and the minor path of movement are clear for the player. Selecting paths is one of the most important parts of dribbling. In our path finding in dribbling, we have a specified goal(as a point), and go toward it while the following conditions are satisfied:

1. Confronting the minimum interference of the opponent.
2. Staying within the minimum possible distance from major path.
3. We should not doubt previous decisions. In other words, we are not allowed to change what we have selected in this cycle, in the next cycle. Many graph based path finding algorithms exist, like D* and A*, but we have used an innovative algorithm, because the obstacles are dynamic, and noise exists. The first thing we have to do, before path finding is to find the major path in the current cycle, so that we could do our path finding based on that. This major path is already defined by "Plan". We have two path finding algorithms, and select which one to use based on the current situation.

First Algorithm In this algorithm, first all players within a specified distance are sensed, then numerous angles around the player are analyzed. Important factors in the selection of the angles are:

1. The amount of crowdedness of the movement angle.
2. The closeness of the movement angle to the plan's major path.
3. The closeness of the movement angle to the player's current body angle.

The amount of crowdedness is analyzed by selecting the opponent players that are close to the ball (e.g., within 12 meters of the ball), and can interfere with the player's movement in that particular angle. Then, using the maximum tangent of the angle between the angle of player movement, and the angle of the opponent player with the ball, along with some details, the best (most uncrowded) angle is selected. This algorithm will only function correctly if the weights that affect these calculations are nearly optimal. This algorithm is normally used for path finding in more "open" spaces.

Second Algorithm In this algorithm, the best movement angle for the player is calculated from the sum of several forces. As pointed out in the beginning of the paper, a major goal for dribbling exists, such that if no obstacle exists, the player will follow that path and move toward that goal. This "goal" drags the player toward itself with a specified amount of force, which can be constant. The opponents and obstacles apply forces on the ball owner too. The amount of this force vector is inversely proportional to the distance of the obstacle and the player. The direction of the vector is perpendicular to the line that connects the obstacle and the ball. From the two possible vectors on the selected line and size, the one that is closer to the major goal is selected. From the (vector)

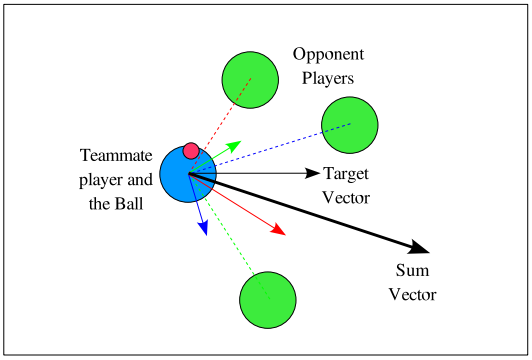

Fig. 5. A situation in which Dribble is finding a path.

summation of all of the forces, one force will be calculated, which shows the player's movement direction. However, what is notable is that in this algorithm, even sub-optimal weights work well. This algorithm is implemented to be used when the player is stuck between several obstacles and opponent players, and open space does not exist for player movement.

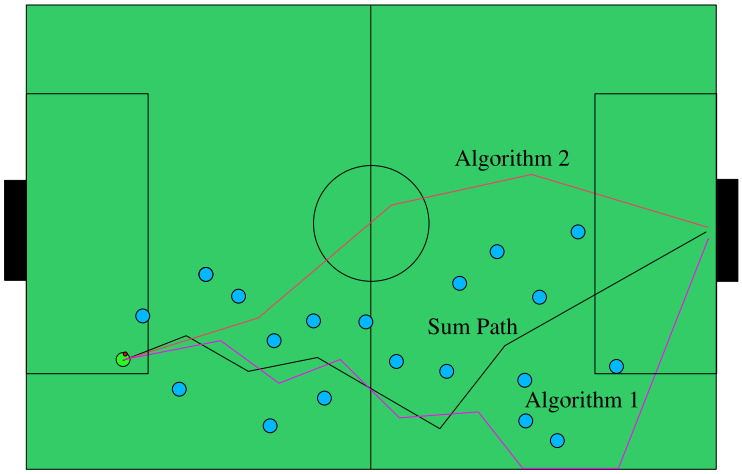

Fig. 6. A situation in which Dribble is finding a path. Path of algorithm 1 and algorithm 2 and sum of them is shown.