CS 420 Artificial Intelligence
Robot Soccer Project
Lafayette College
Department of Computer Science

Wah Loon Keng*

Advised by Prof. Chun Wai Liew

December 5, 2015

1 Introduction

We implement a simple A.I. to play the Robocup Simulation League - a 2D soccer simulation game. The simulator and server are drawn from *2D RoboCup Soccer Simulator*[1].

The server client we use is *jasontbradshaw/soccerpy*[2] written in `Python`. *Soccerpy* handles the client-server communication and most of the backend details. This allows us to focus on writing the A.I. agent to play the game.

For the A.I. components, we draw basic inspirations from *aima-python*[3] and the book *Machine Learning*[4]

2 Description

To make an A.I. that performs well, we observe how a real-life soccer game is played, and how a human player would think and react. It is easy to see that a team of humans exhibit

*Lafayette College, Easton, PA 18042, USA. kengw@lafayette.edu.

some degree of swarm intelligence. They plan and search for the optimal moves on two levels - locally and globally. The local behavior is used when they dribbled, move, overtake nearby enemies; the global behaviour is used when then pass, shoot, and move.

This has inspired us to employ a hybrid of reflex, utility and goal-based agents.

For the environment variables, we consider for each player agent:

1. the distance and angle to the goal posts
2. the distance and angle to the ball
3. the surrounding: the distance and angle of the nearest teammate
4. the surrounding: the distance and angle of the nearest enemy
5. the surrounding: is the path clear for ball, i.e. free of enemy interception
6. is the ball currently owned by our team
7. is the ball currently owned by the enemy team
8. is the ball kickable

The information will allow the use of our heuristics below. Then, we list the basic actions available to a player; these will be extended based on the role of the agent.

1. shoot (to the goal)
2. pass (to teammates)
3. move (run to strategic position, include dribbling)

Then, for the agent roles we striker, defender, and goalie. They implement variations of the actions above based on reflex (e.g. if a ball is shootable), utility (e.g. it is better to pass than shoot), and goal-based (e.g. scoring matters the most in the game).

We describe the 3 types of agent in more details. The heuristics are given from the highest order of importance.

Striker (agent_1.py):

1. *shoot at the goal post*, if the ball is kickable, and agent is close to the goal post, and the path is clear to shoot. The agent finds the ball and the enemy goal post, and scan globally the shooting path is clear of enemies. It then aims at the enemy goal post, and calculate the optimal kicking power, and shoot the ball. This is the most important action as it scores the goal.

2. *pass the ball*, if this is the next-best course of action. The agent does so if it is blocked or approached by enemies. To pass, it scans for nearby ally with a clear shooting path, then calculate the optimal kicking power based on distance, and pass the ball. Passing is also more energy-efficient than dribbling and running.

3. *dribble*, if the above can't be done, then the agent tries to dribble the ball closer to the goal post.

4. *move to the ball*, if enemy team currently owns the ball, and the agent is close enough to it.

5. *move to defend*, the striker moves back a little to get the ball back, if the enemy has the ball, and is close to our goal post.

6. *move to enemy goal post*, if our team has the ball. This is to prepare for attack and to accept a ball-pass from another teammate.

Defender (agent_2.py):

1. *stay back*, always try to remain on our side of the field to prepare to defend.

2. *pass the ball*, the same as from Striker agent.

3. *dribble*, the same as from Striker agent.

4. *move to ball*, the same as from Striker agent.

5. *move to defend*, the same as from Striker agent.

Goalie (agent_3.py):

1. *stay back*, always try to remain close of the goal post to prepare for enemy striker.

2. *pass the ball*, the same as from Striker agent.

3. *move to ball*, when enemy is carrying the ball close to our goal post.

4. *move to defend*, the same as from Striker agent.

3 Implementation

First, we enclose the *soccerpy* client code base into a module for exporting. We make `soccerpy/agent.py` the basic agent class. Then we extend this base agent class polymorphically for the 3 agents described above.

We modify in `soccerpy` the modules `agent, world_model` to allow for the use of our heuristics. For example, we added the methods to get teammate, enemy, scan the agent's surrounding for enemy v.s. teammate, determine if the target path is clear from enemy interception, who owns the ball, distances among objects.

The `main.py` imports these 3 agents, initialize 5 strikers, 5 defenders, and a goalie, then set up the positioning.

4 Result

Our A.I. won the class tournament. From our observation, we attribute this to the striker's heuristics of prioritizing goal-scoring, as well as good aim, kicking power control, and path finding.

The positioning of the 3 different roles of A.I. agents are difficult to spot - it is something which we have not successfully controlled. This could be due to some scaling errors in the distance movement heuristics. However, the other action heuristics still work as expected; the agents pass, move, shoot as specified by their roles, when observed in the debugger output.

5 References

1. akym et. al., *The RoboCup Soccer Simulator*. `http://sourceforge.net/projects/sserver/`
2. jasontbradshaw, *soccerpy*. `https://github.com/jasontbradshaw/soccerpy`
3. Norvig, Peter and Russell, Stuart, *aima-python*. `https://code.google.com/p/aima-python/`
4. Mitchell, Tom M, *Machine Learning*. New York: McGraw-Hill, 1997.