SEU_T 2005 Team Description (2D)

Jiang Dawei

jiang203@jlonline.com
Department of Computer Science and Engineering, Southeast University, Nanjing 210096, P.R.China

Abstract. This paper describes the main features of SEU_T 2005 robot simulation soccer team. After a brief introduction of SEU_T 2004, the main contributions of SEU_T 2005 were presented. They are: hybrid agent architecture, transactions in action sending, compound kick, and heterogeneous agent selection. Finally, we describe our future study directions.

1 Introduction

The SEU_T 2004 robot simulation soccer team has attend the China Robocup 2004 held in Guangzhou. Our team, which is based on UvA Trilearn 2003 source code, ranked 7th in that match, going beyond AmoiensisNQ and CSU_Yunlu.

This year, we made several extensions to our team. Firstly, we applied hybrid agent architecture. Secondly, we use transaction to maintain the consistency of the actions sent to the soccer server with the agent's world model. Thirdly, we implement a new kick skill, which is better than previous one. Finally, we introduce heterogeneous agent to our team.

2 Agent Architecture

The SEU_T 2005 robot simulation soccer team use hybrid agent architecture. It consists of modeling, communications, skills, domain knowledge, and deliberative reasoning. Since the soccer server's low band width limitation, we only use communication to share world model between agents. The modeling module gathers information from the real world, and abstract them to the internal form which is used by reasoning. The deliberative reasoning module is our high level decision component. It reads the information abstracted by the modeling, then use domain knowledge to decide which skill should take. Finally, the skill component decompose the task to several soccer commands and send them. Figure 1 depicts our agent architecture.

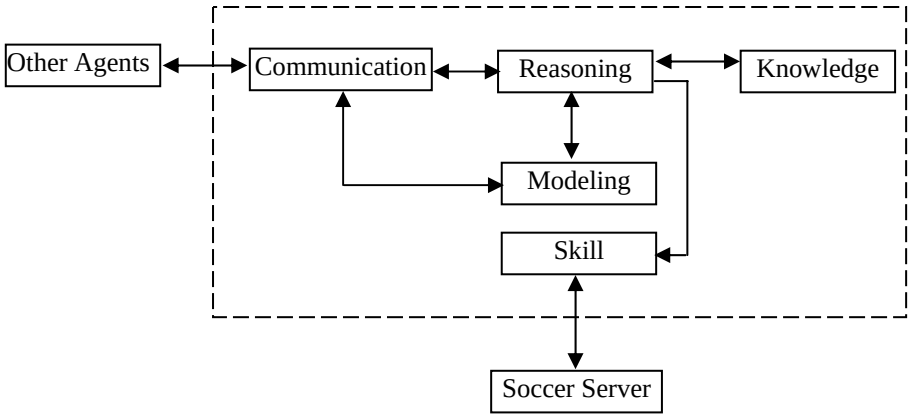

Fig. 1. SEU_T 2005 agent architecture

3 Transactions in Action Sending

Since the soccer server is a discrete real time system, the time when agent should send actions to server is of vital importance.

On one hand, the agent should send actions as soon as possible in order to prevent holes, but on the other hand the agent should make decisions based on the last world model, so he should send actions as late as he can. Assume the actions must be sent to the server before m milliseconds; the visual information will need *n* milliseconds to arrive, the agent decision making last *p* seconds. If $p < m - n$, the agent can make decisions on the last visual information. But in some situations, the agent can not do whole decisions before $m - n$ seconds but only part of them since reasoning process is too complex. What should he send to server? The agent should not send actions in his current buffered command queue, because the reasoning process is not completed, so some actions was made based on the sense body message, but others decided by visual message. The essence is how to maintain the consistency of actions sent by agent with his world model.

Our solution to this problem is using transaction. The idea and term is borrowed from database area. We use transaction to protect the decision making process. When the sense body message arrive, we start the decision making transaction, usually, it completed. When the visual message arrival, we also start the same transaction, if the transaction committed, the agent completed his decision making, the actions in buffered queue sent normally. If not, it means that there are not enough resources to let the agent finish his reasoning, so we roll back the last transaction. Thus the actions sent to the server was fully based on sense body message, the agent will always stay in a consistent state.

4 Compound Kick

Whether to be skillful in completing a kick task, such as pass, shoot, is critical in robocup soccer match. A lot of work has studied the problem. The most promising one is Q-learning combined with adversarial planning introduced by TsinghuAeolus. Generally, we adopt their idea, but made a lot of improvements. Firstly, we optimized for heterogeneous agent's kick rand and kick margin. Secondly, we use a new decision process which is totally different from TsinghuAeolus to map the real continuous space to discrete training state.

A kick task is that given a initial state in current cycle, planning a series of kick to accelerate the ball to desire velocity. The decision process should be efficient, robust, and adversarial. Like TsinghuAeolus, our solution includes two steps. The first one is offline learning. The second one is online planning. In offline learning, the spaces and actions were discretized. The Q-learning method is adopted to evaluate different actions. We use a variable reward method to optimize for heterogeneous agent's kick rand and kick margin. In the second one, we use a mapping method to map the continuous spaces in real math to discrete spaces in offline learning.

Define the player's center as the coordinate origin, the ball's desired velocity direction as the x-axis. In the offline learning phrase, we only consider the ball's relative position and desired speed, all the other things were ignored, i.e. agent body facing, agent global velocity, ball initial speed. The ball was randomly placed in the player's kickable margin area, then the kicker learned how to use a lot of small kicks to accelerate ball to desired speed. In order to optimize for heterogeneous agent's kick rand and kick margin, we use a variable reward method to evaluate the actions agent selected.

If the ball can be accelerated to the desired speed in the next cycle, the reward is 1.0 – cost(*A*). Otherwise the kicker get the reward 0.05 – cost(*A*).

cost(*A*) is the cost of the action agent take action *A*.

$$\text{cost}(A) = \alpha \times bvel + \beta \times \text{fabs}(bdist)/margin \quad (1)$$

bvel is the absolute speed the ball can get when use *A* , and *bdist* is the relative distance the ball travel. In our reward scheme, the higher velocity and further distance the agent kick ball to, the lower the reward he got. Thus we encourage the kicker to kick ball small and short. Use this technique, the influence of the heterogeneous agent's different kick rand and kick margin were decreased to least.

In the online phrase, we use a lot of functions to map the continuous spaces to the discrete training one. We consider all other things ignored by offline learning, i.e. agent body facing, player's move, desired velocity direction. For compensating the agent body facing, we use function f_1 to rotate the ball's position to a corresponding training one, following the criteria that in both cases the effective kick power was the same. With the same idea, we use f_2 to compensate the player's move and f_3 to mediate the real desired velocity direction with the offline learning case. We also use f_4 to avoid side line and f_5 to avoid closing opponent. Finally, we combined the five small mapping functions into one to decide which discrete space should be used. Figure 2 depicts how we compensate agent body facing.

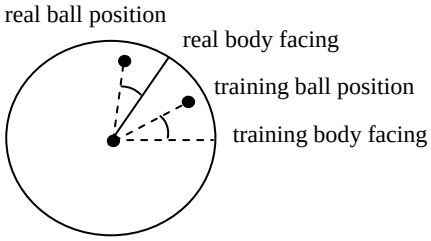

Fig. 2. example for compensating agent body facing

We test our method in 10 real matches, the result is perfect. The agent can accelerate ball to desired speed as fast as possible and with little influences of the kick rand and kick margin. Our implementation was better than TsinghuAeolus, since we take more things into account and we can fully use Q-table, while their implementation can not.

5 Heterogeneous Agent Selection

We abstract 4 criterions from 11 heterogeneous parameters. Firstly, we calculate the agent's start up cycle. Although the player's max speed was fixed to 1.2 in current soccer server, different agent can reach the speed using different time, the smaller the better. Secondly, the stamina factor is taken into consideration. When the player run in max speed, he will lose stamina, the little the better. We consider the kick rand and kick margin factor finally, because our kick skill was influenced little by the two parameters.

To combine the all the criterions, firstly we normalize each one to [0.0, 1.0], then use following formula:

$$F = \alpha \times start\_up\_cyc + \beta \times sta\_lost + \gamma \times kick\_rand + \lambda \times kick\_margin \quad (2)$$

Then all the player types will sort by the value, the highest is the best.

6 Conclusion and Future Work

In this paper, we have discussed our main work done in SEU_T 2005. For future study, we will concentrate on using the transactions to high level decisions, such as wall pass, because the all or nothing semantic fits well in such situation. We are also interested in using reinforcement learning to improve our agent's individual skill, especially in adversarial dribbling.

References

1. Remco de Boer, Jelle Kok. The Incremental Development of a Synthetic Multi-Agent System:The UvA Trilearn 2001 Robotic Soccer Simulation Team[Master Thesis]. Artificial Intelligence and Computer Science, University of Amsterdam. February 2002.
2. Jinyi Yao, Jiang Chen and Zengqi Sun. An application in RoboCup combining Q-learning with Adversarial Planning. The 4th World Congress on Intelligent Control and Automation.