# Architecture of TsinghuAeolus

Jinyi Yao, Jiang Chen, Yunpeng Cai, and Shi Li

State Key Lab of Intelligent Technology and System,
Deparment of Computer Science and Technology,
Tsinghua University , Beijing, 100084, P.R.China
{Yjy01, Chenjiang97, Caiyp01}@mails.tsinghua.edu.cn
lishi@s1000e.cs.tsinghua.edu.cn

**Abstract.** RoboCup Simulation Server provides a wonderful challenge for all the participants. This paper explains key technology implemented by Tsinghuaeolus RoboCup team played in RoboCup environment, including basic adversarial skills, which is developed by using Dynamic Programming in combination with heuristic search algorithm, and reactive strategy architecture. Tsinghuaeolus was the winner of RoboCup2001.

## 1 Introduction

RoboCup has gone through five years, with annual matches becoming more and more attractive. Tsinghuaeolus was established in early 2000. Our goal is to do research on muti-agents environment, especially in RoboCup. How to make agents react as what we intend? How to coordinate them? How to teach and direct them online? All these challenges have been pushing us forward. We have attempted to apply some learning methods in individual players' basic skills. But in high-level strategy decision-making, we put more effort on analytical approaches that seems comparatively easier and more effective than many learning methods.

## 2 Basic Skills

Basic skills include interception, dribble and kick, etc. We experience that these skills are very important to win a match. In Seattle it seems many teams have done it to near perfection. To make an agent competent in these skills alone is easy for most teams. Our main effort is put on adversary management.

### 2.1 Adversarial Dribbling

This skill is totally hand-coded. There are two rules which are strictly followed. First, the ball is kept in the kickable area every cycle. Second, the ball is kept out of the kickable areas of all opponents. The first rule is to ensure that the ball can be adjusted in time when an opponent is closing in. Only after applying these rules, will the agent consider how to move forward.

### 2.2 Adversarial Kicking

When meeting the ball, an agent may hope to kick it out in some desired angle with some desired speed. To complete this task, several steps to adjust the ball might be necessary, such as pulling the ball back first. Meantime, given an opponent or the sideline is close to the left of the agent, then he had better to do these adjust-kicks with the ball to the right. Our approach can be divided into two steps. First, putting the later limitation aside, the task is simplified to adjust the ball by minimum times. Only ball's relative position and velocity are considered here. The position is discretized, and velocity is dealt as from one discretized position to another one. Alternatively, the inputs are translated to kicks that move ball from one position to another. Then Reinforcement Learning is used to evaluate all possible kicks between two discretized positions. Learning process is completed in only several seconds. Second, the former evaluation is taken as heuristic knowledge. With the help of that knowledge, a searching algorithm, similar to A*, is used to find a possible routine for the ball to both avoid the nearby opponents or sidelines and move effectively as intended (See Figure 1). This approach proved to be successful in the games.

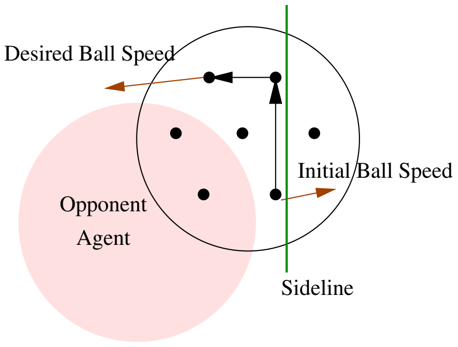

**Fig. 1.** Example of kick routine in adversarial conditions

## 3 Strategy Architecture

Real-time decision-making in a match is a complex task as so many factors have to be considered. We structured the whole task into several modules, including communication, visual control, handle-ball, offense positioning, defense positioning, etc. More attention was paid to the later four. The whole strategy architecture is shown in Figure 2. The following explains the typical components in the architecture.

### 3.1 Action Generation

The task of the action generator is to provide possible actions. The less actions provided, the easier the following decision would be. But the best action

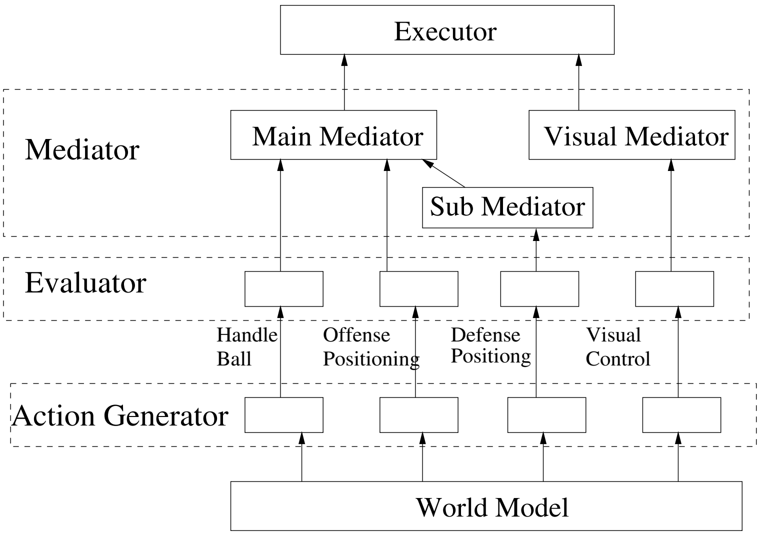

**Fig. 2.** Strategy Architecture

shouldn't be left out. Here we explain how the action generator for passing works. Given that an agent want to pass the ball in the game, he may have many choices. Because it's a tremendous state space consisted of 22 players. In other words, there are so many pass routes, each of which can be represented by a pass angle and kick power. We developed an analytical approach to slashes the useless routes. The angle of pass routes is first discretized. Then along a given angle, the ray that starts from the position of the ball can be divided into many segments, each of which is "controlled" by an agent in the field. "Control" means the agent is nearest to the arbitrary point on this segment of the line (See Figure 3). So when passing along a specific angle to an agent, there exits a velocity section reciprocal to his controlling segment. Alternatively, only if the ball is kicked out in a speed belonging to this velocity section, then it can be reached first by this agent. There is a presupposition that all the agents are homogeneous. Now in a given state that the ball is ready to be passed, we can

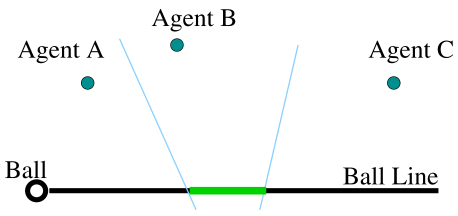

**Fig. 3.** Example of "control" (The green and bold segment is controlled by Agent B).

get a collection of all the possible pass routes. The following strategy is just to evaluate them and select the best one.

### 3.2 Evaluation

Each input action is evaluated and assigned a priority that means how valuable it is . Neural networks do much help here. Human prior knowledge is transferred into codes directly.

### 3.3 Mediation

The mediator gets many actions with reciprocal priority assigned by the previous evaluator from the lower layer. There are two kinds of relationships between these actions, conflicting or compatible. The task of the mediator is to find the optimal combination.
When actions to be mediated are all conflicting with each other, the task is simply to select the best one. Handle-ball belongs to this kind. But more difficult situations may occur in defensive strategy decision-making. For example, three defenders versus three forwards. For each defender, there are four choices: defend against one of the three forwards or keep one's own formation position. So there are totally $3 * 4 = 12$ actions are submitted. We think it's wasteful for two defenders to defend the same forward, and it's impossible for one defender to defend against two forwards. In other words, these actions are conflicting. It's obvious that a combination of compatible actions can includes three actions at most. A fast algorithm is developed for the mediator to find the optimal combination. With the help of this mediator, a reasonable defensive system can be constructed when facing an opponent team's attack.

## 4 Conclusions

Coordination and Counterplot are two basic themes in mutiagent system as RoboCup. With limited communication , it's a convenient and effective way to assume that other agents would reason as oneself. Based on this, analytical approaches are widely applied in Tsinghuaeolus. The architecture of our agents is wholly a reactive and layered one, which seems to be competent for adversarial planning and coordination in mutiagent system. But with the work going on, some obstacles and limitations appeared with this architecture. For example, the information exchange between neighboring layers is commonly only from the lower layer to the upper one. It would obviously be much better if there exits mutual exchange. Furthermore, it seems rather difficult in high-level strategy. We haven't still done any work about online coach and heterogeneous agents, which we'll consider in the future.

**Acknowledgements.** We would in particular like to thank the RoboCup Committee for providing such a wonderful platform. It was a great event in Seattle. We would like to thank CMU for their published source code which was used for reference at our initial stage of development.