Technical Solutions of TsinghuAeolus for Robotic Soccer

Yao Jinyi, Lao Ni, Yang Fan, Cai Yunpeng, and Sun Zengqi

State Key Lab of Intelligent Technology and System
Department of Computer Science and Technology
Tsinghua University , Beijing, 100084, P.R.China
{Yjy01,Noon99,Caiyp01}@mails.tsinghua.edu.cn, albertyang@263.net
szq-dcs@mail.tsinghua.edu.cn

Abstract. TsinghuAeolus is the champion team for the latest two RoboCup simulation league competitions. While our binary and nearly full source code for RoboCup 2001 had been publicly available for the entire year, we won the champion again in Fukuka, with more obvious advantage. This paper describes the key innovations that bring this improvement. They include an advice-taking mechanism which aims to improve agents' adaptability, a compact and effective option scoring policy which is crucial in the option-evaluation framework, and thorough analysis of interception problem which leads to more intelligent interception skill. Although not strongly interrelated, these innovations come together to form a set of solutions for problems across different levels.

1 Introduction

RoboCup Simulation League provides a challenging platform to promote research. By six consecutive years' effort of the whole community, the simulation match has improved amazingly, from being seemingly stupid to so delicate that some people think it's more than real. It should in great part owe to the whole community's great effort on sharing binaries and source codes. As the champion team in Seattle, TsinghuAeolus2001 released both executable and source code. Its effectiveness is convincingly proved by the fact that the newcomer Everest, who started completely from the released TsinghuAeolus2001 source code, won the runner-up in RoboCup 2002.

There was a debate in Fukuka on how to schedule simulation platform's development, alternatively speaking, how to develop the platform to help all the participators do research on it. Since the attraction of RoboCup depends partially on competition, one of the principles should be to encourage winning from better research. On the other hand, RoboCup is so complex an environment that various problems have to be solved to build a competitive team. We feel it's important to summarize what we have reached so that the following researchers can build a strong team easily, on the basis of which they can focus on what they are interested in. For this purpose, in addition to releasing source code, we

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 205–213, 2004.

try to describe our various key innovations in this paper, especially those that make TsinghuAeolus2002 improve from TsinghuAeolus2001.

Taking external advice is an important way toward adaptability. Our first innovation is an advice-taking mechanism, which can be attached to an existing autonomous agent. It works somewhat like case-based architecture. The detail is discussed in Section 2. In RoboCup, the architecture for action selection in ball possession is the key component for a team. But few teams have explained their work on this aspect in detail. Our second innovation focus on the creation and evaluation of pass options. It is discussed in Section 3. The third innovation is further analysis on interception, on the basis of which we develop an adversarial interception skill. More detail is discussed in Section 4.

2 A Scheme Mechanism for Adaptability

Adaptability has been a desirable quality for autonomous agent all along. But in some complex environments, it's not easy for an autonomous agent to improve by itself. RoboCup simulation platform, which is a highly dynamic, adversarial multiagent environment, is such a typical case. The state space of 22 agents in a continuous soccer field seems immense, and it's hard to decide how an action, in a sequence of hundreds of joint actions, is responsible for final success or fail. Besides that, player's local and noisy eyeshot is also a big disadvantage. Recently there is an increasing focus on external advice which is expected to take a quite important role for agent's adaptability in many real applications. In RoboCup simulation, an online coach acts as an advice-giving agent. In human matches, teammates also help each other a lot by advising. Besides these, It will also be quite interesting and meaningful if a domain expert can direct an autonomous agent online by giving advice.

Given external advice, an autonomous agent needs to incorporate them into its existing reasoning process. [9] converts the advice to internal neural hidden units, which are then integrated into the existing knowledge-based neural network. In this way, advice is incorporated seamlessly and can be redressed by later learning. But it depends on a knowledge-based neural network which is not applicable in many domains including RoboCup. [3] converts coach advice to some behavior which competes with other spontaneous behaviors in the same framework of behavior manager. One of its advantages is that advice won't be followed blindly. We develop a compact and efficient mechanism to incorporate advice. It is successfully applied in TsinghuAeolus2002. There are two main roles here: scheme and scheme manager. Programmly each piece of advice is converted into an object called "scheme", which is then inserted into a corresponding scheme manager. When the agent does related reasoning, the scheme manager will be called to run in predefined point. Through the whole process, there are three aspects most concerned:

- **State of Advice**
  Advice is often given as condition-dependent. We define four states for each scheme: sleeping, waiting, working and obsolete. Three conditions are required to update a scheme's state: C_{start}, C_{end} and $C_{activate}$. Before C_{start}

is satisfied, the scheme is sleeping. When C_{end} is satisfied, it is obsolete. Between them, if $C_{activate}$ is satisfied, it's working, otherwise waiting. Only working schemes are really incorporated into the agent's decision-making.

– **Where to Consider Advice**
This is not always a problem. But if you have already an autonomous agent and hope to incorporate advice into its existing decision process, which was not designed to be advice-compatible, you have to consider it. This problem seems domain-dependent and its solution is quite heuristic. A basic idea we use is to classify schemes. Then we heuristically select a point, somewhere in the agent's decision process, for each class instead of each piece of advice. For each class of advice, there is a corresponding scheme manager responsible to update its schemes' states and run all the working ones in some order. Alternatively speaking, the scheme managers also determine what kinds of advice may be incorporated. Those pieces of advice that can't be classified into any existing scheme manager have to be abandoned.

– **Conflicting Issues**
How to handle conflicting issues is concerned by many research work [10] [3]. We mainly handle two kinds of conflicting cases: exclusive and competitive. For exclusive case, according to some heuristic criteria, an value of rank is given to each scheme, which determines the scheme's order of being processed in its manager. The scheme with the highest rank can be considered first and then the other kindred schemes are ignored. For example, it's reasonable to think that an advice from coach is more authoritative then one from some teammate, so we give the coach advice a higher rank which guarantee it can exclude other teammates' kindred advices. For competitive case, we give an value of advice strength S_a which means the extent that the adviser recommend this advice. A typical competitive case is option-evaluation architecture(See Section 3). When an option is advised, its priority is added by an extra value which is proportional with S_a. We'd like to address that the priority is first calculated according to evaluation, which means the detail of this advice is verified by the agent itself. This is a fundamental difference between exclusive case and competitive case. The extra value can also be negative which means this option is discouraged. Generally, we use a piece of advice to encourage a group of options. For example, "pass more to region R", all the pass options whose destinations lie in R are encouraged to same extent.

A simple scheme is a five touple $\{C_{start}, C_{end}, C_{activate}, Rank, Content\}$. The first four have been explained above. The last one is an extendable component, the instances of which often differ across classes of schemes. For example, S_a is extended in $Content$ for classes in competitive cases. The explanation and execution of $Content$ depends on the corresponding scheme manager.

We haven't used online coach to give any advice because we haven't worked on online learning by coach yet. Although some of other teams' online coaches are available, it's not reasonable to assume their advices more advisable than an agent's own autonomous behaviors. To experiment the capability of this mecha-

nism, we give a variety of heuristic advices by ourselves. Some of them are listed here, described in natural language.

- When Playmode is changed, scan the field with wide visual mode in less than 7 cycles
- When Player P_a gets the ball in Region R_A, run to Region R_B and keep some distance from P_a
- Don't pass across our own forbidden area.

It proves to be quite natural and smooth to incorporate them into the existing TsinghuAeolus agent under the scheme mechanism. If using online coach to give advice, we would need to code conditions and actions in some language, for example CLang [15]. And after the interpretation, the left things are just what have been discussed above. In the TsinghuAeolus2002 champion agent, there are totally 5 scheme managers and about 20 pieces of advice that may be given according to the situation. We make a comparison between enabling advice and disabling it. The result is shown in Table 1. It proves to work very well. The improvement should owe to both good advice and the suitable mechanism to incorporate it.

Table 1. Goals in 18000 cycles(30 minutes). To suppress the randomness of heterogenous players across different matches, the parameters for all player types are forced to be same with that of normal type, so all the agents for both sides are homogenous. The another side is TsinghuAeolus2002 in all the cases, with scheme enabled or disabled

Opponent Team	Scheme Enabled	Scheme Disabled
FCP2002	0 - 24	0 - 16
USTC2002	2 - 18	3 - 13
TsinghuAeolus2002	2 - 2	4 - 0

3 Action Selection for Ball Possession

For controlling the player with the ball, the option-evaluation architecture [2] fits well. It involves two layers. The lower layer is in charge of producing options, such as dribbling, passing and shooting and so on. The upper one is to evaluate these options according to various score policies. Then the option with the highest score is to be executed. Although not called exactly as option-evaluation, the corresponding modules in [8], [7] and [12] seem to be quite similar. They differ only on the creation of options and score policies, which determine a team's competence in great part.

As introduced in [2], passing options are created at discreet angle increments and speed increments. For each angle-speed pair, it's rejected if some adversary could intercept the ball before any teammate. Our way to generate pass options was described briefly in [12]. The full implementation source code is available in [14]. We also considering passing the ball at discrete angle increments, but not at speed increments. For each angle, the speed section dominated by each

agent (Any speed in which, paired with the angle, forms such a pass option that the agent can receive the ball in advance of the others.) can be calculated in a simplified model that involves only the positions of all the agents, instead of other things such as their bodyfacing and heterogenous ability. A key technique in this algorithm is reviewed in Section 4. Viewed in the polar coordinate of passing angle and speed, the whole domain is separately dominated by the agents. Here each agent's domain is exactly the set of pass options that will be received by it superior to the others. It's approximated in both dimensions, angle and speed, by discretization in [2]. Our method only needs to discretize the angle dimension, but the cost is that it's specific to the simplified model.

Option's score is best understood as expected reward in the sense of Reinforcement learning[2]. A set of options defined over an MDP constitutes an embed SMDP[11]. [11] gives theoretical insight into the related learning issues. But it's hard to be implemented in RoboCup right now. As we feel, the biggest difficulty is how to automatically give the reward for an executed option accurately. In the whole process from kick-off to goal, there are lots of options executed by different agents. It's hard, even for human, to decide whether it is good or not for each of them. Until now, we haven't known the score policy of any top team is fully dependent on learning. There is much research work that can be done on this subject in the future.

[2] calculates the score of an option by $p_s * v_s + (1 - p_s) * v_f$ where p_s is the possibility of success and v_s and v_f are the values of succeeding and failing respectively. TsinghuAeolus2001 also used this natural form and even took a nearly same way to calculate p_s and v_s. Our improvement for TsinghuAeolus2002 comes from further analysis and approximation of these factors. For p_s, there are at least two factors responsible to the failure of a pass course, the noise of kicking action and the unaccurate knowledge of other agents, such as their positions, body facing and responsiveness and so on. The danger of the first one is approximated by the margin of the option relative to the agent's domain in the polar system of passing angle and speed. Obviously, bigger is this margin, more robust it is as kicking concerned. The other one can be approximated by the margin between its and other agents' interception times. For v_s, we also divide it into two factors, direct reward v_{dr} and future reward v_{fr}. v_{dr} is calculated according to a static potential field which is built and stored in a neural network beforehand. It's a function of location that increases as the ball advances towards the opponent's goal. v_{fr} is approximated by evaluating the suppositional situation when the receiver gets the ball. If there's an open space around it or a good shooting opportunity, we say there is high future reward expected. It's not hard to heuristically combine them to get the final score. In fact, since scores are used only in competition between the options in the same cycle, we don't need to keep them consistent through cycles as what is generally required in Reinforcement Learning.

Our advantage in pass is quite obvious in RoboCup2002. According to the statistical data, the counts of TsinghuAeolus' successful pass are double as many as that of the other top teams'.

Table 2. Pass counts Statistic. The data are generated by Matrix's Team Assistant's analysis on TsinghuAeolus' logs in RoboCup2002

Opponent Team	Success Pass	Score
FCP2002	79 - 198	0 - 7
Everest	80 - 161	0 - 7
rUNSWIFTII	68 - 193	0 - 6

4 Adversarial Soccer Skill

Basic soccer skills, such as kicking, interception, dribbling and so on, are essential for a soccer team. In simulation platform, these problems seem quite fitting the classical machine learning techniques. Various algorithms, such as RL, A* and so on, have been tried and worked well here. A more challenging problem is to exert these skills in an adversarial environment. In [5], RL is used to learn dribbling against an opponent. In [13], Q learning and A star algorithm are combined to provide a fast and efficient approach for kicking in adversarial environment. One of our innovations in TsinghuAeolus2002 is an adversarial interception skill which is based on the further analysis of the interception problem.

Interception problem refers to how to turn and dash, for a given player, until getting the ball given initial physical information of the ball and itself. Supervised learning by collecting successful examples is an empirical way to obtain this skill, which was recommended in [1]. But later, analytical way seems to be more popular for the advantage of efficiency. [2] presents a numerical algorithm for computing interception times. We have done the similar work and made some improvement.

A simplified model of interception involves an arbitrary ball position B_{p0}, initial ball speed B_v and its direction, and receiver position P_r. Other factors, such as the receiver's body facing, velocity and so on, can be considered heuristically later. The ball's moving route is fixed if random noise is not concerned, The reachable area for the receiver in t cycles later is a circle with its initial position as the center. The circle's radius increases linearly by t in the receiver's maximal speed v_p . With t as the third dimension, we can get a 3D view of the interception process, as illustrated in figure 1. The ball's route C_b is a curve and the receiver's reachable area R_r is a cone. The overlay part of C_b with R_p is exactly the set of points where interception may be completed, which we call as Sol. [2] calculates the least time to complete interception. But it's not always the best choice to intercept in the least time. Imagine such a scene, a player lies in front of the ball when the ball is moving forward with a high speed, it's not wise for him to run backward to get the ball and then dribble forward. The better choice is to run forward to meet with the ball. This kind of scene can be found a lot in human's match, especially in anti-offside case. So it's useful to figure out Sol completely. In the 3D coordinate, C_b lies in an vertical plane which intersects the surface of R_r in a hyperbola C_p. As has already been pointed out in [2], there are at most three intersection points between C_b and C_p.

To look into this problem, assume C_p and the direction of the ball's velocity is fixed while B_v is alterable. With varying B_v, C_b may be tangent to C_p in

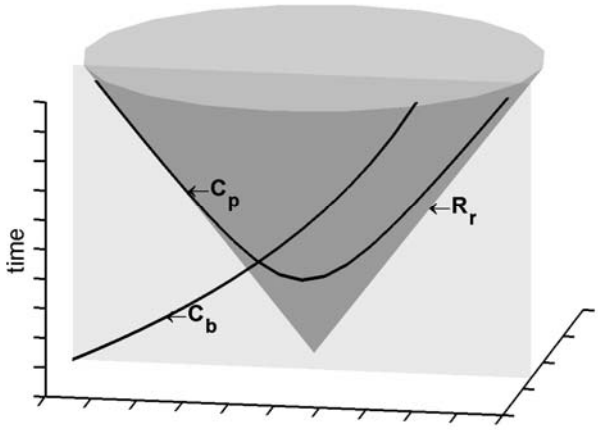

Fig. 1. Interception

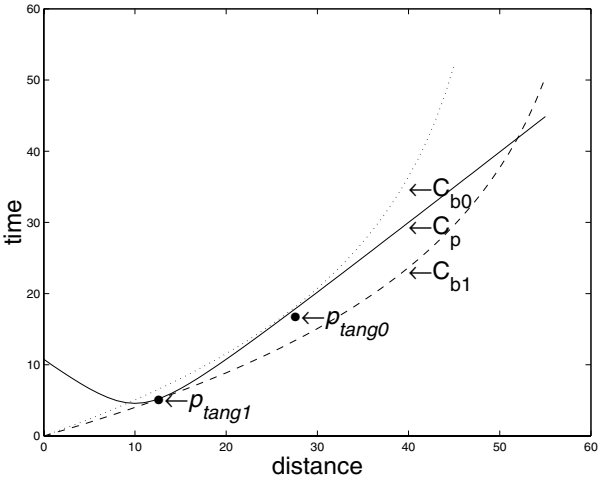

Fig. 2. Two tangent cases

two cases, as illustrated in figure 2. The corresponding values of B_v are b_{tangv0} and b_{tangv1}. The corresponding tangent points are p_{tang0} and p_{tang1}. It's easy to know that there is only one intersection point between C_b and C_p when B_{v0} is in $(0, b_{tangv0})$ or in $(b_{tangv1}, +\infty)$, two when B_v equals b_{tangv0} or b_{tangv1}, and three when B_v is in (b_{tangv0}, b_{tangv1}). Given C_p, p_{tang0} and p_{tang1} can be approximately figured out by Newton methods with appropriate initial values for update. Then the corresponding values of B_v are easy to figure out.

Back to the original problem where both C_p and C_b are given, we use Newton Method to calculate the intersection points, plus some preprocess and tricky treatment. C_p is first divided into two symmetrical parts at its apex p_s. It's obvious that there is not more than one intersection point between the left part

with C_b, which, if exists, can be figured out with standard Newton Method. For the right part, assume there are three intersection points p_0, p_1 and p_2. We only discuss this case in the following. The other cases can be handled in a similar and simpler way. It's obvious that p_0 lies between p_s and p_{tang1}, p_1 between p_{tang1} and p_{tang0}, p_1 between p_{tang0} and $p_{+\infty}$(a point in C_p whose x value is big enough). We use $f_{cb}(x)$ to represent the function of C_b, and $f_{cp}(x)$ for the right half of C_p. The standard Newton update equation is as following.

$$x_{i+1} = x_i - \frac{f_{cp}(x_i) - f_{cb}(x_i)}{f'_{cp}(x_i) - f'_{cb}(x_i)} \tag{1}$$

But it's possible to overshoot. $f'_{cp}(x)$ lies always in $(0, v_p)$. With the maximal and minimal value replacing $f'_{cp}(x)$ in 1, We get the following two modified update rules.

$$x_{i+1} = x_i - \frac{f_{cp}(x_i) - f_{cb}(x_i)}{0 - f'_{cb}(x_i)} \tag{2}$$

$$x_{i+1} = x_i - \frac{f_{cp}(x_i) - f_{cb}(x_i)}{v_p - f'_{cb}(x_i)} \tag{3}$$

To take p_s as the initial point and (2) as update rule, x_i converges to $x(p_0)$. Because in the iterations, $x_{i+1} < x_i, x_i > x(p_0)$, which is not hard to prove and we don't intend to detail here. Similarly, we can reach p_1 by taking p_s as the initial point and (3) as update rule, p_2 by taking $p_{+\infty}$ as the initial point and (2) as update rule.

There is another related problem which is crucial for creating pass options as introduced in Section 3. It originates from the need to know the minimal pass speed to pass across some point before some player can get it. Here C_p, a point p on it, and the direction of the ball's velocity are given, and the problem equals to figure out the maximum B_v when there is at least an intersection point before p between C_b and C_p. The critical case must be that C_p and C_b intersect in either p_{tang0} or p. Assume p_{tang0} and p is the intersection point separately, it's easy to get the corresponding values of B_v. The bigger one is exactly the answer.

5 Conclusion

The interception problem is a basic but important one. We present a thorough view of it in analytical way, which leads to a stronger adversarial skill.

TsinghuAeolus have showed amazing passing skill in Fukuka. The crucial component responsible for that includes the creation and evaluation of pass options. It's the accurate analysis of this problem that leads to our advantage.

An integral advice-taking system for an advised agent involves several steps: receiving, parsing and taking. The scheme mechanism addresses the last step. It can be attached to an existing complex agent without too much work. And it forces few requirements on the existing agent's framework. We also demonstrated its effectiveness. With this mechanism as the basis, we're working on agents' adaptability in the hope of improving the on-line team's behavior.

References

1. Peter Stone. Layered Learning in Multi-Agent Systems. PhD Thesis, Computer Science Dep.,Carnegie Mellon University, December 1998
2. Peter Stone and David McAllester. An architecture for action selection in Robotic Soccer. In Fifth International Conference on Autonomous Agents(Agents'2001)
3. Paul Carpenter, Patrick Riley, Manuela Veloso and Gal Kaminka. Integration of Advice in an Action-Selection Architecture
4. Sebastian Buck and Martin Riedmiller. Learning Situation Dependent Success Rates of Action in A RoboCup Scenario. In PRICAI 2000, Melbourne, published in Lecture Notes in Artificial Intelligence, Springer 2000, p809
5. Martin Riedmiller and Artur Merke. Using machine learning techniques in complex multi-agent domains. In I. Stamatescu, W. Menzel, M. Richter and U. Ratsch (eds.), Perspectives on Adaptivity and Learning (2002), LNCS, Springer
6. Remco de Boer and Jelle R. Kok. The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team. Master's thesis, University of Amsterdam, The Netherlands, February 2002.
7. Jelle R. Kok, Remco de Boer, Nikos Vlassis, and F.C.A. Groen. UvA Trilearn 2002 team description. In G. Kaminka, P.U. Lima, and R. Rojas, editors, RoboCup 2002: Robot Soccer World Cup VI, page 549, Fukuoka, Japan, 2002. Springer-Verlag.
8. Nuno Lau and Luis Paulo Reis. FC Portugal 2001 Team Description: Flexible Teamwork and Configurable Strategy. RoboCup-2001: Robot Soccer World Cup V, Andreas Birk, Silvia Coradeshi, Satoshi Tadokoro editors, LNAI, Springer Verlag, 2002
9. Richard Maclin and Jude W.Shavlik. Incorporating Advice into Agents that Learn from Reinforcements. Proceeding of the Twelfth National Conference on Artificial Intelligence.
10. Benjamin N. Grosof. Conflict handling in advice taking and instruction. Technical report, IBM Research Report 20123, 1995.
11. Sutton, R.S., Precup, D., Singh, S. Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. Artificial Intelligence 112:181-211. 1999
12. Jinyi Yao, Jiang Chen, Yunpeng Cai and Shi Li. Architecture of TsinghuAeolus. RoboCup-2002: Robot Soccer World Cup V, Andreas Birk, Silvia Coradeshi, Satoshi Tadokoro editors, LNAI, Springer Verlag, 2002
13. Jinyi Yao, Jiang Chen, and Zengqi Sun. An application in RoboCup combining Q-learning with Adversarial Planning . In The 4th World Congress on Intelligent Control and Automation(WCICA'2002).
14. TsinghuAeolus2001 Souce Code. Available in http://www.lits.tsinghua.edu.cn/RoboCup.
15. Noda et al, RoboCup Soccer Server Users Manual, RoboCup Federation, June 2001.