Knowledge Evolution in a Dynamic Environment of RoboCup Simulation

T. Nakashima, H. Ishibuchi
Graduate School of Engineering
Osaka Prefecture University
1-1 Gakuencho, Sakai, Osaka 599-8531, Japan
{nakashi, hisaoi}@cs.osakafu-u.ac.jp

M. Nii
Department of Computer Engineering
University of Hyogo
2167 Shosha, Himeji, Hyogo 671-2201, Japan
nii@eng.u-hyogo.ac.jp

Abstract

In this paper we examine the performance of the evolutionary algorithm for a dynamic environment where the opponent team changes during the evolution of strategies. In the dynamic environment, the dash power of opponent players in ball intercept behavior is adjusted. We consider two adjustment modes of the dash power in our experiments. In one mode the dash power of the opponent players is gradually increased over generation of the evolutionary algorithm. The other mode monitors the performance of the evolved team strategies to increase or decrease the dash power of opponent players accordingly. We compare the evolution of team strategies between the two modes and discuss for the future extension to the current experimental settings. We also discuss the possibility of knowledge extraction from the obtained team strategies by the evolutionary algorithm.

1 Introduction

RoboCup soccer is a competition between soccer robots/agents. Its ultimate aim is to win against the human soccer champion team by the year 2050 [1]. Developing RoboCup teams typically involves solving the cooperation of multiple agents, the learning of adaptive behavior, and the problem of noisy data handling. Many approaches have been presented that try to tackle these problems, an example is the application of soft computing techniques.

In this paper we examine the performance of the evolutionary algorithm for a dynamic environment of RoboCup simulation where the opponent team changes during the evolution of strategies. In the dynamic environment, the dash power of opponent players in ball intercept behavior is adjusted. We consider two adjustment modes of the dash power in our experiments. In one mode the dash power of the opponent players is gradually increased over generation of the evolutionary algorithm. The other mode monitors the performance of the evolved team strategies to increase or decrease the dash power of opponent players accordingly. We compare the evolution of team strategies between the two modes and discuss for the future extension to the current experimental settings. We also discuss the possibility of knowledge extraction from the obtained team strategies by the evolutionary algorithm.

2 Team Setup

We use the following action rules to determine player's action

$$\begin{aligned} R_j:\ & \text{If Agent is in Area } A_j \text{ and} \\ & \quad \text{the nearest opponent is } B_j \\ & \text{then the action is } C_j, \quad j = 1, \ldots, N, \end{aligned} \tag{1}$$

where R_j is the rule index, A_j is the antecedent integer value, B_j is the antecedent linguistic value, C_j is the consequent action, and N is the number of action rules. In this paper we evolve action rule sets to find a competitive soccer team strategy.

The antecedent integer value $A_j, j = 1, \ldots, N$ refers to a subarea of the soccer field. We divide the soccer field into 48 subareas as in Fig. 1.

Each subarea is indicated by an integer value. The antecedent value A_j of the action rule R_j is hence an integer value in the interval $[1, 48]$. In this paper 12 actions are available for the consequent action C_j.

Note that each player has a set of action rules. Since there are 48 subareas in the soccer field and *near* and

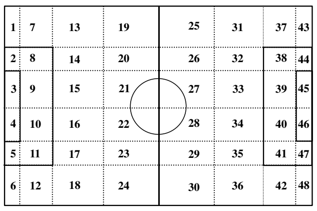

Figure 1: Soccer field

not near are available for the second antecedent part in action rules (i.e., B_j), the number of action rules for a single player is $48 \times 2 = 96$. There are $96 \times 10 = 960$ action rules in total for a single team with ten field players. Action rules for a goal keeper are not considered in this paper.

If the ball cannot be kicked by a player, there are two kinds of actions. One is ball intercept where the player move toward the ball in order to be able to kick it. The other action is positioning where the agent keep its position that is determined from the ball position and its home position. The player determines which action to take based on the positional relation among all the objects including teammates, opponent players, and the ball.

3 Evolutionary Computation

3.1 Encoding

The action of the agents is specified by the action rules in (1) when they keep the ball. Considering that the soccer field is divided into 48 subareas (see Fig. 1) and the position of the nearest opponent agent (i.e., it is *near* the agent or *not near*) is taken into account in the antecedent part of the action rules, we can see that there are $48 \times 2 = 96$ action rules for each player. We apply our evolutionary method to ten soccer agents excluding the goal keeper. Thus, the total number of action rules for a single team is $96 \times 10 = 960$. We use an integer string of length 960 to represent a rule set of action rules for ten players. The task of our proposed evolutionary method is then to evolve the integer strings of length 960 to obtain team strategies with high performance.

On the other hand, the actions of the agent in the case where the nearest opponent agent is not near the agent are shown in the other 48 integers. The value of each integer ranges from an integer interval of [1, 12] as the number of possible actions for each rule is twelve.

3.2 Dynamically Changing Opponent

In our previous studies on the evolutionary computation for RoboCup soccer, the opponent team was fixed during the course of the evolutionary algorithm. Obtained strategies by the evolutionary algorithm are therefore able to successfully perform against the fixed opponent team. However, it is not neccessarily said that the obtained strategies are successfully able to play the soccer game against different team strategies as successfully as against the fixed opponent team. In this paper we consider a dynamically changing environment in order to solve this problem.

In our implementation of the dynamically changing environment, dash power is used as a parameter of dynamically changing opponent strategies. The higher the dash power of opponent agents is, the harder it is for evolutionary teams to defeat the opponent team. In the beginning of the evolutionary algorithm the dash power of the opponent is zero and linearly increase as the number of generations increases.

3.3 Evolutionary Operation

We use one-point crossover, mutation, and ES-type selection as evolutionary operations in our evolutionary method. New integer strings are generated by crossover and mutation, and selection is used for generation update.

In the crossover operation, we first randomly select two integer strings. Then latter part of both strings is exchanged with each other from a randomly selected cut-point. Note that we do not consider any evaluation results when two integer strings for the crossover operation are selected from the current population. In the mutation operation, the value of each integer is replaced with a randomly specified integer value in the interval [1, 12] with a prespecified mutation probability. It is possible that the replaced value is the same as the one before the mutation operation. It should be noted that new integer strings generated by the crossover and the mutation operations do not have their match history. Thus the fitness evaluation of the new integer strings are made by using the game result of only a single game.

Generation update is performed by using ES-type selection in our method. We use a so-called $(\mu + \lambda)$-ES [3] for our generation update scheme. By iterating the crossover and the mutation operations we produce the same number of new integer strings as that of current strings. Then the best half integer strings from the merged set of the current and the new strings are chosen as the next population. The selection is based on the match results. Note that the current strings are also evaluated in this selection process. Thus, it is possible that a current integer string with the best performance at the previous generation update is not selected in the next generation update because the average goals of the integer string after the next performance evaluation may become lower if the result of the game at the next evaluation is poor.

To summarize, our proposed evolutionary method is written as follows:

[Procedure of the proposed evolutionary method]

Step 1. Initialization. A prespecified number of integer strings of length 960 are generated by randomly assigning an integer value from the interval [1, 12] for each integer.

Step 2. Generation of new integer strings. First randomly select two integer strings from the current population. Then the one-point crossover and the integer-change mutation operations are performed to generate new integer strings. This process is iterated until a prespecified number of new integer strings are generated.

Step 3. Performance evaluation. The performance of both the current integer strings and the new integer strings generated by Step 2 is evaluated through the results of soccer games. Note that the performance of current integer strings is also evaluated every generation because the game results are not constant but different game by game.

Step 4. Dynamic environmental change. The dash power of the opponent team is increased. The schedule of the change of the dash power is make so that at the first generation it is zero and becomes full (i.e., 100%) at the final generation.

Step 5. Generation update. From the merged set of the current integer strings and new ones, select best integer strings according to the performance evaluation. In the performance evaluation goals for are used as the first criterion. If multiple individuals have the same goals for, then goals against are used as the second performance criterion. The selected integer strings form the next generation.

Step 6. Termination of the procedure. If a prespecified termination condition is satisfied, stop the procedure. Otherwise go to Step 2.

4 Computer Simulations

The following parameter specifications were used for all the computer simulations in this paper:

The number of integer strings in a population: 5,
The probability of crossover: 1.0,
The probability of mutation for each integer: 5/96,
Generation update: 500.

The population size is specified as five. This is a small number comparing to commonly used parameter specifications. This is because it takes at least five minutes to complete a single soccer game. If the population size is specified large, it is difficult to perform the evolutionary method for a large number of generations. Currently we use a 16-node cluster system for the computational experiments in this paper. It still takes several days to perform a single run of the evolutionary process. The population size will be increased when more powerful computational environments are equippped.

We show the evolution of the soccer teams in Fig. 2. Total scores of goals for and goals against are plotted in Fig. 2. From this figure we can see that the offensive performance degrades as the evolution proceeds as the dash power of the opponent teams increases during the course of the evolution.

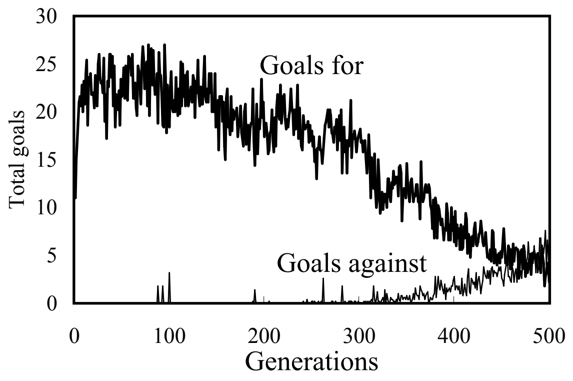

Figure 2: Performance of evolutionary teams.

Now we closely investigate obtained strategy by the evolutionary algorithm. Figure 3 shows the trajectory of the ball by an individual at the initial population. We can see from Fig. 3 that the ball is somewhat kicked randomly at different directions. This is because the initial individual was generated by randomly assigning an action to each action rule. Next we show in Fig. 4 the trajectory of the ball by an individual at the final population (i.e., at the 500-th generation). From

Fig. 4, we can see that the evolved team obtained offensive knowledge where the side forward drives the ball toward the opponent area from the side of the field and place the ball to the center when the ball is near the opponent goal. This strategy is intuitively understandable because human soccer teams normally take this strategy. It is shown that the evolutionary algorithm automatically obtained the human-like strategy from the initial population that was randomly generated.

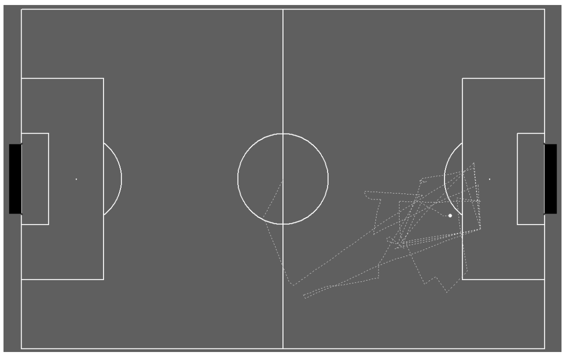
Figure 3: Ball trajectory at the first generation.

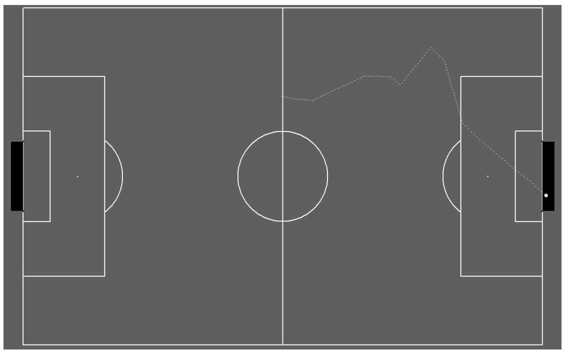
Figure 4: Ball trajectory at the final generation.

5 Conclusions

In this paper we proposed an evaluation method of soccer team strategies by using match history. The match history is used to calculate the average goals and the average goals against. Those teams with high average goals are evaluated as better than those with low average goals. The average goals against are used when the average goals are the same among more than one soccer team strategies. This method avoids the problem caused by uncertainty in the RoboCup soccer such as noise in object movement and the sensing information.

In the evolutionary process of this paper the action of soccer players that keep the ball is determined by a set of action rules. The antecedent part of the action rules includes the positions of the agent and its nearest opponent. The soccer field is divided into 48 subareas. The action of the agent is specified for each subareas. The candidate actions for the consequent part of the action rules form a set of 12 basic actions such as dribble and kick. The strategy of a soccer team is represented by an integer string of the consequent actions. In the evolutionary process, one-point crossover, replacement mutation, and ES-type generation update are used as evolutionary operators. The generation update is performed in a similar manner to the $(\mu + \lambda)$-ES of evolution strategy. That is, the best integer strings are selected from a merged set of current integer strings and new integer strings that are generated from the current integer strings by the crossover and mutation operations. The performance of the soccer team strategies becomes better over generation. For example, the average goals at the end of the evolution process is larger than in the initial population. We also observed that the average goals against did not increase as the evolutionary computation progressed.

In a series of computer simulations, we examined the performance of our evaluation method. We showed that the evolutionary algorithm automatically obtained a human-like strategy from random strategies through the process of evolutional trial-and error.

This paper focused on offensive strategy rather than defensive one. Developing a method for the defensive strategy is left for our future work.

Acknowledgment

This work was partially supported by Grant-in-Aid for Scientific Research on Priority Areas: KAKENHI (18049065) and KAKENHI (1770241).

References

[1] RoboCup official page, http://www.robocup.org/.

[2] T. Nakashima, M. Takatani, M. Udo, H. Ishibuchi, and M. Nii, “Performance Evaluation of an Evolutionary Method for RoboCup Soccer Strategies,” *RoboCup 2005: Robot Soccer World Cup IX*, in press.

[3] T. Bäck, Evolutionary Algorithms in Theory and Practice, Oxford University Press, New York, 1996.