Description of Team 'hana' – RoboCup 2004 Version –

Tomoharu Nakashima, Masahiro Takatani, Masayo Udo, and Hisao Ishibuchi

Department of Industrial Engineering, Osaka Prefecture University
Gakuen-cho 1-1, Sakai, Osaka, 599-8531
{nakashi, takatani, udo, hisaoi}@ie.osakafu-u.ac.jp

1 Introduction

Team 'hana' has been participating in the RoboCup world competition since 2002. Our first trial in 2002 ended in losing in all matches without marking any points. In the second trial in 2003, we survived the first elimination match before losing in the second elimination. We employed YowAI team and UvA Trilearn basic as our base team in 2002 and 2003, respectively. In this competition in 2004, we continue to employ UvA Trilearn basic [1]. We modified some skills such as dribble and pass. We also optimize the team strategy and the selection of heterogeneous players by using the framework of evolutionary computation.

In this team description paper, we first describe our base team. Next two evolutionary computation approaches are described. One is for evolving team strategy, and the other for determining the heterogeneous type of each player. Finally, we describe some future works that are left for the performance improvement.

2 UvA Trilearn Basic Code

UvA Trilearn [1] is the world champion team in the RoboCup 2003 that was held in Padova, Italy. The basic code includes low-level implementation of players such as world model. It also includes a variety of player skills such as dribble, pass, and clearing. The source code is written with detailed description, which can be used for generating html-based document by Doxygen software [2].

3 Evolutionary Computation for Team Srategy

In this section, we describe how the strategy of our team is developed by using evolutionary computation. The soccer field is partitioned into 48 subareas. An action of an agent for each subarea is determined by our evolutionary computation. Note that the actions determined by our evolutionary computation is only when the position of the ball is within the kickable area of players. That is, the behavior of players that currently have a ball is determined by our evolutionary computation. An action set that includes ten basic actions is given for each agent. The agent performs the action that is specified by the bit string when the ball is within the kickable area of the player.

3.1 Action Set for Players

The action set includes the following ten basic players' action:

1) Dribble paralell to X-axis toward the opponent side,
2) Dribble toward the opponent's goal,
3) Dribble forward by carefully avoiding the nearest opponent player,
4) Kick the ball to the nearest team player if its X-axis position is larger than the player. Otherwise clear the ball,
5) Kick the ball to the second nearest team player,
6) Clear the ball inside the field,
7) Clear the ball toward the nearest side line,
8) Kick the ball in front of the opponent's goal (centering),
9) Kick the ball in front of the nearest team player, and
10) Shoot the ball to the opponent's goal.

The above actions are hand-coded and no learning schemes are used. Options 6, 7, 8, and 9 use the functions that are given in the UvA trilearn basic. On the other hand, we implemented for the other options by hand-coding.

3.2 Field Partition

We partitioned the soccer field into 48 subareas as shown in Fig. 1. The task of our evolutionary computation is to select optimal actions for each subarea from the actions in the action set when agents have the ball within their kickable area.

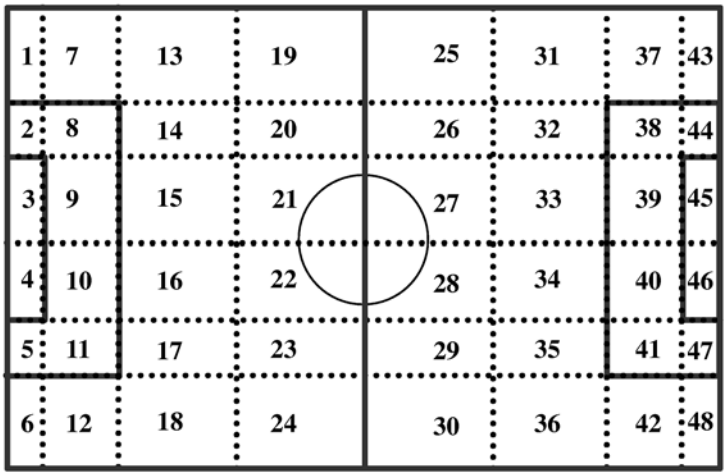

Fig. 1. Parition of the soccer field.

3.3 Genetic Coding

For representing the actions of each player, we use the bit string of the length 96 (Fig. 2). The first 48 bits represent the actions for corresponding subareas in Fig. 1 when the nearest opponent player is near the player. On the other hand, the actions when the nearest opponent player is far from the player are coded in the other 48 bits.

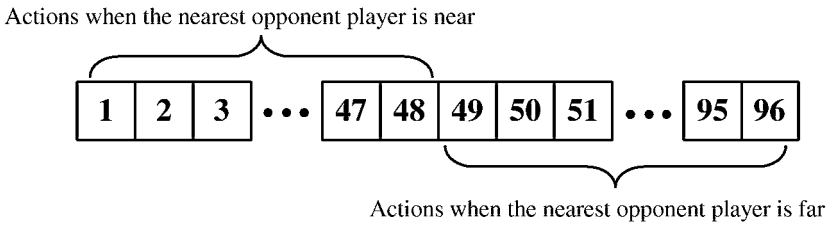

Fig. 2. A bit string for each player.

We optimize the actions of ten players excluding the goal keeper. Thus the total length of the bit string for representing a team is $96 \times 10 = 960$.

3.4 Generation Update

The problem of applying evolutionary computation to the RoboCup domain is that the time necessary for evaluating the performance of a team is long (i.e., ten minites). In order to tackle with this problem, we specify the number of bit strings in a population as one. The generation update is performed in a similar way of (1+1) strategy in evolutionary strategy. The only bit string in the population competes with its mutated version. The bit string that won the match becomes the individual of the next population. This process is iterated until some prespecified stopping condition is satisfied.

4 Evolutionary Computation for Heterogeneous Players

4.1 Rank-Based Weighted Sum Method

For determining the type of heterogeneous players, we propose a rank-based weighted sum method. Let us suppose that each player has a weight vector for nine parameters that are changeable by selecting different heterogeneous types. We first transform the values of each parameter into ranks so that the largest value among all the heterogeneous types has the highest rank and the lowest

rank is assigned to the smallest value. Each agent evaluates the value of each heterogeneous type by a weighted sum of the rank as follows:

$$V_i(j) = \sum_{k=1}^{9} W_{ik} \times R_{jk}, \quad i = 1, \ldots, 10, \quad j = 0, \ldots 6, \tag{1}$$

where i is the index of agent, j is the index of the heterogeneous type, $V_i(j)$ is the value of the j-th heterogeneous type for the i-th agent, W_{ik} is the weight of the i-th player for the k-th parameter, and R_{jk} is the rank of the k-th parameter of the j-th heterogeneous type.

The task of the evolutionary computation in this section is to optimize the weight W_{ik} for maximizing the competitive ability of the team.

4.2 Genetic Coding

We represent the weight W_{ik} as a bit string in our evolutionary computation. For simplicity, we only use integer values in the interval $[-3, 3]$ as a value in the bit strings. Since nine bits are necessary for representing the weights for nine heterogeneous parameters for each player, the total length of the bit string for ten players (excluding the goal keeper) is 90.

Generation update is performed in the same way as in the evolutionary computation for team strategy in Section 3.

5 Future works

The first thing to do in the evolutionary computation for both team strategy and heterogeneous players is the improvement of the generation update. It is clear that the number of the bit strings in a population should be larger. We need parallel computation architecture for solving this problem.

Secondly, diffensive actions should be improved. We only focused on the players action from the perspective of offensive actions this year.

The reconsideration of the member of the action set for team strategy is also included in our future work.

Acknowledgement

We would like to thank the development team of UvA Trilearn for kindly releasing their basic source code. Our team does not exist without it.

References

1. UvA Trilearn, URL at *http://carol.wins.uva.nl/ jellekok/robocup/indexen.html*
2. Doxygen, URL at *http://www.stack.nl/ dimitri/doxygen/index.html*