Behavior Generation for a Mobile Robot Based on the Adaptive Fitness Function

Eiji Uchibe, Masakazu Yanase, and Minoru Asada
Adaptive Machine Systems Graduate School of Engineering
Osaka University, Suita, Osaka 565-0871, Japan
e-mail: uchibe@er.ams.eng.osaka-u,ac.jp

Abstract. We have to prepare the evaluation (fitness) function to evaluate the performance of the robot when we apply the machine learning techniques to the robot application. In many cases, the fitness function is composed of several aspects. Simple implementation to cope with the multiple fitness function is a weighted summation. This paper presents an adaptive fitness function for the evolutionary computation to obtain the purposive behaviors through changing the weights for the fitness function. As an example task, a shooting behavior in a simplified soccer game is selected to show the validity of the proposed method. Simulation results and real experiments are shown, and a discussion is given.

1 Introduction

One of the ultimate goals of Robotics and AI is to realize autonomous robots that organize their own internal structure towards achieving their goals through interactions with dynamically changing environments. In applying some of evolutionary approaches to the robot in order to obtain purposive behaviors, the fitness (evaluation) function should be given in advance. There are two important issues when we attempt to design the fitness function.

First one is that the multiple fitness measures should be considered in order to evaluate the resultant performance. In case the desired behavior is simple one such as wall-following, the fitness function can be also simple. However, many fitness measures are considered when the desired behaviors become more complicated. One simple realization is to create the new scalar function based on the weighted summation of multiple fitness measures. However, this approach faces the essential problem of weighting itself, that is, how to decide the weight values. In order to cope with multiple objective problem, several methods are proposed [2], of which purpose is to obtain the Pareto optimal solutions. However, in robotic applications, it is sometimes meaningless to optimize one of the fitness measures. For example, one of the rational policy of obstacle avoidance in a static environment is not to move, which is not our intentional result.

Second one is the task complexity. If we set up the severe fitness function at the beginning of learning, the robot does not obtain the good evaluation [6, 7]. As a result, the robot can not accomplish the task. Therefore, we have to set up the appropriate fitness function according to the current ability of the robot when the robot can obtain the purposive behaviors in a finite learning time. In general, it seems difficult to

accomplish the complicated task from the beginning. Asada et al. proposed a paradigm called *Learning from Easy Mission* [1]. Yang and Asada [8] proposed Progressive Learning which would learn a motion to be learned from slow to fast and apply it to a peg insertion task. Omata [5] applied GAs to acquire the neural network controller which can drive a bicycle. The designer give an initial velocity to the bicycle so as to control it easily. After the generation proceeded, the assist was slightly decreased.

In this paper, we propose an adaptive fitness function in consideration of the change of the evaluation through the evolution. Based on the given priority, the robot modifies the fitness function to control the task complexity. In order to obtain the policy, we select a Genetic Programming (hereafter GP) method [4]. GP is a kind of genetic algorithms based on the tree structure with more abstracted node representation than gene coding in ordinary GAs. We apply the GP enhanced by the adaptive fitness function to a simplified soccer game. We show how the robot would acquire the purposive behaviors using the proposed method. Finally, the results of computer simulation, real experiments, and a discussion are given.

2 Adaptive Fitness Function

Suppose that n fitness measures f_i $(i = 1, \cdots n)$ are given to the robot. We utilize the standardized fitness representation, that has a positive value between 0 and 1. That is, the smaller is better (0.0 is the best). Then, we introduce a priority function pr, and define the priority of f_i as $pr(f_i) = i$. That is, f_1 is the most important measures that the robot has to consider. Combined fitness function is computed by

$$f_c = \sum_{i=1}^{n} w_i f_i, \tag{1}$$

where w_i denotes the weight for i-th evaluation. The robot updates w_i through the interaction with the environment.

In general, because the robot must consider a tradeoff among all the fitness measures, it seems hard to optimize all the fitness measures simultaneously. We focus on the change of each f_i and correlation matrix so as to modify the weights. Let the fitness measure of the individual j at the generation t be $f_i(j,t)$, and we consider the change of the fitness measures by

$$\Delta f_i(t) = \frac{1}{N} \sum_{j=1}^{N} \{f_i(j,t) - f_i(j,t-1)\}, \tag{2}$$

where N is the number of population. In case of $\Delta f_i > 0$, i-th fitness measure is not improved under the current fitness function. Therefore, the influence of the corresponding weight w_i should be changed. However, other fitness measures f_j $(j = i+1, \cdots n)$ should be also considered since they would be related to each other.

Let $\boldsymbol{C}_i$ be the set of fitness measures which is related to the i-th measure,

$$\boldsymbol{C}_i = \{j \mid |r_{ij}| > \varepsilon, j = i+1, \cdots, n\}, \tag{3}$$

where ε is a threshold between 0 and 1, and r_{ij} is a correlation between f_i and f_j. In case of $\boldsymbol{C}_i = \phi$, w_i can be modified independently because i-th fitness measure does not correlate with other measures. Therefore, w_i is increased so that the i-th fitness measure would be emphasized.

In case of $\boldsymbol{C}_i \neq \phi$, w_{j^*} is updated, where j^* is prior to other evaluation measures in $\boldsymbol{C}_i$, that is,

$$j^* = \arg\max_{j \in \boldsymbol{C}_i} pr(f_i).$$

The reason why w_i is not changed explicitly is that the weight of the upper fitness measure would continue to be emphasized even if the corresponding fitness measure is saturated. As a result, the lower fitness measure related to the upper one is emphasized directly. The update value $\Delta w_{j^*}(t)$ is computed by

$$\Delta w_{j^*}(t) = \begin{cases} 1 & (r_{ij^*} > \varepsilon) \\ -1 & (r_{ij^*} < -\varepsilon) \end{cases} . \tag{4}$$

It is possible for the weight corresponding to improved measures to change for the worse. However, it would be rather unimportant measures because of the given priority. Finally, we summarize the method to modify the fitness function as follows:

1. For $i = 1, \cdots, n$, update the weights as follows:
   A. In case of $\boldsymbol{C} = \phi$, update the i-th weight by $w_i(t+1) = w_i(t) + \alpha \Delta w_i$.
   B. In case of $\boldsymbol{C} \neq \phi$, update the j^*-th weight by Eq.(4).
2. Create the next population, and increment the generation by $t \rightarrow t+1$.

In this study α is equal to 0.02.

3 Task and Assumption

3.1 Environment and Robots

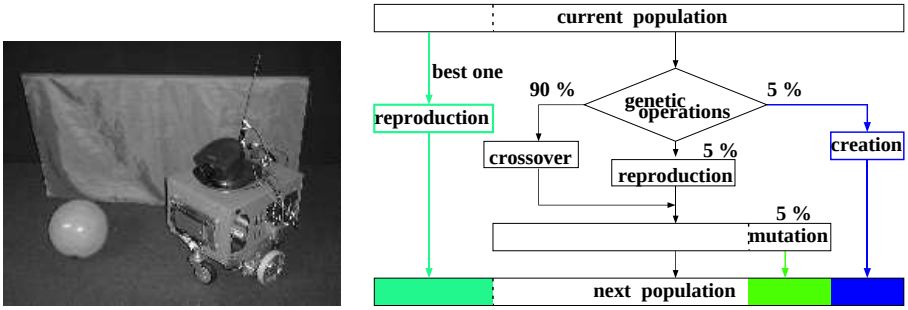
We have selected a simplified soccer game consisting of two mobile robots as a testbed. RoboCup [3] has been increasingly attracting many researchers. The task for the learner is to shoot a ball into the opponent goal without collisions with an opponent. At the beginning, the behavior is obtained in computer simulation, and we transfer the result of simulation to the real robot. Figure 1 (a) shows an our mobile robot, a ball, and a goal.

The environment consists of a ball, two goals, and two robots. The sizes of the ball, the goals and the field are the same as those of the middle-size real robot league of RoboCup Initiative. The robots have the same body (power wheeled steering system) and the same sensor (on-board TV camera). As motor commands, each mobile robot has a 2 DOFs.

3.2 GP Implementation

Each individual has two GP trees, which control the left and right wheel, respectively. A GP learns to obtain mapping function from the image features to the motor command. Then, we select the terminals as the center position in the image plane. For example, in a case of the ball, the current center position $(x_b(t), y_b(t))$ and the previous one $(x_b(t-1), y_b(t-1))$ are considered. Because the objects that the GP robot can observe are the ball, two goals and an opponent, the number of the terminals is 4(objects) $\times$ 4(features) $= 16$. As a function set, we prepare four operators such as $+$, $-$, $\times$ and $/$.

Figure 1 (b) shows a flowchart to create a new generation. The best performing tree in the current generation will survive in the next generation. The size of the population



(a) our mobile robot (b) flowchart of GP

Figure 1: GP Implementation

Table 1: Fitness measures and the priorities

		case A	case B	case C	case D
f_{opp}	the total number of achieved goals	1	1	1	1
f_{own}	the total number of lost goals	2	2	6	2
f_{ov}	the overlapping degree of the ball and the opponent goal in the image plane	3	5	3	3
f_{kick}	the total number of ball-kicking	4	3	4	6
f_c	the total number of collisions	5	4	5	5
f_{step}	the total number of steps until all trials end	6	6	2	4

is set to 150. In order to select parents for crossover, we use tournament selection with size 10. The maximum depths by crossing two trees is 25. We perform 30 games to evaluate each robot. The number of generations for which the evolutionary process should run is 200. One trial is terminated if the robot shoots the ball into the goal or the pre-specified time interval expires. ε is set to 0.5. We perform 10 experiments according to the variety of initial values of weights.

3.3 Fitness Measures

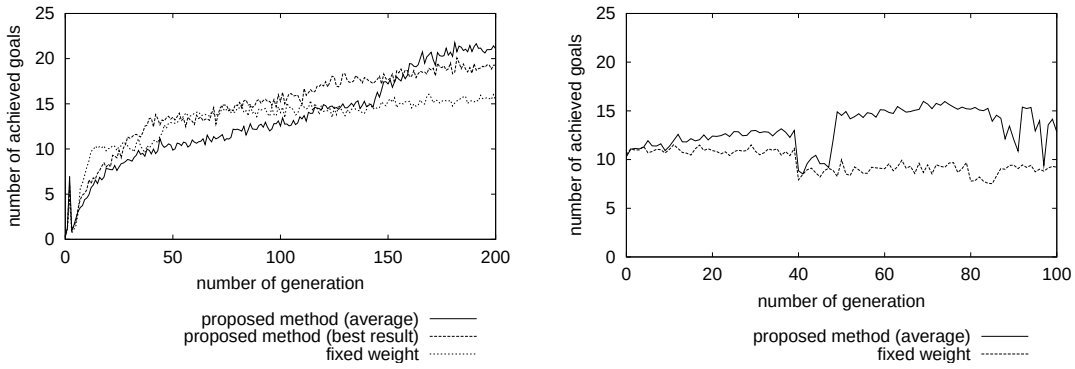
One of the most important issues is to design the fitness measures. In this experiment, we set up six fitness measures described in Table 1. Because it seems difficult to give the optimal priority function, we prepare four priorities (case A, B, C, and D).

The initial weights for the six fitness measures are set as follows: $f_{own} = f_{opp} = 9.0$, $f_{kick} = 8.0$, $f_c = 4.0$, $f_{step} = f_{ov} = 2.0$. They are the best values in our previous experiments using the fixed fitness function. The policy to design the priority is explained as follows. Since the main purpose for this robot is to shoot a ball into the goal, we assume that the most important measure is the number of achieved goals.

4 Experimental Results

4.1 Comparison between the Proposed Method and the Fixed Weight Method

At first, we perform a simulation using a stationary opponent. This experiment can be regarded as an easy situation. We compare the proposed method with the fixed weight



(a) with the stationary opponent (b) with the moving opponent

Figure 2: Average of the number of achieved goals in the first experiment

method. Figure 2 (a) shows the result when the opponent is stationary. In a case of the fixed weight method, the performance is not improved after the 50th generations. On the other hand, the performance based on the proposed method is improved gradually.

Next, we show a simulation results using an active opponent. This experiment can be regarded as a more difficult situation. As an initial population for this experiment, we used the best population which was obtained based on the proposed method described above. Figure 2 (b) shows a histories of f_{opp}. In this experiment, although the opponent just chases the ball, the speed can be controlled by the human designer. Its speed was gradually increased at the 40th and 80th generations, respectively.

According to the increase of the speed of the opponent, the obtained scores f_{opp} was slightly decreased in a case of the fixed weight method. On the other hand, the robot using the proposed method kept the performance same in spite of the increase of the speed.

We checked the obtained behaviors based on both methods, and it was found the following issues: With respect to f_{opp} and f_{own}, both methods achieved the almost same performances. In cases of the number of collisions (f_c) and the steps (f_{step}), the performance of the proposed method is better than that of the fixed weight method. We suppose the reason why the robot would acquire such behaviors as follows. At the beginning of the evolution, the most important thing is to kick the ball towards the opponent goal even if it makes a collision with the opponent. Therefore, the weights for f_c and f_{step} are set to small values in a case of the fixed weight method. After a number of generation, the weight for f_{opp} affected the differences of fitness among individuals because most individuals accomplished the shooting behavior. Consequently, the robot using the fixed weight method did not consider f_c and f_{step} through the whole generations. On the other hand, the robot based on the proposed method changed the weight for f_{step} to be considered. As a result, the robot using the proposed method obtained the shooting behavior more quickly.

4.2 Comparison among the Different Priority Function

Next, we checked how the priority affects the acquired behaviors when we change the order of the priority because the priority of fitness measures must be given to the robot in advance. For the sake of the limitation of the space, we show the results of

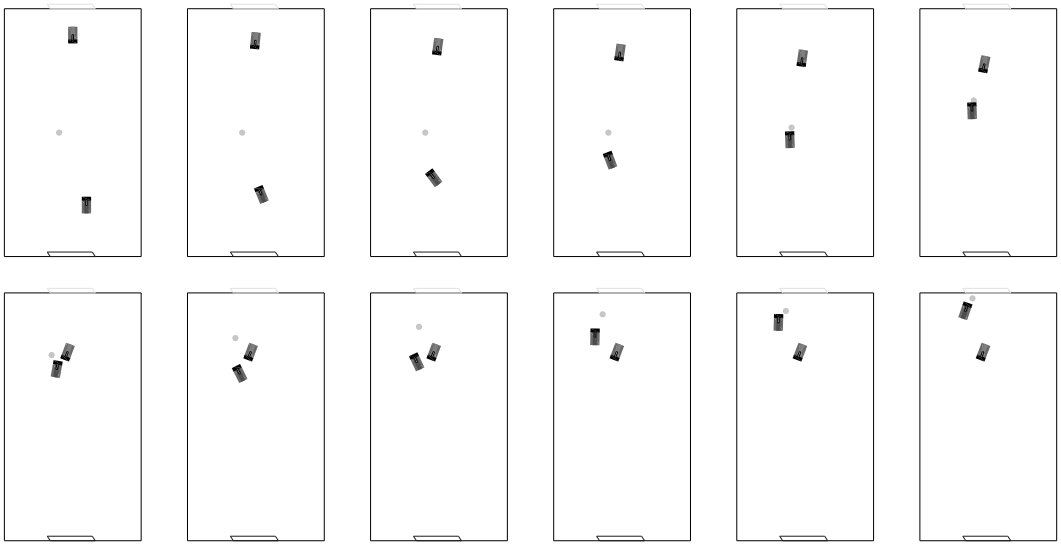
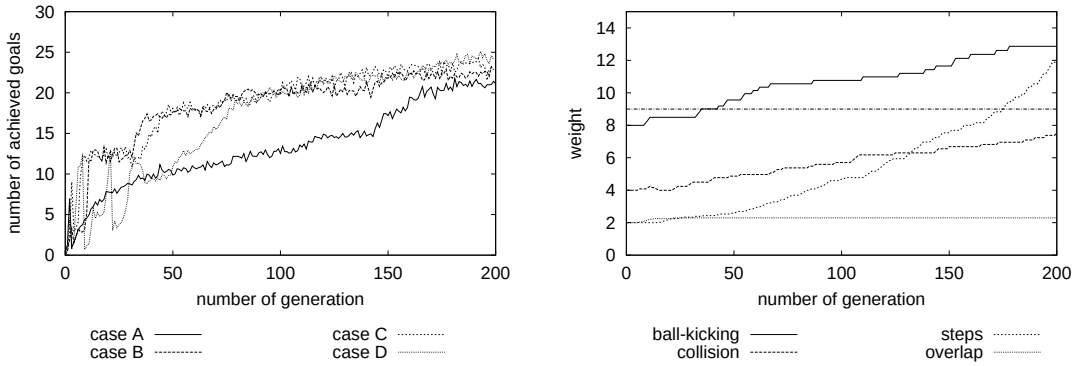


Figure 3: Typical shooting and avoiding behavior in computer simulation



(a) achieved goals (b) weight based on case C

Figure 4: Comparison among four priority functions

four fitness measures in Figure 4 using the four priorities described in Table 1. From Figure 4 (a), although the learning curves are different among four cases, the final values converged to the almost same value.

Figure 4 (b) shows the weights of case C during the evolution. Since the weights for the number of achieved and lost goals are constant, only one line are shown in this figure. It follows from the initial weights described in section 3.3, the important order of fitness measures are described as follows:

$$\text{achieved goals} = \text{lost goals} > \text{ball-kicking} > \text{collisions} > \text{step} = \text{overlap}.$$

Using the priority function in Table 1, the resultant order are described as follows:

$$\text{ball-kicking} > \text{step} > \text{achieved goals} = \text{lost goals} > \text{collisions} > \text{overlap}.$$

In many cases we can see that settlement. That is, ball-kicking is emphasized through the evolution. On the other hand, it was not important for the robot to consider the measure about the overlapping degree f_{ov} directly.

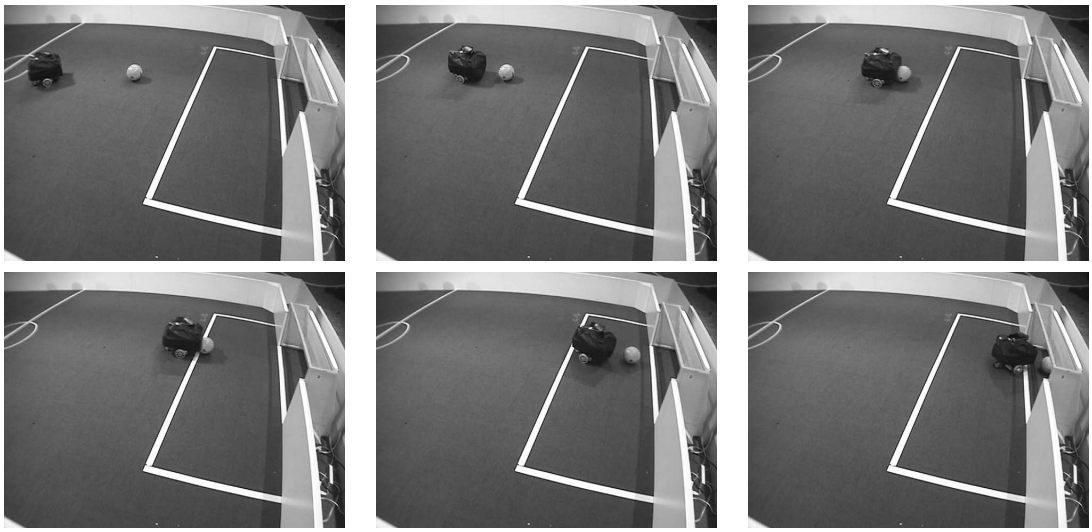


Figure 5: Typical shooting behavior in the real environment

In this experiment, the resultant performance mainly depended on the number of individuals, that was too exiguous for this problem. Although simple extension is to increase the number of individuals, there are alternative approaches. One is to apply a kind of intelligent genetic operations to our problems. We used an standard genetic operation such as crossover and mutation, GP sometimes failed to search the feasible solutions appropriately.

4.3 Real Experiments

We transfer the obtained policy to the real robot. A simple color image processing (Hitachi IP5000) is applied to detect the objects in the image plane in real time (every 33 [ms]). The robot has a TV camera of which visual angles are 35 [deg] and 30 [deg] in horizontal and vertical directions, respectively.

Figure 5 shows a preliminary result of the experiments, that is, one sequence of images where the robot accomplished the shooting behavior. As compared with the behaviors based on our previous methods [1, 7], obtained behavior seems very smooth because of mapping from the continuous sensor space to the continuous action space. This robot participated in the competition of RoboCup 99 which was held in Stockholm. Currently, we perform one-to-one competition in the real environment and check the validity of the proposed method.

5 Conclusion

This paper presented a method of the adaptive fitness function based on the changes of fitness through the evolution. In consideration of the correlation between multiple fitness measures, the weights for the combined fitness function are updated. We applied the adaptive fitness function method to the simplified soccer game, and showed the validity of the proposed method. The processes of evolution among different priority is slightly different, but resultant performance are almost same in this experiment.

As a future work, we hope to challenge simultaneous evolution of multiple robots,

that is co-evolution. We have already reported how the multiple robots could obtain the cooperative behaviors based on GP with the fixed fitness function [7]. Now, we are planning to implement the proposed method to this task.

Acknowledgments

This research was supported by the Japan Society for the Promotion of Science, in Research for the Future Program titled Cooperative Distributed Vision for Dynamic Three Dimensional Scene Understanding (JSPS-RFTF96P00501).

References

[1] M. Asada, S. Noda, S. Tawaratumida, and K. Hosoda. Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning. *Machine Learning*, 23:279–303, 1996.

[2] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[3] H. Kitano, ed. *RoboCup-97 : Robot Soccer World Cup I.* Springer Verlag, 1997.

[4] J. R. Koza. *Genetic Programming I : On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[5] T. Omata. Learning with Assistance based on Evolutionary Computation. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 2180–2186, 1998.

[6] E. Uchibe, M. Asada, and K. Hosoda. Environmental Complexity Control for Vision-Based Learning Mobile Robot. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 1865–1870, 1998.

[7] E. Uchibe, M. Nakamura, and M. Asada. Cooperative and Competitive Behavior Acquisition for Mobile Robots through Co-evolution. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 1406–1413, 1999.

[8] B.-H. Yang and H. Asada. Progressive Learning for Robotic Assembly: Learning Impedance with an Excitation Scheduling Method. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 2538–2544, 1995.