Evolution for Behavior Selection Accelerated by Activation/Termination Constraints

Eiji Uchibe **Masakazu Yanase** **Minoru Asada**
Dept. of Adaptive Machine Systems
Graduate School of Engineering
Osaka University

Abstract

This paper proposes an evolutionary approach to select appropriate behaviors for a mobile robot. We augment each behavior by adding activation/termination constraints to accelerate the evolutionary processes. The former constraints reduce the number of situations where each behavior is executable, and the latter contribute to extract meaningful behavior sequences, each of which can be regarded as one action regardless of its length. We apply the genetic algorithm to obtain the switching function to select the appropriate behavior according to the situation. As an example, a shooting task in a soccer game is given to show the validity of the proposed method. Based on the combination of the proposed architecture and GA, we can obtain the purposive behaviors. Simulation results are shown, and a discussion is given.

1 Introduction

Machine learning techniques such as reinforcement learning [8] and genetic algorithm [4] are promising to obtain purposive behaviors for autonomous robots in complicated environments. Many learning and evolutionary techniques can obtain purposive behaviors such as wall-following [2, 6], shooting a ball into the goal [1], and so on. However, if the robot has no *a priori* knowledge to obtain the complicated behaviors, it takes enormous time. Consequently, the resultant behavior seems trivial in spite of the long learning time. That is, a direct mapping from sensory inputs to motor commands is not tractable.

In order to obtain the feasible solution in the realistic learning time, a layer architecture is often introduced to cope with large scaled problems [7]. Figure 1 shows an example of layered architecture for behavior selection. In this approach, the upper layer learns the switching function to select the suitable behavior already designed or learned. Because the designed behaviors can generate purposive action under the limited situations, they can help the evolutionary computation to search the feasible solutions.

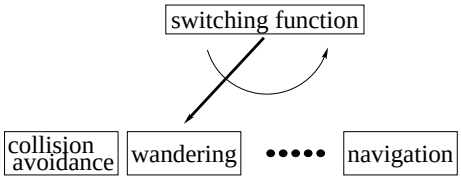
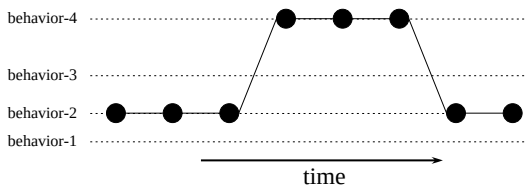


Figure 1: A layer architecture for behavior selection

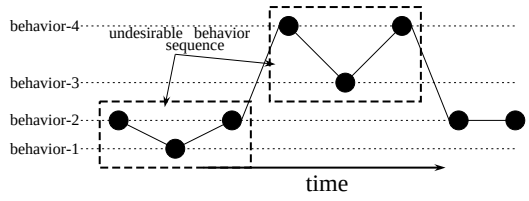
In this approach, we face with the following problems:

1. how to coordinate and switch the behaviors,
2. when to select the behaviors, and
3. when to terminate the currently executing behavior.

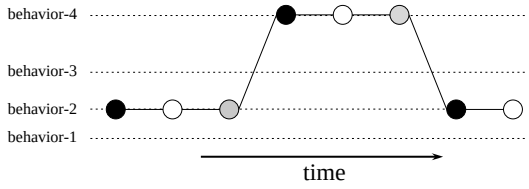
Uchibe *et al.* [9] applied genetic programming to solve the above three problems in the robotic soccer domain. However, the resultant decision tree is not represented in a compact style. In their case, the robot selected the collision avoidance although there were no obstacles near the robot. In addition, the robot did not use the given shooting behavior when it is suitable to be activated. There are two major reasons why a layered approach could not obtain the appropriate behavior sequences: (1) GP does not take account of the precondition of the given behavior explicitly, and (2) the behavior is often switched although the goal of the behavior is not achieved. The first reason prevents GP



(a) ideal behavior sequence



(b) resultant behavior sequence



(c) behavior sequence using a termination constraint

Figure 2: A timing to select the next behavior

from reducing the learning time, and the second causes the scraps of behavior sequence.

An example sequence of the selected behaviors is shown in Figure 2, where the black circle indicates the timing to change the behavior to another one. Figure 2 (a) indicates the ideal behavior sequence, whereas Figure 2 (b) the resultant behavior sequence often obtained by the learning or evolutionary approaches. In a case (a), the robot selects the same behavior for a while. On the other hand, in a case (b), the robot switches the several behaviors according to the situation. Although this can be regarded as an acquisition of the new behavior sequences instead of the given behavior, it causes enormous learning time because the robot does not make good use of the behavior given by the designer.

In order to take advantage of the behavior given by the designer, we have to consider the the precondition and the goal of the behavior. This paper proposes a behavior selection mechanism with *activation/termination constraints*. The former constraints reduce the number of situations where each behavior is executable, and the latter contribute to extract meaningful behavior sequences. We call behavior with activation/termination constraints *module*. These constraints enable the robot to modify the timing to select the next behavior as shown in Figure 2 (c), where the gray circle indicate a vague state whether a new behavior is changed or not. From the case (c), the termination constraint contributes to compression of the behavior sequence. That is, it enables us to deal with heterogeneous modules in the same manner. In this paper, "heterogeneous" means the differences of time to achieve the goal of the module. Once the module is selected, actions are executed until the module terminates stochastically based on the termination constraint. Thus, we can obtain the behavior sequence like Figure 2 (c).

The lower layer consists of multiple modules, while the upper layer selects the appropriate module according to the situation. Genetic algorithm is applied to obtain the switching function to select the appropriate module and the timing to terminate it according to the situation. Activation/termination constraints affect not the genetic operations such as crossover and mutation but the individual representation. Although we utilize standard genetic operations, we can obtain purposive behaviors owing to the activation/termination constraints. The results of computer simulation are shown, and a discussion is given.

2 Behavior Selection with Activation/Termination Constraints

2.1 Lower layer

Suppose that the robot has L modules m_i ($i = 1, \cdots, L$). A module m_i consists of three components: a *behavior* $\pi_i : \boldsymbol{X}$ (state space) $\rightarrow \boldsymbol{U}$ (action space), a *termination constraint* T_i and an *activation constraint* I_i. There are several ways to implement the behaviors, but they must be given to the robot in advance.

The activation constraint I_i gives a set of states where the module should be executable.

$$I_i(\boldsymbol{x}) = \begin{cases} 1 & m_i \text{ is executable at state } \boldsymbol{x}, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

If the designer gives the behavior π to the robot, it is not difficult to give the precondition of the behavior. For example, the collision avoidance behavior is implemented for the mobile robot with sonars, where the

designed behavior is activated only when the obstacles are detected by sonar.

Each module has one termination constraint T_i consisting of a probability to sustain the module or not. In other words, this function gives the time to continue to execute the selected module.

$$T_i(\boldsymbol{x}, t) = \begin{cases} 0 & \text{the goal of the module} \\ & m_i \text{ is achieved,} \\ 0 & t > t_{p,i}, \\ p_i & \text{otherwise,} \end{cases} \quad (2)$$

where t and $t_{p,i}$ denote the cumulative steps and the pre-specified time interval, respectively. If the robot continues to execute the same module $t_{p,i}$ times, it is forced to stop. The robot judges whether the selected module should be terminated or sustained with probability p_i.

2.2 Upper layer

In the upper layer, the modules are switched according to the current situation. Let the value of the module m_i at the state $\boldsymbol{x}$ be $V_i(\boldsymbol{x})$. The robot selects the module of which value is the highest:

$$i^* = \arg \max_{i=1,\cdots,n} V_i(\boldsymbol{x}) I_i(\boldsymbol{x}). \quad (3)$$

Once the module is selected, then actions are executed according to the current behavior π_i until the module terminates stochastically based on T_i.

In order to approximate $V_i(\boldsymbol{x})$, we use the function expressed by

$$V_i(\boldsymbol{x}) = \sum_{j=1}^{N} \exp\left(-(\boldsymbol{x} - \boldsymbol{c}_{ij})^T \boldsymbol{W}_{ij} (\boldsymbol{x} - \boldsymbol{c}_{ij})\right), \quad (4)$$

where $\boldsymbol{c}_{ij} \in \Re^n$ and $\boldsymbol{W}_{ij} \in \Re^{n \times n}$ denote the center position and the symmetric matrix. If $\boldsymbol{W}_{ij}$ is positive definite, Eq.(4) express the Gaussian function.

3 Genetic Operations

In order to obtain the appropriate p_i, $\boldsymbol{c}_{ij}$ and $\boldsymbol{W}_{ij}$, we use the genetic algorithms. In GA, it is an important role to design the genetic operations such as crossover and mutation. A procedure to generate the new offspring is indicated in Figure 3.

Suppose that the robot has L modules, and each module has N parameters (p_i, $\boldsymbol{c}_{ij}$, $\boldsymbol{W}_{ij}$). Figure 4 (a) shows the chromosome of individual. We perform two types of crossover.

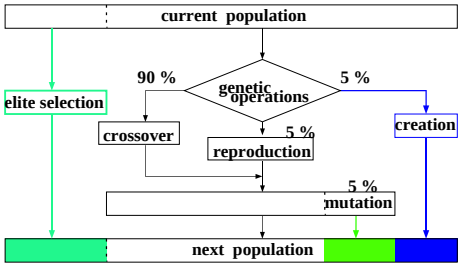


Figure 3: Flowchart of GA

Global crossover : Figure 4 (b) shows a basic idea of global crossover. For each module m_i, a pair of parameters is selected randomly from each parent. Then, we swap two parameters.

Local crossover : At the beginning, we find the parameters of which distance is minimum.

$$(j^*, k^*) = \min_{j,k=1,\cdots N} ||\boldsymbol{c}_{ij}^1 - \boldsymbol{c}_{ik}^2||.$$

1. In case of $\boldsymbol{W}_{ij}$, we perform two point crossover.
2. In case of p_i and $\boldsymbol{c}_{ij}$, we utilize BLX-α [3] based on real-coded GA. Figure 4 (c) shows a basic idea of BLX-α in a case of two dimensional vector. The BLX-α uniformly picks parameter values from points that lie on an interval that extends αI on either side of the interval I between parents. In other words, it randomly generates two children around their two parents by using uniform distribution in the hyper rectangular region whose sides are parallel to axes of the coordinate system.

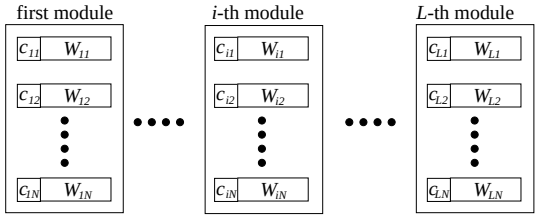
Mutation : One of the elements of $\boldsymbol{c}_{ij}$ or $\boldsymbol{W}_{ij}$ is replaced to a new random value.

Genetic operations used here does not take account of activation/termination constraints explicitly. In other words, activation/termination constraints do not help GA to search the feasible solutions directly.

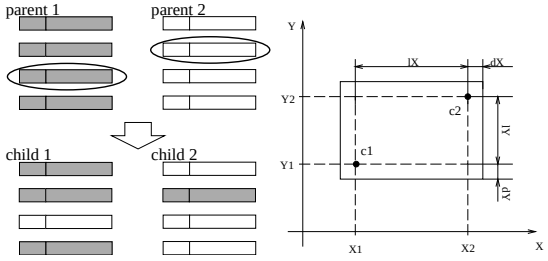
4 Task and Assumptions

4.1 Robot and Environment

We have selected a simplified soccer game as a testbed. The task for the learner is to shoot a ball into the opponent goal. The environment consists of a ball and two goals, and a wall is placed around the field except the goals. 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 [5] that many AI and robotics researchers have been involved in.



(a) chromosome for one module



(b) global crossover (c) local crossover

Figure 4: Crossover

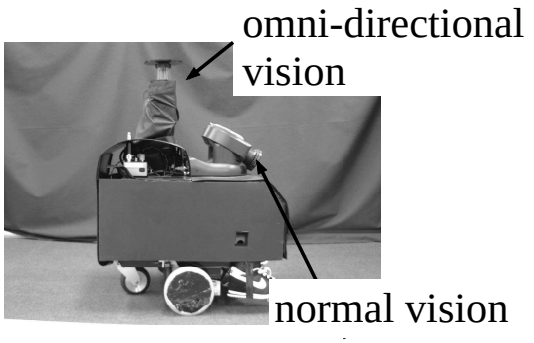
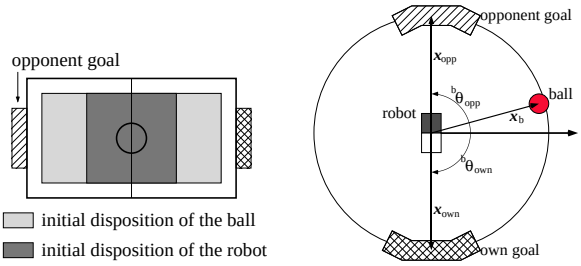


Figure 5: Our mobile robot



(a) top view of the field (b) view captured by an omni-directional vision

Figure 6: The experimental setting.

Figure 5 shows the real robot used for modeling. The robot moves around the field based on the power wheeled steering system. As motor commands, our mobile robot has two degrees of freedom. The input $\boldsymbol{u}$ is defined as a two dimensional vector:

$$\boldsymbol{u}^T = \left[\begin{array}{cc} u_l & u_r \end{array} \right] \quad u_l, u_r \in \{-1,\ 1\},$$

where u_l and u_r are the velocities of the left and right wheels, respectively. In addition, the robot has a kicking device to kick the ball.

The robot has two vision systems; one is a normal vision system to capture the front view, and the other is an omni-directional one to capture the visual information whole around the robot. The omni-directional vision has a good feature of higher resolution in direction from the robot to the object although the distance resolution is poor.

The robot observes the center positions of the ball and two goals in the image plane using two vision systems. Therefore, the number of image features is 12. A simple color image processing is applied to detect the ball and the goal area in the image plane in real-time (every 33 [msec]). Figure 6 (b) shows detected image features to extract the information of the environment based on the omni-directional vision, where $\boldsymbol{x}_b$, $\boldsymbol{x}_{own}$ and $\boldsymbol{x}_{opp}$ are the center position of the ball, the own goal, and the opponent goal, respectively.

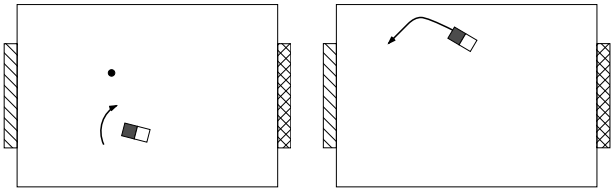
4.2 Module Design

4.2.1 Basic Modules

We prepare five basic modules of which behavior just generates a simple action regardless of sensory information. That is, the motor command generated by each basic module is described as follows:

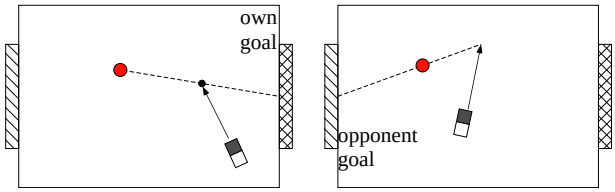
- m_1 : go forward
  $\boldsymbol{u}^T = [1.0,\ 1.0]$
- m_2 : go backward
  $\boldsymbol{u}^T = [-1.0,\ -1.0]$
- m_3 : stop
  $\boldsymbol{u}^T = [0.0,\ 0.0]$
- m_4 : turn left
  $\boldsymbol{u}^T = [-1.0,\ 1.0]$
- m_5 : turn right
  $\boldsymbol{u}^T = [1.0,\ -1.0]$

These modules are always executable, in other words, for all $\boldsymbol{x}$ we set $I_i(\boldsymbol{x}) = 1$ $(i = 1, \cdots, 5)$. Since they have no explicit purpose to achieve, the termination



(a) m_6 (b) m_7

Figure 7: Typical behaviors generated by the reactive modules



(a) m_{10} (b) m_{11}

Figure 8: Typical behaviors generated by the complex modules

constraints only depends on the steps. In this experiment, we set the termination parameters in Eq.(2) as $p = 0.4$ and $t_p = 150$ steps ($= 5$ [sec]).

4.2.2 Reactive Modules

Figure 7 shows typical behaviors generated by the four prepared modules. In order to realize the defending behavior, we design the following four reactive modules.

- m_6 : search the ball
  The purpose of this module is to capture the ball image using the normal vision. Therefore, the robot searches the ball by turning to left or right. T_6 is set to zero when the ball is observed.
- m_7 : avoid collisions
  The purpose of this module is to avoid collisions with the wall. If the wall is not detected, $\boldsymbol{u}^T = [1.0,\ 1.0]$. This module is activated when the robot moves around near the wall.
- m_8 : kick the ball
  In case the ball is in front of the robot and this module is selected, the robot succeeds in kicking the ball. Of course, this module has no effects when the ball is not in front of the robot. This module is activated when the ball image is captured by the normal camera.
- m_9 : shoot the ball
  The purpose of this module is to push the ball into the opponent goal. This module is activated when both the ball and the opponent goal images are captured by the normal camera. The resultant behavior is the same as that of m_1, that is, $\boldsymbol{u}^T = [1.0,\ 1.0]$. This module does not always succeed in shooting behaviors, especially when the ball position is shifted from the goal direction.

In this experiment, we set the termination parameters in Eq.(2) as $p = 0.8$ and $t_p = 150$ for the above four modules.

4.2.3 Complex Modules

We prepare a controller which makes the features on the image plane converge to the desired values. For the desired state $\boldsymbol{x}_d = [x_d\, y_d]^T$, a motor command $\boldsymbol{u}$ is computed by

$$\boldsymbol{u} = \begin{bmatrix} u_r \\ u_l \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \end{bmatrix}, \qquad (5)$$

where u_r and u_l are the velocity of the right and left wheels, respectively. $\boldsymbol{K}$ is a gain matrix. Using the controller based on Eq. (5), we prepare the following two modules.

- m_{10} : move to the defensive position
  The purpose of this module is to move to the place between the ball and the own goal. The desired state $\boldsymbol{x}_d$ is given by
  $$\boldsymbol{x}_d \quad = \quad (1-a)\boldsymbol{x}_b + a\boldsymbol{x}_{own}.$$
- m_{11} : move to the offensive position
  The purpose of this module is to move to the opposite side of the opponent goal to shoot. The desired state $\boldsymbol{x}_d$ is given by
  $$\boldsymbol{x}_d \quad = \quad (b+1)\boldsymbol{x}_b - b\boldsymbol{x}_{opp} \quad (0 \le b \le 1).$$

These two modules can be executed when the desired state is not achieved, that is,

$$I_i = \begin{cases} 1 & ||\boldsymbol{x} - \boldsymbol{x}_d|| \le \varepsilon \\ 0 & \text{otherwise} \end{cases},$$

where ε and $||\boldsymbol{x}||$ denote the norm of $\boldsymbol{x}$, and the small threshold, respectively. In this experiment, we set the termination parameters in Eq.(2) as $p = 0.8$ and $t_p = 150$ for the above two modules.

4.3 GA Settings

The population size is 50, and we perform 30 trials to evaluate each individual. At the beginning of the trial,

the robot and the ball are placed at the dark and light gray areas, respectively shown in Figure 6 (a). One trial is terminated if the robot shoots a ball into the goal or the pre-specified time interval expires. In order to select parents for crossover, we use tournament selection with size 10.

One of the most important issues is to design the fitness measures. In this experiment, we set up four fitness measures as follows:

- f_1 : the total number of obtained goals,
- f_2 : the total number of lost goals,
- f_3 : the total number of steps until all trials end
- f_4 : the total number of ball-kicking,

In order to cope with multiple fitness measures, one simple realization is to create the new scalar function based on the weighted summation of multiple fitness measures by

$$f_c = \sum_{i=1}^{n} w_i f_i, \tag{6}$$

where w_i denotes the weight for i-th evaluation. The problem is to design the value of w_i since we must consider the tradeoff among all the fitness measures. In this experiment, we use the adaptive fitness function [10] to decide the weights. Based on this method, the weights are modified considering the relationships among the changes of the four evaluations through the evolution process.

5 Experimental Results

In order to show the validity of the proposed method, we perform the following four experiments; (1) without activation/termination constraints, (2) with termination constraints, (3) with activation/termination constraints, and (4) proposed method. In cases of (2) and (3), a probability p_i for each module is fixed. Figure 9 shows the averaged scores during the evolutionary processes. Since we perform 30 trials to evaluate one individual, the maximum value of the averaged score is 30.

5.1 Simulation Results

Without activation/termination constraints

Since the robot selects the new module in real time (every 33 [msec]), the robot changed the module frequently, especially from six to ten seconds. Figure 9 shows that this approach did not fulfill the goal of shooting behavior.

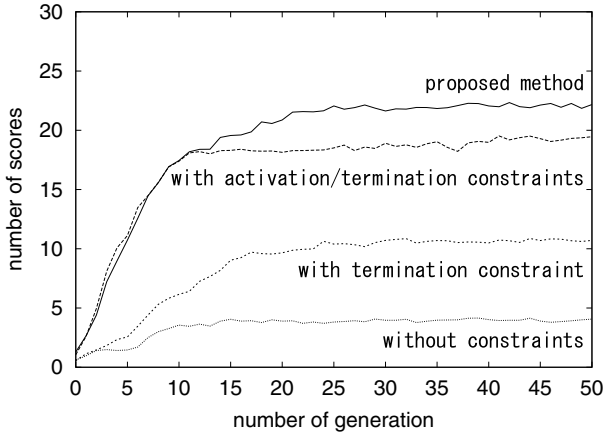


Figure 9: Average of scores

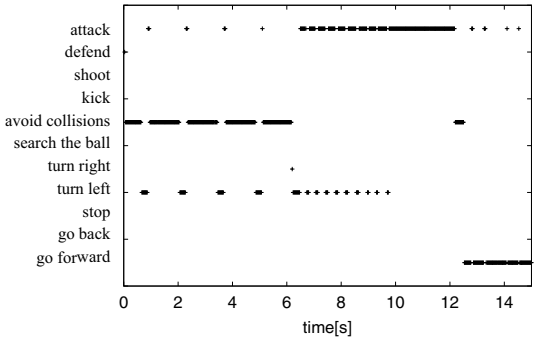
Figure 10 (a) shows the transition of the selected modules. At the beginning, the robot selected the avoiding module m_7 although the robot is not located near the wall. In this case, the robot utilized m_7 to approach the ball since this module generated the backward action when the ball is not observed. Then, the two modules m_{11} and m_4 are selected frequently from six to ten seconds. As a result, the robot failed to shoot the ball into the goal until the pre-specified time interval expired.

With termination constraints

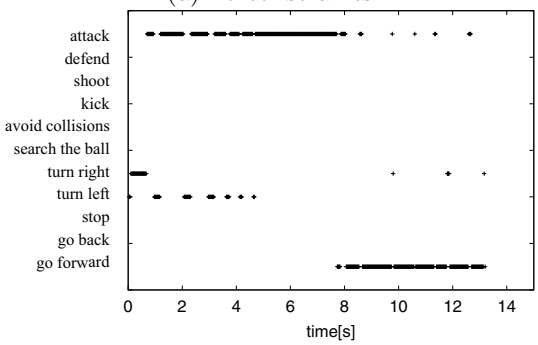
This approach caused the successful shooting behavior, and took shorter learning time than the case of no constraints described in the previous section. However, this approach took longer time to evolve than the case with both constraints. Figure 10 (b) shows the transition of the selected modules. In this experiment, m_1 was selected to shoot the ball into the goal instead of m_9.

With activation/termination constraints

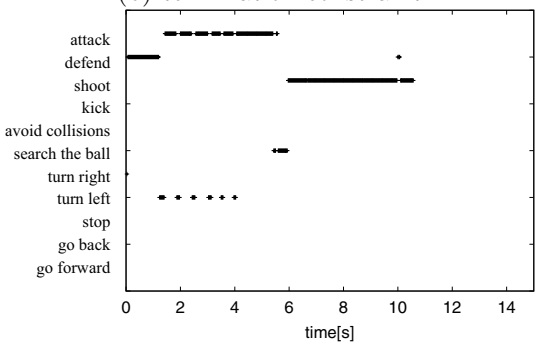
Figure 10 (c) shows the transition of the selected module. Until six seconds, the robot used three modules m_4, m_{10}, and m_{11} to go back to the offensive position. In this situation, the pre-defined behavior π_{11} can not succeed in moving to the offensive position since this behavior is implemented by a local linear feedback controller. After the robot moved to the front of the ball, the robot succeeded in shooting the ball into the goal.



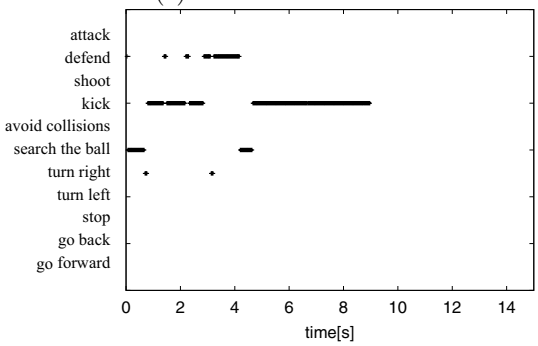
(a) no constraints



(b) termination constraint



(c) both constraints



(d) proposed method

Figure 10: Sequences of the selected modules

Figure 11: Example behavior sequences based on the proposed method

Proposed method

Figure 9 shows that this approach took the shortest learning time to obtain the shooting behavior, and got best scores. In a case of this method, the basic module, for example, m_5 (turn right) was terminated quickly. One of the successful behaviors based on the proposed method is shown in Figure 11, where the numbers in the figure represents the elapsed time.

The learning processes of the case (c) and (d) are almost same shown in Figure 9. In this experiment, the probability p_i did not converged because the optimal probability depends on the switching function.

5.2 Real Experiments

We show a result to demonstrate how the proposed method works. We transfer the result of computer simulation to the real robot. A simple color image processor (Hitachi IP5000) is applied to detect the ball and the goal area in the image in real-time (33 [msec]).

Figure 12 shows an example sequence of obtained behavior in the real environment. Because of the low image resolution of the omni-directional vision system, the robot sometimes failed to detect the objects at a long distance. In this case, the module could not be performed appropriately. In spite of those troubles, our robot could accomplish the given task.

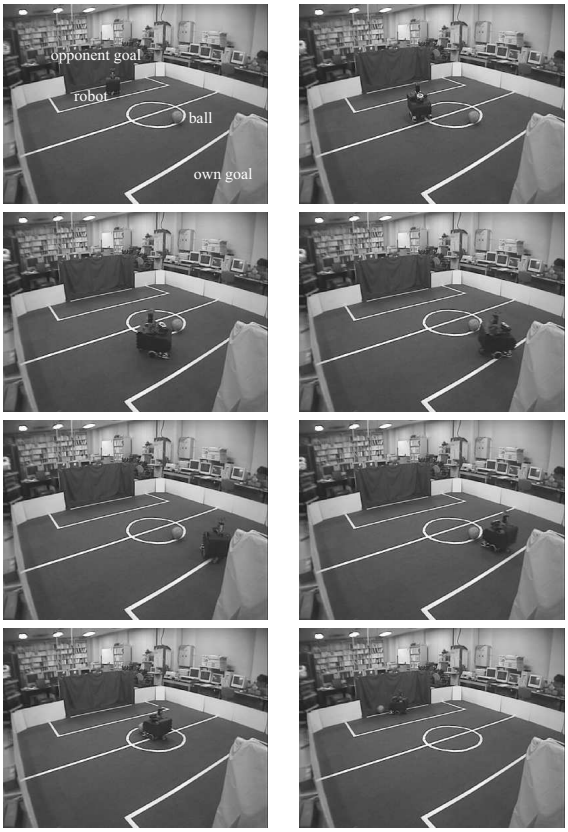


Figure 12: Obtained behavior in a real environment

6 Discussion and Future Works

This paper presented a architecture for behavior selection with activation/termination constraints. We applied the proposed method to a soccer situation, and demonstrated the experiments on a simulated robot.

In the current version of our method, the genetic operations are applied only to learning the upper layer, that is, $\boldsymbol{c}$ and $\boldsymbol{W}$. One interesting extension is to learn the appropriate termination constraints since it should be better to set the appropriate T according to the situation.

As future work, we will apply the proposed method to co-evolution for cooperative behavior acquisition in the context of RoboCup.

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 et al. Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning. *Machine Learning*, 23:279–303, 1996.

[2] M. Ebner and A. Zell. Evolving a behavior-based control architecture – From simulations to the real world. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 1009–1014, 1999.

[3] L. J. Eshleman and J. D. Schaffer. Real-Coded Genetic Algorithms and Interval-Schemata. In *Foundations of Genetic Algorithms 2*, pages 187–202. 1993.

[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison–Wesley, 1989.

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

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

[7] P. Stone. *Layered Learning in Multiagent Systems.* The MIT Press, 2000.

[8] R. S. Sutton and A. G. Barto. *Reinforcement Learning.* MIT Press/Bradford Books, March 1998.

[9] 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.

[10] E. Uchibe, M. Yanase, and M. Asada. Behavior Generation for a Mobile Robot Based on the Adaptive Fitness Function. In *Proc. of Intelligent Autonomous Systems (IAS-6)*, pages 3–10, 2000.