INVITED PAPER

Incremental Coevolution With Competitive and Cooperative Tasks in a Multirobot Environment

Soccer-playing robots can develop skills based on the success or failure of previous behavior, and skill-development is enhanced when all team members adopt successful behavior.

By Eiji Uchibe and Minoru Asada, *Fellow IEEE*

ABSTRACT | Coevolution has been receiving increased attention as a method for simultaneously developing the control structures of multiple agents. Our ultimate goal is the mutual development of skills through coevolution. The coevolutionary process is, however, often prone to settle into suboptimal strategies. The key to successful coevolution has thus far been unclear. This paper discusses how several robots can emerge cooperative and competitive behavior through coevolutionary processes. In order to realize successful coevolution, we propose two ideas: multiple schedules for incremental evolution and fitness sharing based on the method of importance sampling. To examine this issue, we conducted a series of computer simulations. We have chosen a simplified soccer game consisting of two or three robots as a testbed for analyzing a problem in which both competitive and cooperative tasks are involved. We show that the proposed fitness evaluation allows robots to evolve robust behaviors in cooperative and competitive situations.

KEYWORDS | Arms race; coevolution; fitness sharing; importance sampling; incremental evolution; multiple schedules; RoboCup

Manuscript received June 1, 2005; revised June 1, 2006.
E. Uchibe is with Neural Computation Unit, Okinawa Institute of Science and Technology, Promotion Corporation, Okinawa 904-2234, Japan (e-mail: uchibe@oist.jp).
M. Asada is with the Department of Adaptive Machine Systems, Graduate School of Engineering, Osaka University, Osaka 565-0871, Japan (e-mail: asada@ams.eng.osaka-u.ac.jp).
Digital Object Identifier: 10.1109/JPROC.2006.876918

I. INTRODUCTION

A. Motivation

One of the ultimate goals of robotics and artificial intelligence is the realization of autonomous robots that organize their internal structures to accomplish tasks through interactions with their environments. In particular, the emergence of cooperative behaviors among robots has been receiving increased attention. Coevolution, simultaneous evolution of two or more populations with coupled fitness, is one of the promising and natural approaches for acquiring cooperative and competitive behaviors in the multirobot domain. Evolutionary robotics [1], [2] supported by a Genetic Algorithm (GA) [3] and/or Genetic Programming (GP) [4] has been investigated for many years as the tool for evolving a robotic controller in a single-robot domain. Some researchers have also reported how coevolution has emerged through experiments with a predator and its prey.

Although evolutionary computation techniques have been applied to coevolutionary tasks, their inner workings are still only poorly understood. For example, emerging patterns in the predator and prey task can be placed in three categories [5], [6].

1) *Cycles of switching fixed strategies*. This pattern can be observed when a predator shifts its strategy drastically to catch its prey and its prey shifts its strategy drastically to escape from the predator. The same strategies alternate many times and no improvements on either side takes place.

0018-9219/$20.00 ©2006 IEEE

2) *Trap of suboptimal equilibria*. Although this is not a problem peculiar to coevolution, we should note it more seriously. If the evolution of a robot's opponents progresses much earlier than its own evolution, the robot could not improve its strategy given the difficult circumstances that its opponents created. When one side overwhelms its opponents, both sides reach a stable but low level of skill after which no change occurrs. This problem occurs mainly caused when the task complexity differs for each robot.
3) *Mutual skills development*. In certain conditions, every robot can improve its strategy against ever-changing environments caused by the improved strategies of the other robots. This is true coevolution by which all agents evolve effectively. This development is known as an *arms race*.

Human designers hope that all robots mutually develop their own behaviors over the course of evolution. The coevolutionary process is, however, often prone to settle into suboptimal strategies such as 1) and 2) above. The key to successful coevolution thus far has been unclear.

Recently, Asada *et al.* [7] proposed cognitive developmental robotics as a new paradigm for the design of intelligent robots including humanoids. In their argument, the conventional robot design principle puts much more emphasis on the embedded control structure than on environmental issues. Environmental design issues are also essential for a robot with an embedded structure to gradually adapt itself to more complicated environments. In a single-robot domain, we can change a task complexity by changing one parameter that is given by the human designers with a prior knowledge. For example, Asada *et al.* [8] claimed that the robot should learn how to respond to easy situations before tackling progressively more difficult situations. They showed that the scheduled arrangements of the robot and the ball allow the robot to learn shooting behaviors more efficiently in a robot soccer domain. Omata [9] applied GA to acquiring a neural network controller that could drive a bicycle. The designer gave an initial velocity to the bicycle in order to control the bicycle easily. After some generations, the velocity was gradually decreased. However, it is very difficult to reproduce the design issues suggested by Asada *et al.* in coevolution because some easy situations for one robot are sometimes difficult situations for other robots.

B. Related Work

Evolution of collective behaviors in a multirobot domain has been discussed before. Baldassarre *et al.* [10] showed that multiple robots with identical controllers obtained coordinated navigation behaviors based on the evolutionary method, and Baldassarre *et al.* [11] extended their work to more complicated tasks. Quinn [12] and Quinn *et al.* [13] showed that multiple robots could evolve coordinated behaviors without dedicated communication channels. In their studies, all the robots were controlled by the same controller, and a single population was used for evolution. Therefore, one controller had to generate a variety of different behaviors in order to assign different roles to the robots. Although their approach is efficient and simple from a viewpoint of evolutionary computation, a single population is not a natural assumption in a competitive situation.

Several researchers have investigated coevolutionary processes in the context of prey and predator problems [5], [14]–[16], is a typical example of coevolution, which a prey and predator problem can be regarded as a simple competitive task for the robot. In another example, Hornby and Mirtich [15] showed that coevolving agents against a diversity of opponents produced agents with more general strategies. Nolfi and Floreano [5] used a *hall of fame* approach [7] to encourage the arms race. In this approach, a set of past good individuals are used to ensure that the individuals in the current population remain competitive with past individuals. Stanley and Miikklainen [18] proposed an *evolutionary complexification* approach. This involves an incremental development of solutions by adding new components to the neural controllers in the duel domain of the two simulated robots. In this domain, the robot with higher energy wins when it collides with its opponent. Each robot can charge its battery by consuming food items in order to keep the energy high. We begin our discussion with predator–prey relationships because they are sufficiently complex situations to study coevolution. However, we think that this problem is still simple, because: 1) the number of agents is limited to two and 2) the relationship between agents is always competitive. In nature we can see various aspects of behaviors emerge in multiagent environments—not only competition but also cooperation, ignorance, and so on.

RoboCup (robot soccer) [19] gives us a new problem setting instead of studying simple predator-prey relationships. RoboCup has increasingly attracted researchers as benchmark. Behavior acquisition in a multirobot environment has been especially tackled by means of reinforcement learning and/or evolutionary computation [20]–[29]. Luke *et al.* [25] showed that coevolved teams consisting of 11 soccer players could obtain cooperative behaviors using coevolution. However, coevolving cooperative agents have not been considered as a design issue for individual players only to teams. Ciesielski *et al.* [22] also applied GP to behavior acquisition for the soccer game. Their approach was to evolve one player's program and to copy that program to the ten remaining players. Keep-away is a subtask of robot soccer, where one team of agents (keepers) tries to maintain possession of the ball while another agent tries to get it. Since keep-away is more tractable, it has been selected as a new testbed by many researchers [23], [24], [27], [28]. During the game, the keepers learn to pass the ball to each other as much as

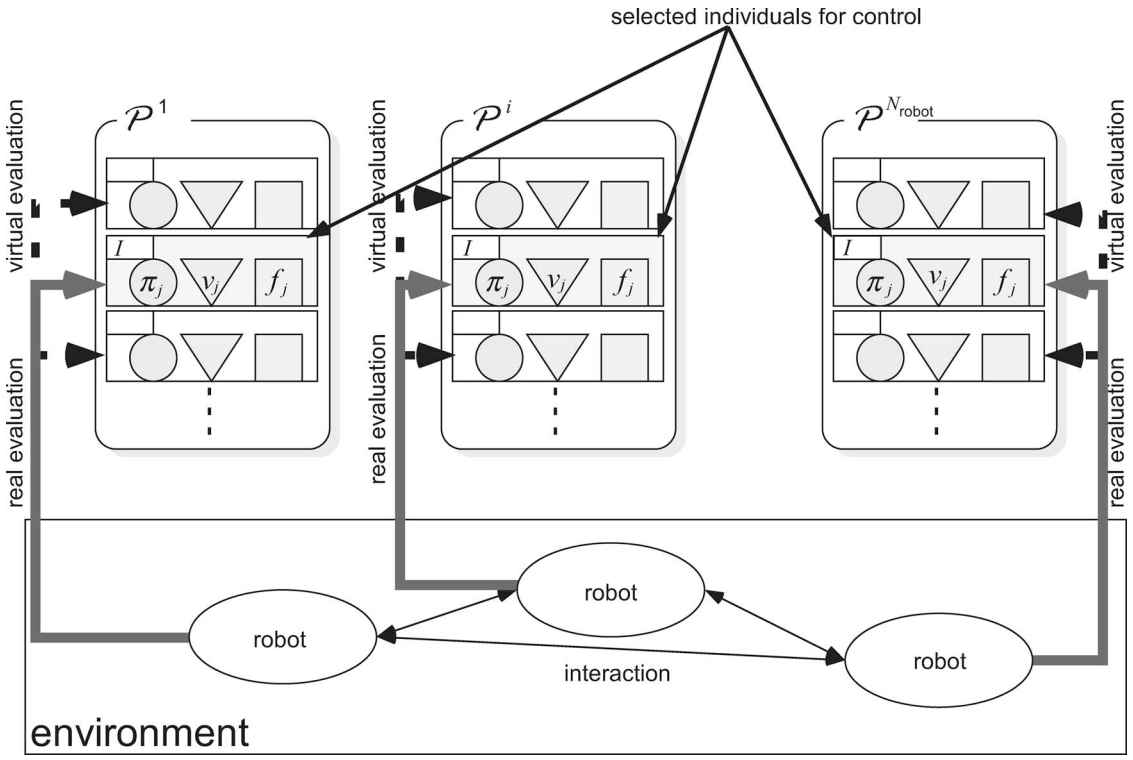


Fig. 1. ***Entire coevolutionary system in our method. Each robot has its own population, and there is no migration among the populations. The selected individuals obtain the real evaluations calculated by (2) while the individuals that are not selected obtain the virtual evaluation based on the fitness sharing discussed in Section II-B. The superscript i is omitted for simplicity.***

possible while the opponent attempts to intercept the ball. In this task, all keepers are controlled by the same evolved neural controller, while the opponent is controlled by a fixed controller.

We believe that aside from one-to-one individual competition and team competition, noncompetitive multirobot behaviors could coevolve. In the RoboCup domain, we are challenged to evolve both competitive and cooperative behavior. We have shown preliminary results when two or three robots evolved simultaneously in our previous work [6]. Our previous experiments revealed that the existence of a competitor is essential for cooperative robots to emerge cooperative behaviors by coevolution.

C. Our Approach

The purpose of this study is to investigate how multiple robots evolve controllers simultaneously during coevolutionary processes. We claim that there are two critical issues for successful coevolution. The first one is the *Multiple Schedules* approach for incremental evolution, which means that multiple schedules are tested simultaneously in each generation. The other is a method of *fitness sharing*. That is, the robots evolve robust and versatile stochastic policies (controllers). The straightforward implementation of multiple schedules requires a prohibitively long time to evolve behaviors since each schedule must be sequentially evaluated. Fitness sharing is based on the method of importance sampling [30]. We can modify fitness values followed by one schedule, then those followed by another schedule based on the similarity of the stochastic policies. Consequently the evolutionary process can depart from the problems in the cases of 1) and 2) described above. To examine the above issues, we conduct a series of computer simulation of a simplified soccer game in which both competitive and cooperative tasks are involved at the same time.

The remainder of this paper is organized as follows. Section II describes our method in detail. Section III explains the simulation settings about the simplified soccer game [6]. Section IV presents the results of a series of our simulation experiments. Section V concludes, and discusses future work.

II. SUCCESSFUL COEVOLUTION BY FITNESS SHARING

A. Notations

Fig. 1 shows an overview of our coevolutionary system. Suppose that there are N_{robot} robots in the environment.

Each robot R^i has its own population $\mathcal{P}^i (i = 1, \ldots, N_{\text{robot}})$ and attempts to acquire desired behaviors. There is no migration among the populations. The number of each population $\mathcal{P}^i$ is N_{pop} for simplicity, although we can set it arbitrarily. Each individual $I^i_j (j = 1, \ldots, N_{\text{pop}})$ has a performance value v^i_j and a stochastic policy (controller) π^i_j that assigns a probability distribution over actions to each state.

L games are held in each generation. After L games are over, the new populations are generated by applying genetic operations. At the beginning of each game, the robot R^i chooses one individual from its own population. The probability for R^i to select the jth individual is given by

$$\Pr(i, j) = \frac{\exp\left(\beta v^i_j\right)}{\sum_{j=1}^{N_{\text{pop}}} \exp\left(\beta v^i_j\right)} \tag{1}$$

where β is a positive parameter called the inverse temperature. Low β causes (nearly) equi-probable selection of all individuals, while high β causes selection of the individual with the highest performance with probability close to one.

Let us denote I^i_{selected} the selected individual. When a new game starts, each robot is controlled by the stochastic controller of the individual. Let h^i_l be the sequence of states $\boldsymbol{x}$, actions $\boldsymbol{u}$, and evaluations $\boldsymbol{e}$ of the ith robot in the lth game

$$h^i_l = \left\{ \boldsymbol{x}^i_1, \boldsymbol{u}^i_1, \boldsymbol{e}^i_1, \boldsymbol{x}^i_2, \boldsymbol{u}^i_2, \boldsymbol{e}^i_2, \ldots, \boldsymbol{x}^i_{M_l}, \boldsymbol{u}^i_{M_l}, \boldsymbol{e}^i_{M_l}, \boldsymbol{x}^i_{M_l+1} \right\}$$

where M_l is the length of the lth game. From the sequence of evaluation $\{\boldsymbol{e}^i_1, \boldsymbol{e}^i_2, \ldots, \boldsymbol{e}^i_{M_l}\}$, the fitness value for the selected individuals is computed by

$$f^i_l(\text{selected}) = f\left(h^i_l\right) = \sum_{m=1}^{M_l} \boldsymbol{w}^T \boldsymbol{e}^i_m \tag{2}$$

where $\boldsymbol{w}$ is a weight vector, and $\boldsymbol{w}^T$ denotes a transpose of $\boldsymbol{w}$. Note that $f^i_l(\text{selected})$ is the function of the episode h^i_l. We call f^i_l *real* the fitness value for the selected individual I^i_{selected} at the lth game. Later, we may omit the superscript i because fitness evaluation and genetic operations are performed separately in each robot.

B. Fitness Sharing Based on Importance Sampling

Standard evolutionary computation methods use the fitness value f_l calculated by (2) just for evaluation of the individuals that attended the game. This means that every individual of one population plays every member of the other. Consequently, computing time in coevolution will be prohibitively long. Here, we consider how the real fitness can be used for evaluating all individuals $I_j (j = 1, \ldots, N_{\text{pop}})$ that do not attend the game. Since the probability that the sequence h_l occurs depends on the stochastic policy used by the robot, we have to modify the real fitness values according to the difference between the target policy $\pi_j (j \neq \text{selected})$ and selected policy π_{selected}. In order to compensate the mismatch between them, we can use the method of importance sampling [30] that is a well-known classical technique for obtaining accurate tail quantiles of a bootstrap distribution more quickly.

Let us denote $p_j(h)$ to be the probability that the sequence h occurs by following the stochastic policy π_j. A simple fitness estimator based on importance sampling is given by

$$\begin{aligned} E_{\pi_j}\{f\} &= \int_h f(h) p_j(h) dh \\ &= \int_h f(h) \frac{p_j(h)}{p_{\text{selected}}(h)} p_{\text{selected}}(h) dh \\ &= E_{\pi_{\text{selected}}} \left\{ f \frac{p_j(h)}{p_{\text{selected}}(h)} \right\} \\ &\approx \frac{1}{L} \sum_{l=1}^{L} f_l w_l(j), \end{aligned} \tag{3}$$

where E_π denotes the expectation with respect to the stochastic policy π, and

$$w_l(j) = \frac{p_j(h)}{p_{\text{selected}}(h)}. \tag{4}$$

$w_l(j)$ is called the *importance sampling weight* for the jth individual at the lth game. Another implementation of the importance sampling estimator is given by

$$f(j) = E_{\pi_j}\{f\} = \frac{\sum_{l=1}^{L} f_l w_l(j)}{\sum_{l=1}^{L} w_l(j)}. \tag{5}$$

Although the estimator given by (5) is consistent but biased [30], this estimator is often faster and more stable in practice than the estimator given by (3). Later, we call this fitness value by using (5) *virtual* fitness.

It is easy to calculate the importance sampling weight [31] because the system knows all stochastic policies. The probability that the sequence h_l occurs by following the stochastic policy π is represented by

$$p(h_l) = \Pr(h_l|\pi) = \Pr(\boldsymbol{x}_1)\prod_{m=1}^{M_l}\pi(\boldsymbol{x}_m,\boldsymbol{u}_m)\Pr(\boldsymbol{x}_{m+1}|\boldsymbol{x}_m,\boldsymbol{u}_m) \quad (6)$$

where $\Pr(\boldsymbol{x}_1)$ and $\Pr(\boldsymbol{x}_{m+1}|\boldsymbol{x}_m,\boldsymbol{u}_m)$ are the initial state probability and the single-step transition probability, respectively. Then, $w_l(j)$ is given by

$$w_l(j) = \prod_{m=1}^{M_l}\frac{\pi_j(\boldsymbol{x}_m,\boldsymbol{u}_m)}{\pi_{\text{selected}}(\boldsymbol{x}_m,\boldsymbol{u}_m)} \quad (7)$$

where we assume that the behavior policy should satisfy $\pi_{\text{selected}}(\boldsymbol{x},\boldsymbol{u}) > 0$ if $\pi_j(\boldsymbol{x},\boldsymbol{u}) > 0$. Here, the knowledge of the environmental dynamics $\Pr(\boldsymbol{x}_{m+1}|\boldsymbol{x}_m,\boldsymbol{u}_m)$ is not needed to compute $w_l(j)$. We only need the ratio of the stochastic policies. The standard fitness evaluation without importance sampling is obtained if we set the weights as follows:

$$w_l(j) = \begin{cases} 1 & \text{if } \pi_j(\boldsymbol{x},\boldsymbol{u}) = \pi_{\text{selected}}(\boldsymbol{x},\boldsymbol{u}) \text{ for all } \boldsymbol{x} \text{ and } \boldsymbol{u}, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

In this case, the fitness value of π_{selected} is not used for evaluating others.

C. Multiple Schedules for Successful Coevolution

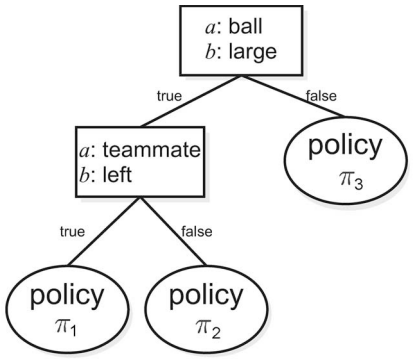
Incremental evolution has been used with various evolutionary methods to solve complex problems. A complex problem is challenged by first evolving for a simplified situation of the environment. Then, the difficulty of the problem is gradually increased by modifications of the meta-issues such as the initial configurations or the fitness function. Although this incremental evolution approach works well in the single robot domain [8], [9], it is not promising in the multiple robot domain. Suppose that the evolutionary schedules A and B are well established for the robot R1 and R2, respectively. If evolution is followed by the single schedule A, R1 and R2 do not have even task complexities. This unfair schedule leads the coevolutionary process to suboptimal strategies. The Multiple Schedules approach means that multiple schedules are tested simultaneously in each generation. In other words, both R1 and R2 are evaluated in the situation followed by the schedule A and B.

So we propose a Multiple Schedules approach to overcome the above problem. Each robot evolves behaviors under the environmental settings scheduled not only for its own but also for others in each generation. Since each population has a chance to evolve behaviors in a well-scheduled environment, coevolution such as arms race can be emerged as a result. One disadvantage of the Multiple Schedules approach is that naive implementation requires a long time for evaluation. Fortunately, we can reduce the computing cost by using the fitness sharing shown in Section II-B.

D. Representation of the Stochastic Policy

We can consider several representations to implement stochastic policies. Here, we use a tree structure to implement the reconfigurable stochastic policy shown in Fig. 2. In our policy representation, each leaf of the decision tree represents the hand-coded stochastic policy that can be regarded as a basic skill. The set of the hand-coded stochastic policies are called a *terminal set*. One conditional branching function divides the state space into two regions. The form of the conditional branching function is given by (`if`(= `a` `b`)`c` `d`), which executes its first branch `c` if the condition "`a` is `b`" is true; otherwise it executes its second branch `d`. In our experiment, `a` and `b` denote an object and its category, respectively. Then, multiple branching functions give complex switching function of stochastic policies according to the environmental state. We call the set of branching functions the *function set*.

Crossover is one of the most important genetic operations. We consider genetic operations used in GP. Then, a leaf node is regarded as a terminal while a conditional branching function is a special case of the




```
(if (= ball large)
    (if (= teammate left) π1 π2)
    π3)
```

Fig. 2. *Stochastic policy represented by tree structure. Box and ellipsoid represent the conditional branching function and the hand-coded stochastic policy, respectively.*

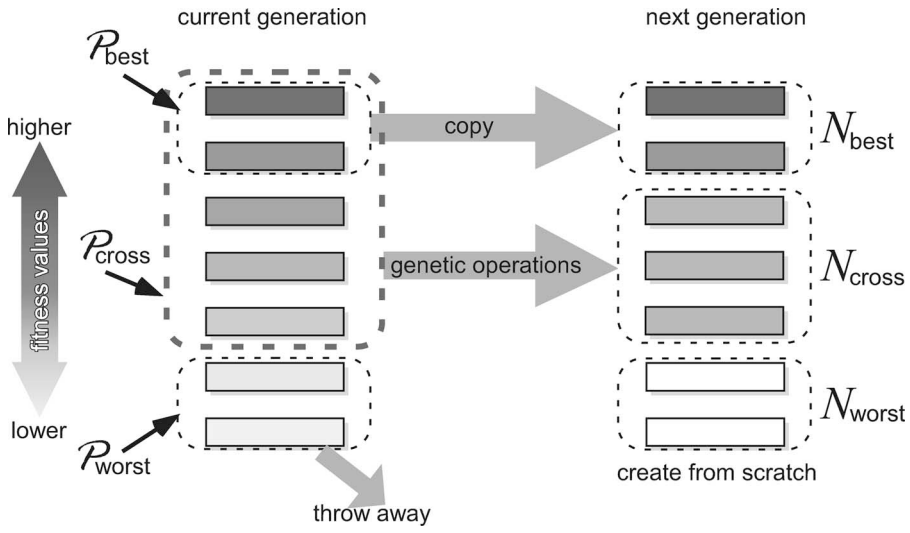


Fig. 3. *Genetic operations to generate the next population. In order to make N_{cross} individuals, two parents are selected from $N_{best} + N_{cross}$ individuals.*

normal function. In crossover, two cut points are chosen and whole subtrees are swapped. If the depth of one or both of the resulting trees is larger than the maximum depth D_{max}, the subtrees are swapped back and another cut point is chosen. The initial depth of the new tree is strictly limited to D_{init}, where $D_{init} \leq D_{max}$. Moreover, mutation should take place with a prespecified probability $P_{mutation}$. In mutation, one node is exchanged with another node that is randomly chosen from the function and terminal sets.

E. Genetic Operations

We explain the genetic operations (selection, crossover, and mutation) to generate the next generation pool shown in Fig. 3. According to the estimated fitness values, the population $\mathcal{P}$ is divided into three groups. The first is the *best* subpopulation $\mathcal{P}_{best}$ consisting of the best N_{best} performing individuals. The individuals in $\mathcal{P}_{best}$ are moved unchanged to the new population. The second is the *worst* subpopulation $\mathcal{P}_{worst}$ consisting of the worst N_{worst} individuals. The individuals in $\mathcal{P}_{worst}$ are thrown away, and N_{worst} individuals are created for the new population. The last is the *cross* subpopulation $\mathcal{P}_{cross}$ that constitutes the best $N_{best} + N_{cross}$ individuals. By applying genetic operations such as crossover and mutation into $\mathcal{P}_{cross}$, N_{cross} individuals are generated to the next population. Note that $N_{best} + N_{worst} + N_{cross} = N_{pop}$.

Since each robot selects one individual from its own population according to (1), the performance values v_j^i must be assigned before actual evaluation. We set fitness values at the previous generation to the performance values for individuals in $\mathcal{P}_{best}$ because they are not changed before and after generations. The performance values for individuals in $\mathcal{P}_{worst}$ are initialized as random values. Consequently, the values v_j at the generation $t+1$ are computed from the fitness values f_j at the generation t by

$$v_j(t+1) = \begin{cases} f_j(t) & j \in \mathcal{P}_{best}, \\ \tilde{f} & j \in \mathcal{P}_{worst}, \\ \left[f_{j_1}(t) + f_{j_2}(t)\right]/2 & j \in \mathcal{P}_{cross} \end{cases} \quad (9)$$

where j_1 and j_2 are the indexes to represent parents. $\tilde{f}$ is a random variable that has a uniform distribution in the range 0 to $\min_{j \in \mathcal{P}_{best}} f_j$.

We use *Tournament Selection* in order to select two parents for crossover. $N_{tournament}$ individuals are selected from the population $\mathcal{P}_{cross}$ at random. Selection pressure is adjusted by changing the tournament size. Smaller tournament size gives weak individuals a better chance of being selected.

III. TASK AND ASSUMPTIONS

A. RoboCup Task

We have chosen a simplified soccer game consisting of two or three robots as a testbed for the problem because both competitive and cooperative tasks are involved as stated in the RoboCup Initiative [19]. We consider the setting of the middle-sized robot league of RoboCup Soccer. The environment consists of a ball and two goals, and a wall is placed around the field except at the goals. Fig. 4 shows one of the real robots used for modeling. The robots have the same body (power wheeled steering system) and the same sensor (onboard TV camera); that is, they are homogeneous agents. In our simulator, the robot cannot obtain complete information about its environment

because of the limitation of its sensing capability and occlusion of the objects.

As described before, the branch of the tree is a simple conditional function "(`if`(= `a b`)`c d`)" that executes its first branch `c` if the condition "`a` is `b`" true, otherwise it executes its second branch `d`, where `a` and `b` denote an object and its category, respectively. In our task, there are five objects (the ball, the goal, the opponent's goal, and the other two robots), and seven categories (left, center, right, small, medium, large, and lost). These image features are the same ones that were used in our previous work [6], [8]. All robots must be recognized uniquely because each has a different role. The number of function sets is $35(=7\ \text{(ball)} + 2 \times 7\ \text{(two goals)} + 2 \times 7\ \text{(other two robots)})$.

Leaf node is a stochastic policy that gives a probability to select the action according to the current state. We prepare the following four stochastic hand-coded policies.

1) π_{shoot}: the robot shoots a ball into the opponent's goal.
2) π_{pass}: the robot kicks a ball to a teammate.
3) π_{avoid}: the robot avoids collisions with other robots.
4) π_{search}: the robot searches for the ball by turning to the left or right.

The stochastic policies described above were formulated by a combination of a linear feedback controller and a Gaussian noise as follows:

$$\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{K}(\boldsymbol{x}_d - \boldsymbol{x})\Sigma),$$

where η is a normalizing constant, $\boldsymbol{K}$ is a feedback controller gain, $\boldsymbol{x}_d$ is the desired state, and Σ is the covariance matrix of noise, respectively. In other words, $\boldsymbol{u}$ is a normally distributed with mean $\boldsymbol{K}(\boldsymbol{x}_d - \boldsymbol{x})$ and covariance Σ.

The task for the robots is to play a game to obtain as many points as possible without losing points. We first consider the following parameter to evaluate team behaviors such as cooperation between teammates and competition with opponents:

$$e_1^i(m) = \begin{cases} 1 & \text{if the goal is scored by the team to which the robot } R^i \text{ belongs at the } m\text{th time step,} \\ -0.5 & \text{if the goal is lost by the opponent team,} \\ 0 & \text{otherwise.} \end{cases}$$

With the above evaluation only, it will take a prohibitively long time to evolve policies because almost all individuals have zero fitness value at the early stage of evolution. To address this, we introduced the following more individualized evaluation to encourage robots to interact with each other while minimizing the number of collisions:

$$e_2^i(m) = \begin{cases} 1 & \text{if } R^i \text{ kicks the ball at the } m\text{th time step,} \\ 0 & \text{otherwise,} \end{cases}$$

$$e_3^i(m) = \begin{cases} -1 & \text{if } R^i \text{ makes a collision with the objects,} \\ 0 & \text{otherwise.} \end{cases}$$

Parameters used are shown in Table 1. These parameters are decided by performing the preliminary experiments.

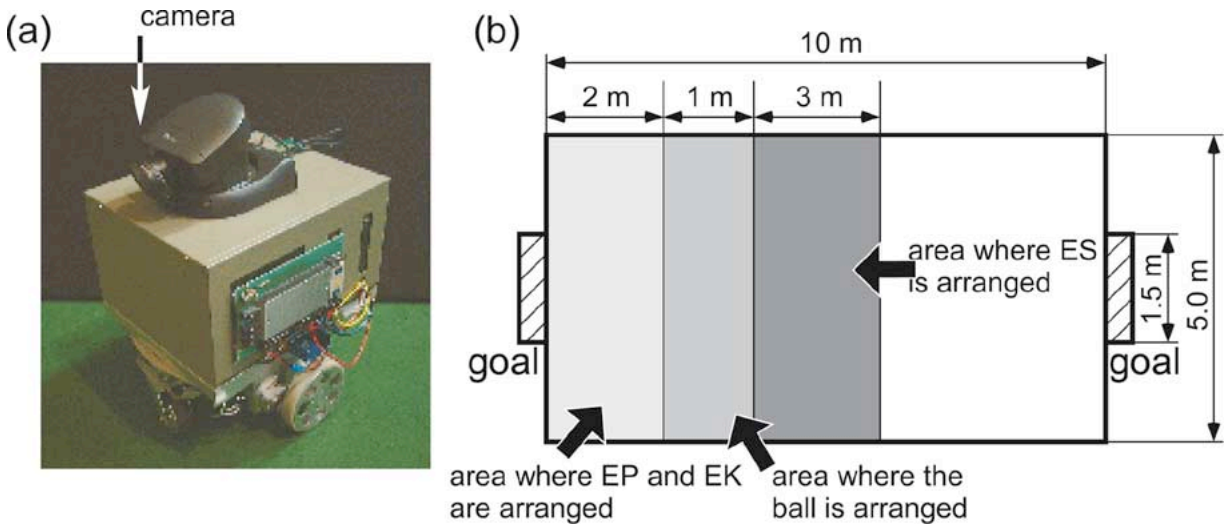


Fig. 4. *Robot and environment. (a) Our robot for modeling. (b) The size of the environment and the initial areas where the robots and the ball are located. EP, ES, and EK mean the robots that are expected to be the passer, shooter, and keeper, respectively.*

Table 1 Parameter Settings Used in the Experiments

parameters	setting
population size	$N_{\mathrm{pop}} = 100$, $N_{\mathrm{best}} = 10$, $N_{\mathrm{worst}} = 10$, $N_{\mathrm{cross}} = 80$
weight for fitness function	$w_1 = 1.0$, $w_2 = 0.1$, $w_3 = 0.1$
depth of the tree	$D_{\mathrm{init}} = 2$, $D_{\mathrm{max}} = 10$
tournament size	$N_{\mathrm{tournament}} = 10$
inverse temperature	$\beta = t/100$ at the generation t
the number of games	$L = 600$

There are two important parameters β and $D_{\max}$. Since β controls the probability to select the randomness of the probability to select the individual that actually collects experiences, β must increase gradually. We conducted some experiments with different parameter settings; we found no significant differences at the end of evolution.

B. Evolutionary Schedule

We can consider a variety of evolutionary schedules, since the soccer task involves not only competitive but also cooperative situations. Here, we prepare the three schedules named Cooperative Schedule, Competitive Schedule, and No Schedule. For readers' understanding, we named the three robots EP, ES, and EK, which signifies the robots that are expected to be the passer, shooter, and keeper, respectively.

Cooperative Schedule

A-1 From 0 to 1000 generations. EP and ES evolve behaviors. There are no competitors in the environment.

A-2 From 1001 to 2000 generations. One robot, EK, is added to the environment. EK does not move at all in this phase. Both EP and ES continue to evolve behaviors.

A-3 From 2001 to 3000 generations. EK starts to evolve behaviors. This task includes cooperative and competitive relations among the robots.

The purpose of the Cooperative Schedule is regarded as the patronized schedule for ES and EK. Note that EK has to evolve behaviors in the difficult situations because we expect that EP and ES obtain good cooperative behaviors at the end of **A-2**.

Competitive Schedule

B-1 From 0 to 1500 generations. ES and EK evolve behaviors. Since the relationship between ES and EK is competitive, this phase is regarded as competitive coevolution.

B-2 From 1501 to 3000 generations. EP is added to the environment. All the robots evolve behaviors.

ES and EK are practically even in **B-1**.

No Schedule

C-1 From 0 to 3000 generations. All the robots evolve behaviors. Then, no incremental evolution is considered at all.

No Schedule is the most straightforward way for applying coevolution. Multiple Schedule combines the above three schedules. In this case, we can summarize the evolutionary schedule as follows.

Multiple Schedules

D-1 From 0 to 1000 generations. The initial configuration is selected from **A-1**, **B-1**, or **C-1** at random.

D-2 From 1001 to 1500 generations. The initial configuration is selected from **A-2**, **B-1**, or **C-1** at random.

D-3 From 1501 to 2000 generations. The initial configuration is randomly selected from **A-2** or **C-1**, since **B-2** and **C-1** are the same setting.

D-4 From 2001 to 3000 generations. The initial configuration is **C-1**. That is, all the robots evolve behaviors simultaneously.

IV. EXPERIMENTAL RESULTS

Experimental results are summarized in Table 2.

A. Results Followed by Cooperative Schedule

Fig. 5(a) and (b) show the transitions of the average fitness values and the depths of the best ten individuals followed by the Cooperative Schedule. Average fitness values gradually increased and remained almost constant from the 700th generation to the 1000th generation.

Table 2 Summary of the Experimental Results

Cooperative Schedule	**EP**	mutual skills development
	ES	
	EK	trap of suboptimal equilibria
Competitive Schedule	**EP**	trap of suboptimal equilibria
	ES	mutual skills development
	EK	
No Schedule	**EP**	trap of suboptimal equilibria
	ES	mutual skills development
	EK	
Multiple Schedules	**EP**	mutual skills development best performance
	ES	
	EK	

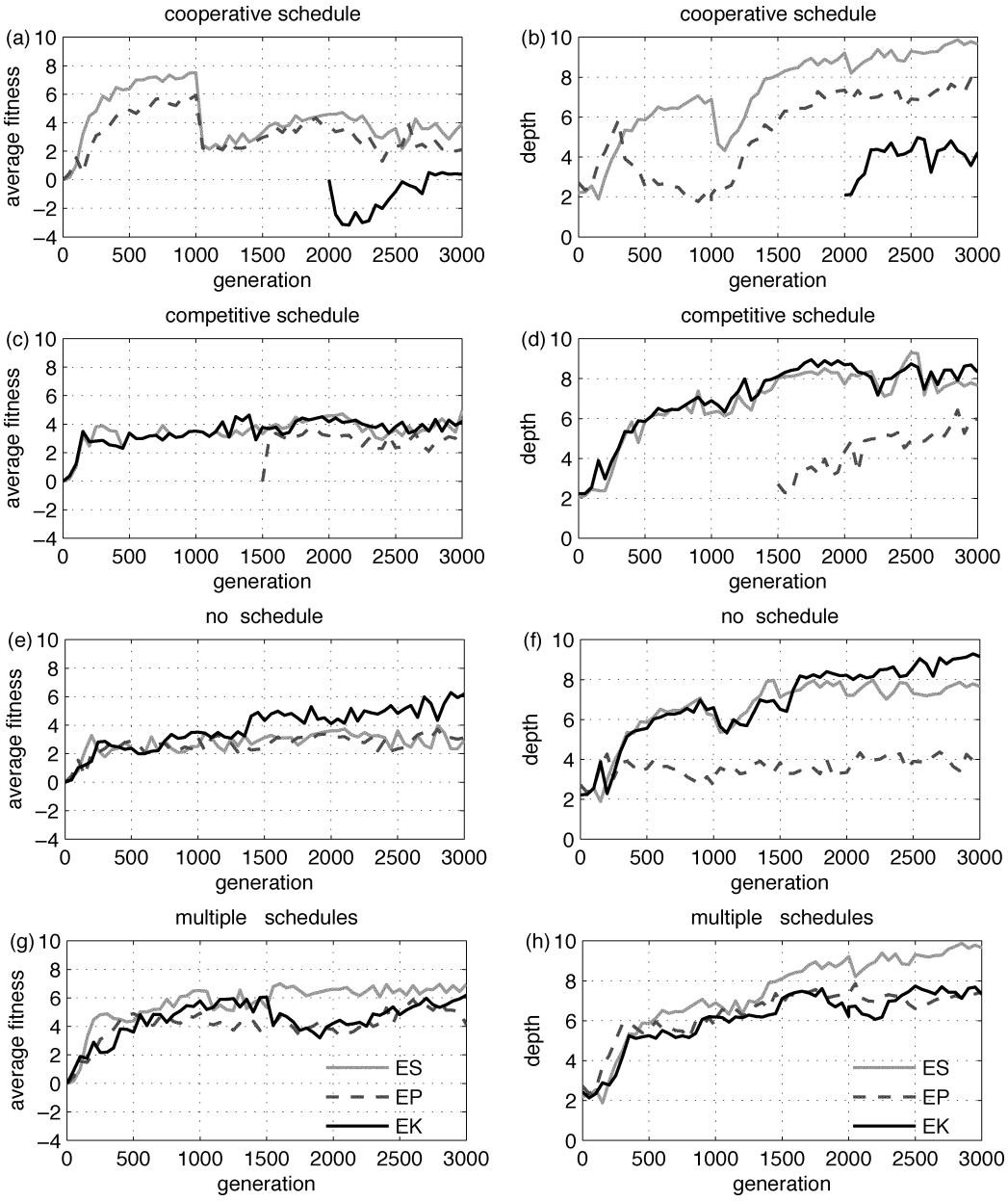


Fig. 5. ***Experimental results. (a), (c), (e), and (g) show the transition of the average fitness values followed by Cooperative Schedule, Competitive Schedule, No Schedule, and Multiple Schedules, respectively. (b), (d), (f), and (g) show the depths of the best ten individuals followed by the above schedules.***

Since both ES and EP received the same evaluations with respect to the numbers of obtained and lost scores, the difference of fitness values was mainly caused by the ball-kicking scores. There was a gradual increase in the average depth for ES. Conversely, the average depth for EP reached a peak of six in the 300th generation, and it then decreased rapidly after about 300 generations. These results suggested that the evolved stochastic policies obtained by ES were much more complicated that those obtained by EP.

Adding EK caused a rapid decrease of fitness values and the depths of EP and ES. This suggested that the evolved populations in **A-1** were not good initial ones in **A-2**. After 1000 additional generations, ES and EP obtained cooperative behaviors as follows:

- EP passed the ball in front of ES;
- ES shot the received ball into the goal while avoiding collisions with stationary EK.

EK started to evolve behaviors after 2000 generations. Because EP and ES had already obtained efficient

cooperative behaviors for scoring, the environmental setting was very difficult for EK to evolve behaviors from scratch. Although some individuals could evolve basic goalkeeping behaviors at the end of evolution, none was able to obtain goal scoring behaviors. The depth of EK's tree was not deeper than those of the trees of ES and EP at the end of generation. That is, the stochastic policies obtained by EK were not as complicated as those of ES and EP. In fact, EK failed to evolve a good behavior for goal saving.

B. Results Followed by Competitive Schedule

Fig. 5(c) and (d) show the transitions of the average fitness values and the depths of the best ten individuals followed by the Competitive Schedule. Since the initial configurations were almost the same for ES and EK, the transitions of their average fitness values of them had the same tendency and remained constant after about 100 generations. By seeing the obtained behaviors, we found that ES defeated the novice EK and *vice versa*. In addition, the game with the best EP and EK was drawn.

ES seldom moved randomly, since most individuals in this role could perform purposive shooting behaviors by themselves at the end of **B-1**. We expected that EP could easily evolve appropriate behaviors because EP only had to kick the ball toward ES. The experimental results were, however, different from our expectation. EP did not affect the evolutionary processes of ES and EK against our expectation. The depth of EP's tree followed by this schedule was smaller than that followed in the Cooperative Schedule, as shown in Fig. 5(b) and (d). We found that EP ran into empty space along the wall to avoid collisions while ES and EK competed keenly with each other. Clearly, these are not the cooperative behaviors that we intended before conducting experiments.

C. Results Followed by No Schedule

Fig. 5(e) and (f) show the transitions of the average fitness values and the depths of the best ten individuals followed by No Schedule. The transitions of average fitness values were similar to those shown in the Section IV-B. After about 1300 generations, average fitness values of EK became larger than those of ES and EP.

We could not find the increase of the average depth of EP's tree over the generation while those of ES and EK gradually increased as shown in Fig. 5(f). This result implied that this cooperative and competitive task had degenerated into competitive one like the predator–prey problem described in the introduction.

D. Results Followed by Multiple Schedules

Fig. 5(g) and (h) show the transition of the average fitness values and the depths of the best ten individuals in the evolutionary process, respectively. We found that a good synchronization among EP, ES, and EK could be seen by following the Multiple Schedules. That is, there were no significant differences among evolutions in the three average fitness values of EP, ES, and EK.

Evolved behaviors proved to be very effective. Fig. 6 shows the results of the game when the game was held by the robots followed by the different schedule. The game was played 100 times for each schedule, and the pair of EP and ES showing the best performance was selected. For example, the top left figure [Fig. 6(a)] shows the obtained and lost scores by the best pair of ES and EP followed by No Schedule when they competed with EK followed by Multiple Schedules. Fig. 6(a), (b), (c), and (d) suggest that EK followed by Multiple Schedules could defend the goal regardless of the evolutionary schedule for ES and EP. Fig. 6(d), (h), (l), and (p) also shows the effectiveness of the Multiple Schedules approach. That is, the best pair of EP and ES followed by Multiple Schedules could obtain better scores as compared with those following the other three schedules.

Cooperative Schedule was very effective for evolving behaviors of ES and EP, since Fig. 6(c), (g), (k), and (o) had the same tendencies in Fig. 6(d), (h), (l), and (p), respectively. Conversely, Cooperative Schedule was ineffective for evolving behaviors of EK because EK allowed many scoring opportunities to be lost.

V. CONCLUSION AND FUTURE WORK

This paper has shown how coevolutionary techniques can emerge not only competitive behaviors but also cooperative behaviors through a series of experiments in which two or three robots play a simplified soccer game. For cooperative and competitive robots to coevolve successfully, they must synchronize their evolutionary processes to avoid getting trapped in suboptimal equilibria (suboptimal strategies). In addition, the evolving robots must interact with a variety of cooperators and competitors to estimate robust fitness values.

Incremental evolution is one of the effective approaches for evolving controllers in the single robot domain. However, it is almost impossible to prepare the single effective evolutionary schedule for the problem in which both competitive and cooperative tasks are involved simultaneously. Experimental results suggest that the efficiency of the schedule such as Cooperative Schedule and Competitive Schedule depends on the robot. Therefore, the Multiple Schedules approach is very essential for realizing successful coevolution. Our method does not show the efficiency of the best schedule, if one exists, for all robots. In this case, the best schedule will give us better results, but we think that it is a rare case in cooperative and competitive tasks. The proposed fitness sharing based on the technique of importance sampling can reduce the computing time required for evaluation. Experimental results showed that our fitness evaluation is very efficient for cooperative and

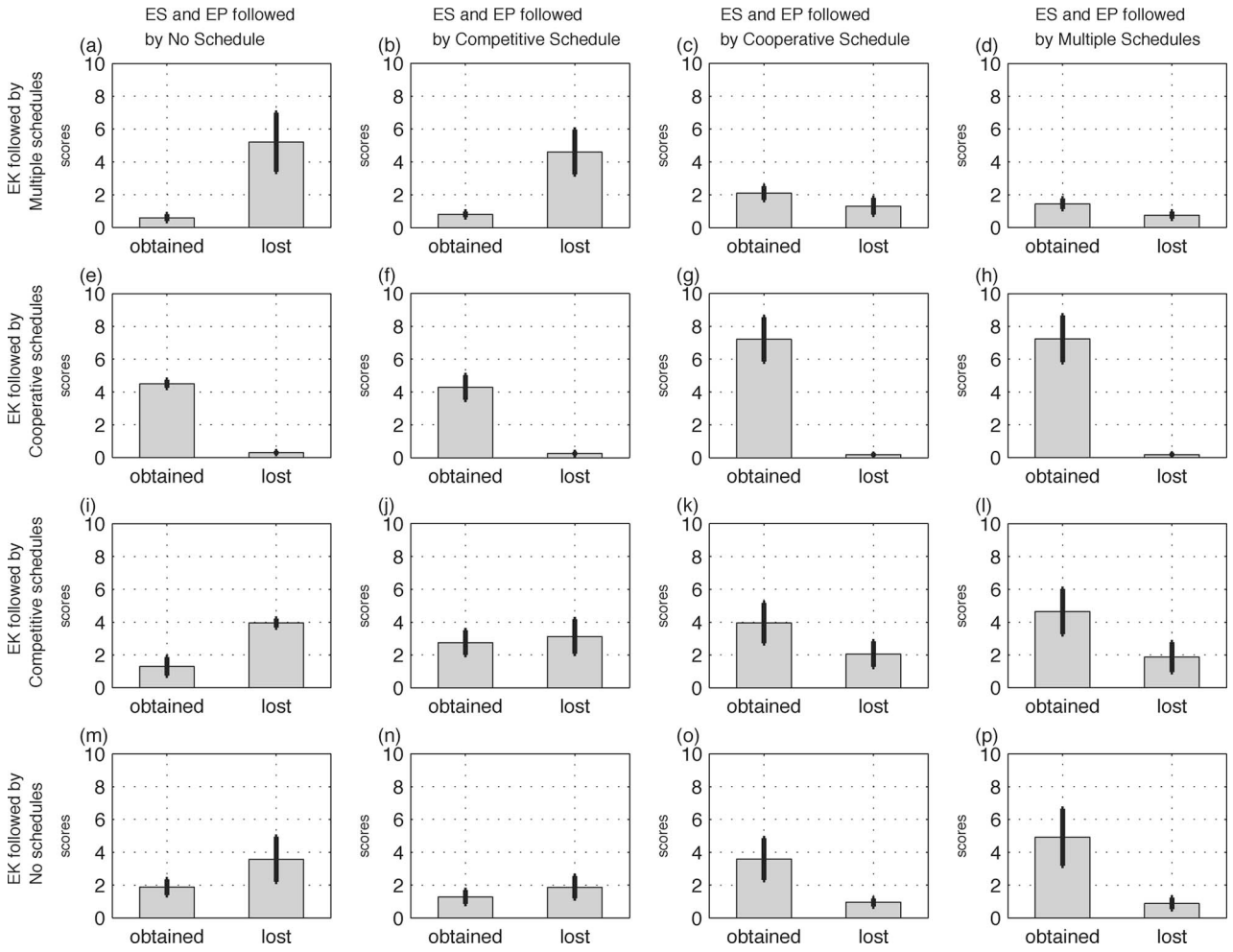


Fig. 6. ***Experimental results of the game. The boxes and the error bars mean the average and the standard deviation of obtained and lost scores, respectively. (a) shows obtained and lost scores by the best pair of ES and EP followed by No Schedule when they competed with EK followed by Multiple Schedules.***

competitive coevolutionary tasks. Importance sampling has received increased attention in the field of reinforcement learning [32], [33].

Multiple Schedules can be regarded as a method to collect randomized training samples. If the individuals are evaluated in order of the Cooperative Schedule and Competitive Schedule, only one schedule will be considered in practice. Randomizing samples is a natural approach in the field of machine learning society, since it is well known that the order of training patterns (instances) affects the results of learning. For example, a random presentation of training samples may help a neural network escape from poor local minima in the error function. To combine the method developed with the machine learning literature is one of the challenging issues.

Since the number of evaluations L is not directly related to the population size, our fitness sharing method can be applied to larger populations. However, we think that the more sophisticated strategy would be required to select the individual that actually collects experiences. Because the same individual with the best performance is usually selected at later generations, searches for other individuals are not performed well regardless of the population size. This is one of our future research topics.

Another important topic we do not mention is designing the fitness function. That is, the robot should change its own fitness function dynamically by taking into account whether the task could be accomplished. We have developed an adaptive fitness function [34] to control the complexity of the task. This method can modify the weights used in the linear combination of fitness measures according to a statistical analysis of those measures.

We have not yet conducted the real robotic experiments. One of the difficult problems for evolution in the

real environment is to collect important and interesting experiences used for evaluating individuals. Our fitness sharing is a powerful tool because it can transfer weighted values of evaluation to all individuals according to the differences among the stochastic policies. This means that our method can greatly shorten the experiment time. ■

REFERENCES

[1] S. Nolfi and D. Floreano, *Evolutionary Robotics, The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA: MIT Press, 2000.

[2] J. Walker, S. Garrett, and W. Wilson, "Evolving controllers for real robots: A survey of the literature," *Adapt. Behav.*, vol. 11, no. 3, pp. 179–203, 2003.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

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

[5] S. Nolfi and D. Floreano, "Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution?" *Artif. Life*, vol. 4, pp. 311–335, 1998.

[6] E. Uchibe, M. Nakamura, and M. Asada, "Cooperative and competitive behavior acquisition for mobile robots through co-evolution," in *Proc. Genetic and Evolutionary Computation Conference*, 1999, pp. 1406–1413.

[7] M. Asada, K. F. MacDorman, H. Ishiguro, and Y. Kuniyoshi, "Cognitive developmental robotics as a new paradigm for the design of humanoid robots," *Robot. Auton. Syst.*, vol. 37, pp. 185–193, 2001.

[8] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Purposive behavior acquisition for a real robot by vision-based reinforcement learning," *Mach. Learn.*, vol. 23, pp. 279–303, 1996.

[9] T. Omata, "Learning with assistance based on evolutionary computation," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1998, pp. 2180–2186.

[10] G. Baldassarre, S. Nolfi, and D. Parisi, "Evolving mobile robots able to display collective behavior," *Artif. Life*, vol. 9, pp. 255–267, 2003.

[11] G. Baldassarre, D. Parisi, and S. Nolfi, "Coordination and behavior integration in cooperating simulated robots," in *Proc. 8th Int. Conf. Simulation of Adaptive Behavior: From Animals to Animats 8*, S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, Eds., 2004, pp. 385–394.

[12] M. Quinn, "Evolving communication without dedicated communication channels," in *Advances in Artificial Life: Proc. 6th Eur. Conf. Artificial Life (ECAL 2001)*, J. Kelemen and P. Sosik, Eds., pp. 357–366.

[13] M. Quinn, L. Smith, G. Mayley, and P. Husband, "Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors," *Philos. Trans. R. Soc. Lond. A, Math., Phys. Eng. Sci.*, vol. 361, pp. 2321–2344, 2003.

[14] D. Cliff and G. F. Miller, "Co-evolution of pursuit and evasion II: Simulation methods and results," in *Proc. 4th Int. Conf. Simulation of Adaptive Behavior: From Animals to Animats 4*, 1996, pp. 506–515.

[15] G. S. Hornby and B. Mirtich, "Diffuse versus true coevolution in a physics-based world," in *Proc. Genetic and Evolutionary Computation Conf.*, 1999, pp. 1305–1312.

[16] D. Floreano and S. Nolfi, "Adaptive behavior in competing co-evolving species," in *Proc. 4th Eur. Conf. Artificial Life*, 1997, pp. 378–387.

[17] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evol. Comput.*, vol. 5, no. 1, pp. 1–29, 1997.

[18] K. O. Stanley and R. Miikklainen, "Competitive coevolution through evolutionary complexification," *J. Artif. Intell. Res.*, vol. 21, pp. 63–100, 2004.

[19] M. Asada, H. Kitano, I. Noda, and M. Veloso, "Robocup: Today and tomorrow—what we have learned," *Artif. Intell.*, 1999.

[20] M. Asada, E. Uchibe, and K. Hosoda, "Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development," *Artif. Intell.*, vol. 110, pp. 275–292, 1999.

[21] A. Bajurnow and V. Ciesielski, "Layered learning for evolving goal scoring behavior in soccer players," in *Proc. 2004 Congr. Evolutionary Computation*, 2004, pp. 1828–1835.

[22] V. Ciesielski, D. Mawhinney, and P. Wilson, "Genetic programming for robot soccer," in *Proc. RoboCup 2001 Int. Symp.*, 2001, pp. 319–324.

[23] W. H. Hsu and S. M. Gustafson, "Genetic programming and multi-agent layered learning by reinforcements," in *Proc. Genetic and Evolutionary Computation Conf.*, 2002, pp. 764–771.

[24] W. H. Hsu, S. J. Harmon, E. Rodriguez, and C. Zhong, "Emprical comparison of incremental reuse strategies in genetic programming for keep-away soccer (late breaking paper)," presented at the *2004 Genetic and Evolutionary Computation Conf.*, Seattle, WA, 2004.

[25] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler, "Co-evolving soccer softbot team coordination with genetic programming," in *Proc. 1st RoboCup-97 Workshop at IJCAI'97*, pp. 115–118.

[26] P. Stone, *Layered Learning in Multiagent Systems*. Cambridge, MA: MIT Press, 2000.

[27] P. Stone, R. Sutton, and G. Kuhlmann, "Reinforcement learning for RoboCup-soccer keepaway," *Adapt. Behav.*, vol. 13, no. 3, pp. 165–188, 2005.

[28] S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone, "Evolving soccer keepaway players through task decomposition," *Mach. Learn.*, vol. 59, no. 1, pp. 5–30, 2005.

[29] M. Wiering, R. P. Salustowicz, and J. Schmidhüber, "Reinforcement learning soccer teams with incomplete world models," *Auton. Robots*, vol. 7, no. 1, pp. 77–88, 1999.

[30] R. Rubinstein, *Simulation and the Monte Carlo Method*. New York: Wiley, 1981.

[31] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. Cambridge, MA: MIT Press/ Bradford Books, 1998.

[32] D. Precup, R. S. Sutton, and S. Dasgupta, "Off-policy temporal-difference learning with function approximation," in *Proc. 18th Int. Conf. Machine Learning*, 2001, pp. 417–424.

[33] E. Uchibe and K. Doya, "Competitive-cooperative-concurrent reinforcement learning with importance sampling," in *Proc. 8th Int. Conf. Simulation of Adaptive Behavior: From Animals to Animats 8*, S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, Eds., 2004, pp. 287–296.

[34] E. Uchibe, M. Yanase, and M. Asada, "Behavior generation for a mobile robot based on the adaptive fitness function," *Robot. Auton. Syst.*, vol. 40, no. 2–3, pp. 69–77, 2002.

ABOUT THE AUTHORS

Eiji Uchibe received the Ph.D. degree in mechanical engineering from Osaka University, Osaka, Japan, in 1999.

He worked as a Research Associate with the Japan Society for the Promotion of Science, in a Research for the Future Program titled Cooperative Distributed Vision for Dynamic Three-Dimensional Scene Understanding. Then he joined Advanced Telecommunication Research Institute International (ATR) as a Researcher in 2001. Since 2004, he has been a researcher at the Neural Computation Unit, Okinawa Institute of Science and Technology, Promotion Corporation, Okinawa, Japan. His research interests are in learning robots, reinforcement learning, evolutionary computation, and computational neuroscience, and their applications.

Dr. Uchibe received the best philosophical paper award from the Simulation of Adaptive Behavior Conference (SAB 2004).

Minoru Asada (Fellow, IEEE) received the B.E., M. E., and Ph.D. degrees in control engineering from Osaka University, Osaka, Japan, in 1977, 1979, and 1982, respectively.

From 1982 to 1988, he was a Research Associate of Control Engineering, Osaka University. In April 1989, he became an Associate Professor of Mechanical Engineering for Computer-Controlled Machinery, Osaka University. In April 1995 he became a Professor in the same department. Since April 1997, he has been a Professor in the Department of Adaptive Machine Systems at the same university. From August 1986 to October 1987, he was also a Visiting Researcher at the Center for Automation Research, University of Maryland, College Park. He has also been a Research Director for the JST ERATO Asada Synergistic Intelligence Project since September 2005.

Dr. Asada received the 1992 best paper award of the IEEE/Robotics Society of Japan (RSJ) International Conference on Intelligent Robots and Systems (IROS92) and the 1996 best paper award of the RSJ. In 2001, he received the Commendation by the Minister of Education, Culture, Sports, Science and Technology, Japanese Government, as a person of distinguished service to enlightening people on science and technology. Since 2002, he has been the President of the International RoboCup Federation. He has been a research director for JST ERATO Asada Synergistic Intelligence project since September 2005.