Cooperative Behavior Acquisition in A Multiple Mobile Robot Environment by Co-evolution

Eiji Uchibe, Masateru Nakamura, Minoru Asada

Dept. of Adaptive Machine Systems, Graduate School of Eng., Osaka University, Suita, Osaka 565-0871, Japan

Abstract. Co-evolution has been receiving increased attention as a method for multi agent simultaneous learning. This paper discusses how multiple robots can emerge cooperative behaviors through co-evolutionary processes. As an example task, a simplified soccer game with three learning robots is selected and a GP (genetic programming) method is applied to individual population corresponding to each robot so as to obtain cooperative and competitive behaviors through evolutionary processes. The complexity of the problem can be explained twofold: co-evolution for cooperative behaviors needs exact synchronization of mutual evolutions, and three robot co-evolution requires well-complicated environment setups that may gradually change from simpler to more complicated situations so that they can obtain cooperative and competitive behaviors simultaneously in a wide range of search area in various kinds of aspects. Simulation results are shown, and a discussion is given.

1 Introduction

Realization of autonomous robots that organize their own internal structures to accomplish given tasks through interactions with their environments is one of the ultimate goals of Robotics and AI. Especially, emergence of cooperative behaviors between multiple robots has been receiving increased attention as a problem of multi agent simultaneous learning. Because, it seems difficult to apply conventional learning algorithms such as reinforcement learning to co-evolve cooperative agents since the environment including other agents may cause unpredictable changes in state transitions for learning agents.

Uchibe et al. proposed a reinforcement learning supported by system identification [10] and learning schedule [9] in multi agent environments. Their method estimates the relationships between learner's behaviors and other robot ones through interactions. However, in their method, only one robot may learn and other robots should have fixed policy in order for the learning to converge.

Recently, co-evolution has been receiving increased attention as a method for multi agent simultaneous learning. Existing methods have mostly focused on two competing individuals such as a prey and a predator. Cliff and Miller [2] have analyzed the relationship between a prey and a predator, and Floreano and Nolfi [3] have implemented real robot experiments which co-evolved prey and predator robots of which skills gradually leveled up under certain conditions. Luke et al. [7] apply the co-evolution technique to the soccer game to evolve teams each of which can be regarded as an individual and attempts to beat other teams, that is, co-evolution for competition.

In the realm of nature, we can see, however, various aspects of behaviors emerged from multi agent environments, not only competition but also cooperation, ignorance, and so on. That means there could be artificial co-evolution for other than competition. This paper discusses how multiple robots can obtain cooperative behaviors through the co-evolutionary process. As an example task, a simplified soccer game with three learning robots is selected and a GP (genetic programming) method [5, 6], a kind of genetic algorithms based on tree structure with more abstracted node representation than gene coding in ordinary GAs, is applied so as to experimentally evaluate obtained behaviors in the context of cooperative and competitive tasks. Each robot has its own individual population, and attempts to acquire desired behaviors through interactions with environment that is ever changing in the co-evolutionary process. The complexity of the problem can be explained twofold: 1) co-evolution for cooperative behaviors needs exact synchronization of mutual evolutions, and 2) three robot co-evolution requires well-complicated environment setups that may contribute to providing a wide variety of searching area from simpler to more complicated situations in which they seek for better strategies so that they can emerge cooperative and competitive behaviors simultaneously.

The rest of this article is organized as follows. First, we describe our views on co-evolution in the context of cooperative and competitive tasks. Next, we explain our example task, a simplified soccer game in which cooperative and competitive tasks are involved. Then, we give a brief explanation of the GP and setting parameters. Finally, the preliminary results of computer simulation are shown, and a discussion is given.

2 Co-evolution in cooperative tasks

Generally, we have following difficult problems in multi agent simultaneous learning:

1. **Unknown Policy**
   Learning agents do not know other agents' policies in advance, therefore they need to estimate them through observations and actions. What's the worse is that the agent policies may change through a learning process.
2. **Synchronized Learning**
   Mutual learning robots have to improve their learned policies simultaneously. If the opponent learning converged much earlier than itself, one robot could not improve its strategy against the difficult environment its opponent has already fixed.
3. **Credit Assignment**
   Credit assignment to learning robots for cooperation seems difficult. If the

credit involves group evaluation only, one robot may accomplish a given task by itself and others do just actions irrelevant to the task as they do not seem to interfere the one robot's actions. Else if only individual evaluation is involved, robots may compete others each other. This trade-off should be carefully dealt.

Co-evolution is one of potential solutions for the first problem by seeking for better strategies in a wide range of searching area in parallel. The second and third ones might be solved by careful designs of environmental setups and fitness functions. Emerging patterns by co-evolution can be categorized into three ones.

1. **Cycles of switching fixed strategies**
   This pattern can be often observed in case of a prey and predator which often shift their strategies drastically to escape from or to catch the opponent. The same strategies iterates many times and no improvements on both sides seem to be seen.
2. **Trap to local maxima**
   This corresponds to the second problem stated above. Since one side overwhelmed its opponents, both sides reached to one of stable but low skill levels, and therefore no change happens after this settlement.
3. **Mutual skill development**
   In certain conditions, every one can improve its strategy against ever-changing environments due to improved strategies by other agents. This is real co-evolution by which all agents evolve effectively.

As a typical co-evolution example, a competitive task such as prey and predator has been often argued [2, 3] where heterogeneous agents often change their strategies to cope with the current opponent one. That is, the first pattern was observed. In case of homogeneous agents, Luke et al. [7] co-evolved teams consisting of eleven soccer players among which cooperative behavior could be observed. However, co-evolving cooperative agents has not been addressed as a design issue on fitness function for individual players since they applied co-evolving technique to teams.

We believe that between one to one individual competition and team competition, there could be other kinds of co-evolution than competition. Thus, we challenge to evaluate how the task complexity and fitness function affect co-evolution processes in case of multi agent simultaneous learning for not only competitive but also cooperative tasks through a series of systematic experiments. First, we show the experiments for a cooperative task, that is, shooting supported by passing between two robots in 4.1 where unexpected cooperative behavior that can be regarded as the second pattern was emerged. Next, we add a stationary obstacle before the goal area into the first experimental set up in 4.2 where the complexity is higher and expected behavior was observed after longer generation changes than the previous one. Finally, we add an active learning opponent instead of the stationary obstacle to evaluate how both cooperative and competitive behaviors are emerged in 4.3. We have tried several fitness functions, and we may conclude that the same level fitness function among them seems better to co-evolve cooperative and competitive agents, and other ones tend to evolve only one side, that is the second pattern. In the following, we describe them in detail.

3 Task and assumptions

3.1 Environment and robots

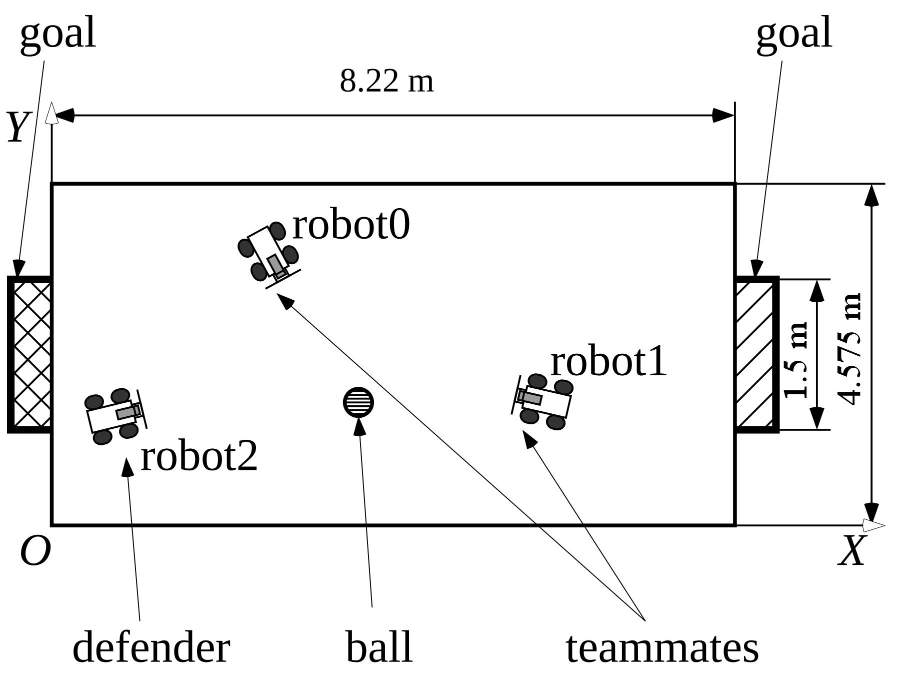


Fig. 1. Environment

Before explanation of the proposed method, we show a concrete task for reader's understanding of the method. We have selected 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 RoboCup Initiative [4]. We built an original soccer simulator which models real mobile robots we have been using so far in [1, 8, 9]. The environment consists of a ball and two goals, and a wall is placed around the field except the two goals. The sizes of the ball, the goals and the field are the same as those of the middle league of RoboCup. Figure 1 shows the size of the environment.

The robots modeled have the same body (power wheeled steering system) and the same sensor (on-board TV camera), that is, homogeneous agents. In this simulator, the robot can not obtain the complete information because of limitation of its sensing capability and occlusion of the objects.

3.2 Function and terminal sets

As sets of functions, we prepare the simple conditional branching function "IF_a_is_b" that executes its first branch if the condition "a is b" is true, otherwise executes its second branch, where a is a kind of image features, and b is its category. Table 1 shows the details of this function "IF_a_is_b".

Table 1. Function sets

a	ball, goal, other robot 0, other robot 1, $\cdots$
b	left, middle, right, small, medium, large, lost

Terminals in our task are actions that have effects on the environment. A terminal set consists of the following four behaviors :

1. `shoot` : the robot shoots a ball into the opponent goal based on the visual information about the ball and the opponent's goal.
2. `pass` : the robot kicks a ball to one teammate based on the visual information about the ball and other robots including the teammate.
3. `avoid` : the robot avoids collisions with other robots based on the visual information about them.
4. `search` : the robot searches the ball by turning to the left or right based on the visual information about the goal.

Although we design these behaviors by hand in this experiments, these primitive behaviors can be acquired by other learning algorithms such as ones in [1, 8, 9].

3.3 Fitness measure

One of the problems to apply an evolutionary algorithm is the design of fitness function which leads robots to purposive behaviors. We utilize the standardized fitness representation, that has a positive value. The smaller is the better (0.0 is the best). We first consider the following parameters to evaluate team behaviors such as cooperation between teammates and competition with opponents:

- $G(i)$: the total number of achieved goals for the team to which robot i belongs,
- $L(i)$: the total number of lost goals for the team to which robot i belongs.

With these parameters only, most robots tends to be idle (passive cooperation) except one that attempts at achieving the goal for itself, and therefore no active cooperation can be seen. Then, we introduce the following more individual e-valuation to encourage robots to interact with each other while to minimize the number of collisions:

- $K(i)$: the number of ball-kicking by robot i,
- $C(i)$: the number of collisions between robot i and other ones.

In addition to the above, the following is involved to make robots achieve the goal earlier.

- *steps* : the number of steps until one trial ends, where a step is defined as a time period for one action execution against the sensory input of a robot (1/30 [msec]).

The fitness function is calculated by linear combination of these parameters. In our case, the fitness value which the robot i receives is given by :

$$\begin{aligned} f_s(i) &= \alpha_k h(K(i), \beta) + \alpha_g h(G(i), T_{max}) + \alpha_l * L(i) \\ &\quad + \alpha_c * C(i) + \alpha_s * steps \qquad (1) \\ h(x,y) &= \begin{cases} y - x & \text{if } x < y \\ 0 & \text{otherwise} \end{cases}, \end{aligned}$$

where T_{max} denotes the maximum number of trials, and $\alpha_k \sim \alpha_s$ and β are constants. In the following experiments, we set $\alpha_k = \alpha_g = 1$, $\alpha_l = 0.5$, $\alpha_c = 0.05$, $\alpha_s = 0.0001$, and $\beta = 10$. If two or more individuals have the same fitness value, we prefer to one with more compact tree depth.

3.4 Other parameters in genetic programming

Other parameters in GP here are: the size of each population is 80, the number of generations for which the evolutionary process should run is 60, the maximum depth that must not be exceeded during the creation of a genetic tree is 10, and the maximum tree depth by crossing two trees is 25.

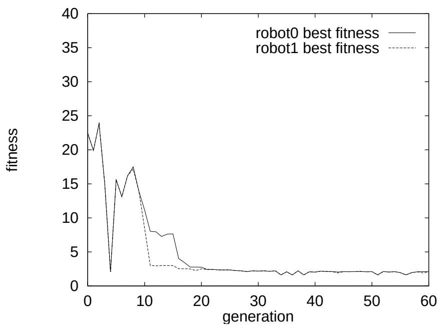
The best performing tree in the current generation will be moved unchanged to next generation. In order to select parents for crossover, we use tournament selection with size 10. The crossover probability is set to 95 %, reproduction probability is set to 5 % mutation probability is set to 10 %.

After each population selects one individual separately, the selected individuals participate in the game. We perform 20 games to evaluate them. One trial is terminated if the robot shoots a ball into the goal or *steps* exceed 1000. As a result, it needs 1600 trials to alter a new generation. The hardware used for the simulation is Sun SPARC Station Ultra2, which takes about one day to evaluate one experiment.

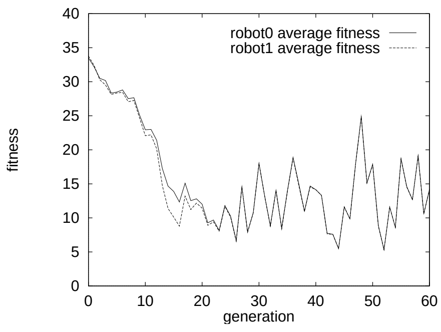
4 Simulation results

4.1 Two learners

At first, we demonstrate the experiments to acquire cooperative behaviors between two robots. Both robots belong to the same team, and they obtain the score if they succeed in shooting a ball into the goal. The number of function sets is 28(= 7(ball) + 2 × 7(two goals) + 7(teammate)).



(a) best fitness



(b) average fitness

Fig. 2. fitness in case of two learners

Figures 2 (a) and (b) show the results of evolution process in the case of two robots. The fitness values of the best individuals converged in generation 20 (See (a)). The tree depths and the numbers of nodes of the best-robot 0 and 1 are (29, 637) and (21,611), respectively. In this case, robot 0 does not kick the ball by itself but shakes its body by repeating the behaviors `search` and `avoid`. On the other hand, robot 1 approaches the ball and passes the ball to robot 0. After robot 0 receives the ball, it executes `shoot` behavior to shoot the ball into the goal. However, robot 1 approaches the ball faster than robot 0. As a result, robot 0 shoots the ball into the goal while robot 1 avoid collisions with robot 0. The successful behaviors are shown in Figure 3.

Although we tested several fitness functions, the resultant behaviors are similar to the behavior shown in Figure 3. In this task, robot 0 does not kick the ball toward robot 1 through all the generation. We suppose that the reasons why they acquire the cooperative behaviors as shown in Figure 3 are as follows :

– In order for robot 0 to pass the ball to robot 1, robot 1 has to shoot the ball which is passed back from robot 0. This means that in this situation the development of both robots needs to be exactly synchronized. It seems very difficult for such a synchronization to be found.

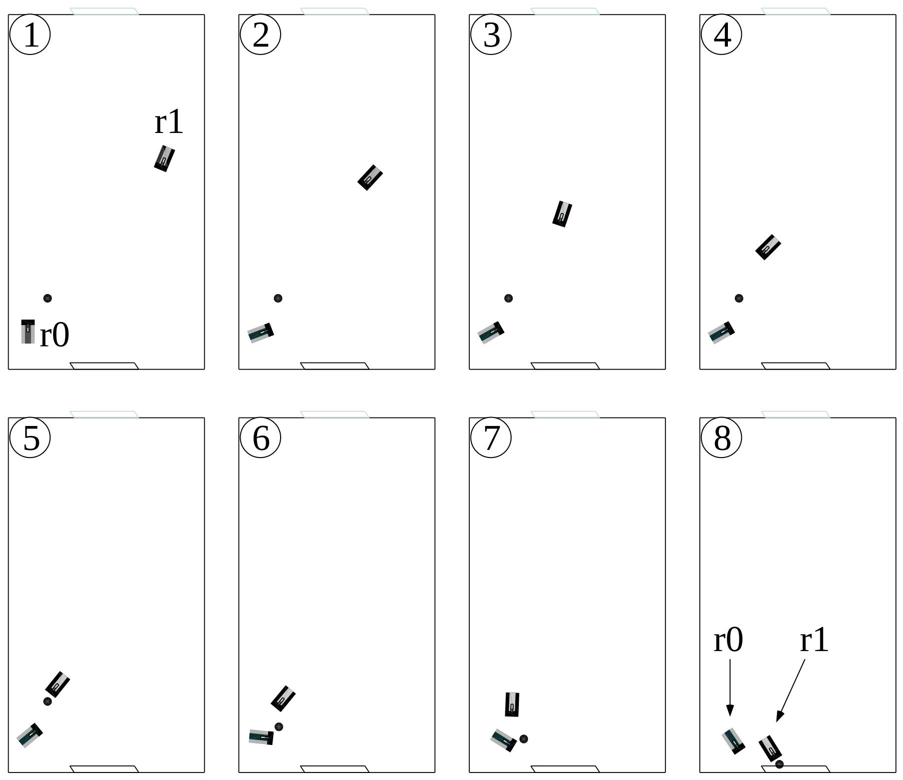


Fig. 3. Two robots (r0 : robot 0, r1 : robot 1) succeed in shooting a ball into the goal

– Robot 1 may shoot the ball by itself whichever robot 0 kicks the ball or not. In other words, robot 1 does not need the help by robot 0.

In this task, robots 0 and 1 do not have even complexity of the tasks. As a result, the behavior of robot 1 dominates this task while robot 0 does not improve its own behavior. This is the second pattern explained in 2

4.2 Two learners and one stationary robot

Next, we add one robot as an stationary obstacle to the environment described in section 4.1. The number of function sets is $35(= 7(\text{ball}) + 2 \times 7(\text{two goals}) + 2 \times 7(\text{teammate and opponent}))$.

Figures 4 (a) and (b) show the results of evolutionary process where a good synchronization between the best individuals of robots 0 and 1 can be seen (See (b)). The tree depths and the numbers of nodes of the best-robots 0 and 1 are (11,63) and (19, 577), respectively. Although both learning robots are placed in

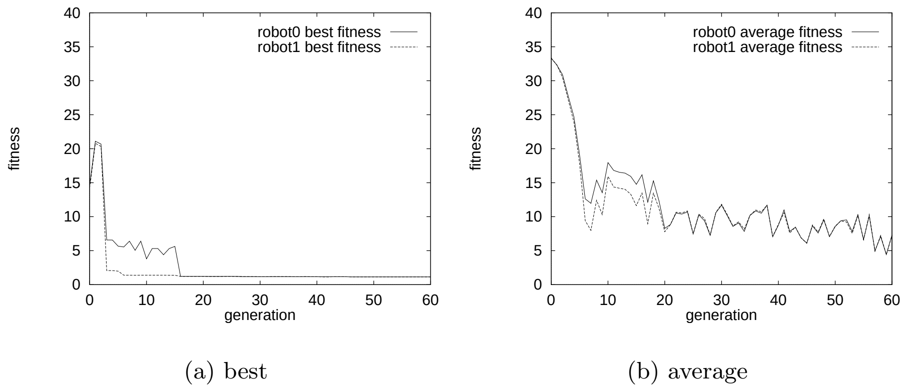


Fig. 4. fitness in case of the two learners and one stationary robot

the same way as in the previous experiments, the acquired cooperative behaviors are quite different because of the one stationary opponent. Since it becomes more difficult for robot 1 to shoot the ball for itself because of the existence of robot 2, robot 1 has to evolve behaviors with robot 1 synchronously. In other words, the complexity of the task for robot 0 increased around the same level of robot 1.

A history of evolution is as follows. Although both robots 0 and 1 chase after the ball and kick the ball until generation 4, robot 0 begins to kick the ball towards robot 1. However robot 1 can not shoot the ball from the robot 0 directly because robot 0 can not pass the ball to the robot 1 precisely. Therefore, robot 1 kicks the ball to the wall and continues to kick the ball to the opponent's goal along the wall until generation 15. After a number of generations, both robots improve their own behaviors and acquire cooperative behaviors shown in Figure 5 in generation 61, where robot 0 kicks the ball to the front of robot 1, then robot 1 shoots the ball into the opponent's goal. Although it intends to shoot the ball for itself, robot 0 makes a way for robot 1 to avoid collisions with other robots. As a result, both robots improve the cooperative behaviors synchronously. This is a kind of the third pattern described in 2

4.3 Three learners

Finally, we test the co-evolution among three robots. That is, robot 2 added in section 4.2 evolves its behavior with robots 0 and 1 simultaneously. The difference from sections 4.1 and 4.2 is involvement of competition between robot 2 and robots 0 and 1. The number of function sets is as many as the case of section 4.2.

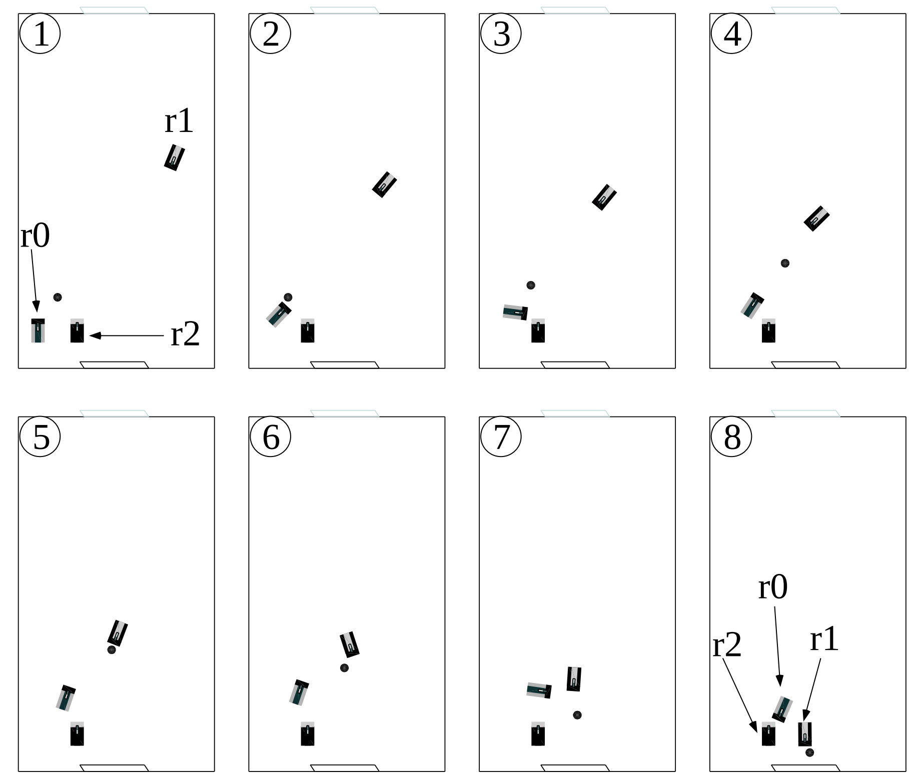
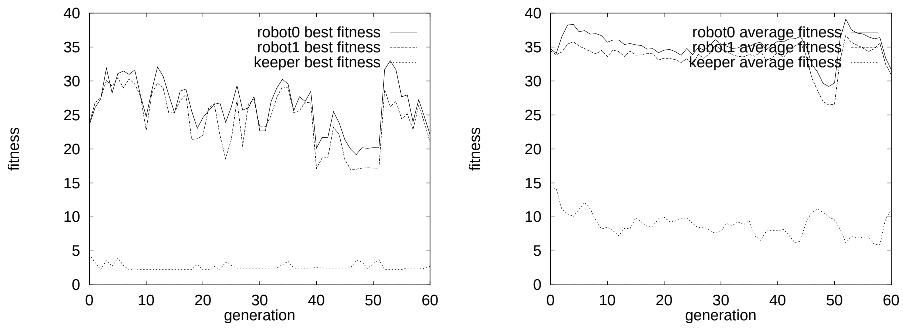


Fig. 5. Two robots (r0 : robot 0, r1 : robot 1) succeed in shooting a ball into the goal against the stationary keeper (r2 : robot 2)

We prepare a fitness function in which $\alpha_g = 0$ (no term for achieving goals) in eqn. (1) to evolve robot 2 as a keeper. Figures 6 (a) and (b) show the results of evolution process in case of this fitness function. Because it seems simple for robot 2 to save the ball from the robots 0 and 1 by shaking its body in front of the goal, the behavior of robot 2 comes to dominate the game in the early generation. Therefore, both robots 0 and 1 obtain the suboptimal behavior because of the low fitness. This is also the second pattern.

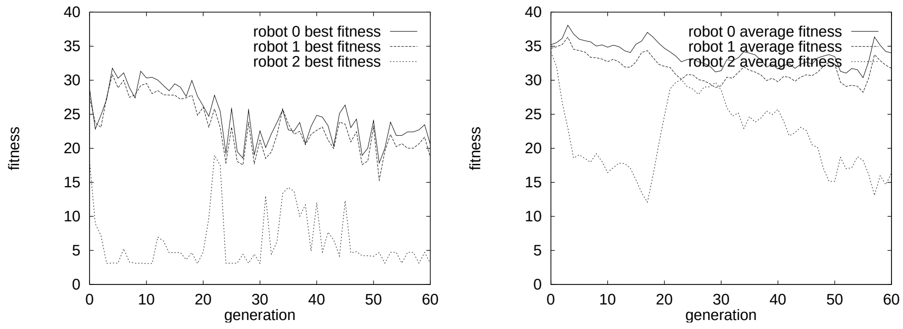
Then, we setup the same fitness function (eqn. (1)) so that we make robots 0,1 and 2 equal. The results are shown in Figure 7. As compared with the only cooperative tasks in section 4.2, fitness values rather oscillate than stay stable. The tree depths and the numbers of nodes of the best-robot 0, 1, and 2 are (24,1143), (15, 1093) and (21, 749), respectively.

We can see two typical settlements in this three-robot soccer game. One is the same behaviors described in section 4.2 : robot 0 kicks the ball toward robot



(a) best (b) average

Fig. 6. fitness in case of three learners (different fitness function)



(a) best (b) average

Fig. 7. fitness in case of three learners (same fitness function)

1, then robot 1 shoots the ball into the goal avoiding collisions with robot 2 (See Figure 8). The other one is that robot 2 intercepts the ball and shoots the ball into the goal (See Figure 9). The ratio between the former and the latter is about 25 % : 75 %. The aim of robot 0 is to pass the ball to robot 1 while the aim of robot 2 is going to intercept the ball. It depends on each other for robots 0 and 2 to achieve each goal. However, robot 2 can observe the ball and the opponent's goal at the same time and it may shoot the ball by itself while robot 0 needs to pass the ball to robot 1. As a result, we suppose that the predominance of robot 2 may be caused by the different complexity of the given tasks, that is, task complexity for robots 0 and 1 is higher than that for robot 2.

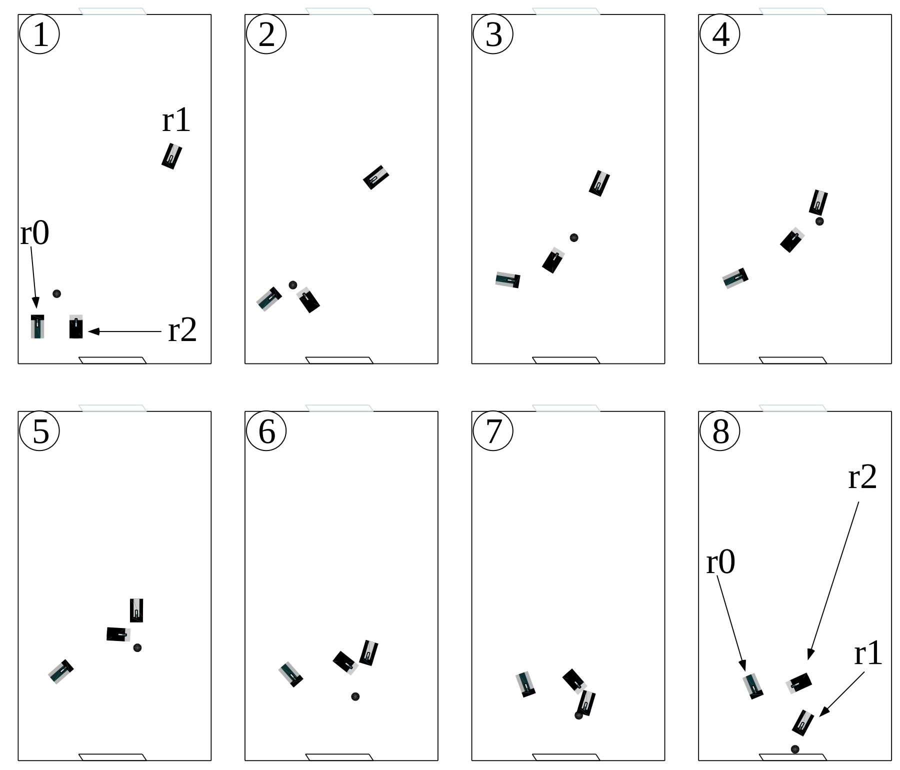


Fig. 8. Two robots (r0 : robot 0, r1 : robot 1) succeed in shooting a ball into the goal against the keeper (r2 : robot 2)

5 Concluding remarks

This paper showed how co-evolution technique could emerge not only competitive behaviors but also cooperative ones through a series of experiments in which two or three robots play a simplified soccer game. In order to co-evolve cooperative agents, it should be noted that robots must synchronize their evolutionary processes. Otherwise, there are many traps to local maxima (suboptimal strategies) as we can see in 4.1.

In case of more complicated situation (three agents and both cooperation and cooperation are involved), the task complexity should be equal to all agents

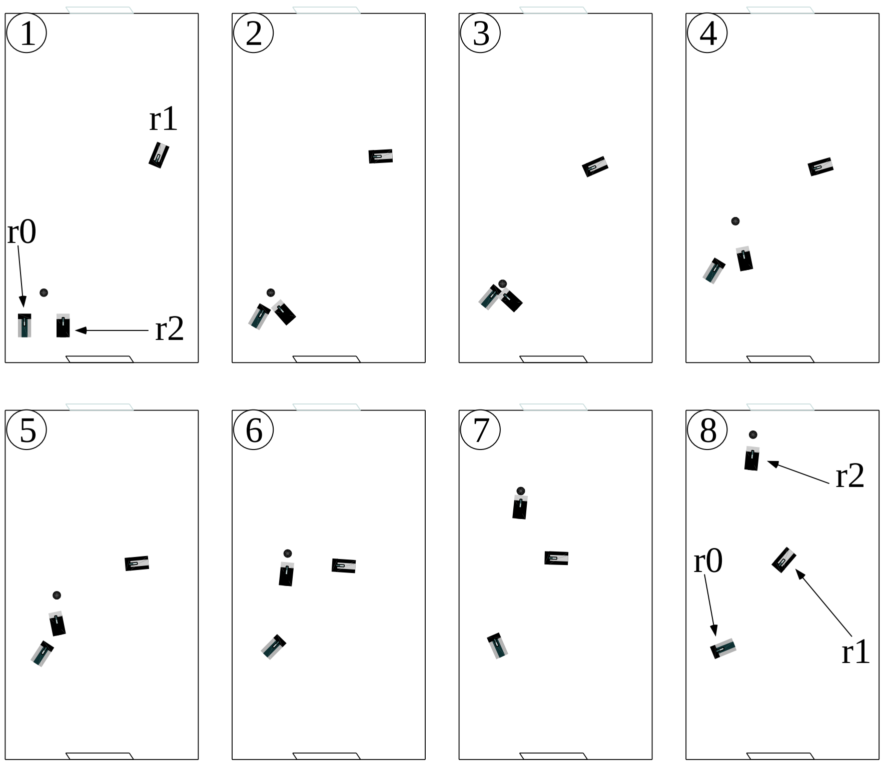


Fig. 9. The keeper (r2 : robot 2) succeeds in shoot a ball into the goal against the two robots (r0 : robot 0, r1 : robot 1)

so as to co-evolve cooperative and competitive agents simultaneously. This also suggests that the environment itself should co-evolve from simpler to more complicated situations to assist the development of desired skills of cooperations and competitions. Otherwise, co-evolution is prone to be settled into suboptimal strategies as shown in 4.3.

More systematic understanding is, however, needed to make clear what are necessary and sufficient conditions to lead co-evolutionary processes to successful situations. Design issues of environments including agents, tasks, and fitness functions are our future work. Also, we are planning to implement real experiments to check the validity of the proposed method and the obtained behaviors.

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. D. Cliff and G. F. Miller. Co-evolution of pursuit and evasion II : Simulation methods and results. In *Proc. of the 4th International Conference on Simulation of Adaptive Behavior: From Animals to Animats 4.*, pages 506–515, 1996.
3. D. Floreano and S. Nolfi. Adaptive behavior in competeing co-evolving species. In *Fourth European Conference on Artificial Life (ECAL97)*, pages 378–387, 1997.
4. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. Robocup a challenge problem for ai. *AI Magazine*, 18(1):73–85, 1997.
5. J. R. Koza. *Genetic Programming I : On the Programming of Computers by Means of Natural Selection.* The MIT Press, 1992.
6. J. R. Koza. *Genetic Programming II : Automatic Discovery of Reusable Programs.* The MIT Press, 1992.
7. S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *Proc. of the RoboCup-97 Workshop at the 15th International Joint Conference on Artificial Intelligence (I-JCAI97)*, pages 115–118, 1997.
8. E. Uchibe, M. Asada, and K. Hosoda. Behavior coordination for a mobile robot using modular reinforcement learning. In *Proc. of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1329–1336, 1996.
9. E. Uchibe, M. Asada, and K. Hosoda. Cooperative behavior acquisition in multi mobile robots environment by reinforcement learning based on state vector estimation. In *Proc. of IEEE International Conference on Robotics and Automation*, 1998.
10. E. Uchibe, M. Asada, and K. Hosoda. State space construction for behavior acquisition in multi agent environments with vision and action. In *Proc. of International Conference on Computer Vision*, pages 870–875, 1998.

This article was processed using the LaTeX macro package with LLNCS style