International Journal of Fuzzy Logic and Intelligent Systems, vol. 4, no. 3, December 2004 pp. 327-336

Learning of Emergent Behaviors in Collective Virtual Robots using ANN and Genetic Algorithm

Kyung-Dal Cho

Dept. of Computer Science & Engineering, Chung-Ang University, Korea

Abstract

In distributed autonomous mobile robot system, each robot (predator or prey) must behave by itself according to its states and environments, and if necessary, must cooperate with other robots in order to carry out a given task. Therefore it is essential that each robot have both learning and evolution ability to adapt to dynamic environment. This paper proposes a pursuing system utilizing the artificial life concept where virtual robots emulate social behaviors of animals and insects and realize their group behaviors. Each robot contains sensors to perceive other robots in several directions and decides its behavior based on the information obtained by the sensors. In this paper, a neural network is used for behavior decision controller. The input of the neural network is decided by the existence of other robots and the distance to the other robots. The output determines the directions in which the robot moves. The connection weight values of this neural network are encoded as genes, and the fitness individuals are determined using a genetic algorithm. Here, the fitness values imply how much group behaviors fit adequately to the goal and can express group behaviors. The validity of the system is verified through simulation. Besides, in this paper, we could have observed the robots' emergent behaviors during simulation.

Key words : Virtual Robot System, Pursuit Domain, Artificial Neural Network(ANN), Artificial Life, Multi-robot

1. Introduction

When facing difficult problems, we can often find the solution from creatures in nature. From this kind of attempts, a research area of artificial life (AL) came into being by C.G. Langton in 1987, in purpose of unifying hitherto individually achieved research results of living creatures and further activating those researches [1-4].

In engineering aspects, the goal of the artificial life is to incarnate the unique behaviors or phenomena of living creatures in nature onto artifacts like computers. We expect the artificial life can provide a useful methodology for the virtual robot learning which is full of autonomy and creativity. One of basic concept of the artificial life is "emergence". According to its mechanism, the emergence can be divided into several classes. Most of them determine behaviors and interaction rules of lower-level components. The local interaction between components generates global orders or behaviors [5-13].

The present paper develops a pursuit system that is mostly applied to the area of robot and robot in order to apply an emergent characteristic of the artificial life into machine learning. The pursuit system contains robots and preys. In the system, robots catch the preys escaping from them through evolution. For the simulator of the system, we have modeled the structure of the robot with the artificial neural network and evolved the structure of the neural network with the genetic algorithm. The prey is given only pre-knowledge of direction identification to move.

Numerous researches about the autonomous mobile robot control in the pursuit system have been studied. Nolfi and Foreano [23] simulated pursuit system with two robots (a predator and a prey) in real environments. They evolved both robots reciprocally with genetic algorithm. Yasuo and Shin [15] controlled the autonomous mobile agents using reinforcement learning. Kam-Chuen [18] evolved the autonomous mobile agents with the genetic algorithm. By Il-Kwon [19], both fuzzy controller and genetic algorithm were used. The fuzzy function displayed agent's position, and the genetic algorithm was used for learning. The reinforcement learning method is to develop agent's behaviors by means of the interrelationship with environment and resulting reinforcement signals. It can guarantee learning and adaptability without precise pre-knowledge about environments. However, its critical weak point is the difficulty of learning when the rewards of taken actions are not instantly computed. Takayuki and Yasuo [14,15] have a problem in applying to the open environments which is dynamically changing since once a agent's role is decided, then it is fixed. By Il-Kwon [19], both agents and prey move randomly, and the genetic algorithm helps to prevent the collision between agents. However, agents were not intelligent to do learning for themselves. Also, since the gene structure was represented by fuzzy member functions, it took much time to resolve pursuing problems.

Therefore, in this paper, to resolve those problems, we apply the artificial life method in our system. We model the robot

Manuscript received Nov. 18, 2003; revised Jun. 17, 2004

structure with the artificial neural network that is easy to model both robots and use the genetic algorithm for robot learning, which results in giving intelligence to robots in their pursuing movement toward the prey. The prey has only pre-knowledge to identify distance to avoid the robots. We have designed a virtual environment where 20 robots and a prey. Furthermore, various selection methods of the genetic algorithm are considered for simulation.

The rest of this paper is organized as follows. In chapter 2, the related works of this paper and is described. Chapter 3 describes the evolution of virtual robot in pursuit system. In chapter 4, simulation and evaluation are described. Finally, conclusion is presented in chapter 5.

2. Related works

2.1 Genetic Algorithm

A genetic algorithm (GA) is a method to obtain an optimal solution by applying a theory of biological evolution [21,22]. GAs generally consist of three fundamental operators: reproduction, crossover and mutation. Given an optimization problem, simple GAs encode the concerned parameters into finite bit strings, and then run iteratively using the three operators in a random way but based on the fitness function evolution to perform the basic tasks of copying strings, exchanging portions of strings as well as changing some bits of string, and finally find and decode the solutions to the processing of a GA. The following paragraphs explains the details:

(1) Coding the parameters

It has become a common way to translate the parameters into binary bit strings, Several parameters are coded into one long string.

(2) Initial generation.

It always begins by randomly generating an initial population of N strings which length is m bits. The population size N is a compromising factor. Large N increases the possibility of including the solution in the first few generations, but decreases the running speed of the GAs mentioned above. The string length m determines the resolution. It has long been recognized that because genetic algorithms perform a global search of a solution space, the encoded bit string lengths should be kept as short as possible, since the size of the search space increases exponentially with string size.

(3) Fitness evaluation

In the current generation, each of the strings is decoded into its corresponding actual parameter, Then, these parameters are sent to a judgment machine which yields a measure of the solution's quality, evaluates with some objective functions and then assigns individually with fitness values.

(4) Reproduction

Reproduction is a process by which the strings with larger fitness values can produce, accordingly with higher probabilities, large number of their copies in the new generation.

(5) Crossover

Crossover is a process by which the systematic information exchange between two strings is implemented using probabilistic decisions. In a crossover process, two newly reproduced strings are chosen from the mating pool and arranged to exchange their corresponding portions of binary strings at a randomly selected partitioning position along them. This process can combine better qualities among the preferred good strings.

(6) Mutation

Mutation is a process by which the chance for the GA to reach the optimal point is reinforced through just an occasional alteration of a value at a randomly selected bit position. The mutation process may quickly generate those strings which might not be conveniently produced by the previous reproduction and crossover processes. Mutation may suddenly spoil the opportunity of the current appropriate generation, so, this process usually occurs with a small probability and is complementary to reproduction and crossover.

(7) Iteration

The GA runs iteratively repeating the process (3)-(7) until it arrives at a predetermined ending condition. The speed of iteration depends on not only the population size N and the string length m but also the selection of probabilities. Finally, the acceptable solution is obtained and decoded into its original pattern from the resulting binary strings.

2.2 Multi-robot System

Multi-robot systems differ from single-robot systems in that several robots exist which model each other's goals and actions. In the fully general multi-robot scenario, there may be direct interaction among robots. From an individual robot's perspective multi-robot systems differ from single-robot systems most significantly in that the environment's dynamics can be affected by other robots. In addition to the uncertainty that may be inherent in the domain, other robots intentionally affect the environment in unpredictable ways. Thus, all multi-robot systems can be viewed as having dynamic environments [24, 25].

In the pursuit domain, communication creates new possibilities for predator behavior. Since the prey acts on its own in the pursuit domain, it has no other robots with which to communicate. However the predators can freely exchange

information in order to help them capture the prey more effectively. The current situation is illustrated in Figure 1. When using a "distributed" strategy, the robots are still homogeneous, but they communicate to insure that each moves toward a different capture position. In particular, the predator farthest from the prey chooses the capture position closest to it, and announces that it will approach that position. Then the next farthest predator chooses the closest capture position from the remaining three, and so on. This simple protocol encourages the predators to close in on the prey from different sides. A distributed strategy, it is much more effective than the local policy and does not require very much communication.

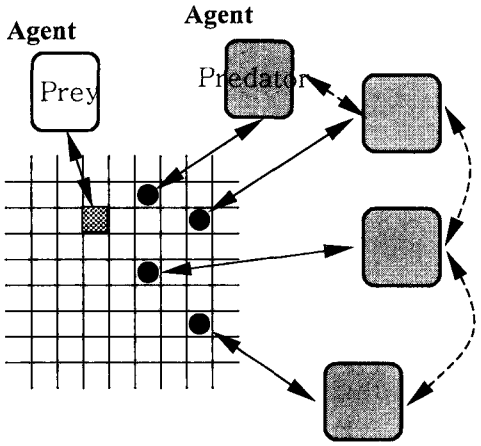


Figure 1. The pursuit domain with homogeneous communication robots

2.3. Artificial Neural Network(ANN)

The artificial neural network is consisted of many units that represent neurons. Each unit is a basic unit of information process. Units are interconnected via links that contain weight values. Weight values help the neural network to express knowledge. The neural network is divided into three layers: input layer, hidden layer, and output layer. The input layer transfers input signals into the hidden layer or the output layer. The hidden layer transfers signals from the input layer into the output layer. The output layer outputs transferred signals. Units can be regarded as nodes consisting of the neural network. One node can receive signals from other nodes and transfer specific signal into other nodes. The transfer rules can be described as next [12].

Step 1) All input signals and corresponding weight values are summed.

$g(x)=\Sigma x_i \bullet w_i$ (x_i : input signal, w_i : weight value)

Step 2) Output value is computed by using activation function.

$y=g(x)$ (y: output signal, g(x): activation function)

3.Emergent behavior implementation of virtual robots

3.1 Virtual environment and the structure of the system

Virtual environment of the pursuit system is a 10 by 10 lattice where up-and-down and right-and-left are connected each other as shown and robots and preys co-exist, as shown in Figure 2. Accordingly, if an robot or a prey move continuously toward one direction, it comes back to its origin. One any spot, there exists only one robot or one prey. In this paper, the prey is given the pre-knowledge that can identify and avoid robots in 4 directions. If it cannot move any more, it can stay on a spot.

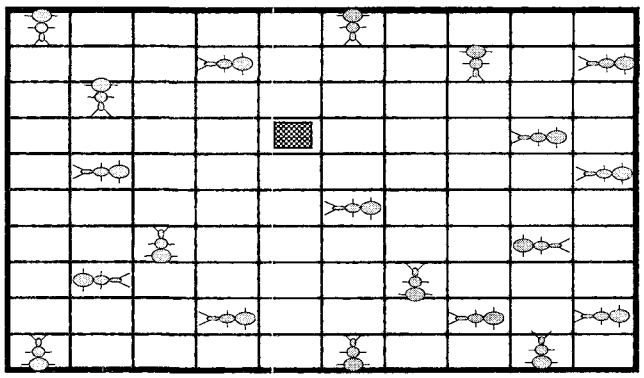


Figure 2. A lattice environment

The structure of whole system is described in Figure 3.

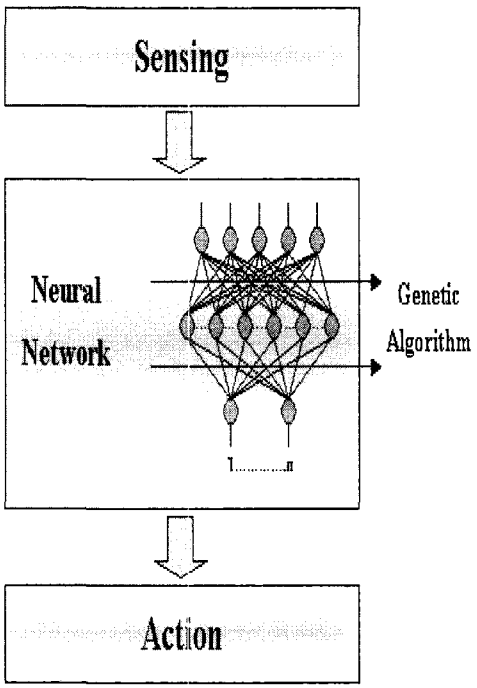


Figure 3. Structure of the pursuit system

An robot can contain several sensory organs. It perceives limited environments with those organs and behaves. In figure 3, at sensing, the input data of the positions of the robot input. The second stage constructs the learning structure of robot the artificial neural network. In simulation, 32 nodes are used: 8 nodes for prey identification, 8 nodes for robot identification, and 16 nodes that are exactly duplicated from the nodes of robot. All nodes are connected in parallel. Thus, this paper can apply a learning algorithm with parallel search structure. From the input nodes, the nodes in the medium layer use weight values that are transmitted to the output layer to generate robot's behaviors. Output layer consists of 5 nodes. The neural network makes the learning of weight values in the medium layer. In addition, the genetic algorithm makes the learning of the structure of the neural network. The learning results generate the robot's behaviors in the action stage where there exist 5 actions: catching prey and movements of left, right, up, and down.

3.2 Robot's structure

Any robot can identify the existence of other robots or preys within a defined distance, and its behavior is to catch and encircle a prey or to move by 1 toward 4 directions. Each robot contains several sensory organs, and its behavior uses those organs.

Figure 4 briefly describes the processes of an robot's identifying and behaving with the environments within a limited distance with its sensory organs.

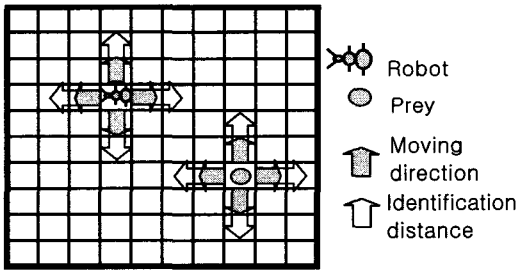


Figure 4. Identification distance and moving direction of artificial organisms. Both robot and prey can identify distance by 2 spaces and move one space at a time.

3.3 Genetic information

A robot possesses its genetic information that is used to construct a neural network. The neural network determines each robot's behaviors. The input nodes of the neural network are connected to sensory organs so that they receive inputs from robot's environments. The output nodes output signals that can activate or inhibit specific behaviors. We assume that the genetic information is made up of binary string, and its length is fixed. Shown in Figure 5, the genetic information is interpreted and utilized to construct a neural network that generates unique behavior patterns.

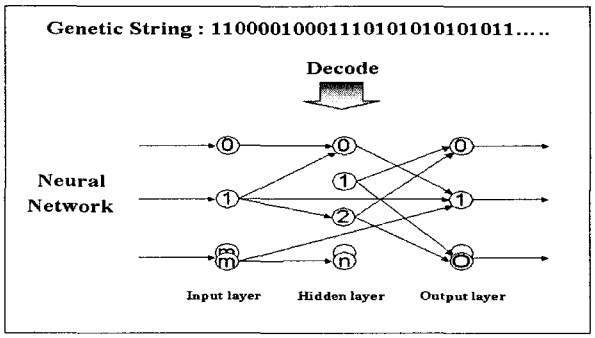


Figure 5. Construction of a neural network from genetic information

Figure 5 displays the neural network consisting of 32 input units, 21hidden units, and 5 output units. In simulation, the neural network contains 32 input units. Among them, 8 nodes are utilized for the robot to check the position of robots within 2 spaces of left, right, up, and bottom. Another 8 nodes are utilized for robot to check the position of the prey within 2 spaces of left, right, up, and bottom. The distributed genetic algorithm where each robot exchanges genes through communication can improve the robot's capability from superior genes learned in different environments. Before their moving, robots repeat 1) inputting sensor values to the neural network, 2) computing output values, and 3) behaving the action corresponding to the maximum output value.

The construction of a neural network with binary string uses connection descriptors as in Collins work [12]. One connection descriptor connects one node to another node. To represent one connection descriptor needs three fields;

1) From-Field: represent the number of unit (node) to receive initial stimuli
2) To-Field: represent the number of node to transfer the stimuli
3) Weight-Field: represent how much the stimuli can be amplified or inhibited.

We assume the structure of a connection descriptor as in Table 1.

<Table 1> The structure of a connection descriptor

From-Field	To-Field	Weight-Field
7bits	7bits	6bits

In <Table 1>, from-field and to-field are represented by 7 bits that can express 128 nodes. The weight field can be

represented by 6 bits. The first bit is the sign bit. If gene information bits become longer, the complexity increases. Proper size of weight bits is required to save operation time and memory.

When a genetic string has next values, each field of the first connection descriptor can be expressed as:

Genetic string: 0000000 0000001 10001.....

From-Field: 0000000 (indicate unit 0)
To-Field: 0000001 (indicate unit 1)
Weight: 100001 (indicate connection weight -1).

Figure 6 describes the connection between units.

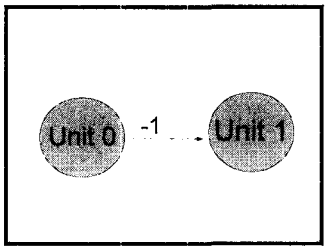


Figure 6. Connection between nodes

If the length of a genetic string is n, the length of a connection descriptor is m, n/m number of connections can be described between units.

Table 2 describes an algorithm generating the structure of the neural network from the input of gene information.

<Table 2> Neural network structure generation algorithm

```
neural_network_construct(chromosome)
from_length=7;
to_length=7;
weight_length=6;
matrix[*][*]=0;
for each descriptor in chromosome do
from=from_f(chromosome);
to=to_f(chromosome);
weight=weight_f(chromosome);
matrix[from][to]+=weight;
endfor
return(matrix);
```

3.4 Emergent behavior evolution strategy

In simulation, we consider total and tournament competition methods to know of the evolution for emergent behaviors. The total competition method runs all kind of robots consisting of their specific generations during a fixed time interval and generates offsprings as next generation according to the proportion to accumulated fitness values. As operators to generate next generation robots, reproduction, crossover, and mutation are used. The algorithm of total competition to generate next generation robots is described next.

<Total competition algorithm>
Step 1) Run all robot classes during a fixed time interval.
Step 2) Select some classes survivable in next generation according to the proportion to fitness values.

$$F^i = \sum_{t=0}^{T} f_1^i(t) + f_2^i(t)$$

where, T= life time
f_1^i*={1 if ith robot is adjacent (up, down, right, and left) to the prey, 0 otherwise.},*
f_2^i*={1 if ith robot is in 'catch' action with other robots, 0 otherwise}*
'catch' denotes the state where robots are encircling the prey.

Step 3) Reproduce or crossover selected classes.
Step 4) Mutate newly generated classes.

Tournament competition method constructs 4 groups where each group consists of 4 robots from current generation, selects one superior robot from each group in the first two groups, and selects one inferior robot from each group in the rest two groups. Then, subsist the selected two superior robots (via reproduction, crossover, or mutation) and exterminates the selected two inferior robots. The tournament competition method, where parents and offsprings co-exist in one generation, is a kind of steady-state algorithm, and its feature is the competition of robots in only a small group. The algorithm to generate next generation in the tournament competition method is described as follow.

<Tournament competition algorithm>
1) Randomly select 4 groups where each group consists of 4 robots.
2) Select two groups, run during the fixed time interval, and select one superior robot from each group.
3) Select the rest of two, run during the fixed time interval, and delete one inferior robot from each group.
4) Reproduce or crossover those superior elements and generate their off-springs.
5) Mutate newly-generated robots.

Figure 7 describes the tournament competition method.

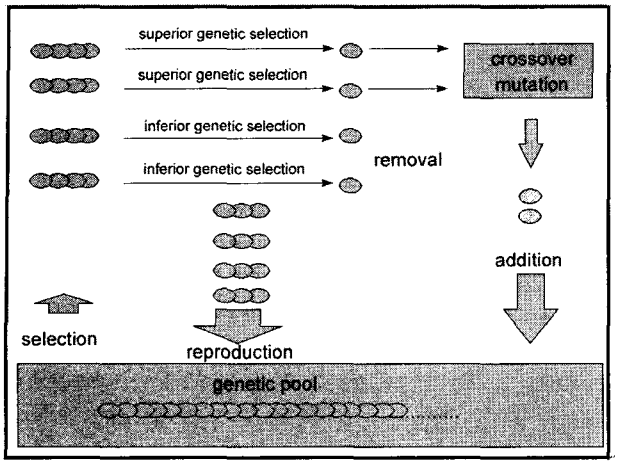


Figure 7. Tournament competition method

This paper simulates as considering 3 methods for the tournament competition method: 1) random selection, 2) linear area selection, and 3) 2-D area selection. Random selection selects randomly some classes from the general group. Linear area selection is to select robots adjacent to a specified range in a robot group arranged in one-dimension. 2-D selection is to select robot adjacent to a specified surrounding in 2-dimension space. Figure 8 describes the tournament competition method, and the white colored robots can be selected as same tournament group.

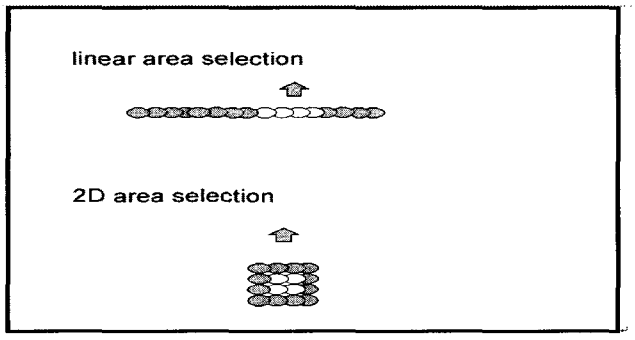


Figure 8. Linear, 2D area selection method

4. Simulation and evaluation

4.1 Simulation

This chapter describes emergent behaviors of autonomous mobile robots in pursuit system. The parameters used for simulation are described in Table 3.

The prey is given a primitive priority in moving directions as left, right, up, and down with decreasing order of priority. Since robots are prey are supposed to be placed on different spots (one on one) on the lattice, there will not occur any collision among them. Each robot continuously evolves during its life time (20 movements), regardless of whether the prey is caught or not.

<Table 3> Simulation parameters

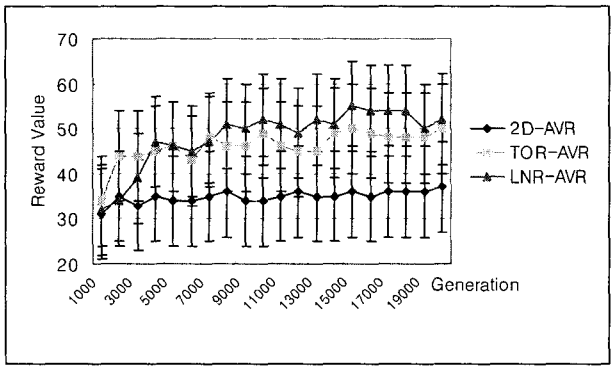
• Number of specified classes of robots :20
• Range of robot sensor : up-and-down or right-and-left by 2 from its location
• Number of prey : 1 (always escape from robots in 4 directions)
• Escape strategy of prey : Move into the position where escaping robots as much as possible from current location or when moving to up-and-down or right-and-left by 1
• Value reward of robot : Get one point when the robot is adjacent to prey, and get one point when the adjacent robot acts 'EAT'
• Activity time : 20 (robot acts 20 times)
• Crossover rate : 0.6 (the rate that the selected two can be crossover)
• Mutation rate : 0.03 (the rate that the least unit of genetic information can be mutated)
• Maximum number of input unit at the neural network : 32
• Maximum number of hidden unit at neural network :21
• Maximum number of output unit at the neural network : 5
• Environment : 10X10 lattice (connected to up-and-down and right-and-left)

The general simulation procedure of the pursuit system is described as below.

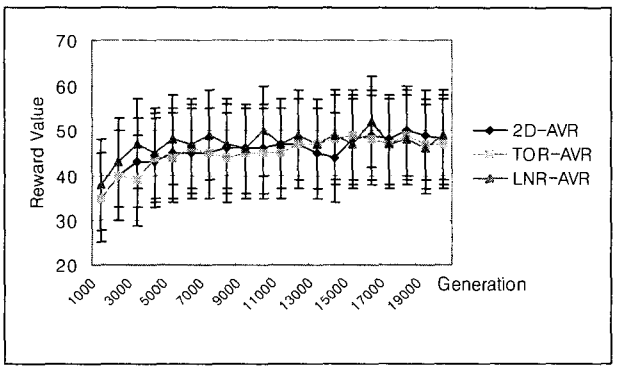
<Step 1> Initialize all genetic information of the gene pool.
<Step 2> Interpret all kinds of genetic information, construct neural network, initialize environments (position the prey and robots on the lattice), and construct N of robot brains.
<Step 3> Move N robots during fixed time. Robots can obtain fitness values according to proper actions.
<Step 4> When run time is completed, the fitness values of the corresponding class are cumulated based on each robot's fitness values.
<Step 5> The fitness values of all robots are obtained through <Step 2-4>.
<Step 6> With the rate proportional to the fitness values, gene classes are selected, and next generation robots are re-produced by crossover and mutation.
<Step 7> Go to Step 2.

In simulation, the paper assigned different rates of mutation and crossover operators of genetic algorithm and applied to different competition methods. Figure 9 represents the evaluation of different mutation rates (0, 0.03, and 0.07) when crossover rate is fixed to 0.6. On the contrary, figure 10 represents the evaluation of different crossover rates (0.3, 0.6, and 0.8) when the mutation rate is fixed to 0.03. For the efficiency at the total competition method, we force one of the selected superior robots to be reproduced in the next generation. For the tournament competition method, we simulate with three

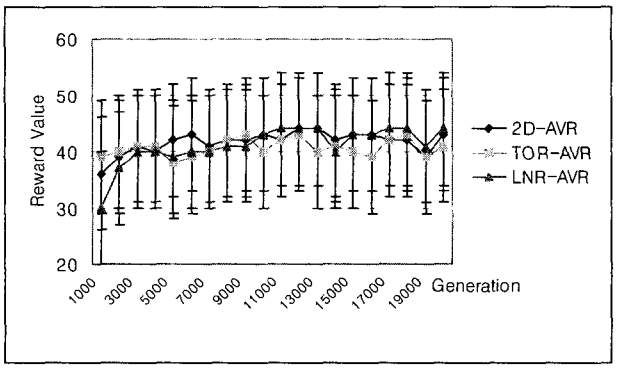
selection methods of random selection, linear area selection, and 2-D area selection. In figure 9 and 10, the horizontal axis represents the number of generation, and the vertical axis represents the average number of caught prey. In figure 9, 2D-AVR plot represents the average number of caught prey using 2-dimension area selection method. TOR-AVR plot implies the use of the random selection method. LNR-AVR plot implies the use of linear area selection method.



(a) mutation rate:0, crossover rate: 0.6



(b) mutation rate: 0.03, crossover rate 0.6



(c) mutation rate: 0.07, crossover rate: 0.6

Figure. 9 Simulation results of mutation rate

Figure 9 (a) plots when no mutation occurred. As shown in figure 9 (a), without mutation, the catching prey was performed well. When the mutation rate was given 0.03, the performance was better than others. The mutation much affected on the initial stage. After medium stage, it doesn't seem to affect on the performance.

Next is the evaluation about crossover rate. Figure 10 showed the simulation results of 2-D area selection method. It showed the crossover rate was not proportionally related to the performance. Properly chosen crossover rate may result in good performance. Fundamentally, the manipulation of various variables follows the general features of genetic algorithms. In the pursuit system domain, the variables of genetic algorithm didn't differentiate the general features, since this paper used the mutation and crossover operators.

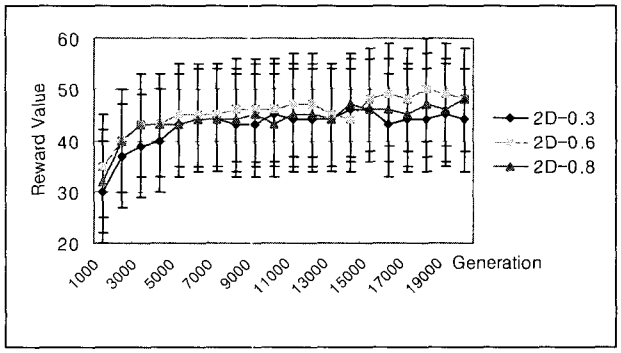


Figure 10. Results of crossover rate (mutation rate: 0.03)

Table 4 describes simulation results of each selection method in tournament competition method with crossover rate of 0.6 and mutation rate of 0.03.

Table 4. Simulation results of tournament competition method

	Generation	Reward value (cp=0.6)	Reward value (mp=0.03)	Reward value (cp=0.6mp=0.03)
2D area selection	20000	37	48	48
Linear area selection	20000	48	53	52
Random selection	20000	52	46	45

We simulated 20,000 times of generation and averaged 50 times of simulations for each selection method. Among the three selection methods, the linear area selection method displayed slightly better results. However, most simulation results didn't show big differences.

Figure 11 represents the evaluation of competition methods using 0.6 of crossover rate and 0.03 of mutation rate.

4.2 Evaluation

Simulation results represented that the tournament competition method showed better evolution and excellent emergent behavior results than the total competition method did. The reason was that in the total competition method, the whole gene pool needed to be considered, which resulted in decreasing processing speed. On the contrary, in the

tournament competition method, small numbers of robots were selected for the next generation. Thus, the tournament competition method could be very effective evolution method in emergent behavior evolution of autonomous robots that were similar to living creatures in nature. To improve the evolution speed at the total competition method, heuristics that can distinguish intelligent behavior patterns ought to be added into adaptation function. However, we could find that there was a limit in the evolution. Such adaptation function could not be regarded as an artificial life approach since it could be pre-defined by system designers.

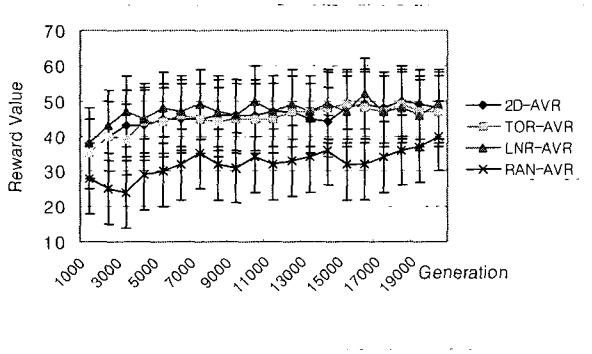


Figure 11. Reward value in each generation

In addition, we could observe that robots' behavior patterns were divided into three stages. In the first stage, robots were the least intelligent and didn't intend to approach or to catch (or encircle) the prey. In the second stage, robots were somewhat intelligent. Robots could catch the prey in front of themselves or move toward to the direction of the prey. However, robots often fell into the infinite-loop so that they could not catch preys continuously (This stage appeared only in global competition method.). In the third stage, robots were most intelligent. Robots never fell into the infinite-loop. Our simulation results have been obtained based on the third stage.

Figure 12 illustrates sequentially the evolution process of robot's behavior with the simulation of 2-D area selection method. The circles denote the robots, and the rectangular denotes the prey.

In figure 12, the term "time 11" denotes the status that both all robots and the prey moved 11 times after environments were initialized. Through Time 10 to 14, the prey is encircled (i.e. caught) by the robots. However, after then, the robots still keeps on evolving during the given life time. At Time 20, the state of 'catch' is temporarily broken, which may occur due to the evolution of robots. At Time 21, the prey is re-encircled by the robots, and no more break-down has occurred. Thus, the Time 22 is called the optimal state at the viewpoint of the robots to obtain the reward values continuously.

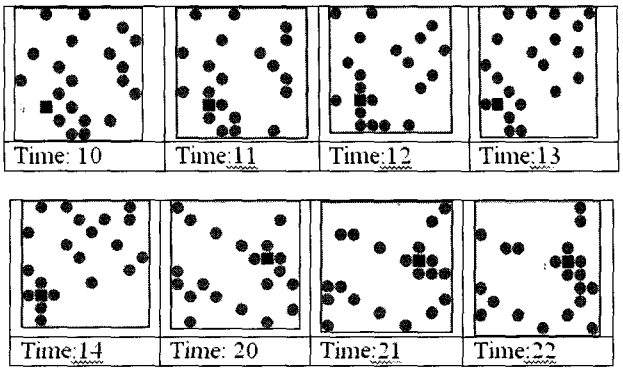


Figure 12. Process of robot's behavior evolution

In the present paper, we could have observed the communication and cooperation between robots. Without the communication and cooperation, robots cannot catch the prey continuously. For example, as shown in Time 20 of Figure 12, the prey can find an escape passage to the bottom, but with the communication and cooperation of another robot at the next time, the prey can be caught continuously. Thus we can define the communication and cooperation as to identify the location among robots and behave proper actions to catch the prey. More simply, we consider the communication as the positioning between robots. We further consider the cooperation as a method to resolve problems in parallel. These characteristics are advantageous to stably accomplish common goal, as applied to replace with another when a specific robot becomes defected.

5. Conclusion

This paper merges the artificial life method in emergent behavior evolution of virtual robot as modeling the pursuit system with the artificial neural network and genetic algorithm. In doing so, in virtual environments, we construct a neural network representing robot and prey and propose a model to evolve its structure. Also, we have compared evolution strategies of several selection methods varying the crossover and mutation rates in simulation and investigated the variation of behavior during the evolution process. As a result, the tournament competition method has showed higher realization of robot's emergent behavior than others. Furthermore, we have observed that even in a virtual environment, intelligent behavior patterns can be emergent for a living creature to exist through evolution. The robot's evolution similar to this kind of creature has a merit in that no human being's help is needed.

Compare to the reinforcement learning method where the reward of robots' behaviors is not instantly computed, this paper has not shown this kind of problem. The genetic algorithm method showed that robots could face some limitation to solve problems, or robots' roles were not dynamic even when each robot was needed to communicate and share its

capability and knowledge with others. That is, the genetic algorithm method is not proper to be applied to the open environments. In this paper, we can observe the communication and cooperation between robots for a common goal. To express the evolution of robots conveniently, we have merged with artificial neural network. Fuzzy function method reveals a time consuming problem since the gene structure is expressed with the fuzzy membership functions. However, this paper doesn't need either fuzzification or defuzzification, which results in reducing time consumption. Therefore, this paper concludes the usefulness of the emergent behavior evolution.

For future works, further research is needed to see if the proposed method in this paper can be applied into more various and complicated application areas in real world.

References

[1] Floreano, D.,Evolutionary Agentics in Artificial Life and Behavior Engineering. In T. Gomi (ed.), Evolutionary Agentics II, Ontario (Canada): AAI Books, 1998.

[2] Gomez, F., & Miikkulainem, R., "Incremental evolution of complex general behavior", Adaptive Behavior, 5, pp. 317-342, 1997.

[3] Floreano, D., Nolfi, S. and Mondada, F. (2001) Co-Evolution and Ontogenetic Change in Competing Agents. In M. Patel, V. Honavar, and K. Balakrishnan (eds.), Advances in the Evolutionary Synthesis of Intelligent Agents, Cambridge (MA): MIT Press.

[4] Brooks and Maes ed., Artificial Life , MIT Press, 1994.

[5] A. Asama et. Al. eds, Distributed Autonomous Agentic Systems I, II, Springer-Verlag, 1994.

[6] D. W. Lee, K. B. Sim, "Behavior Learning and Evolution of Collective Autonomous Mobile Agents using Distributed Genetic Algorithms," Proc. of 2nd AScian Control Conference, Vol.2, pp.575-678, 1997.7.

[7] D.W. Lee, K.B. Sim, "Development of Communication System for Cooperative Behavior in Collective Autonomous Mobile Agents,", *Proc. of 2nd AScian Control Conference*, Vol.2, pp.615-618, 1997.7.

[8] D.W. Lee, H. B. Jun, and K.B. Sim, "Artificial Immune System for Realization of Cooperative Strategies and Group Behavior in Collective Autonomous Mobile Agents,", Proc. of Fourth Int'l Symp. On Artificial Life and Agentics, pp.232-235, 1999.

[9] K.B Sim, "Realixation of Intelligent agent system Based on Artificial Life", *Journal of Korean Electronics*, Vol.24, No.3, pp. 70-82, 1997. 3.

[10] Lisa, M.J. Hogg and Nichola R. Jennings, "Socially Intelligent Reasoning for Autonomous Agents", *IEEE Trans. On Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol.31, No.5, pp. 381-393, September, 2001.

[11] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni, "Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Trans. On Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol.26, No.1, pp. 29-41, February, 1996.

[12] Collins, Robert J. and David R, "An artificial Neural network representation for artificial organisms", Proceedings of the First workshop on Parallel Problem Solving, Number 496, In Lecture Notes in Computer Science, pp. 259-263, Springer-Verlag, 1992.

[13] Victor R. Lesser, "Cooperative Multiagent systems: A personal view of the state of the art", *IEEE Transactions on knowledge data engineering*, Vol.11, No. 1, pp.133-142.1999.

[14] Takayuki Kohri and Matsubayashi , "An Adaptive Architecture for Modular Q-Learning", In Proceedings of the 10th International Confernece on Simulation of Adaptive Behavior, MIT Press, pp.1-6, 2000.

[15] Yasuo Nagayuki, Shin Ishii, "Multi-Agent Reinforcement Learning: An Approach Based on the Other Agent's Internal Model", From Animals to Animates:Proceedings of the 8thConfernece on the Simulation of Adaptive Behavior, MIT Press , pp.478-485, 1999.

[16] H. Asama et al., Distributed Autonomous Agentic Systems, Springer- Verlag, 1994.

[17] H. Asama et al., Distributed Autonomous Agentic Systems 2, Springer-Verlag, 1996.

[18] Kam-Chuen Jim, C.Lee Giles, "Talkin Helps: Evolving Communicating Agents for the Predator-Prey Pursuit Problem", Artificial Life 6, pp. 237-254, 2000.

[19] Il-Kwon Jeong and Ju-Jang Lee, "Evolving Fuzzy Logic Controllers for Multiple Mobile Agents Solving a Continuous Pursuit Problem", *IEEE International Fuzzy Systems Conference Proceedings*, pp.685-689, 1999.

[20] Carlos Andres Pena-Reyes and Moshe Sipper, "Fuzzy CoCo: A Cooperative Co-evolutionary Approach to Fuzzy Modeling", *IEEE Trans. On Fuzzy Systems*, Vol.9, No.5, pp.727-737, October, 2001.

[21] Genetic Programming : On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.

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

[23] Nolfi, S. and Floreano, D. ,"Co-evolving predator and prey agents: Do 'arm races' arise in artificial evolution?", Artificial Life, 4(4), 311-335,1998.

[24] Peter Stone and Manuela Veloso,"Multiagent Systems: A survey from a Machine Learning Perspective", Autonomous Robots, 8, 345-383, 2000.

[25] Haynes, T. and Sen, S., "Learning cases to resolve conflicts and improve group behavior", *International Journal of Human Computer Studies*, 48, pp. 31-49, 1998.

Kyung-Dal Cho received his B.S. degree in Computer Science from Kyonggi University, Kyonggi, Korea, in 1990.
He received the M.S and Ph. D. degree in Computer Science and Engineering from Chung-Ang University, Seoul, Korea, in 1992 and 2004, respectively. His research interests include Fuzzy System, Artificial Life, Machine Learning, Artificial Neural networks.

E-mail : kdcho88@daum.net