Evolutionary Design of a Fuzzy Logic Controller for Multi-Agent Robotic Systems

Il-Kwon Jeong and Ju-Jang Lee

Abstract : It is an interesting area in the field of artificial intelligence to find an analytic model of cooperative structure for multi-agent system accomplishing a given task. Usually it is difficult to design controllers for multi-agent systems without a comprehensive knowledge about the system. One of the way to overcome this limitation is to implement an evolutionary approach to design the controllers. This paper introduces the use of a genetic algorithm to discover a fuzzy logic controller with rules that govern emergent co-operative behavior. A new modified genetic algorithm is applied to automating the discovery of a fuzzy logic controller for multi-agents solving a pursuit problem in a continuous world. Simulation results indicate that, given the complexity of the problem, an evolutionary approach to find the fuzzy logic controller seems to be promising.

Keywords : multi-agent system, fuzzy logic controller, evolutionary algorithm

I. Introduction

Studying computational models of co-operative structures accomplishing a given task is an interesting area in the field of artificial life. However, generally it is difficult to design such models by analysis. As the problem size grows, it becomes more difficult. In the field of self-learning reactive systems it is not even desirable to have a clear idea of a computational model. Autonomous agents being adaptable implies an minimally pre-programmed systems. The general aim is that agents learn to accomplish tasks by interacting with the environment and adapt future behavior on the basis of feedback from present (or past) action[1].

Genetic algorithms are search methods based on natural selection and genetics. GAs are used in various problems including control problems nowadays. It has been empirically proved that GAs are especially suitable for solving combinatorial optimization problems[6].

In this paper, we use a genetic algorithm to evolve a team consisting of mobile robots in order to accomplish a given task while showing emergent cooperative behavior. The GA constructs a fuzzy logic controller for the mobile robots. The task considered in this paper is solving a pursuit game in continuous world. The objective of each robot is to capture a stading/moving prey in cooperation with the other robots to satisfy a pre-specified capture condition while avoiding collisions with them. To the author's knowledge, this is the first attempt to deal with the pursuit problem in continuous domain with mobile robots.

There have been several works related to the work dealt here. They can be categorized into two groups. The first group deals with the methods for evolving a team. Patel and Maniezzo solved a soccer-playing agent problem using neural networks to control agents, and genetic algorithms (GAs) to train the neural networks[1]. Lund and Miglino also used a GA to evolve neural network controllers for real robots[2]. They solved a rather simple obstacle avoidance problem. Recently, it has been shown that GAs are not suitable for training neural networks, and most GA-based learning algorithms for neural networks use hybridization of GAs and another calculus-based algorithm (e.g. GA+Back Propagation) or modified GA specially designed for performance improvement[3].

Haynes and Sen presented several crossover mechanisms in a genetic programming in order to reduce the time needed to evolve a good team solving a pursuit problem[9]. Iba showed the emergence of the cooperative behavior for the multiple agents solving the pursuit problem by means of genetic programming[10]. However, these two works were done in discrete domain. Lohn and Reggia used GAs to discover automata rules that govern self-replicating processes, which was also done in a grid world[7].

The second group of works concerns controlling group behaviors of multi-agents such as gathering, lining, and circulating. Fukuda generated a group behavior for real small mobile robots using reflex movements[12]. Balch presented a behavior-based approach to robot formation-keeping[14]. These works differ from ours in that they used a pre-designed strategy.

Though not dealing with multiple agents, there are some works dealing with robot's behavior evolved by using GA or GP. Lee et. al applied a behavior-based approach to design the controller of a mobile robot solving a box pushing problem by using genetic programming[11]. They also developed a method for co-evolving controllers and structures of a robot to achive a obstacle avoidance task.

The continuous pursuit problem here is much more difficult than the problems solved in the works mentioned above, because it involves a moving obstacle avoidance problem and the movement of a mobile robot is determined by the fuzzy logic controller associated with it while a movement of a robot can be directly assigned by rules in a discrete pursuit problem.

A new modified genetic algorithm (NMGA) is used to discover a fuzzy logic controller that govern emergent co-operative behavior in a continuous world. Though genetic

Manuscript received : Jan. 18, 1999., Accepted : Nov. 9, 1999.
Il-Kwon Jeong : Motion Information Team, VR center, CSTL, ETRI
Ju-Jang Lee : Department of Electrical Engineering, KAIST
※The authors wish to acknowledge the financial support of the Korea Research Foundation made in the program year of 1997.

programming can be successfully applied to evolving a fuzzy logic controller, GA is used here because the problem we are solving can be represented in a form of combinatorial optimization problem and using GP requires some additional steps to construct a fuzzy logic controller[8].

The paper is organized as follows. In section 2, a pursuit model is described. In section 3, discovery of the fuzzy logic controller using NMGA is described. Simulation results are provided in section 4.

II. Pursuit model

A pursuit problem or predator-prey problem is a test bed in distributed artificial intelligence (DAI) research to evaluate techniques for developing cooperation strategies. The objective of an agent (predator) is catching a prey in cooperation with other agents. The problem domain is similar to the effector automata (EA) model[7] except that the domain in this paper is a continuous world. In the EA model, a cellular space is defined where individual processing units (automata or agents), operating in parallel, receive input from their local neighborhood, and produce an output using a pre-defined rule. Each cell is a location in space, and agents (automata) are entities that can occupy cells. The output in a pursuit model is an action command to effect, such as moving or turning speed.

Time is discretized in the pursuit model due to the sampling interval, and space is an two-dimensional square of w (width) $\times l$ (length). An agent is assumed to be a two-wheeled mobile robot with radius 0.5. It can move to anywhere in the space. Each agent is represented by a symbol $T_i, i = 1,2,\Lambda\ ,n$, indicating the ith agent, where n is the number of agents. T_p represents the prey. It is assumed that all agents use identical rules, i.e. the homogeneous strategy.

Fig. 1 shows a pursuit model with two agents and a prey (in the center). It is assumed that each agent can detect other objects (agents and the prey). An agent can detect the positions of other agents. The behavior of each agent is governed by a fuzzy logic controller. It is the rule table that should be designed by using a genetic algorithm in order to complete the fuzzy logic controller.

The actions possible for the pursuit model are changing velocity and angular velocity. In this paper, the velocity of an agent is assumed to be a constant. Therefore the fuzzy logic controller only determines the angular velocity of a robot.

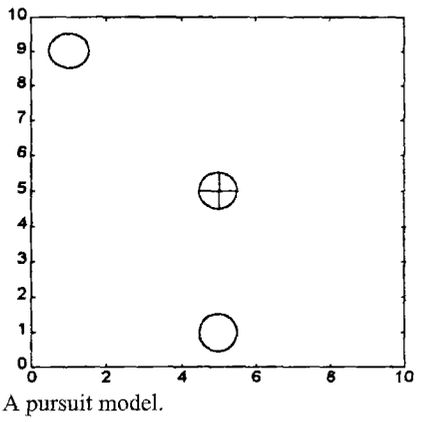


Fig. 1. A pursuit model.

We need a simulator for the pursuit model. The simulator simulates the movements of agents and the prey according to the fuzzy logic controller for a given time steps, and scores the result. The simulation stops when the prey is captured or the given time steps are over. When an action or movement of an agent is outside the world the simulator disables the action. A collision policy should be specified to address the possibility of two or more agents attempting to occupy the same region. Two example policies are mutual annihilation which results in all agents being disabled to move, and the random winner policy which randomly selects one agent to occupy the region in question[7]. We use mutual annihilation policy here.

III. Design of a fuzzy logic controller using a new modified genetic algorithm

1. Problem description

Our objective is to investigate how relatively simple agents can adaptively learn to solve a complex problem. Each agent should learn simple behaviors which are collectively sufficient to solve the problem. Agents have to decompose the problem effectively but this decomposition should be an emergent property of adaptive learning and not preprogrammed. It is an important motivation of this work that a problem should be solved with the minimal possible direction from the programmer or the trainer. We apply a new modified genetic algorithm to find a fuzzy logic controller (FLC) for agents solving a continuous pursuit problem.

The predators (agents controlled by FLC) have to learn to catch the prey. The motion of the prey is determined to run away from the nearest predator in the opposite direction to him with the pre-specified velocity when the predator is in the threat region.

The task of the agents is to capture the prey in a limited period while at the same time to satisfy the constraint in (2) and (3) that each agent should catch the prey in an appropriate direction. Fig. 2 show the regions in which each agent should be placed to satisfy the constraint for the case of 2 and 3 agents. The prey is considered to be captured when the following two conditions are satisfied.

$$distance(T_i, T_p) \le 1.5, \quad \forall i \tag{1}$$

$$\frac{3\pi}{2n} \le (angle(T_p, T_{m_{i+1}}) - angle(T_p, T_{m_i})) \le \frac{5\pi}{2n}, \quad \text{for } i \in [1,n] \tag{2}$$

where $m_i (\in [1,n])$ are indices such that $(angle(T_p, T_{m_i}) \le$

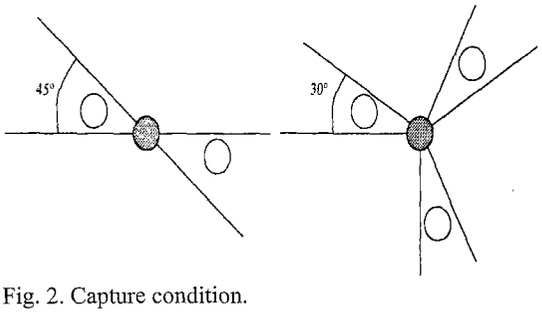


Fig. 2. Capture condition.

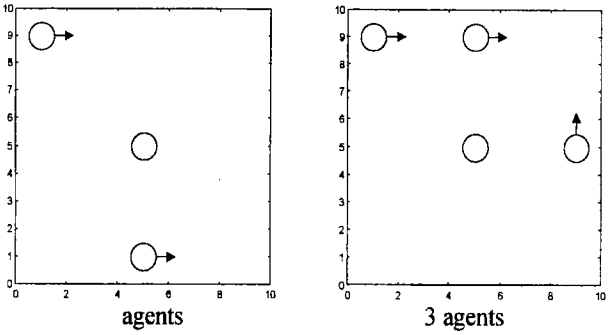


Fig. 3. Initial conditions.

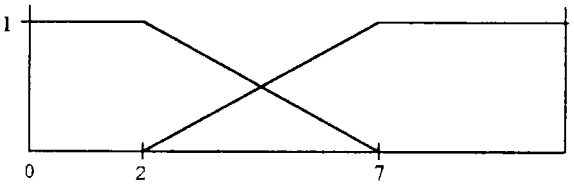


Fig. 4. Membership functions for r.

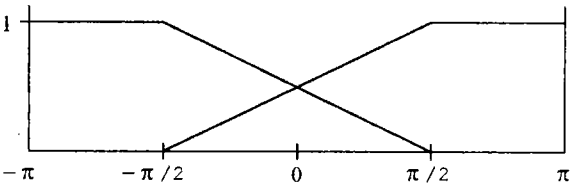


Fig. 5. Membership functions for θ.

$angle(T_p, T_{m_{i+1}}))$, $i = 1,2,\Lambda\ n-1$ and $m_{n+1} = m_1$. *distance*() returns the distance between two agents and $angle(T_p, T_i)$ returns the relative angle of T_i with respect to the heading angle of T_p.

Success of the task depends on agents learning to co-operate in order to catch the prey. Each agent behaves independently of the other, and only knows about the existence and relative position of other agents and the prey as described in the previous section. So there is no direct communication between agents, and their knowledge of the aims of other agents is also indirect. Hence each agent interacts with a highly dynamic environment. Learning (modifying the rule table of FLC) takes place through feedback gained from actions in the environment.

In this paper, the pursuit model has the size of 10×10, that is, w = 10 and l = 10. Initial positions and orientations of the agents and the prey are shown in Fig. 3 for the case of 2 agents and 3 agents. The threat region of the prey is defined as a circle centered at the prey with a radius of 3. The translational velocities for agent and prey are set to 3/sec and 1/sec, respectively.

We use a zero-order Sugeno fuzzy model for FLC to reduce the computation time[15]. Every linguistic variable has two fuzzy sets. Fig. 4 and Fig. 5 show the membership functions of the fuzzy set for r_i and θ_i, respectively. An entry of a rule table is a condition-action rule of the form:

$$\text{If } \theta_1 \text{ is } A \text{ and } r_2 \text{ is } B \text{ and } K \text{ and } r_n \text{ is } D \text{ and } \theta_n \text{ is } E \text{ and } r_p \text{ is } F \text{ and } \theta_n \text{ is } G \rightarrow \text{action} \quad (3)$$

where θ_1 is the heading angle of the robot which is being controlled. r_i and θ_i, $i = 2,3,\Lambda\ ,n$, represent the distance between T_1 and T_i, and relative angle of T_i with respect to the heading angle of T_1, respectively. A, B, and Λ are fuzzy sets corresponding to each linguistic variables.

The position of each agent is updated using the following approximations.

$$\Delta x = (v\cos\theta)\Delta T \quad (4)$$

$$\Delta y = (v\sin\theta)\Delta T \quad (5)$$

$$\Delta\theta = w\Delta T \quad (6)$$

where v and w are the robots' translational and angular velocities, respectively, and ΔT is the sampling period, which is set to 0.05 sec in the simulation. We have simulated the following two situations with 2 and 3 agents.

Case 1: the prey is fixed during the simulation.

Case 2: the prey is moving to run away from the predators.

2. Methodology

We used a new modified genetic algorithm (NMGA) which orginated from MGA[4]. The NMGA is described briefly here. The NMGA consists of the fitness modification and the modified mutation probability. The fitness value for a certain string is determined by the following rule.

$$fitness' = \begin{cases} k \times fitness_{avg}, \text{if } fitness \geq k \times fitness_{avg} \\ fitness, \text{other case} \end{cases} \quad (7)$$

where *fitness* is the original fitness value and *fitness* ' is the modified value. $fitness_{avg}$ is the average of fitness values and k is a constant greater than 1. The modified mutation probability, p_m is given as

$$p_m(i_{gen}+1) = \begin{cases} p_m(i_{gen}), \text{if the fittest is not the same for recent } N_{reset} \text{ generations} \\ p_{m0}, \text{if } p_m(i_{gen}) \times k_1 \leq p_{m\_low} \\ p_m(i_{gen}) \times k_1, \text{other case} \end{cases} \quad (8)$$

where i_{gen} is the generation number. p_{m0}, p_{m_low}, and k_1 is a positive constant less than 1. N_{reset} is an integer constant. NMGA differs from MGA in the mutation probability. NMGA is able to hold the mutation probability when there is a change in the fittest recently, which is considered that the mutation probability at the time is appropriate for the GA under the problem.

Some aspects to be considered to use the modified genetic algorithm are as follows:

- *Chromosome representation and Population size*: a rule table of condition-action rules is indexed implicitly by the pattern in the condition part. 2 bits are used to represent the output (angular velocity) in {-1, 0, 1, 2}. Since there are 5 linguistic variables and two fuzzy sets for each variable in the case of 2 agents, a rule table encoded in a binary string requires 64 bits since (2 × 2 × 2 × 2 × 2) × 2 = 64. In the case of 3 agents, a rule table is encoded in a 256 bit long binary string. A population consists of 50 chromosomes in our simulation.
- *Fitness*: the fitness function is defined as

$$fitness = \frac{1}{\sum_{t=0}^{t_{final}} (F_1(t) + F_2(t))} + F_3 \quad (9)$$

$$F_1(t) = t \times \sum_{i=1}^{n} distance(T_p, T_i) \quad (10)$$

$$F_2(t) = \sum_{\forall i,j \text{ s.t. } distance(T_i,T_j) \le 1.5} \frac{150}{distance(T_i, T_j)} \quad (11)$$

$$F_3 = \begin{cases} 1, \text{ when the prey captured} \\ 0, \text{ otherwise} \end{cases} \quad (12)$$

where t_{final} is the time when a simulation stops. When no goal occurs t_{final} is set to 100, i.e. 5 sec. $F_1(t)$ rewards the agent that moves closer to the prey. $F_2(t)$ prevents the agents from collisions.

- *Reproduction*: first, the fitness values are normalized by dividing them by the average fitness value of the current population. A chromosome is reproduced j times, where j is an integer part of the normalized fitness. The remaining fractions are used to generate additional offsprings using the standard roulette wheel selection method. We used the elitist strategy, that is, the best chromosome is always reproduced without any alterations. Each simulation consists of 200 generations.
- *Crossover*: we used one point crossover. From experimentation, we found that a crossover probability of 0.8 yielded best results.
- *Mutation*: we used the modified mutation probability. The MGA parameters are as following: p_{m0}=0.5, p_{m_low}=0.01, N_{reset} = 5, k = 2.5, and k_1 = 0.9.

IV. Simulation results and discussion

Fig. 6 and Fig. 8 show the typical curves of fitness values for the case 1 and case 2 with 2 agents, respectively. Each graph illustrates the maximum fitness value at each generation. In the case of the present experimental task the increase over generations indicates that the agent is learning more and more appropriate behavior. NMGA successfully found solutions to the both cases. The agents have displayed co-operative behavior to capture the prey while satisfying the constraint.

In the case 1, the NMGA found a solution (FLC capable of making the agents captures the prey) after about 30 generations. In the case 2, the NMGA found a solution after 45 generation due to the moving prey. In both cases, the agents capture the prey after about 4 seconds.

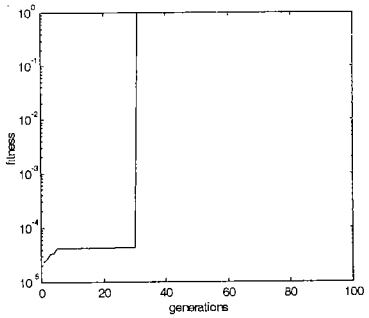


Fig. 6. Best fitness results for the case 1.

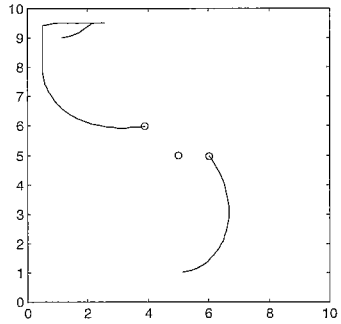


Fig. 7. Agent trajectories for the case 1.

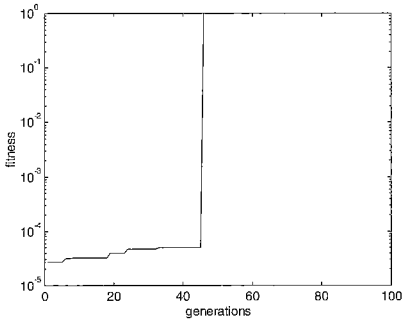


Fig. 8. Best fitness results for the case 2.

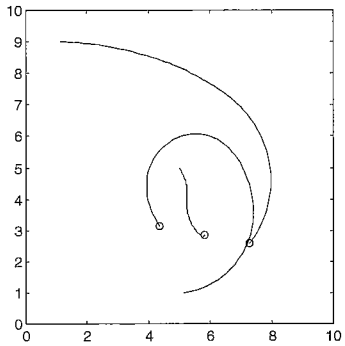


Fig. 9. Agent trajectories for the case 2.

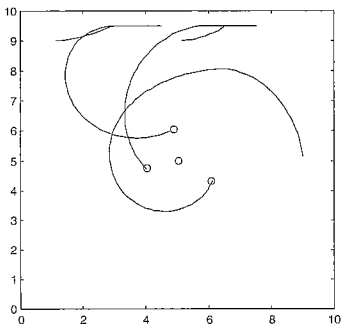


Fig. 10. Agent trajectories for the case 1.

Fig. 7 and Fig. 9 show the typical successful trajectories of the agents for the case 1 and 2 with 2 agents, respectively. 'o' represents the end position of an agent. In the case 1, the upper agent shows a complicated motion due to the wall. The other agent is waiting till other agent approaches in the right

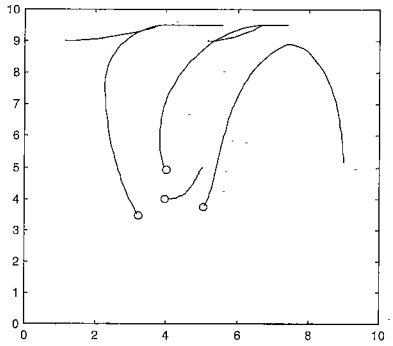


Fig. 11. Agent trajectories for the case 2.

direction. Although the translational velocities are equal the lengths of the trajectories are different because the agent learns to know how to stop using the mutual annihilation policy. In the case 2, the agents successfully captured the moving prey by predicting the motion of the prey.

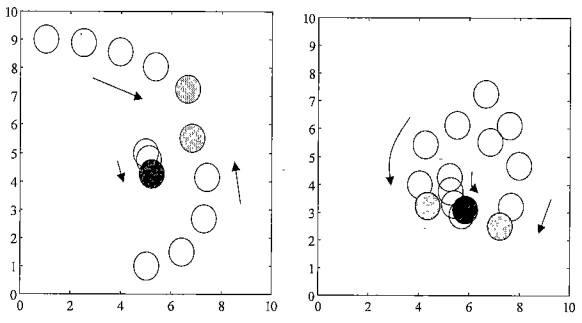
Fig. 10 and Fig. 11 shows the typical successful trajectories of the agents for the case 1 and 2 with 3 agents, respectively. The predators successfully captured the prey after some complicated movements. Due to the increased complexity, NMGA sometimes failed to find a solution. In order to validate the use of NMGA we simulated the same cases so far with a standard GA(SGA) with p_m = 0.05 which was tuned manually. All the cases were simulated 20 times. Table 1 and 2 summarize the results. where SR means the success ratio and

Table 1. Performance comparison between SGA and MGA for 2 agents problem.

2 Agents	case 1, SR	case 1, Avg. Gen. #	case 2, SR	Case 2, Avg. Gen. #
SGA	1.0	21.1	0.95	50.4
NMGA	1.0	29.7	1.0	51.3

Table 2. Performance comparison between SGA and MGA for 3 agents problem.

3 Agents	case 1, SR	Case 1, Avg. Gen. #	case 2, SR	Case 2, Avg. Gen. #
SGA	0.3	104.7	0	N/A
NMGA	1.0	83.5	0.15	168.3



(a) 0–2 sec. (b) 2–4 sec.

Fig. 12. Agent trajectories for the case 2 with ΔT=0.01sec.

Avg. Gen. # means the average number of generations required to find a solution among the successful executions.

SGA is slightly better in the problems with 2 agents. However, in the 3 agents problems NMGA shows much better performance than SGA, which shows the suitability of NMGA for complex problems due to the increased diversity from the mutation process. In the case 2 with 3 agents, SGA could not find a solution.

In order to check if the sampling period (0.05 sec) affects the motion of the agents, we simulated the case 2 with 2 agents and the same fuzzy logic controller as in Fig. 9 and a sampling period of 0.01 second. Fig. 12 show the stroboscopic agent trajectories at every 0.5 second for the sampling period of 0.01 sec. It shows the almost same trajectories as in Fig. 9, which indicates that the obtained fuzzy logic controller is robust to a sampling period variation.

V. Conclusion

We have implemented an evolutionary approach using a genetic algorithm to design a fuzzy logic controllers for multi-agent system solving a pursuit problem in a continuous world. Each agents is modeled as a two-wheeled mobile robot. Solving a continuous pursuit problem using FLC is more difficult than its discrete counterpart, because it includes a moving obstacle avoidance problem and FLC indirectly controls the position of a robot. A new modified genetic algorithm was applied to automating the discovery of rule table of the fuzzy logic controller for multi-agents solving a pursuit problem. Simulation results with 2 and 3 agents showed emergent co-operative behaviors and the validity and generality of the proposed method. Developing a way of effectively decreasing the complexy of the problem as the number of agents increases remains as a further study.

References

[1] M. J. Patel and V. Maniezzo, "NN's and GA's: Evolving cooperative behavior in adaptive learning agents," *IEEE International Conference on Evolutionary Computation*, pp. 290-295, 1994.

[2] H. H. Lund and O. Miglino, "From simulated to real robots," *IEEE International Conference on Evolutionary Computation*, pp. 362-365, 1996.

[3] I. K. Jeong and J. J. Lee, "Adaptive simulated annealing genetic algorithm for control applications,'" *Int. J. Sys. Sci.*, vol. 27, no. 2, pp. 241-253, 1996.

[4] I. K. Jeong and J. J. Lee, "A modified genetic algorithm for neurocontrollers," *IEEE International Conference on Evolutionary Computation*, pp. 306-311, 1995.

[5] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 46-53, 1993.

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

[7] J. D. Lohn and J. A. Reggia, "Automatic Discovery of self-replicating structures in Cellular Automata," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 3, pp. 165-178, Sep., 1997.

[8] I. K. Jeong and J. J. Lee, "Design of a Rule Based Controller using Genetic Programming and Its Application to Fuzzy Logic Controller," *J. of Control, Automation and System Engineering*, vol. 4, no. 5, pp. 624-629, Oct., 1998.

[9] T. Haynes and S. Sen, "Crossover Operators for Evolving A Team," *Proc, of the Annual Con- ference on Genetic Programming*, pp. 162-167, 1997.

[10] H. Iba, "Multiple-Agent Learning for a Robot Navigation Task by Genetic Programming," *Proc. of the Annual Conference on Genetic Programming*, pp. 195-200, 1997.

[11] P. Lee, J. Hallam, and H. H. Lund, "Applying Genetic Programming to Evolve Behavior Primitives and Arbitrators for Mobile Robots," *IEEE International Conference on Evolutionary Computation*, pp. 501-506, 1997.

[12] T. Fukuda, H. Mizoguchi, K. Sekiyama, and F. Arai, "Group Behavior Control for MARS(Micro Autonomous Robotic System)," *IEEE International Conference on Robotics and Automation*, pp. 1550-1555, 1999.

[13] W. P. Lee, J. Hallam, and H. H. Lund, "A Hybrid GP/GA Approach for Co-evolving Controllers and Robot Bodies to Achieve Fitness-Specified Tasks," *IEEE International Conference on Evolutionary Computation*, pp. 384-389, 1996.

[14] T. Balch and R. C. Arkin, "Behavior-Based Formation Control for Multirobot Teams," *IEEE Trans. Robotics and Automation*, vol. 14, no. 6, pp. 926-939, Dec., 1998.

[15] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Upper Saddle River, NJ, Prentice-Hall, 1997.

[16] L. Davis, *Handbook of Genetic Algorithms*, New York, Van Nostrand Reinhold, 1991.

Il-Kwon Jeong

was born in 1970. He received the B. S., and the M. S. degree in electrical engineering from KAIST in 1992 and 1994, respectively. He received the Ph. D. degree in electrical engineering from KAIST in 1999. He is currently a senior researcher at ETRI. His research interests include intelligent system and virtual reality.

Ju-Jang Lee

was born in 1948. He received the B. S., and the M. S. degree in electrical engineering from Seoul National University in 1973 and 1977, respectively. He received the Ph. D. degree in electrical engineering from University of Wisconsin in 1984. From 1978 through 1980 he worked as a project engineer for G.T.E. Automatic Electric Company. He is currently a professor in electrical engineering at KAIST and the head of the robotics research group of the Institute of Control, Automation and Systems Engineers. He is a senior member of IEEE.