Proceedings of the 13th
KACC, October 1998

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

Il-Kwon Jeong and Ju-Jang Lee
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
373-1 Kusong-dong Yusong-gu Taejon 305-701 Korea
Fax: +82-42-869-3410
Email: jik@odyssey.kaist.ac.kr

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 modified genetic algorithm was applied to automating the discovery of a fuzzy logic controller for multi-agents playing a pursuit game. Simulation results indicate that, given the complexity of the problem, an evolutionary approach to find the fuzzy logic controller seems to be promising.

1 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].

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 + BackPropagation)[3][4] or modified GA specially designed for neural network training[5].

Genetic algorithms are search methods based on natural selection and genetics. GAs are used in various problems including control problems. In control area, the GA has been used in identification, adaptation and neural network controller[5][6][7][8]. Calculus-based search methods usually assume a smooth search space, and most of them use the gradient following technique. The GA is different from conventional optimization methods in several ways. The GA is a parallel and global search technique that searches multiple points so it is more likely to obtain the global solution. It makes no assumption about the search space, so it is simple and can be applied to various problems[9]. It has been empirically proved that GAs are especially suitable for solving combinatorial optimization problems.

Haynes and Sen presented several crossover mechanisms in a genetic programming in order to reduce the timed

needed to evolve a good team[11]. Iba showed the emergence of the cooperative behavior for the multiple agents by means of genetic programming[12]. Lohn and Reggia used GAs to discover automata rules that govern self-replicating processes[10]. However, all these works were done in a grid world. Lee et. al. applied a behavior-based approach to design control systems of mobile robots using genetic programming.

In this paper, we use a modified genetic algorithm (MGA)[5] to discover a fuzzy logic controller that govern emergent co-operative behavior in a continuous world. The genetic algorithm is applied to automating the discovery of the fuzzy logic controller for multi-agents playing a pursuit game. The paper is organized as follows. In section 2, a pursuit model is described. In section 3, discovery of the fuzzy logic controller using MGA is described. Simulation results are provided in section 4.

2 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[10] 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. It can move to anywhere in the space. Each agent is represented by a symbol T_i, $i = 1, 2, \cdots, n$, 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 distance and relative orientation 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. An entry of a rule table is a condition-action rule of the form:

$$\text{If } \theta_1 \text{ is } A \text{ and } r_2 \text{ is } B \text{ and } \theta_2 \text{ is } C \text{ and } \ldots \text{ and } r_n \text{ is } D \text{ and } \theta_n \text{ is } E \text{ and } r_p \text{ is } F \text{ and } \theta_p \text{ is } G \rightarrow action \quad (1)$$

where θ_1 is the heading angle of the robot which is being controlled. r_i and θ_i, $i = 2, 3, \cdots, n$, 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. $A, B, \cdots$ are fuzzy sets corresponding to each linguistic variables.

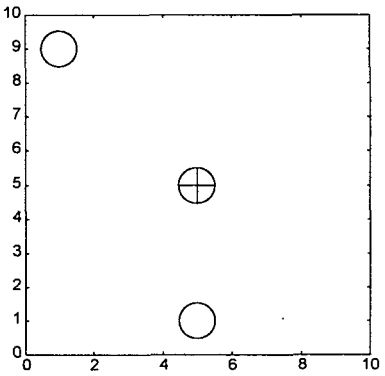

Fig 1. A pursuit model

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.

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[10]. We use mutual annihilation policy here.

3 Design of a Fuzzy Logic Controller Using a Modified Genetic Algorithm

3.1 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 pre-programmed. 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 modified genetic algorithm to find a fuzzy logic controller (FLC) for agents.

The experimental problem is a pursuit game. The predators (agents controlled by FLC) have to learn to catch the prey. Two agents and one prey are used in our simulation. The motion of the prey is determined to run away from the nearest predator with the pre-specified velocity when the predator is in the threat region.

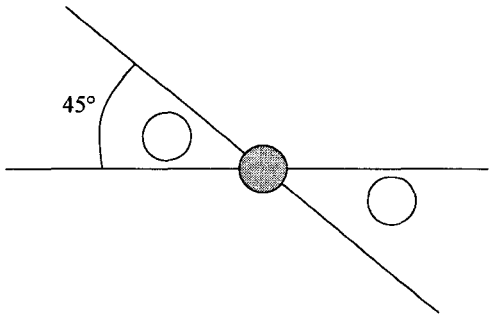

Fig 2. Capture condition

The task of the agents is to capture the prey in a limited period while at the same time to satisfy that constraint that each agent should catch the prey in the opposite direction. Fig. 2 shows two regions in which each agent should be placed to satisfy the constraint. The prey is considered to be captured when the following two conditions are satisfied.

$$distance(T_i, T_p) \leq 1.5, \quad \forall i \tag{2}$$

$$3\pi/4 \leq |angle(T_p, T_1) - angle(T_p, T_2)| \leq 5\pi/4 \tag{3}$$

where $distance()$ 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 cooperate 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.

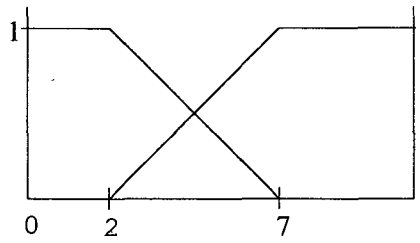

Fig 3. Membership functions for r

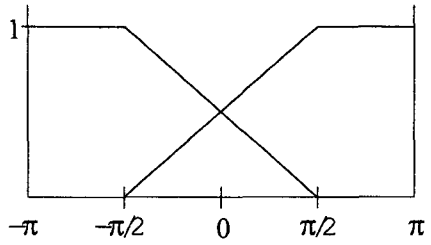

Fig 4. Membership functions for θ

In this paper, the pursuit model has the size of 10×10, that is, $w = 10$ and $l = 10$. Initial positions of the agents and the prey is shown in Fig. 1. 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 3/sec and 1/sec, respectively. We use a zero-order Sugeno fuzzy model for FLC to reduce computation time[15]. Every linguistic variables has two fuzzy sets.

Fig. 3 and Fig. 4 show the membership functions of the fuzzy set for r_i and θ_i, respectively. The position of each agent is updated using the following approximations.

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

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

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

where v and ω are the robots' translational and angular velocity, respectively, and ΔT is the sampling period, which is set to 0.05 sec in the simulation. We have simulated the following two situations.

Case 1: the prey is fixed during the simulation.
Case 2: the prey is moving to run away from the predators.

3.2 Methodology

We used a modified genetic algorithm (MGA)[5]. For a full description and performance of the algorithm readers are referred to the reference. The MGA is described briefly here for convenience. The MGA 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

$$p_m(i_{gen}+1) = \begin{cases} p_{m0}, \text{ if the fittest is the same for} \\ \quad N_{reset} \text{ generations} \\ p_{m\_low}, \text{ if } p_m(i_{gen}) \times k_1 \leq p_{m\_low} \\ p_m(i_{gen}) \times k_1, \text{ other case} \end{cases} \quad (8)$$

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.

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, a rule table encoded in a binary string requires 64 bits since $(2 \times 2 \times 2 \times 2 \times 2) \times 2 = 64$. A population consists of 50 chromosomes in our simulation.

• *Fitness*: the fitness function is defined as

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

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

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

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

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 100 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 $k_1 = 0.9$.

4 Simulation Results and Discussion

Fig. 5 and Fig. 7 show the typical results for the case 1 and case 2, 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. MGA successfully found solutions to the both cases. The agents have displayed co-operative behavior to capture the prey while satisfying the constraint.

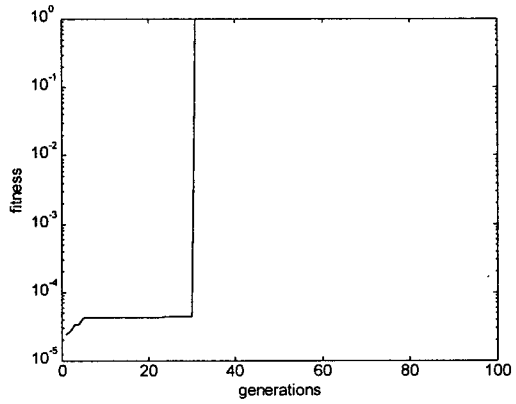

Fig 5. Best fitness results for the case 1

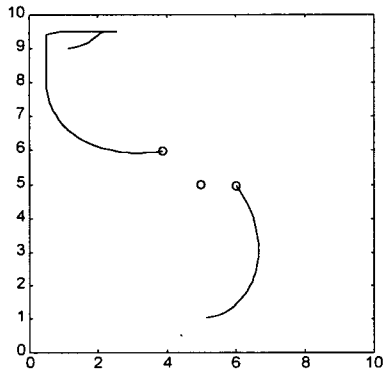

Fig 6. Agent trajectories for the case 1

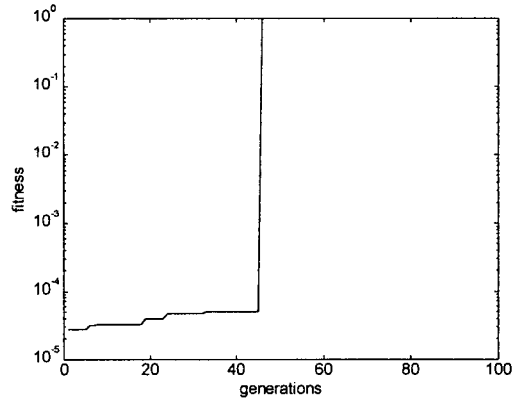

Fig 7. Best fitness results for the case 2

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

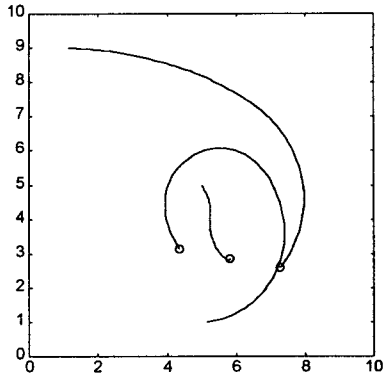

Fig 8. Agent trajectories for the case 2

Fig. 6 and Fig. 8 show the best trajectory of each agent for the case 1 and 2, 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 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.

5 Conclusion

We have implemented an evolutionary approach using a genetic algorithm to design a fuzzy logic controllers for multi-agent system. For that purpose, a pursuit model consisting of two-wheeled mobile robots was used. A modified genetic algorithm was applied to automating the discovery of rule table of the fuzzy logic controller for multi-agents solving a pursuit problem. Though we did not use comprehensive knowledge about the system the genetic algorithm successfully discovered a rule table that govern emergent co-operative behavior. Simulation results indicate that, an evolutionary approach to find the appropriate rules for emergent co-operative behavior seems to be promising.

References

[1] M. J. Patel and V. Maniezzo, "NN's and GA's: Evolving co-operative 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] M. McInerney and A. P. Dhawan, "Use of genetic algorithms with backpropagation in training of feed-forward neural networks," *IEEE International Conference on Neural Networks*, pp. 203-208, 1993.

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

[6] K. Kristinsson and G. A. Dumont, "System identification and control using genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, Vol. 22, No. 5, pp. 1033-1046, 1992.

[7] 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.

[8] Y. Ichikawa and T. Sawa, "Neural network application for direct feedback controllers," *IEEE Trans. Neural Networks*, Vol. 3, No. 2, pp. 224-231, 1992.

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

[10] J. D. Lohn and J. A. Reggia, "Discovery of self-replicating structures using a genetic algorithm," *IEEE International Conference on Evolutionary Computation*, pp. 678-683, 1995.

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

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

[13] W. 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.

[14] H. Heider and T. Drabe, "Fuzzy System Design with a Cascaded Genetic Algorithm," *IEEE International Conference on Evolutionary Computation*, pp. 585-588, 1997.

[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.