Mobile robot learning by evolution of fuzzy controller [1]

Sung-Bae Cho and Seung-Ik Lee
Dept. of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120–749, Korea
Tel.: +82 2 361 2720; Fax: +82 2 365 2579; E-mail: [sbcho, cypher]@csai.yonsei.ac.kr

In this paper we describe an evolutionary fuzzy system to control a mobile robot effectively, and apply it to the simulated mobile robot called Khepera. The system gets input from eight infrared sensors and operates two motors according to fuzzy inference based on the sensory input. In order to robustly determine the shape and number of membership functions in fuzzy rules, genetic algorithm has been utilized. This approach reduces the burden of human operators to decide the structure of fuzzy rules. With the simulation of Khepera robot, we confirm that the evolutionary approach might find out a set of optimal fuzzy rules which make the robot to reach the goal point, as well as to solve autonomously several subproblems such as obstacle avoidance and passing-by narrow corridors.

1. Introduction

Artificial intelligence (AI) has produced several promising techniques to control mobile robot, especially in static environments about which the robot has an explicit representation. It would be rather easy to move the robot to the goal point by planning the optimal path based on the representation. However, almost all the approaches could not obtain a perfect mobile robot of even an insect's intelligence yet due to the lack of adaptability to changing environments.

As a reaction to this problem, a novel approach called behavior-based robotics [2, 3, 4] has recently appeared. Whereas conventional AI is more concerned with a high level definition of the environment and knowledge required by the system, it stresses the importance of continuous interaction between the robot and its own environment for the dynamic development of the control system and the assessment of its performance [4]. It also emphasizes the autonomy of the system which should be completely self-contained and find the most appropriate solutions to satisfy simultaneously several goals.

One of the key points of this approach is not to give the robot information about the movement but to let the robot find the knowledge by itself. With this approach, a number of researchers have successfully employed an evolutionary procedure to develop the control system of simulated robots [1]. The rich variety of structures that have appeared during evolution and the large number of evolved behaviors have empirically demonstrated the power and generality of the evolutionary algorithm. However, this approach suffers from the difficulty of analyzing the control system evolved, which prohibits the designer from making use of some domain knowledge to design the control system by an evolutionary approach.

This paper attempts to develop a fuzzy system for a behavior-based robot, and presents an evolutionary approach to determine the parameters in the fuzzy controller. The proposed system has also been applied to the simulated robot called Khepera. An analysis of the fuzzy rules obtained by evolution presents the emergence of the strategies to optimally guide the robot to the goal point.

2. Khepera: A mobile robot

The Khepera robot employed in our experiment is circular, compact and robust (Fig. 1(a)). This is a miniature robot that has diameter of 55 mm, height of 30 mm, and weight of 70 g. The robot is supported by two wheels and two small Teflon balls placed under its platform. The wheels are controlled by two DC motors with an incremental encoder (12 pulses per mm of robot advancement) and can rotate in both directions. The geometrical shape and the motor

[1]The authors wish to acknowledge the financial support of the Korea Research Foundation made in the program year of 1997.

Journal of Intelligent and Fuzzy Systems 6 (1998) 91–97
ISSN 1064-1246 / $8.00

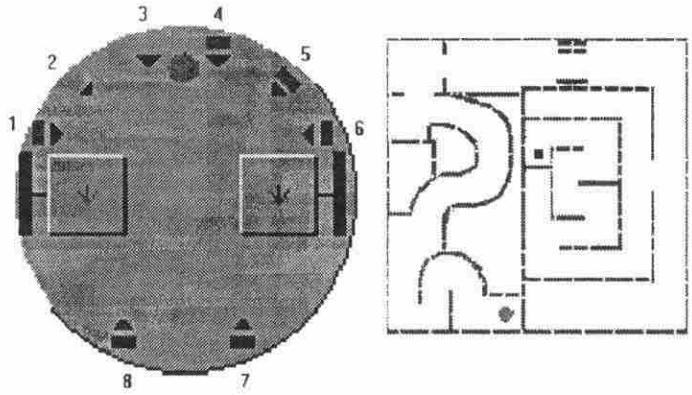

Fig. 1. (a) Khepera robot; (b) A simulated world.

layout of Khepera make the robot to navigate in sophisticated environment even when its control system is immature.

It is provided with eight infrared proximity sensors placed around its body which are based on emission and reception of infrared light. Each receptor can measure both the ambient infrared light and the reflected infrared light emitted by the robot itself. Several new single sensors and complete modules, such as a stereo-vision module and a gripper module, can be easily added, due to the hardware and software modularity of the system.

Dedicated to Khepera, the simulated mobile robot [9] includes eight infrared sensors allowing it to detect by reflection (small rectangles) the proximity of objects in front of it, behind it, and to the right and left sides of it. Each sensor returns a value ranging between 0 and 1023 represented in gradual color levels. 0 means that no object is perceived whereas 1023 means that an object is very close to the sensor (almost touching the sensor). Intermediate values may give an approximate idea of the distance between the sensor and the object. Each motor can take a speed value ranging between −10 and +10. The size of arrows on the motors in Fig. 1(a) indicates the amount of speed.

The simulation in this paper aims at explicitly evolving the ability to navigate from a start point to a goal point in the environment. The robot is located in the environment consisting of a sort of maze of which external size is 1×1 m^2 large (Fig. 1(b)). The robot can sense the walls with the IR proximity sensors. Because the corridors are rather narrow, some sensors are slightly active most of the time.

3. Genetic fuzzy controller

In order to operate the robot introduced at the previous section, we have developed a fuzzy controller of which the internal parameters are adapted with genetic algorithm. In this section we shall describe them in detail.

3.1. Fuzzy controller

A fuzzy system is basically an expert system which uses fuzzy logic for inferring outputs. Especially, rules in any fuzzy control system can be represented by IF (condition) THEN (action), where condition part specifies the condition of parameters encoded by fuzzy membership with eight inputs and action part with two outputs. The generic definition of a rule is as follow:

IF ([X_0 is i_0]) and ([X_1 is i_1]) and ([X_2 is i_2]) and ([X_3 is i_3]) and ([X_4 is i_4]) and ([X_5 is i_5]) and ([X_6 is i_6]) and ([X_7 is i_7])
THEN
Y_0 is o_0 and Y_1 is o_1.

In order to facilitate the controller of Khepera robot,

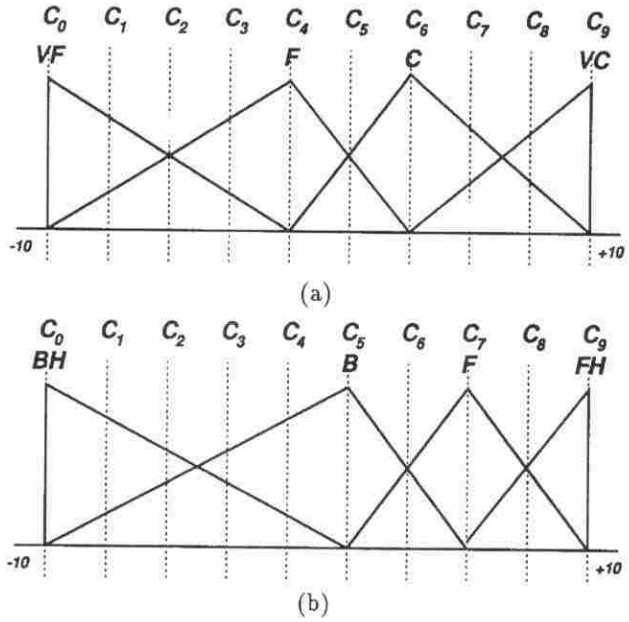

Fig. 2. Membership functions for (a) input; (b) output.

we use the following four fuzzy sets for the input and output parameters:

Input : 8 values from infrared sensors (0 ~ 1023)
Fuzzy set: $I = \{VF, F, C, VC\}$
VF (Very Far)
F (Far)
C (Close)
VC (Very Close)

Output : 2 values from motors (-10 ~ +10)
Fuzzy set: $O = \{BH, B, F, FH\}$
BH (Backward High)
B (Backward)
F (Forward)
FH (Forward High)

Triangular shapes specify the membership function. A parameter value divides the range (0 ~ 1023 for input and −10 ~ +10 for output) by ten equidistance segments.

Figure 2 shows the membership functions used for input and output values, respectively. Correlation minimum method is used for fuzzy inference, and centroid defuzzification method is adopted. Because the standard methods are used to design the basic fuzzy controller, refer to the reference [7] for more details.

3.2. *Genetic adjustment of fuzzy rules*

Genetic algorithm (GA) is considered as an effective method for optimization [6, 10], and several hybrid methods with fuzzy logic have been recently proposed. For instance, Lee and Takagi [8] attempted to optimize fuzzy system using GA and adjust the internal parameters in the GA with the optimized fuzzy system. Fukuda et. al. [5] utilized GA to optimize fuzzy system with RBF membership functions.

Evolution is a remarkable problem-solving machine [6]. First proposed by John Holland in 1975, GAs as one of the computational implementations are an attractive class of computational models that mimic natural evolution to solve problems in a wide variety of domains. A GA emulates biological evolutionary theories to solve optimization problems.

A GA comprises a set of individual elements (the population) and a set of biologically inspired operators

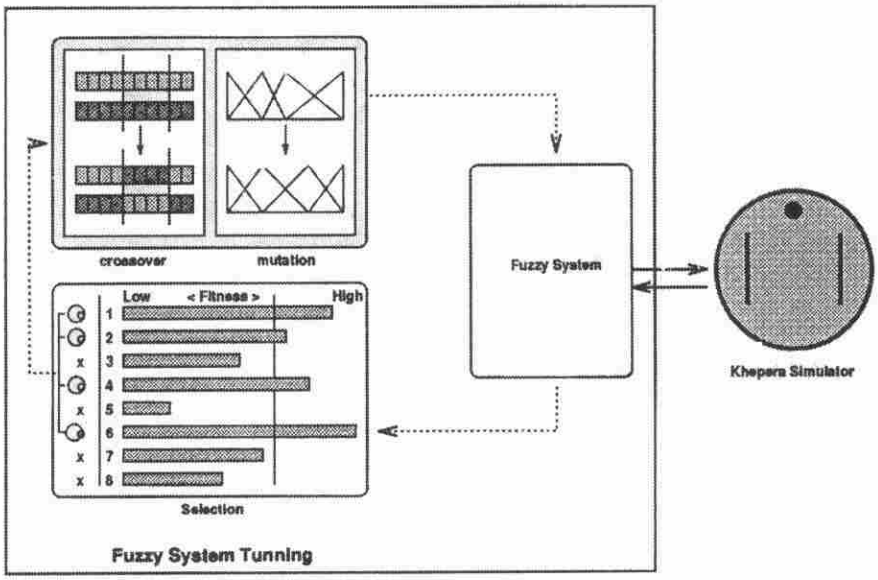

Fig. 3. Schematic diagram of the genetic fuzzy system.

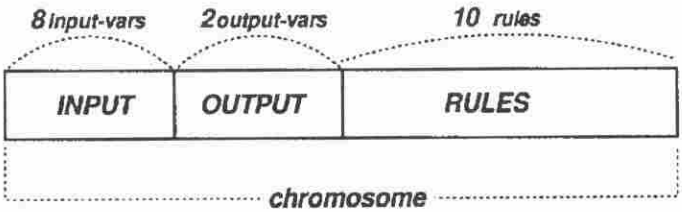

Fig. 4. Gene code for encoding the fuzzy system.

defined over the population. According to evolutionary theories, only the most suited elements in a population are likely to survive and generate offspring, thus transmitting their biological heredity to new generations.

Figure 3 shows the overall diagram of the proposed system. The system parameters in the fuzzy system are represented as a gene, and the performance with the Khepera simulator decides whether it can produce offsprings with the genetic operators. In the figure, four genes of number 1, 2, 4 and 6 are selected as candidates for the next generation, and the crossover is applied to them.

One of the key issues in using the genetic algorithm is to adopt a gene coding scheme appropriate to the problem at hand. In this paper, we should incorporate the input and output membership functions and the rules as a gene code, as shown in Fig. 4 which encodes the eight input parameters, two output parameters and maximum 10 rules. A membership function for each input or output parameter can be specified as two vertices, each of which requires three bits asshown in Fig. 5. This is because the vertices of two boundary triangles are already fixed. Therefore, the total number of bits required to represent the input and output membership functions in a gene code comes to 60 $(6 \times (8+2))$.

The coding scheme for the rules is naturally derived from that for the membership functions. Figure 6 depicts how each rule is encoded. The condition part in a rule can be represented with eight input parameters, and the consequence part with two output parameters. Because each parameter can specify one of the four fuzzy sets, two bits are enough to represent one. In order to indicate whether the parameter is active or not at the corresponding rule, one bit is

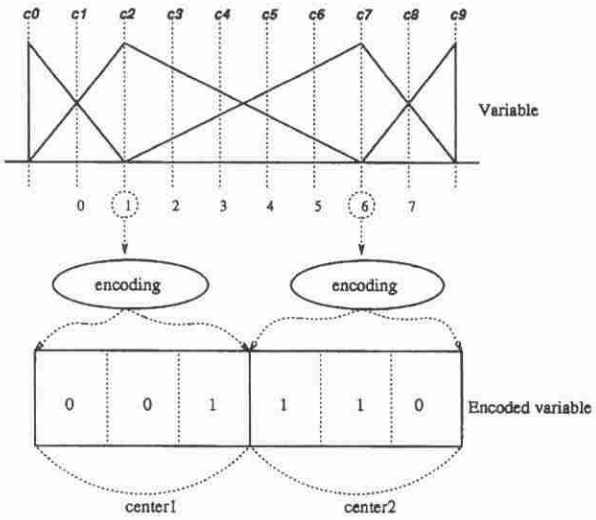

Fig. 5. Encoding method for a membership function.

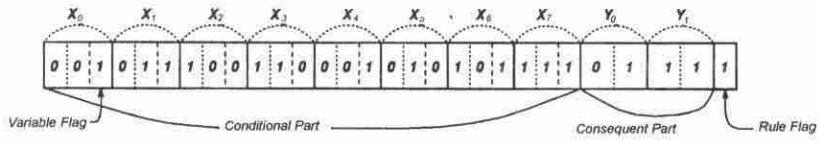

Fig. 6. Encoding method for a rule.

added as variable flag. Similarly, one more additional bit as rule flag is included to indicate whether the corresponding rule is active or not. Therefore, 29 bits get to be required to represent a rule. For an instance, the gene code of Fig. 6 encodes the following rule:

IF $(X_0 = VF)$ and $(X_1 = F)$ and
$(X_4 = VF)$ and $(X_6 = C)$ and $(X_7 = VC)$
THEN $(Y_0 = B)$ and $(Y_1 = FH)$.

The other key issue to make use of the genetic algorithm is to determine the fitness measure as appropriate to the problem at hand. In this paper we make the fitness function decreases as the robot has collided with the walls, and increases as it has moved farther from the start point. In addition, a couple of factors are included to induce the compact fuzzy system by preferring to the smaller number of rules and membership functions. The fitness function is as follows:

$$\begin{aligned}\text{fitness} = {} & \alpha \times \text{number of collisions} \\ & + \beta \times \text{distance moved} \\ & + \gamma \times \text{number of rules} \\ & + \delta \times \text{number of membership functions} \\ & + \varepsilon \times \text{number of check points reached},\end{aligned}$$

where α, γ and δ are negative, while β and ε are positive.

The coefficients might be determined by another optimization technique, but in this paper we just select them by trial-and-error. The fitness would increase as the robot goes farther from the start point while passing by more check points. The fitness would decrease as the robot collides with the walls or the numbers of rules and membership functions get larger. In order to expedite the evolution, we put several check points along with the pathways which will be removed later.

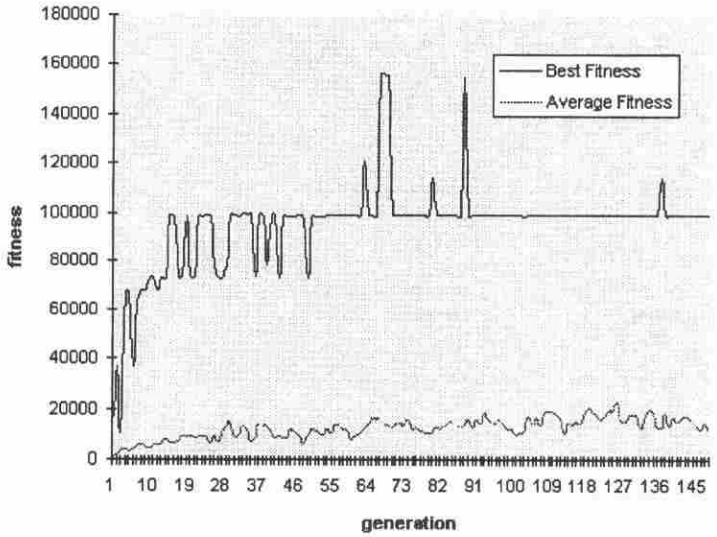

Fig. 7. Fitness change.

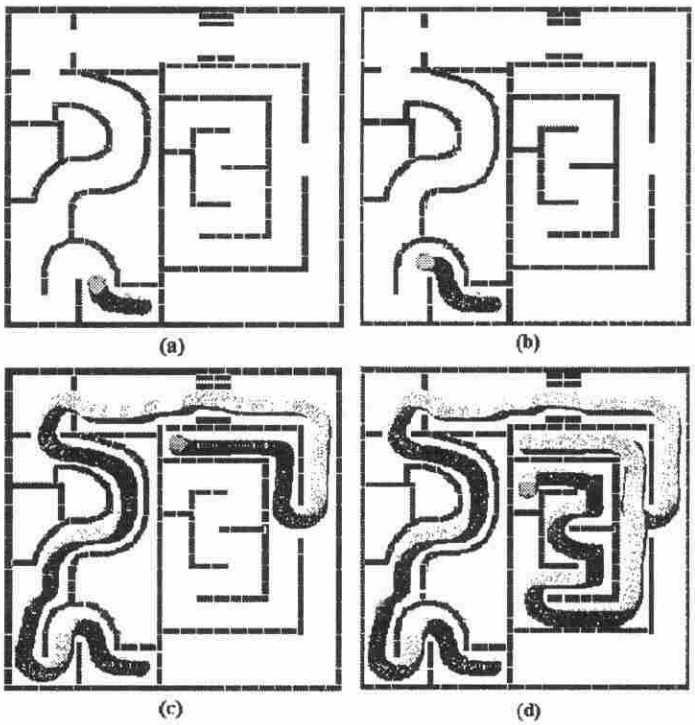

Fig. 8. The trajectories of the robot; (a) right-turn; (b) left-turn; (c) 180-turn; and (d) final status.

4. Simulation results

The Khepera simulator has been written in *C++* [9], and the simulation has been conducted in SUN Sparc 10 workstation. We initialized 200 chromosomes at random, each of which was developed to a fuzzy controller for the robot. Each robot operates within the limit of 5000 unit time, and produces the

performance value according to the fitness function.

Figure 7 shows the best and average fitness changes in the course of simulation. As the figure depicts, the performance increases clearly as the generation goes, and a robot finally arrived at the goal point has been obtained at less than 100 generations. It can be seen that the fitness is radically increased at the beginning stage, but there is nearly no change after 90 generations. Around the 67th generation the best individuals already perform a near optimal behavior. They navigate so smoothly that they do not bump into walls and corners, and maintain a straight trajectory when possible. They finally make a success in complete navigation from the start to the goal point effectively.

Figure 8 shows the trajectories that the robot has made during the simulation. These results are highly reliable and have been replicated in many runs of the experiment. In the beginning of the evolution the individuals evolved a frontal direction of motion, corresponding to the side where more sensors are available. Those individuals that moved in the other direction got stuck in a corner without being able to detect it and soon disappeared from the population. Note that the robot finds out several useful rules, such as right-turn (Fig. 8(a)), left-turn (Fig. 8(b)) and 180-turn (Fig. 8(c)), even though no specific rules for them were given explicitly. The controller for this robot consists of only seven effective rules, which are generated through the evolutionary process, as follows.

(Rule 1) IF $(X_2 = C)$ and $(X_5 = VF)$ and $(X_7 = VC)$
THEN $(Y_0 = BH)$ and $(Y_1 = B)$
(Rule 2) IF $(X_4 = VF)$
THEN $(Y_0 = FH)$ and $(Y_1 = F)$
(Rule 3) IF $(X_1 = VC)$ and $(X_2 = F)$ and
$(X_4 = C)$ and $(X_7 = VC)$
THEN $(Y_0 = BH)$ and $(Y_1 = B)$
(Rule 4) IF $(X_2 = F)$ and $(X_3 = F)$ and $(X_6 = VC)$
THEN $(Y_0 = F)$ and $(Y_1 = FH)$
(Rule 5) IF $(X_4 = FC)$
THEN $(Y_0 = BH)$ and $(Y_1 = F)$
(Rule 6) IF $(X_2 = VF)$ and $(X_4 = F)$ and $(X_6 = VC)$
THEN $(Y_0 = F)$ and $(Y_1 = VH)$
(Rule 7) IF $(X_0 = VF)$ and $(X_4 = F)$ and $(X_5 = C)$
THEN $(Y_0 = BH)$ and $(Y_1 = F)$

Even though we did not give any hints to the system, several effective rules to control the mobile robot appropriately at a number of different cases have emerged through the evolution. This result dictates that the evolutionary approach is quite useful to design a flexible and efficient fuzzy systems to control mobile robot.

5. Concluding remarks

In this paper, we have proposed a fuzzy system to control a mobile robot, and utilized genetic algorithm to optimize the internal parameters in the system. A successful controller generated consists of only seven effective rules, which shows the evolution finds out the optimal set of rules to control the robot. Therefore, we can assert that the evolutionary concept can be an effective vehicle to develop a fuzzy system with many parameters for mobile robot control.

References

[1] R.D. Beer and J.C. Gallagher, Evolving dynamical neural networks for adaptive behavior, *Adapt. Beh.* **1** (1992), 91–122.
[2] R.A. Brooks, A robust layered control system for a mobile robot, *IEEE Trans. Robotics and Automation* **2**(1) (1986), 14–23.
[3] R.A. Brooks, Intelligence without representation, *Artif. Intell.* **47** (1991), 139–159.
[4] M. Dorigo and U. Schnepf, Genetic-based machine learning and behavior based robotics: A new synthesis, *IEEE Trans. Syst. Man. Cybern.* **23** (1993), 141–154.
[5] T. Fukuda, A. Kawamoto and K. Shimojima, Acquisition of swimming motion by RBF fuzzy neuro with unsupervised learning, in: *Proc. 1995 Int. Workshop Biologically Inspired Evolutionary Systems,* 1995, pp. 118–123.
[6] D.E. Goldberg, *Genetic Algorithms in Search Optimization & Machine Learning,* Addison-Wesley, 1989.
[7] B. Kosko, *Neural Networks and Fuzzy Systems,* Prentice-Hall, 1992.
[8] M.A. Lee and H. Takagi, Dynamic control of genetic algorithms using fuzzy logic techniques, in: *Proc. Fifth Int. Conf. Genetic Algorithms,* 1993, pp. 76–83.
[9] O. Michel, *Khepera Simulator Version 1.0 User Manual,* 1995.
[10] M. Srinivas and L.M. Patnaik, Genetic algorithms: A survey, *IEEE Computer* (1994),17–26, June.