Evolving Neural Network Controllers based on Cellular Automata*

Sung-Bae Cho
Department of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea
Tel: +82-2-2123-2720 Fax: +82-2-365-2579
E-mail: sbcho@csai.yonsei.ac.kr

Abstract

There has been extensive work to construct an optimal neural network for controlling a mobile robot by evolutionary approaches such as genetic algorithm, genetic programming, and so on. However, evolutionary approaches have a difficulty to obtain the controller that conducts complex and general behaviors. In order to overcome this shortcoming, we propose a method of combining several evolved modules by a rule-based approach. The multi-modules integration method can make complex and general behaviors by combining several modules evolved or programmed to do simple behavior. Experimental results show the potential of the multi-modules integration method as a sophisticated technique to make the evolved neural network to do complex and general behaviors.

Keywords: Artificial life, neural networks, cellular automata, mobile robot control

1 Introduction

There are some difficulties to construct a sensory-motor controller for an autonomous mobile robot such as coordinating the mechanics and control system parts of the robot, and managing interaction with external environments [1]. To remedy the shortcomings there have been many studies of constructing a mobile robot controller by evolutionary approaches, such as evolving neural network by genetic algorithm, using genetic programming, and combining fuzzy controller with genetic algorithm, because they have the possibility of automatic construction of the controller without explicit design. In particular, evolving neural networks is the most promising way for this problem: Neural networks can easily exploit various forms of learning during life-time and they are resistant to noise that is massively presented in interaction between robot and environments.

There are more than one hundred publications that discuss evolutionary design methods applied to neural networks [2]. One of the important advantages of evolutionary neural networks is their adaptability to a dynamic environment, and this adaptive process is achieved through the evolution of connection weights, architectures and learning rules [2]. Designing the optimal architecture of neural networks can be formulated as searching in the space in which each point represents an architecture. The performance level of all the architectures forms a surface in the space. There are several characteristics in such a surface which make evolutionary algorithms promising candidates with respect to conventional learning models.

In this paper, we evolve the CAM-Brain [3, 4], neural networks based on cellular automata, to control an autonomous mobile robot. CAM-Brain model develops neural networks composed of neurons, axons and dendrites on cellular automata. Cellular automata can represent complicated structures and functions with simple rules as a biological brain composed of billions of simple nerve cells does. This indicates that the cellular automata might be a good platform of complex neural networks developed based on simple rules. Moreover, due to the features of cellular automata it is possible to evolve enormous neural networks very quickly on parallel hardware [3, 4].

*This research was supported by Brain Science and Engineering Research Program sponsored by Korean Ministry of Science and Technology.

However, this traditional evolutionary method cannot be easily used to obtain the controller for complex and general behaviors. We propose a method for a sensory-motor controller to do complex behaviors with neural networks based on cellular automata. This is to combine several modules evolved or programmed to do simple behavior by a rule-based approach. Experimental results of this method show the feasibility of integration of multi-modules evolved neural networks based on CA.

2 Neural Network Evolved on CA

CAM-Brain is the model to create neural networks based on cellular automata, and finally aims at developing an artificial brain. In particular, due to the features of cellular automata the neural networks composed of millions of neurons can be evolved very quickly on parallel hardware. It is already proved that it can be implemented on CAM-Brain Machine (CBM) at MIT and ATR [3]. CBM is programmable logical device implemented with FPGA (Field Programmable Gate Array). It is basically designed to create and evolve CA based neural network modules composed of ten thousands of modules in real time [3].

CAM-Brain model makes neural networks based on cellular automata with its own chromosome that has information about CA-cell structure and state. Therefore, it is possible to evolve and adapt the structure of the neural network to a specific task by genetic algorithm. Figure 1 shows the evolution process of this model. In general, genetic algorithm generates the population of individuals and evolves them with genetic operators such as mutation and crossover. We have used the genetic algorithm to search the optimal neural network. At first, a half of the population that has better fitness value is selected to produce new population. Two individuals in the new population are randomly selected and parts of them are exchanged by one-point crossover. The crossover is occurred at the same position in the chromosomes to maintain the same length in chromosomes. Mutation is operated in the segment of chromosome. The genetic algorithm generates a new population from the fittest individuals for the given problem.

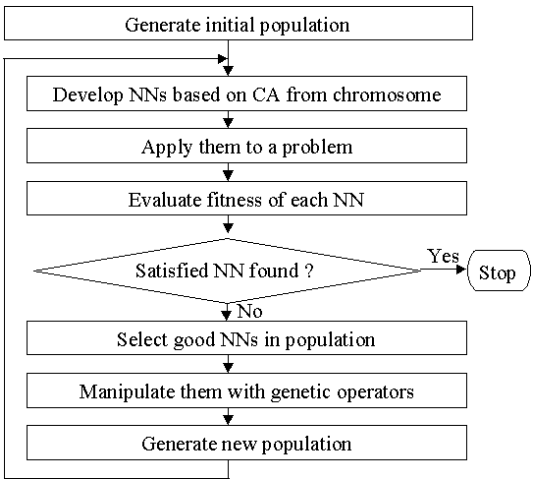

Figure 1: The process of evolving neural networks based on CA

2.1 Chromosome Representation

CAM-Brain's neural network structure composed of blank, neuron, axon and dendrite is grown inside 2-D or 3-D CA-space by states, neighborhoods and rules encoded by chromosome. Roles of each cell are as follows.

- Blank: If cell state is blank, it represents empty space and cannot transmit any signals.
- Neuron: It collects signals from surrounding dendrite cells which are accumulated. If the sum of collected signals is greater than threshold, neuron cells send them to surrounding axon cells.
- Axon: It sends signals received from neurons to the neighborhood cells.
- Dendrite: It collects signals from neighborhood cells and passes them to the connected neuron in the end.

Neighborhood cells of one cell mean surrounding cells (north, south, west and east in 2-D CA space and top and bottom added to them in 3-D CA space). The states of each cell and program (or rules) deciding it with those of neighbors are determined by chromosome (see Figure 2). The information encoded in a chromosome determines a neural network architecture. To represent the whole structure of a neural network, chromosome has the same number of segments with the cells in CA-space and each segment has information of

each cell. A segment can change blank cell to neuron cell (NS bit of Figure 2), and decide the directions of sending received signals to neighborhood cells (N, S, E, W, T and B bits of Figure 2). The signal can be only sent to the direction in which the bit corresponds to 1.

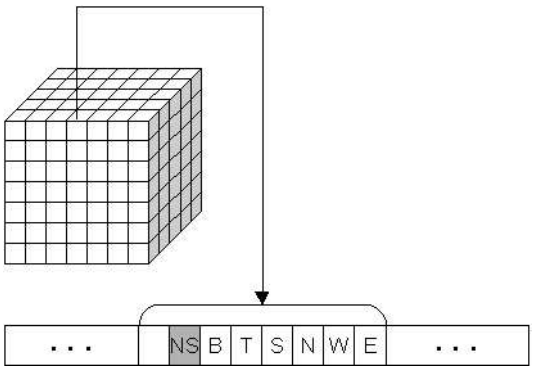

Figure 2: Information encoded in chromosome.

2.2 Growth Phase

The growth phase organizes neural structure and makes the signal trails among neurons. Neurons are seeded in CA-space by chromosome. The neural network structure grows by sending two kinds of growth signals (axon and dendrite) to neighborhood cells. A neuron sends axon growth signal to two opposite directions decided by chromosome and dendrite growth signal to the remaining four directions. The detailed procedure is as follows.

- Step 1: A chromosome is randomly made and the states of all cells are initialized as blank. At this point some of the cells are specified as neuron with some probability.
- Step 2: A neuron cell sends axon and dendrite growth signals to the direction decided by chromosome. Axon growth signal is sent to two directions and dendrite growth signal is sent to the remaining directions.
- Step 3: The blank cell received growth signal changes to axon or dendrite cell according to the type of growth signal. It sends the signals received from other cells to the direction determined by chromosome.
- Step 4: Every blank cell goes through Step 3. Repeating this process, the final neural network is obtained when the state of every cell changes no longer.

2.3 Signaling Phase

Signaling phase transmits the signal from input to output cells continuously. The trails of signaling are performed with evolved structure at the growth phase. Each cell plays a different role according to the cell type. If the cell type is neuron, it gets the signal from connected dendrite cells and gives the signal to neighborhood axon cells when the sum of signals is greater than threshold. If the cell type is dendrite, it collects data from the faced cells and eventually passes them to the neuron body. If the cell type is axon, it distributes data originating from the neuron body.

The position of input and output cells in CA-space is decided in advance. At first, the signal produced by input cells is sent to the faced axon cells which distribute that signal. Then, neighborhood dendrite cells belonged to other neurons collect this signal and send it to the connected neurons, which send it to axon cells. Finally, dendrite cells of output neuron receive this signal and send it to the output neurons. During signaling phase, the fitness evaluation is executed. The detailed procedure of signaling is as follows.

- Step 1: If the state of input cell is neuron, it receives signals from outside and accumulates them.
- Step 2: If the sum of signals from outside is greater than threshold, it sends $+1$ to excitatory axon and -1 to inhibitory axon.
- Step 3: Axon cell received from neuron sends the signal to surrounding cells except the cell that sends the signal. Repeating this process, axon cell distributes the signal to neighborhood cells continuously.
- Step 4: When dendrite cells belonged to another neuron receive the signals, they collect these signals and send them to neuron.
- Step 5: A neuron cell received signals from dendrite cell goes through Step 2 and it sends the signals to surrounding cells. Repeating this process, the signal from input cell is passed to the neuron cells and finally arrives at output neuron.

Fitness value is evaluated by the output value in this process. Depending on the task, several methods can be used such as the number of activated cells, Hamming distance of the target and output vectors, and some function to evaluate the fitness. Figure 3 shows the directions of signals after neuron, axon and dendrite are made. In this figure, neuron sends excitatory signal $(+1)$ to neighborhood cell that has grown into excitatory axon, and inhibitory signal (-1) to the neighborhood cell that has grown into inhibitory axon. Dendrite cell collects signals from neighborhood cells and sends them to neuron, and axon cell distributes the signals originated from neuron to neighborhood cells.

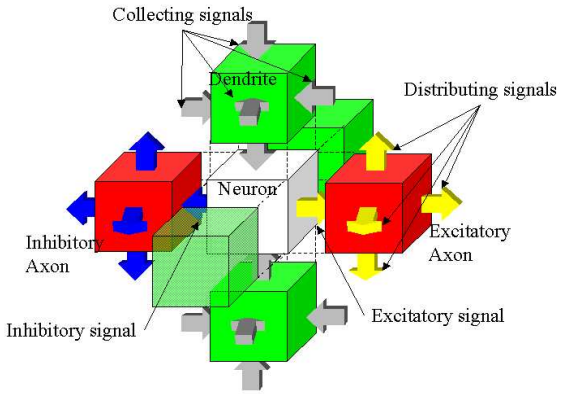

Figure 3: The process of signaling of neural networks.

3 Multi-Modules Integration

The controller composed of only one neural network module has a difficulty to make the robot to perform complex behaviors. Some researchers attempt to overcome this shortcoming by combining several modules evolved or programmed to a simple behavior such as "going straight," "avoiding obstacles," "seeking object," and so on [5, 6]. They expect the controller combined with several modules can do complex behaviors. In this section, basis behaviors and the IF-THEN rules for combining them are presented. Four basis behaviors used in this paper are defined as follows.

- Battery Recharge: If a robot arrives at battery recharge area, battery is recharged. This module enables the robot to operate as long as possible.
- Following Light: The robot goes to stronger light. It must operate this module to go to the "Battery Recharge" area because the light source exists in that area.
- Avoiding Obstacles: If the obstacles exist around the robot, it avoids obstacles without bumping against them. This module enables it to go to "Battery Recharge" area safely by avoiding obstacles.
- Going Ahead: If there is nothing around the robot, it goes ahead. This module makes it to move continuously without stopping.

These basis behaviors are needed to adapt to a given environment shown in Figure 4. In order to be alive for a long time in a given environment the robot must avoid bumping against obstacles and go to the "Battery Recharge" area for recharging battery occasionally.

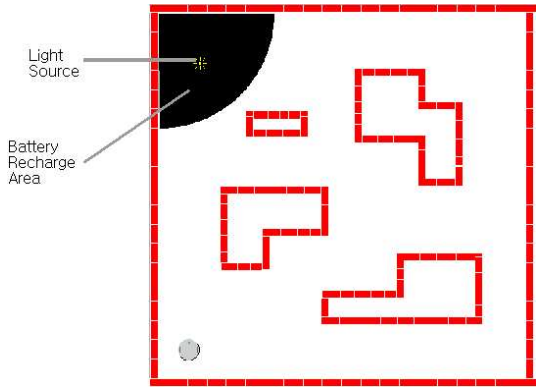

Figure 4: Simulation environment.

We use IF-THEN rules for combining the four basis behaviors properly. Using IF-THEN rules, we can decide an operating module according to the situation which is judged by sensor values of the robot. We expect that this method adapts to a given environment if the rules are defined properly. The rules used in this paper are as follows.

IF ("Battery Recharge" area)
 Execute Battery Recharge module
ELSE
 IF (Battery sensor $< \alpha$) AND
  (Minimum of light sensors $\leq \gamma$)
  IF (Maximum of distance sensors $\leq \beta_1$)
   Execute Following Light module

ELSE
Execute Avoiding Obstacles module
ELSE
IF (Maximum of distance sensors $\leq \beta_2$)
Execute Going Ahead module
ELSE
Execute Avoiding Obstacles module

The robot moves differently according to the constant values (α, β, γ). In these rules, constants are defined as follows.

- α: If battery sensor value is less than α, battery is needed recharging.
- β_1 and β_2: If the maximum value of distance sensors is greater than β_1 and β_2, the robot perceives obstacles.
- γ: If the minimum value of light sensors is less than γ, the robot perceives light.

The robot moves differently according to these constant values. For example, the robot moves around "Battery Recharge" area if α is large, while the robot consumes the battery before reaching the "Battery Recharge" area if it is small. Also, if β is large, the robot bumps against the obstacles frequently and if γ is very small, the robot far from light source cannot perceive "Battery Recharge" area.

4 Experimental Results

This approach is simulated in a modified Khepera simulator. There are two kinds of modules. One is programmed modules such as "Battery Recharge" and "Going Ahead," and the other is evolved CAM-Brain modules such as "Following Light" and "Avoiding Obstacles".

"Battery Recharge" module is programmed because this behavior is very simple. Algorithm of this module is that the robot recharges battery if it is in the "Battery Recharge" area shown in Figure 4. We can decide whether it is in "Battery Recharge" area or not by measuring the distance from the light source to the location of robot. This method has a problem in real world because x-y coordinate values are not available. In this case, we can use another sensor which can recognize floor painted black [7].

"Going Ahead" module is also programmed. If there is nothing around the robot, it goes ahead. This module is made by controlling the velocity of the left and right wheels. The velocity of left and right wheels is fixed as 5.

"Following Light" module is evolved to go to the light source. The light source tells the robot about "Battery Recharge" area. Input values of CAM-Brain module are light sensor values of the robot. The fitness value is defined as follows.

$$Fitness = S * (1 - \sqrt{V}) * (c * D) \qquad (1)$$

S : Average speed of the two wheels
V : Difference between the velocity of two wheels
c : 1 if the robot does not bump against obstacles and
1/2 otherwise
D : Distance from the robot to the goal position

The fitness evaluation leads the robot to go straight quickly and reach nearby the light source. Population size is 50 and the fitness of individual is computed by the average of four runs. Of course, the direction of the robot is different in each run. Figure 5 (a) and (b) show the change of the best and average fitnesses respectively.

"Avoiding Obstacles" module is incrementally evolved, because the behavior, avoiding bumping against obstacles, is complex and difficult to evolve. Incremental evolution approach is expected to evolve the neural network module to do complex behavior efficiently.

The operating module is selected by the rules explained at the previous section and sensor values of the robot. Figure 6 shows the trajectory of the robot by combined modules in a given environment. According to starting position of the robot, the trajectory and moving behavior are different. Battery decreases as the robot moves and battery becomes 2500 when it is recharged: The robot can move without recharging for 2500 times. α is 2/3 of the maximum battery, β_1 is 200, and β_2 is 250.

5 Concluding Remarks

This paper has attempted to evolve neural networks based on cellular automata to control an

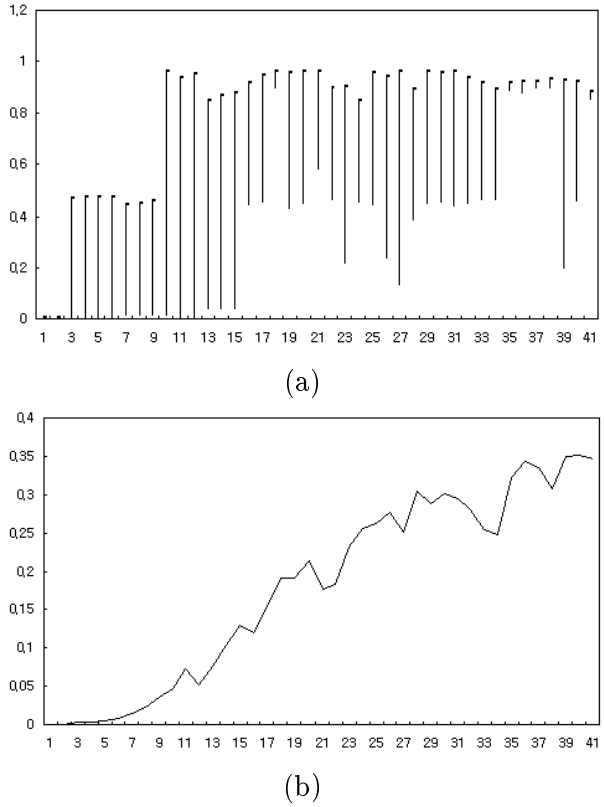

Figure 5: The change of the fitness. (a) Best fitness. (b) Average Fitness.

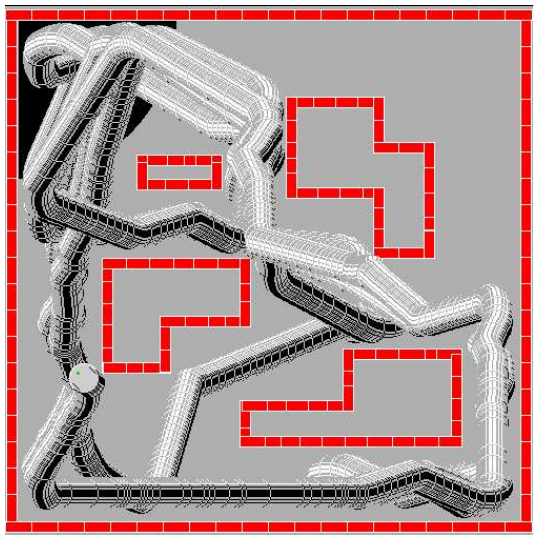

Figure 6: The trajectory of the robot with multi-modules.

autonomous mobile robot. In particular, in order to remedy the shortcoming of conventional evolutionary neural networks and obtain the sensory-motor controller doing complex and general behaviors, we have proposed a multi-module integration method. We have combined multi-modules evolved or programmed to do simple behaviors using IF-THEN rules to generate complex behaviors, so that we show the potential of combining multiple neural network modules. We hope that the sophisticated technique proposed in this paper makes the evolutionary algorithms to be used in larger variety of applications in robotics.

References

[1] I. Harvey, P. Husbands and D. Cliff, "Issues in evolutionary robotics," *Proc. of the Second Int. Conf. on Simulation of Adaptive Behavior (SAB92)*, pp. 364-373, Cambridge MA, 1993.

[2] X. Yao, "Evolutionary artificial neural networks," *Int. Journal of Neural Systems*, vol. 4, no. 3, pp. 203–222, 1993.

[3] H. de Garis, "CAM-Brain: ATR's billion neuron artificial brain project," *Proc. of Int. Conf. on Evolutionary Computation*, pp. 886-891, Nagoya, Japan, 1996.

[4] N.E. Nawa, H. de Garis, F. Gers and M. Korkin, "ATR's CAM-Brain Machine (CBM) simulation results and representation issues," *Proc. of Genetic Programming Conf.*, USA, 1998.

[5] W. Banzhaf, P. Nordin and M. Olmer, "Generating adaptive behavior using function regression within genetic programming and a real robot," *Int. Conf. on Genetic Programming*, pp. 35-43, 1997.

[6] M.J. Mataric, "Designing and understanding adaptive group behavior," *Adaptive Behavior*, Vol. 4, No.1, pp. 51-80, 1995.

[7] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Trans. Systems, Man, and Cybernetics*, Vol.26, No.3, pp. 396-407, 1996.