# Behavior Analysis of Evolved Neural Network based on Cellular Automata

Geum-Beom SONG and Sung-Bae CHO

Computer Science Department, Yonsei University
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea
Tel: +82-2-361-2720 Fax: +82-2-365-2579 E-mail: goldtiger and sbcho@candy.yonsei.ac.kr

**Abstract**

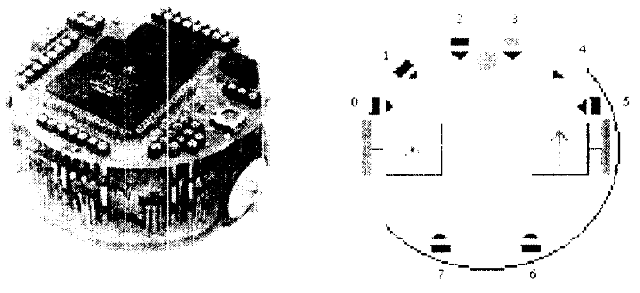
CAM-Brain is a model to develop neural networks based on cellular automata by evolution, and finally aims at a model as an artificial brain. In order to show the feasibility of evolutionary engineering to develop an artificial brain we have attempted to evolve a module of CAM-Brain for the problem to control a mobile robot. In this paper, we present some recent results obtained by analyzing the behaviors of the evolved neural module. Several experiments reveal a couple of problems that should be solved when CAM-Brain evolves to control a mobile robot, so that some modification of the original model is proposed to solve them. The modified CAM-Brain has evolved to behave well in a simulated environment, and a thorough analysis proves the power of evolution.

**Keywords**: CAM-Brain, evolved neural network, cellular automata, mobile robot control, a-life

## 1. Introduction

Recently, there have been vigorous attempts to understand and reconstruct the functions of brain. One of these attempts is to investigate engineering-based brain. A brain builder group at ATR Human Information Processing Research Laboratories in Japan has attempted to develop artificial brain called CAM-Brain [1]. This system is based on Cellular Automata (CA) and controls the growth and signaling of neuron. In particular, due to the features of CA it is possible to evolve very quickly on parallel hardware such as CAM-8 at MIT and CBM at ATR [2].

Evolutionary Engineering (EE) is an approach to combine neural network modules that has evolved with particular functions to develop artificial brain. It has been extensively exploited to apply each neural network module to a specific problem. In the previous work, that a module of CAM-Brain could be evolved to control a mobile robot to navigate well in a simulated environment with sophisticated behaviors such as avoiding to bump against well [3]. This paper illustrates the power of the model by analyzing the robot behaviors.



(a) Khepera robot (b) Simulator
Fig. 1 Khpera

## 2. Khepera: Mobile Robot

Fig. 1(a) shows real Khepera robot, which contains 8 infrared sensors to detect by reflection the proximity of objects in front of it, behind it, and to the right and left sides of it, and to measure the level of ambient light all around the robot. Also, the robot has two motors to control left and right wheels. Khepera simulator (see Fig. 1(b)) also features the ability to drive a real Khepera robot, so that we can very easily transfer our simulation results to the real robot.

Each sensor of Khepera simulator returns a value ranging between 0 and 1023. 0 means that no object is perceived, while 1023 means that an object is very close to the sensor. Intermediate values may give an approximate idea of the distance between the sensor and the object. These sensors also return a value ranging between 50 and 500. When a sensor is very close to light source, it returns 50. This paper uses only distance value returned by only 4 sensor (two of front, left and right). Each motor can take a speed value one of 5, 0, 5 [4].

## 3. Neural Networks based on CA

One neural network module of CAM-Brain is defined on CA-space that is 2-D or 3-D, by two different processes. One of that is growth phase and the other is signaling phase. Growth phase makes the neural network structure by letting every cell in CA-space be one of four types (Neuron, Axon, Dendrite and Blank). Signaling phase transmits the signal through neural networks that are made in growth phase.

## 3.1 Growth phase

The growth phase organizes neural structure and makes the signal trails among neurons. Neurons are seeded in CA-space by chromosome (if chromosome is not exist then it is decide randomly). Neural network structure grows by sending growth signals to neighborhoods cells (N, S, W, E, Top and Bottom in 3-D CA-space). Neuron sends axon growth signal to opposite two directions by chromosome. Type of neighborhood cells that are received axon growth signal becomes axon. They propagate axon growth signal to neighborhood cell.

Also, neuron sends dendrite growth signal to the rest four directions. Then type of neighborhood cells that are received dendrite growth signal becomes dendrite. They propagate dendrite growth signal to neighborhood cells. If type of cell is decided, it is not change any more. So, axon and dendrite grow from neuron cells. Neural network is constructed in this process. This neural network structure is encoded to chromosome, and it is evolved by genetic algorithm.

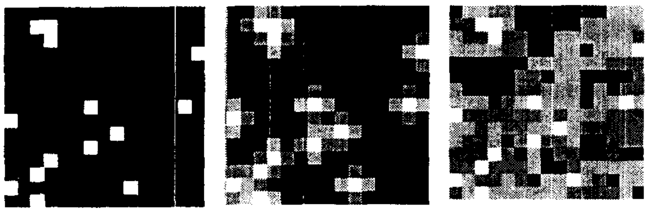

(a) Initial state (b) Intermediate state (c) Final state
Fig. 2 Growth phase

Fig. 2 shows that process of growth phase in 2-D CA-space. Initially, all cells are set to blank type, and some cells are decided as neuron-seed cells by chromosome information. Neuron cells are made only at the initial state as shown in Fig. 2(a). Neuron cells send two kinds of growth signals to their neighbors, either "grow a dendrite", or "grow an axon". The blank neighbors, that receive a neural growth signal, turn into either an axon cell or a dendrite cell. Fig. 2(b) shows the growing axon and dendrite by growth signals. Fig. 2(c) shows the completion of evolving one neural network module. Each axon and dendrite cell belongs to exactly one neuron cell.

## 3.2. Signaling phase

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

Position of input and output cells in CA-space is decided in advance. First, if input cells give the signal (only neuron cell gives and takes input and output signal), it sends the signal to faced axon cells, which distributes that signal. Then, neighborhood dendrite cells that are belonged to another neurons collect this signal and send it to neurons. So, neurons that received the signal from dendrite cells send to axon cells. So, signal transmits more and more. At last, dendrite cells of output neuron receive this signal, and send it output neurons. Output value can be obtained from output neurons.

During signaling phase, fitness evaluation is executed. According to given task, various methods can be used, such as the number of activated cells, hamming distance of target and output vectors, or function to evaluate fitness. This fitness is used for evolving chromosome [5].

## 3.3 Applying CAM-Brain to robot control

In order to apply CAM-Brain to robot control, following process is needed. Neural network structure is made in growth phase. In signaling phase, sensor values from Khepera simulator are used as inputs of the input cells of CAM-Brain. CAM-Brain transmits signal from input cells to output cells. When output values of CAM-Brain are inputted to Khepera simulator, the robot moves. When the robot bumps against the obstacle or reaches the goal, it's fitness is computed. Chromosomes are recreated according to it.

## 3.4. Problems

There are a couple of problems to apply the model to controlling robot. One is that CAM-Brain does not perfectly utilize activation values of robot sensors. After a sensor value (scaled between 1 to 32) enters CAM-Brain, the corresponding input neurons send 1 and -1 to axons if input value is greater than threshold. This can be thought of activation value of a robot sensor as scaled between -1 and 1. It cannot reflect various situations because input neurons only decide whether it is greater than threshold without regard to various size of input value. The other problem is that delay time is needed until CAM-Brain makes output value. After sensor values are inputted to input cells some steps are needed until this value arrives at output cells. It hinders the robot from reacting promptly.

## 3.5 Solution

Dividing input range has solved the first problem. The number of reacting input cells varies according to the size of input value. When input value is greater than 21, the region of input cells gets to be the largest. On the other hand, when input value is less than 11, the region of input cells is the smallest. The other problem is solved by executing dummy signaling phase for some duration until signals started from input cells arrive at output cells. This enables the timely reaction of robot according to situation.

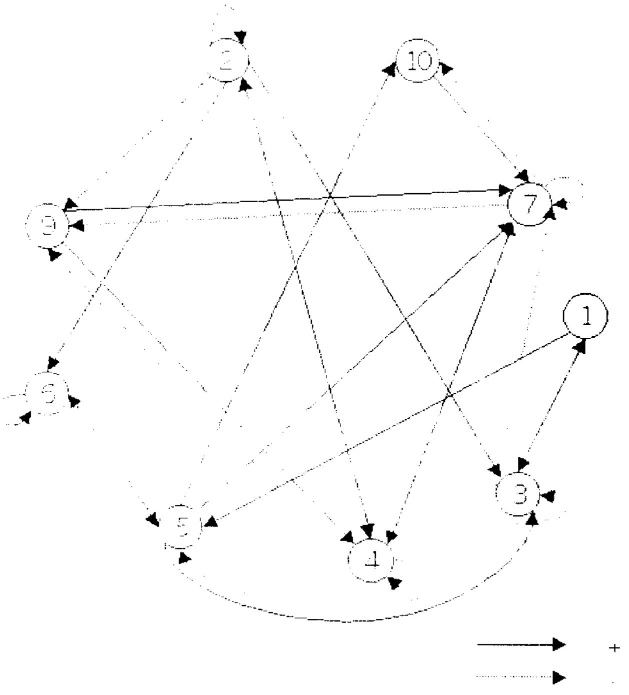


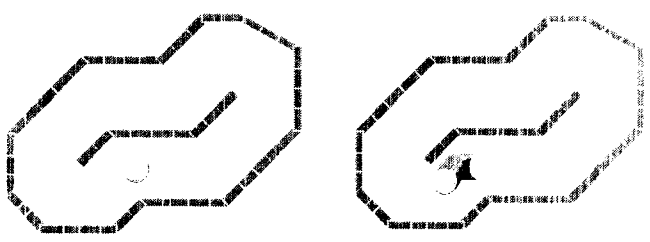
Fig. 3 Schematic diagram of the evolved neural network

## 4. Simulation

Khepera robot simulator has been programmed with C++ [4], and the experiment has been performed on Sun UltraSparc. The population size is 100, and maximum step of sensor sampling step is 30000 to find the fittest one.

### 4.1 Environment

We use 5X5X5 CA-space that solves the problem. Only four sensors are used in this simulation and each input cell is in the center region of four faces of hexahedron CA-space. Output cells are in the top and bottom faces of hexahedrons. Output cell of top and bottom faces produces the speed of left and right motors, respectively. One robot completes the growth phase with a chromosome and then starts receiving inputs. Only neuron cells can take inputs. A signaling phase is performed for some steps per one sensor sampling time unit.



(a) Initial step (b) 250th step
Fig. 4 Avoid bumping

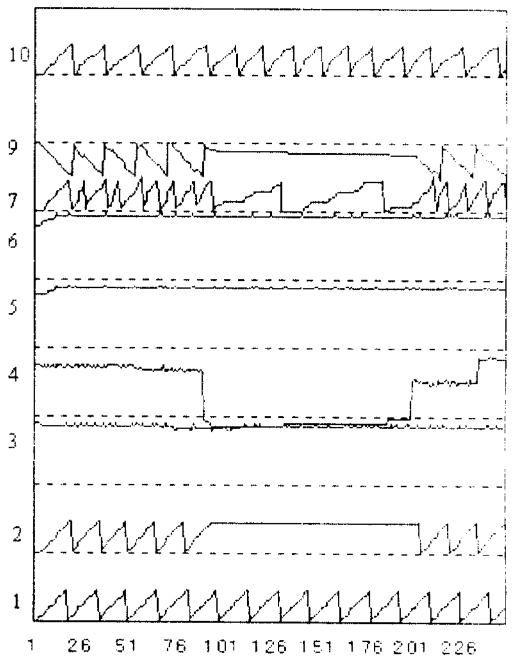


Fig.5 Activation values of neurons when avoiding bumping step

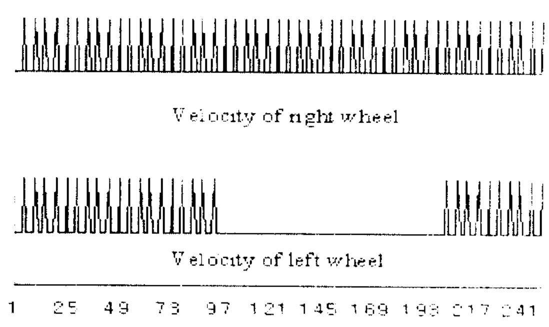


Fig 6 Velocity of robot wheels when avoid bumping step

A simple method is used for fitness evaluation as follows. Let the center of simulation environment be $O$, robot's starting point be $S$, and robot's present location be $N$, then the fitness is given like this.

$$fitness = \angle SON / 2\pi$$

This fitness leads robot's clockwise rotation. If the robot goes round the simulation space completely, the fitness becomes 1.0 and the robot stops moving. If robot crashes to the wall then it stops movement and evaluates the fitness on that position. Even if we do not give any knowledge about the movements, such as "turn right", "turn left" and "avoid bumping", the evolution guides CAM-Brain naturally to solve the problems.

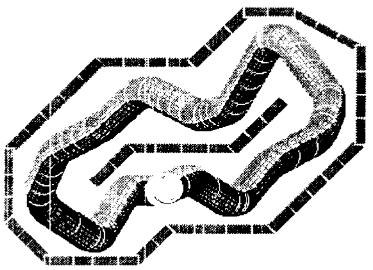


Fig. 7 Trajectory of the robot

4.2 Analysis of results

Fig. 3 shows the architecture of the neural network evolved. Dotted line represents inhibitory connection, and fully line represents exhibitory connection. This has been obtained by tracing activation values of each neuron. The number of neurons is 12, but neurons 8, 11 and 12 are dummy neuron because these neurons are not in the path of input to output neurons. Neuron 2 and 10 are output neurons to produce the velocity of left and right wheels, and neurons 3, 4, 5 and 6 are input neurons. Neuron 3 is for front sensor of the robot, neurons 5 and 6 are for left sensor of the robot, and neuron 5 is for right sensor of the robot.

Fig. 4 shows the intermediate state of the robot when it avoids bumping against wall. Fig. 4 shows the activation values of all neurons and Fig. 5 shows the velocity of left and right wheels. The velocity of left and right wheels gets 5 if the value of output neuron is greater than 0 and it gets -5 if it is less than 0.

Because the activation value of neuron 3 increases, the velocity of right wheel is 5, while the activation value of neuron 2 does not change to 0 from the 100th to the 200th steps so that the velocity of left wheel is 0. This time, neuron 2 is directly influenced by neuron 4 (see Fig. 3). The activation value of neuron 4 is very low because the activation value of neuron 4 is for right sensor value and right sensor is very close to the wall. Originally, sensor value of the robot simulator is high when the robot is close to the obstacle. However, we scaled sensor values inversely so that the activation value of neuron 4 is low though right sensor is very close to the obstacle. Eventually, the robot avoids bumping by turning left when it becomes very close to the wall.

The trajectory of the robot that reached the target spot is shown in Fig. 7. The robot avoids bumping by turning left and right when it is close to the wall. Trajectory of the robot is not good than that of shown [3]. However this clearly shows movements of the robot when the robot close to the wall, because active values of neurons is shown well due to less number of neuron in each step.

## 5. Concluding Remarks

This paper simply presented CAM-Brain which is the evolved neural network based on CA, and analyzed some recent result which is obtained by applies a module of CAM-Brain to control a mobile. Also, we showed that a module of CAM-brain is evolved to behave well in a given environment. And architecture of evolved neural network that has been made by CAM-Brain was shown with schematic diagram. We could observe operating processes of neural network in order to avoid bumping against the obstacle.

CAM-Brain model could organize the neural structure to solve the problem, so that we expect that solving more complex problem is possible. In order to achieve this goal, we must investigate possible mechanisms for learning and evolving the CAM-Brain model. Furthermore, we should devise a method to integrate the many neural network modules evolved.

## Reference

1. H. de Garis, "CAM-Brain : ATR's billion neuron artificial brain project: A three year progress report," *Proc. Int. Conf. on Evolutionary Computation*, pp 886-891, Nagoya, Japan, May 1996.
2. F. Gers and H. de Garis, "Porting a cellular automata based artificial brain to MIT's cellular automata machine 'CAM-8'," *Proc. SEAL '96*, pp. 321-330, Taejeon, Korea, November, 1996.
3. S. B. Cho, G. B. Song, J. H. Lee and S. I. Lee, "Evolving CAM-Brain to control a mobile robot," *Proc. AROB '98*, pp. 271-274, Beppu, Japan January, 1998.
4. O. Michel, *Khepera Simulator Version 1.0 User Manual*, 1995.
5. F. Gers, H. de Garis and M. Korkin, "CoDi-1Bit: A cellular automata based neural net model simple enough to be implemented in evolvable hardware," *Proc. AROB III' 98*, pp. 263-266, Beppu, Japan Feburary, 1998.