Coordination of Multiple Behavior Modules Evolved on CAM-Brain

Kyung-Joong Kim
Department of Computer Science
Yonsei University
Seoul 120-749, Korea
Email : uribyul@candy.yonsei.ac.kr

Sung-Bae Cho
Department of Computer Science
Yonsei University
Seoul 120-749, Korea
Email : sbcho@csai.yonsei.ac.kr

Abstract- In behavior-based robotics the control of a robot is shared between a set of purposive perception-action units, called behaviors. A major issue in the design of behavior-based control systems is the formulation of effective mechanisms for coordination of the behaviors' activities into strategies for rational and coherent behavior. Early, there has been extensive work to construct an optimal controller for a mobile robot by evolutionary approaches such as genetic algorithm, genetic programming, and so on. In this line of research, we have also presented a method of applying CAM-Brain, evolved neural networks based on cellular automata (CA), to control a mobile robot. However, this approach has a limitation to make the robot to perform appropriate behavior in complex environments. The multi module coordination method can make complex and general behaviors by combining several modules evolved or programmed to do a simple behavior. In this paper, we have attempted to coordinate several modules evolved to do a simple behavior by Maes's Action Selection Mechanism. Maes has proposed a mechanism for action selection, which is reviewed here and is evaulated using a simulated environment. Experimental results show that this approach has potential to develop a sophisticated evolutionary neural controller for complex environments.

1 Introduction

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 indicated 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 [1].

There are many studies of constructing mobile robot controller with different approaches such as evolving neural network by genetic algorithm [2], using genetic programming [3], combining fuzzy controller with genetic algorithm [4] and programming [5]. In previous work [6], we presented CAM-Brain, evolved neural networks based on cellular automata [1,6], and applied it to controlling a mobile robot.

Because this evolutionary approach 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. The method is coordinating several modules evolved or programmed to do a simple behavior by action selection mechanism. Some researchers combine several modules evolved or programmed to do a simple behavior such as "going straight," "avoiding obstacles," "seeking object," and so on. They expect the controller combined with several modules can do complex behaviors [5,7,8].

Each neural network can be evolved or programmed. Evolved neural network is based on CAM-Brain model, and a programmed module controls the robot directly. We apply Pattie Maes's Action Selection Mechanism (MASM) [9] to coordinate the modules and control a mobile robot in simulated environments. The rest of this paper introduces CAM-Brain model and basic behaviors, and presents the coordination method in detail. The detailed description of simulation follows, and the results of simulation are given.

2 Neural Networks Evolved on CA

2.1 CAM-Brain

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 [1]. CBM is programmable logical device implemented with FPGA (Field Programmable Gate Array). It can create and evolve CA based neural network modules composed of ten thousands of modules in real time [1].

There are few simulation results of this model to some problems such as binary comparator, timer, sine curve generator, and so on [10]. Especially, they show that this model can solve XOR problem which is often used for

benchmarking neural networks and some more complex problems such as multiple timer and pattern detector [11]. Recently, they have attempted to make tens of thousands of evolved neural network modules to control a lifesize kitten robot, "Robokoneko," for showing off the capacities of an artificial brain.

CAM-Brain is developed by its own chromosome: One chromosome is mapped to exactly one neural network module. Therefore, with genetic algorithm working on this chromosome, it is possible to evolve and adapt the structure of the neural network for a specific task. Figure 1 shows the evolution process of CAM-Brain. It is the basic idea of CAM-Brain that brain-like system can be constructed by combining many neural network modules of various functions [1,6]. This section illustrates a design of CA-space for developing a neural network module.

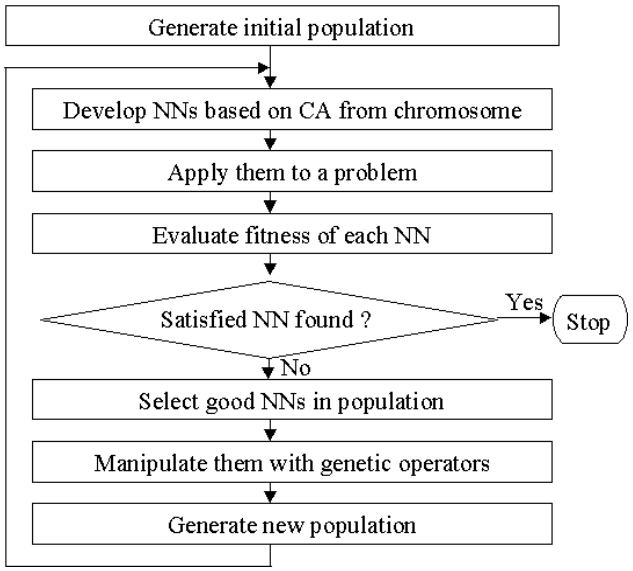

Figure 1: The evolution process of CAM-Brain.

2.2 CoDi Model

The CoDi model is a simplified cellular automata based neuron model. It is used to evolve cellular automata based neural network modules for ATR's artificial brain project "CAM-Brain" [1]. In this model, the neural network structure composed of blank, neuron, axon and dendrite grows inside 2-D or 3D CA-space by state, neighborhoods and rules encoded by chromosome. If cell state is blank, it represents empty space and cannot transmit any signals. Neuron cell collects signals from surrounding axon cells. Axon cell sends signals received from neurons to the neighborhood cells. Dendrite cell collects signals from neighborhood cells and passes them to the connected neuron in the end [1,6].

A chromosome leads to exactly one neural network. Figure 2 shows chromosome representation. The CoDi model uses a distributing chromosome to encode its structure. The chromosome is initially distributed throughout the CA-space, so that every cell in the CA-space contains one instruction of the chromosome, i.e, one growth instruction, so that the chromosome belongs to the network as a whole. To represent whole structure of a neural network, a 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), and decides the directions of sending received signals to neighborhood cells (N, S, E, W, T and B bits). The signal can be only set to the direction in which the bit corresponds to 1.

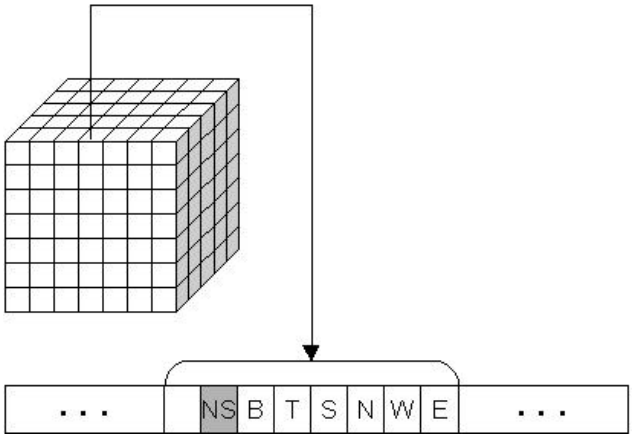

Figure 2: Information encoded in choromosome.

To decide in which directions axonal or dendritic trails should grow, the grown cells consult their chromosome information which encodes the growth instructions. These growth instructions can have an absolute or a relative directional encoding. An absolute encoding masks the 6 neighbors of a 3D cell with 6 bits. The cell contiguous to the face whose bit is set will itself become an axon cell or dendrite cell. For a relative encoding, 5 bits are sufficient, as the direction from which the initial growth signal comes is known to a cell. The growth information can be interpreted according to this direction, so that one bit is saved. A relative encoding saves one bit in the cell state, which can be crucial for CAM (Cellular Automata Machine). 5 bits of chromosome information in each cell offer 32 different growth instructions. Using its chromosome instructions, a cell can determine the neighbors to which it transmits the incoming growth signals.

The growth phase organizes neural structure and makes the signal trails among neurons. First, 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. 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. Next 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. Finally, repeating this process, the neural network is obtained when the state of every cell changes no longer.

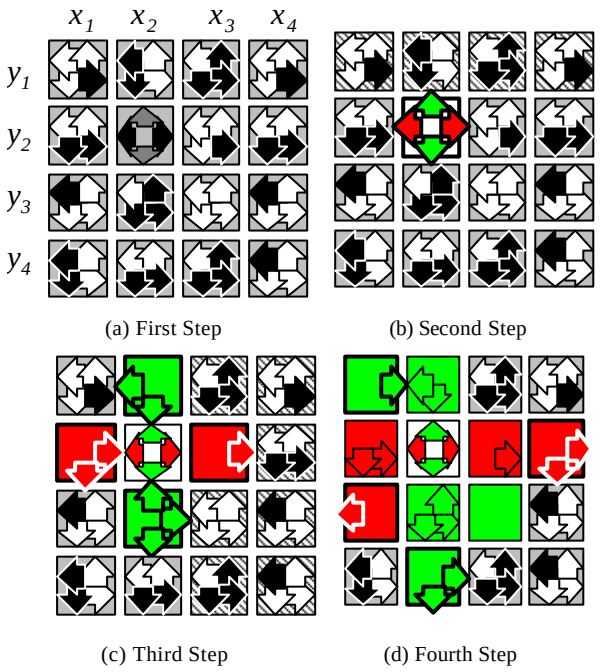

Figure 3: Growth phase. (a) Black arrow represents a neuron is located in (x_2,y_2). (b) The neuron sends growth signals. (c) The cell state is decided according to the type of growth signals. (d) Propagating growth signals, blank cells become axon or dendrite.

Figure 3 shows the growing process in 4 × 4 2-D CA-space. In this figure, the cell which has oblique lines is blank cell, and the blank arrows show the direction of signaling decided by chromosome. Figure 3(a) shows the process of sending a neuron in blank cells, where a neuron is located in (x_2,y_2). Figure 3(b) shows that the neuron cell sends growth signal to surrounding cells. Figure 3(c) shows the cell state is changed by growth signal. Figure 3(d) shows that blank cells grow into axon or dendrite. In a neuron, the dendrite collects signals and sends to the neuron, and the axon distributes signals originated from the neuron.

Figure 4 shows growing process of neural network inside 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 only made at the initial state as shown in Figure 4(a). Neuron cells send two kinds of growth signals to their neighbors, either "grow a dendrite" and "grow an axon". Figure 4(b) shows the growing axon and dendrite by growth signals. The grown cells never change their type and send the signal to their neighbor blank cells. Figure 4(c) shows the progress of growing axon and dendrite. Figure 4(d) shows the completion of one neural network module.

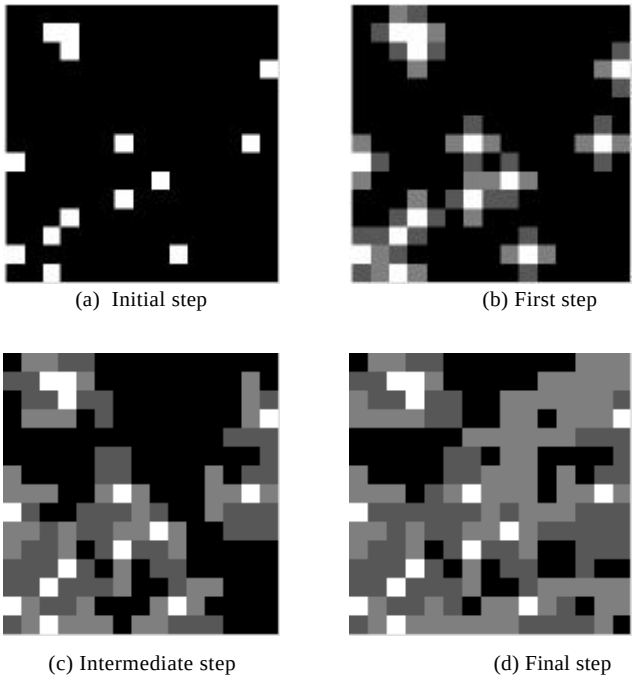

Figure 4: The process of growth of neural structure.

Signaling phase transmits the signal from input to output cells continuously. The trails of signaling are transmitted with evolved structure at the growth phase. Each cell plays a different role according to the type of cells. 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 the threshold. If the cell type is dendrite, it collects data from the faced cells and eventually passes them to the neuron body. The position of input and output cells in CA-space is decided in advance. At first, if input cells produce the signal, it is sent to the faced axon cells, which distribute that signal. Then, neighborhood dendrite cells belonged to other neurons collect and send this signal to the connected neurons. The neurons that have received the signal from dendrite cells send it to axon cells. Finally, dendrite cells of output neuron receive and send this signal to the output neurons. Output value can be obtained from output neurons.

During signaling phase, the fitness is evaluated by the output 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 5 shows the

directions of singals after neuron, axon and dendrite are made. In this figure, neuron sends excitatory singal (+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 cell and sends them to neuron, and axon cell distributes the signals originated from neuron to neighborhood cells.

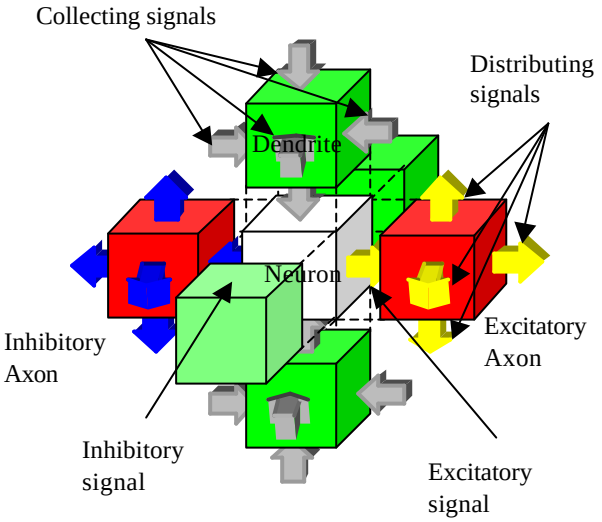

Figure 5: Signaling phase.

2.3 Evolution of CAM-Brain

In general, simple genetic algorithm generates the population of individuals and evolves them with genetic operators such as selection, mutation, and crossover [12]. Because a chromosome leads to exactly one neural network in CoDi model, it has a unique fitness value and hence do not have to spend time calculating an average fitness for each chromosome. The advantage of CoDi model is that the structural integrity of the basic components (the neurons) is preserved in the presence of mutations (or alternations in the chromosome), so a consistent structure of the neural network can be guaranteed.

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 fit individuals on the given problem.

3 Coordination of Multiple Modules

In behavior-based robotics the control of a robot is shared between a set of purposive perception-action units, called behaviors. Based on selective sensory information, each behavior produces immediate reactions to control the robot with respect to a particular objective, i.e., a narrow aspect of the robot's overall task such as obstacle avoidance or wall following. Behaviors with different and possibly incommensurable objectives may produce conflicting actions that are seemingly irreconcilable. Thus a major issue in the design of behavior-based control systems is the formulation of effective mechanisms for coordination of the behaviors' activities into strategies for rational and coherent behavior. This is known as the action selection problem (also refered to as the behavior coordination problem) [13].

To solve coordinating problem of CAM-Brain behavior modules, Maes's Action Selection Mechanism (MASM) proposed originally as an improvement over previous approaches to action selection in the field of AI is used. In particular, it was proposed as an improvement on traditional planning systems and reactive systems [8,9].

3.1 Maes's Action Selection Mechanism

MASM is a distributed, non-hierarchical network. There are two waves of input to the network: from the sensors of the external environment (mostly external stimuli) and from the motivations or goals (mostly derived from internal stimuli such as body temperature). Nodes of network are behavior modules such as obstacle avoidance, wall following.

MASM is composed of nodes, internal links and external links. Each node has a set of preconditions. The preconditions are logical conditions about the environment that are required to be true in order for the node to be executable. The add list consists of conditions about the environment that the node is likely to make true. The delete list consists of conditions that are likely to be made false by the execution of the node. The final two components of the node are the activation level and the code that gets run if the node is executed. The internal links are specified in Table 1. The external links providing input to the network are specified in Table 2. Table 1 and 2 contain the descriptions of links.

Iterative procedure of MASM selects one proper action among many behavior modules at each time. This procedure is as follows.

1. Calculate the excitation coming in from the environment and the motivations (or goals).
2. Spread excitation along the predecessor, successor, and conflictor links.
3. Normalize the node activations so that the average activation becomes equal to the constant π.

4. If any nodes are executable, choose the one with the highest activation and execute it. Set global threshold as original value and go to 1.
5. If no node is executable, reduce the global threshold and go to 1.

A node is executable if all its preconditions are true and if its activation is greater than the global threshold. If more than one node is executable after a cycle, the one with the highest activation is chosen. Figure 6 shows the procedure of MASM.

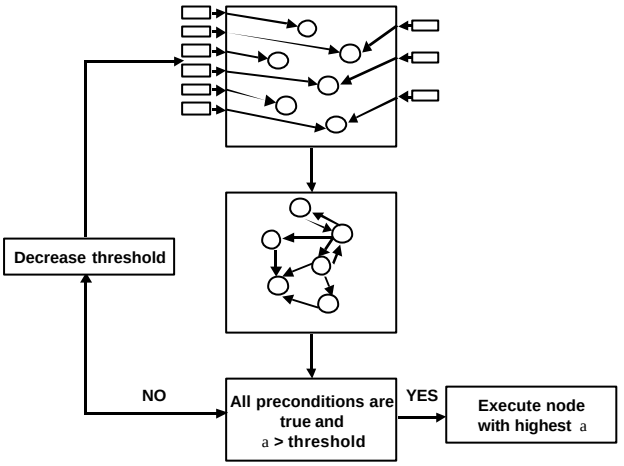

Figure 6: The procedure of MASM (α=activation).

Table 1: Internal links.

Type	Description
Predecessor link	If (proposition X is false) and (proposition X is a precondition of node A) and (proposition X is in the add list of node B), then there is an active predecessor link from A to B.
Successor link	If (proposition X is false) and (proposition X is in the add list of node A) and (proposition X is a precondition of node B) and (the node A is executable), then there is an active successor link from A to B.
Conflictor link	If (proposition X is true) and (proposition X is a precondition of node A) and (proposition X is in the delete list of node B), then there is an active conflictor link from A to B.

3.2 Tunning of MASM

Five global parameters are used to tune the performance of the ASM to a particular environment. π is the mean activation value used in normalization. θ is the initial value of the global threshold which is reduced by an amount (e.g., 10%) after each cycle if there is no node to be executable. ϕ is used to determine the weights of environmental sensor inputs and successor link. δ is used to determine the weights of protected goal inputs and conflictor links. γ is used to determine the weights of the inputs from the goals, successor links and conflictor links. MASM has six types of inputs to a node. Each input is multiplied by the different weights. They are as follows.

- External link from environmental sensor: ϕ
- External link from goals: γ
- External link from protected goals: δ
- Internal successor link: ϕ/γ
- Internal predecessor link: 1
- Internal conflictor link: δ/γ

Table 2: External links.

Type	Description
From sensors of the environment	If (proposition X about the environment is true) and (proposition X is a precondition of node A), then there is an active link from the sensor of the proposition X to node A.
From goals	If (goal Y has an activation greater than zero) and (goal Y is in the add list of node A), then there is an active link from the goal Y to node A.
From protected goals	If (goal Y has an activation greater than zero) and (goal Y is in the delete list of node A), then there is an active link from the goal Y to node A.

4 Experimental Results

Khepera Simulator Ver 2.0 is used to show the performance of coordinating multiple CAM-Brain behavior modules using MASM. Two CAM-Brain behavior modules such as avoiding obstacle and following light are developed in previous work [6,14]. MASM model is designed for navigating in simulation environment long time.

4.1 Khepera Simulator

Khepera robot contains 8 infrared sensors to detect by reflection the proximity of objects in front of it, behind it, and to the right and the left sides of it, and to measure the level of ambient light all around the robot. Also, the robot has two motors to control the left and right wheels. Khepera Simulator Ver 2.0 which is developed by Olivier Michel is used for experiment [15]. Khepera simulator also features the ability to drive a real Khepera robot, so that we can easily transfer the simulation results to the real robot [3,4,5]. Distance sensor 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. Light sensor returns a value ranging between 50 and 500. 50 means that light source is very close to the sensor.

Figure 7 shows the interface of robot simulator. Left side of the interface shows trajectory of the khepera robot. Right side of the interface shows the status of action selection model and the status of khepera mobile robot. While the robot moves, the interface draws the trajectory of robot and the snapshot of action selection model.

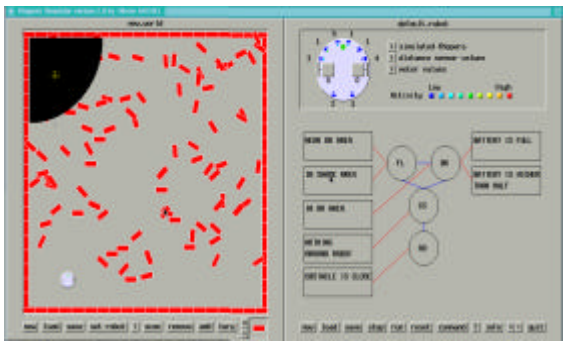

Figure 7: Khepera simulator.

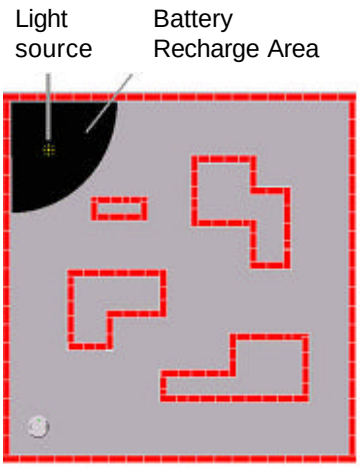

Figure 8: Simulation environments.

4.2 Basic Behaviors

In this paper, four basic behaviors are defined as follows. They are mapped to the nodes of MASM.

- Recharging Battery: 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. This module must be operated to go to the battery recharge area because the light source exists in that area.
- Avoiding Obstacle: If the obstacles exist around the robot, it avoids them without bumping against them.
- Going Straight: If there is nothing around the robot, it goes ahead. This module takes it to move continuously without stopping.

Recharging Battery and Going Straight modules are programmed because they are very simple. Figure 8 shows simulation environments. At first, robot has initial battery level. When the robot moves one step, robot's battery level decreases one level. To navigate long time in this environment, robot must go to the battery recharge area if robot's battery is low.

4.3 Coordination Model

MASM needs the definition of environment sensors and goals. Also, the relationships between behavior modules (nodes) are required. Figure 9 shows the MASM model for simulation environment. After designing MASM model, parameter setting is conducted by trial and error. We set the values of π, θ, γ, ϕ, and δ in MASM model as follows: $\pi = 4.5$, $\theta = 3.0$, $\gamma = 0.8$, $\phi = 1.2$, and $\delta = 1.0$. In this environment, there are five environment sensors and two goals. Inputs from environment sensors are binary-valued and those from goals are real-valued [8].

In this environment, five environment sensors are set for action selection. They are as follows.

- "In battery recharge area" : Checks if robot is in battery recharge area.
- "Obstacles are close" : Checks if the maximum value of distance sensors is larger than 700.
- "Near battery recharge area" : Checks if the distance from robot to light source is less than 800. Light source is the center of battery recharge area.
- "Light is low": Checks if the minimum value of light sensors is larger than 400.
- "Nothing around the robot" : Checks if the maximum value of distance sensors is less than 700.

We set two goals such as "Full battery," and "Not zero battery." Because robot's battery decreases while robot moves, the robot tries to maintain high battery value to live long. They are set as follows.

- "Full battery" :

$$c = \frac{m - n}{m}$$

c : Value of "Full battery"
m: Maximum battery

n : Robot's battery

- "Not zero battery" : Checks if battery is less than half of the maximum battery.

If robot has low battery level, "Full battery" gives high activation to recharging battery. If robot's battery value is lower than half of the maximum, "Not zero battery" gives activation to recharging battery.

Between the node and sensors in precondition, there exists external link. Between the goal and a node which directly helps to achieve it, there exists an external link. Table 3 describes preconditions of the nodes. Each node has one or two preconditions. The relationships among nodes are decided by successor links or predecessor links. If predecessor link is from A node to B node, successor link from B node to A node exists. If node A and node B are connected, they exchange their activation values while activation spreads. Table 4 describes add lists of nodes and

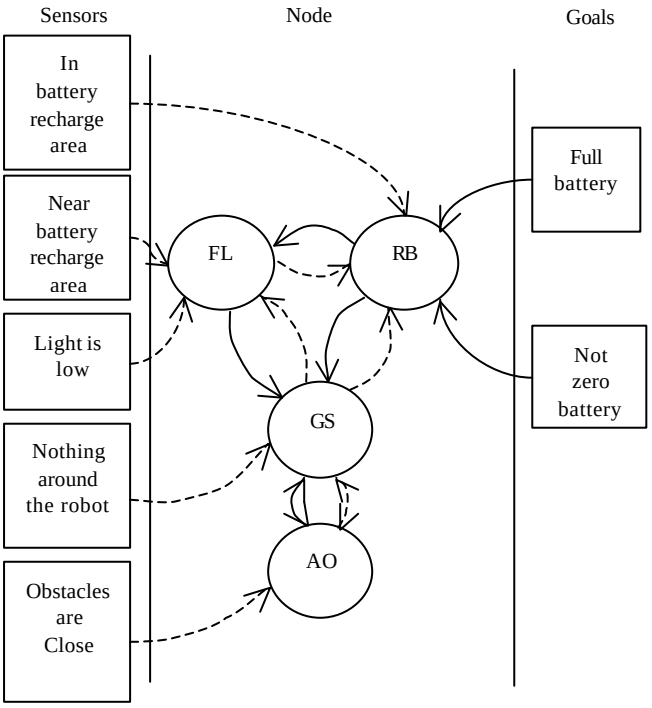

Figure 9: MASM model. Solid lines denote goal or predecessor connections, and dashed lines denote sensor or successor connections.

Table 3: Preconditions of nodes.

Node	Preconditions
Recharging Battery	In battery recharge area
Following Light	Light is low, Near battery recharge area
Avoiding Obstacle	Obstacles are close
Going Straight	Nothing around the robot

Table 4: Add lists of nodes.

Node	Add lists
Recharging Battery	Full Battery, Not zero battery
Following Light	In battery recharge area
Avoiding Obstacle	Nothing around the robot
Going Straight	Obstacles are close, In battery recharge area, Near battery recharge area

4.4 Result Analysis

Robot moves 4942 times in simple environment when initial battery level is 2500. Figure 10 shows robot's trajectory in simple and chaotic environments. We can analyze robot's behavior: When robot is in battery recharge area, sensor "In battery recharge area" becomes true, and recharging battery module can be selected. Because "In battery recharge area" is a precondition of recharging battery module. When battery is low, robot's goals increase and following light module is selected more frequently than other behavior modules.

Robot selects avoiding obstacle and following light modules successively before battery recharge area. This helps robot go battery recharge area without bumping against obstacles. By selecting lower level behavior, robot can make high level behavior which we cannot make easily. Figure 11 shows action selection of robot. Robot selects going straight and avoiding obstacle many times. Robot selects following light and recharging battery very little. Robot chooses following light and then recharging battery. Going to battery recharge area, robot must choose following light first. This shows our model works properly. If robot is not near battery recharge area, robot wanders. To prevent from bumping, robot chooses going straight and avoiding obstacle frequently. Our model helps robot go battery recharge area when robot's battery is low. This shows that action selection model can help robot archieve goals.

To see the effect of noise, we have modified the source code of Khepera Simulator from the information of Khepera Simulator Homepage [15]. After adding some noise to distance sensor, the movement of the robot is not better than before, but robot can achieve goals and navigate long time even in the noisy environment. Figure 12 shows the simulation results of Khepera Simulator in the original and noisy environments. This shows the coordination capability of the robot is also guaranteed in the real Khepera robot.

5 Conclusions

In this paper, we have applied Maes's Action Selection Mechanism to coordinate multiple CAM-Brains for robot control. Robot selects basic behavior modules from behavior networks. MASM gets signals from environments and internal motivations, and spreads the activation among

internal links. We have four basic behaviors which are evolved on CAM-Brain or programmed. MASM model can be used to control mobile robot by modeling environments. Robot achieves goals by selecting basic behaviors, which shows MASM has potential to coordinate several neural network modules for robot control.

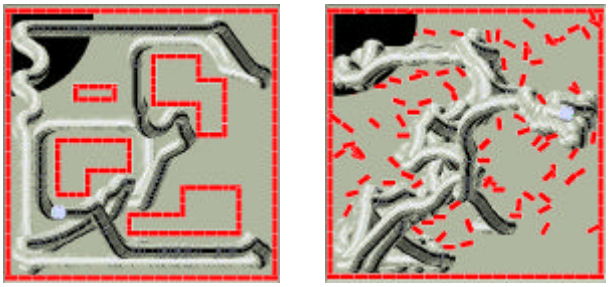

Figure 10: Robot's trajectory in simple and chaotic environments.

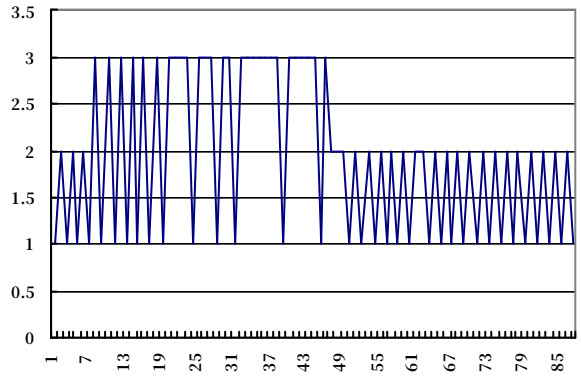

Figure 11: Action selection of robot (0=Recharging Battery, 1=Following Light, 2=Avoiding obstacle, and 3=Going Straight).

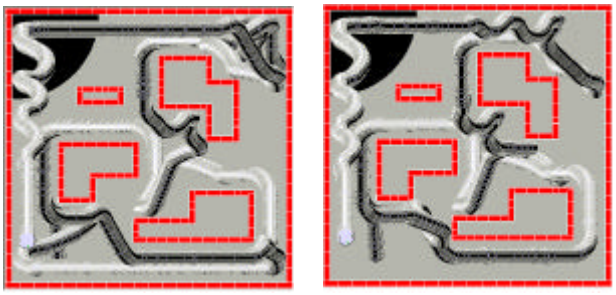

Figure 12: Simulation of Khepera robot in noisy environments. Left is original environment, and right is noisy environment.

Bibliography

[1] F. Gers, H. de Garis and M. Korkin, "CoDi-1Bit: A simplified cellular automata based neural model," *Proc. Conf. on Artificial Evolution*, Nimes, France, Oct, 1997.

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

[3] P. Nordin and W. Banzhaf, "Real time control of a Khepera robot using genetic programming," *Cybernetics and Control*, Vol. 26, No. 3, pp. 533-561, 1997.

[4] S.-B. Cho and S.-I. Lee, "Evolutionary learning of fuzzy controller for a mobile robot," *Proc. Int. Conf on Soft Computing, pp*. 745-748, Iizuka, Japan, 1996.

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

[6] G.-B. Song and S.-B. Cho, "Incremental evolution of CAM-Brain," *Proc. AROB99*, pp. 297-300, Beppu, Japan, 1999.

[7] 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.

[8] T. Tyrrell, "An evaluation of Maes's bottom-up mechanism for behavior selection," *Adaptive Behavior*, Vol. 2, pp. 307- 348, 1994.

[9] P. Maes, "How to do the right thing," *Connection Science Journal*, Vol 1, No. 3, pp 291-323, 1989.

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

[11] H. de Garis, M. Korkin, F. Gers, E. Nawa, and M. Hough, "ATR's artificial brain (CAM-Brain) project: A sample of what individual CoDi-1Bit model evolved neural net modules can do," *Proceedings of International Conference on Artificial Life and Robotics*, Japan, 1999.

[12] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, 1989.

[13] P. Pirjanian, "Behavior coordination mechanism–state-of-the-art," *Tech-report IRIS-99-375*, School of Engineering, University of Southern California, Oct, 1999.

[14] S.-B. Cho and G.-B. Song, "Evolving CAM-Brain to control a mobile robot," *Applied Mathematics and Computation*, vol. 111, pp. 147-162, 2000.

[15] Khepera Simulator Homepage, http://diwww.epfl.ch/lami/team/michel/khep-sim/.