

Selection of Neural Architecture and the Environment Complexity

Genci Capi¹, Eiji Uchibe^{1,2} and Kenji Doya^{1,2}

¹ CREST, Japan Science and Technology Cooperation

² ATR, Human Information Science Laboratories

“Keihanna Science City”, Kyoto, 619-0288, Japan

E-mail:gcapi@atr.co.jp

Abstract. In this paper we consider how the complexity of evolved neural controllers depends on the environment using foraging behavior of the Cyber Rodent in two different environments. In the first environment, each fruit can be seen from limited directions and different groups of fruits become ripe in different periods. In the second environment, fruits inside a zone are rewarding and those outside are aversive. After evolution, agents with recurrent neural controller outperformed those with feed-forward controllers by effectively using the memory of border passage. Simulation and experimental results with the Cyber Rodent robot confirmed the selection of appropriate complexity of neural controller, both in size and structure, through evolution.

1 Introduction

The neural architectures found in different animal species, at different stages of development, have tremendous diversity. While the sensory-motor mapping and pattern generation networks are exactly identified in some invertebrates, simple movement by a human is dependent on the complex hierarchical networks of the spinal cord, the brain stem, the cerebellum, the basal ganglia, and the cerebral cortex. What is the origin of such diversity in the network architecture?

It is intuitively expected that the more complex the environment, the more complex neural architecture the animals therein should have. Conversely, animals that have more complex neural architecture can perform more challenging behaviors needed for survival and reproduction. The goal of this study is to test such a hypothesis in a evolutionary experiments of simulation and hardware experiments of Cyber Rodent.

We focus on the foraging behavior in two different environments. This behavior underlies many more complex behaviors such as following a target or another agent, object manipulation. In the first environment the fruits are randomly scattered and they can not be seen from all directions. Also, they are ripe only for a short time. In this way the agent must combine exploration and directed approach the environment to efficiently eat fruits. In the second environment some fruits are inside and some outside a surrounded zone. The agents get a positive reward if it eats the fruits inside the zone and a negative reward if the fruit eaten is outside. All the fruits have the same characteristics (color and shape). Therefore, the agent must remember if it is inside or outside the zone.

We compare the results for a feed-forward Neural Network (FFNN) controller and a recurrent Neural Network (RNN) with two memory units. The simulation and experimental results show that the FFNN controller works well with the first environment but the performance in the second environment is poor. On the other hand the RNN performance in the second

environment is much better compared with the FFNN. The Neural Networks are evolved using Genetic Algorithm (GA), which has recently become an interesting research topic for many researchers (Nolfi, 1997, Mondada 1995, Floreano 1994). Its great strength as a tool for the construction of complex behaviors, is that it is in the nature of the GA to be able to evolve successful and functional systems without caring about the internal complexity of the dynamical or interactions which determine the quality of the performance.

In our work, the inputs of the neural controller are obtained by utilizing the visual sensor of the Cyber Rodent (CR) robot. The experimental results show that neural controller evolved by simulation gives good results when it is implemented in the physical mobile robot. Also, the input data used in the simulation are well obtained by the physical visual system of CR robot. In most of the previous approaches in evolving neural controllers that used physical mobile robots, the values of the proximity sensors are used as inputs of the neural network.

This paper is organized as follows. In section 2 the CR robot is presented. Section 3 discusses the problem and the proposed algorithm. Simulation and experimental results are given in Section 4. Conclusions and future works are given in Section 5.

2 CR Robot

The CR robot is a two wheel driven mobile robot as shown in Fig.1. It is 250mm long and weights 1.7 kg. The CR is capable of moving wheelie and it is equipped with:

- Omni-directional CCD camera.
- Ultrasonic range sensor.
- Seven infrared proximity sensors.
- 3-axis acceleration sensor.
- 2-axis gyro sensor.
- Red, green and blue LEDs for visual signaling.
- Audio speaker and two microphones for acoustic communication.
- Infrared port to communicate with a near-by-agent.
- Wireless LAN card and USB port to communicate with the host computer.

The CR has a Hitachi SH-4 CPU with 32 MB memory. The FPGA graphic processor is used for video capture and image processing at 30 Hz. 5 proximity sensors are in the front of robot, 1 behind and 1 under the robot pointing downwards. The proximity sensor under the robot is needed when the robot moves wheelie.

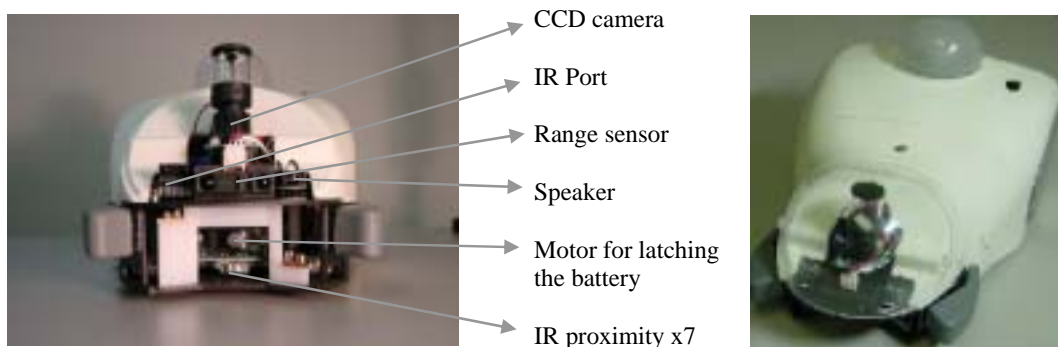


Figure 1. CR robot.

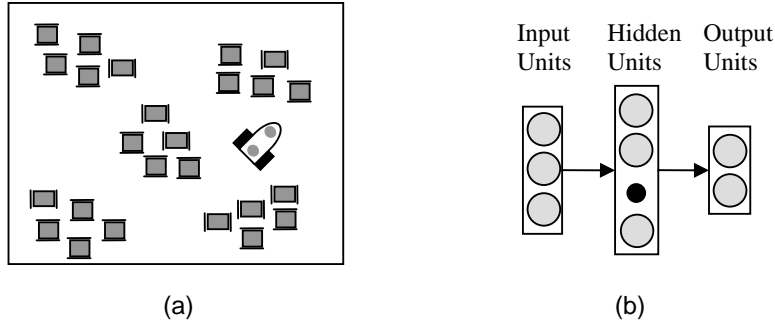


Figure 2. Environment and FFNN used in the experiment 1.

3 Algorithm

3.1 Experiment 1

Figure 2(a) shows the simulated robot in the environment used in experiment 1 which is a square of 5m x 5m. The red color fruits are distributed in the environment randomly and the CR robot is placed initially in a random position and orientation. In real life fruits are harder to find and are only ripe for a brief period of time. Therefore, the fruits in environment 1 can not be seen from all directions and stay in the environment for a short period of time. During the agent's life five groups of fruits ripe randomly one after the other.

In the environment 1, we considered a single layer FFNN, as shown in Figure 2(b). The FFNN receives 3 inputs: the angle to the nearest fruit, the fruit color and a constant bias input. The robot uses the visual sensor to provide the angle and the color of the nearest fruit. The angle input varies from -45° to $+45^\circ$. The output units directly control the right and left wheel velocities. The output units use a sigmoid activation function where 0 corresponds to no motion and 1 corresponds to full speed forward rotation. Also, the hidden units use a sigmoid activation function.

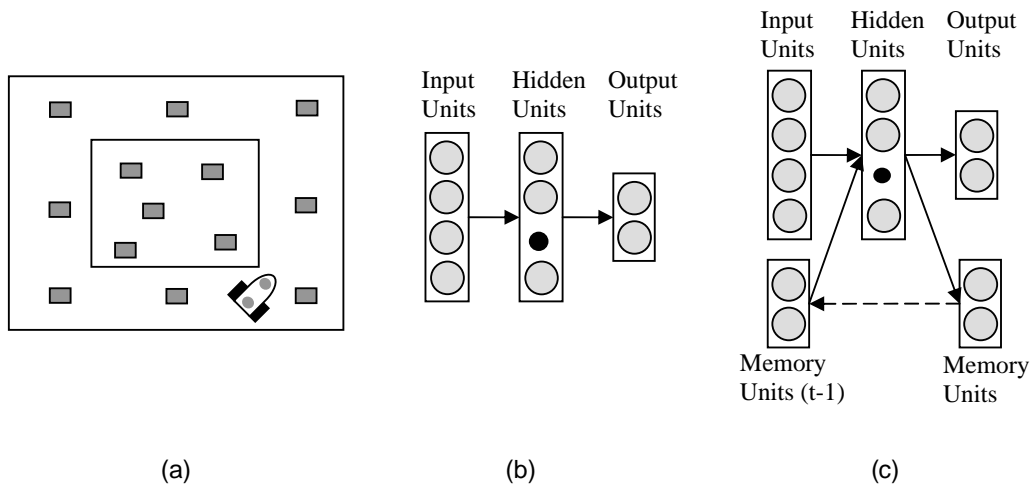


Figure 3. Environment, FFNN and RNN architectures used in the experiment 2. The solid arrows represent fully connected layers. The dashed arrows represent one for one basis with 1 weight connection.

3.2 Experiment 2

The second environment contains 13 fruits with the same color and shape. The fruits can be seen from all directions. Five of them are inside a zone and eight of them outside, as shown in Figure 3(a). First the robot is placed outside the zone. The robot get a positive reward when eats the fruits inside the zone and a negative reward when eats the fruits outside it. In order to get a higher reward, the agent has to remember if its position is inside or outside the zone.

In the experiment 2, the number of input units is 4. We added another input unit which is fully activated in the moment when robot crosses the border of the rewarding. The neural architectures are a FFNN and a RNN with two memory units (Elman 1990), which are used as extra inputs to hidden units in the next time step (Figure 3(b), 3(c)). The hidden units in FFNN use the sigmoid activation function. In both architectures, the number of hidden nodes is evolved by GA.

3.3 GA

A real-value GA was employed in conjunction with the selection, mutation and crossover operators. Many experiments, comparing real value and binary GA, show that the real value GA generates better results in terms of the solution quality and CPU time (Michalewicz 1994). Initially, 100 individuals are randomly created. The GA optimizes the weight connections and the number of hidden nodes of the neural controllers. Rather than use variable-length genotypes to allow varying numbers of hidden neurons in the network, we use fixed-length genotypes with the maximum searching number of the hidden nodes. When the number of hidden neurons is smaller than the maximum number, some parts of genome are ignored during the crossover and mutation operations. The number of hidden neurons in the initial population is randomly generated by GA between 1 and 8. For the FFNN and RNN in the experiment 2 the genotype of each individual encodes all the connection weights. The weights are encoded as real numbers between -10 and 10. Each individual of the population, which presents a possible neural controller, controls the CR robot during a trial period of 20 seconds. The fruits in the environments disappear after the agent eats them. Maximum number of generations is used as a termination function. The GA selection, mutation and crossover functions and parameters are given in Table 1.

Function Name	Parameters
Arithmetic Crossover	2
Heuristic Crossover	[2 3]
Simple Crossover	2
Uniform Mutation	4
Non-Uniform Mutation	[4 GNmax 3]
Multi-Non-Uniform Mutation	[6 GNmax 3]
Boundary Mutation	4
Normalized Geometric Selection	0.08

Table 1. GA functions and parameters.

4 Simulation and Experimental Results

4.1 Experiment 1

In the environment 1 the evolved FFNN architecture has 4 hidden nodes. Figure 4 shows the fitness value of the best individual (number of fruits eaten) and the mean of the fitness for every generation. In Figure 5 is shown the agent performance in a simulated environment with three groups of fruits. First the group of fruits in the left is ripe, then the group on the right and at the group in the center. As can be seen the agents rotates around and search when there is no fruit in the visual information.

An experimental result with the CR robot hardware is shown in Fig. 6. The robot could capture colored paper fruits in a smooth motion despite the noises in the visual sensor. A captured image by the visual sensor during the experiments is shown in Figure 7. From the image captured, the red color is extracted and the angle to the nearest fruit is calculated. In order to reduce the time to calculate the angle and the wheels angular velocities during the experiments, we used look-up tables for the sigmoid and inverse tangent functions. The sensor data is updated every 100 μ s.

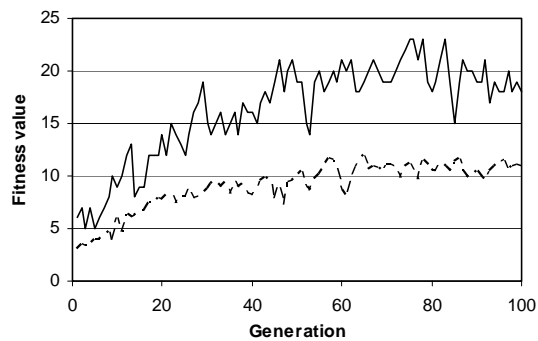


Figure 4. Fitness value of the best individual (solid line). Dotted line is the mean value of every generation.

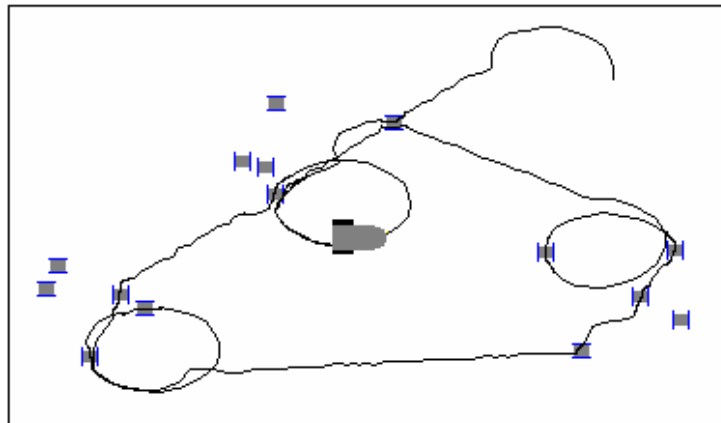


Figure 5. FFNN performance in experiment 1.



Figure 6. Video capture of CR robot during the experiment.

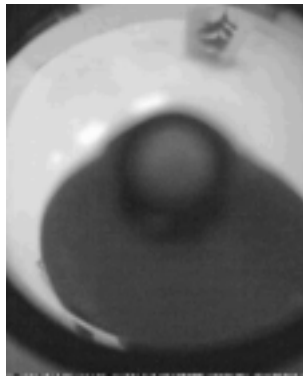
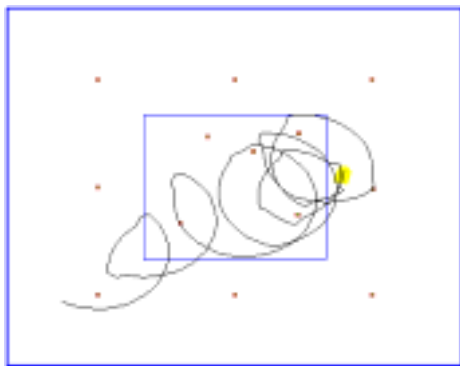


Figure 7. Image captured during the experiment.



(a)



(b)

Figure 8. FFNN and RNN performances in experiment 2.

4.2 Experiment 2

In the second environment we used the same architecture for the FFNN evolved in the first environment and evolved all the weight connections. But, its performance is bad because of its lack of feedback (Figure 8(a)). The agent controlled by FFNN can not remember whether it

has passed the zone border or not. Because the number of fruits outside the zone is larger than the fruits inside the agent avoids eating fruits.

RNN architecture quickly evolved to robustly solve the task. The RNN controller has two hidden neurons. The performance of the RNN is shown in Figure 8(b). The robot initial position is outside the zone. Due to the strong connection weights of the right wheel, the robot rotates right when there is no fruit in the visual sensor. The agents avoid eating fruits outside the zone and stays inside the zone most of the time. The best agent ate 4 fruits inside the zone during the life time. Although the agent avoids eating fruits outside the zone, its performance inside the zone may still be improved by considering other RNN architectures.

5 Summary

In this paper we presented the results of evolved neural controllers for foraging behavior. FFNN and RNN were tested in two different environments. In the environment 1 where the fruits were randomly scattered, the FFNN controller performed well. But in the environment 2, where the agent must remember its position relative to a surrounded zone the RNN controller gave better results. The performance evaluation was carried out by simulation and experiments. Based on the simulation and experimental results we conclude:

- 1- The complexity of the environment influences the size, structure and architecture of the neural controller.
- 2- The controllers developed in the simulation direct the CR robot very well during the experiments. This indicates that the technique can be used to create a controller which can steer the robot in more complex environments.

In the future we plan to consider:

- 1- The agent performance for fixed or learned policy.
- 2- Evolution of behaviors in real hardware by utilizing the CR robot.

References

- [1] **S. Nolfi**, Evolving Non-Trivial Behaviors on Real Robots: A Garbage Collecting Robot, *Robotics and Autonomous Systems* **22**, pp. 187-198, 1997.
- [2] **F. Mondada, and D. Floreano**, Evolution of Neural Control Structures: Some Experiments on Mobile Robots, *Robotics and Autonomous Systems* **16**, pp. 183-195, 1995.
- [3] **D. Floreano, and F. Mondada**, Automatic Creation of Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot, *Proceedings of 3^d International Conference on Simulation of Adaptive Behavior*, MIT Press, pp. *** 1994.
- [4] **J. Elman**, Finding structure in time, *Cognitive Science*, 14, pp. 179-211, 1990.
- [5] **Z. Michalewicz**, *Genetic Algorithms + Data Structures = Evolutionary Programs*, Springer-Verlag, 1994.