Investigation of Simply Coded Evolutionary Artificial Neural Networks on Robot Control Problems

Yoshiaki Katada and Jun Nakazawa

Abstract— **One of the advantages of evolutionary robotics over other approaches in embodied cognitive science would be its parallel population search. Due to the population search, it takes a long time to evaluate all robot in a real environment. Thus, such techniques as to shorten the time are required for real robots to evolve in a real environment. This paper proposes to use simply coded evolutionary artificial neural networks for robot control to make genetic search space as small as possible and investigates the performance of them using simulated robots. Two types of genetic algorithm (GAs) are employed, one is the standard GA and the other is an extended GA, to achieve higher final fitnesses as well as achieve high fitnesses faster. The results suggest the benefits of the proposed method.**

I. Introduction

Embodied cognitive science[1] have attracted much research interest in recent years, where a robot must interact with a real world environment for obtaining successful and robust behaviors. One of approaches in embodied cognitive science is evolutionary robotics[2], where robot control systems are designed by using evolutionary techniques.

It has been pointed out that evolutionary approaches are potentially advantageous over other approaches due to their parallel population search. However, they make difficult to use physical robots without any consideration, especially in the case of tasks with human intervention because they take a long time to evaluate all robots in a real environment.

One approach to overcoming this problem would be the use of simulations. Computer simulations may be helpful to reduce the amount of experimental time to evaluate individuals in a population. However, the controllers evolved in a simulated environment do not always work well in the real one because of uncertain effects, e.g., noise and differences in the electronics and mechanics of robots [3]. Therefore, the validity of simulations is a particularly relevant problem. Some researchers claimed that the problem described above can be overcome by carefully designing simulators [4][5]. Others proposed to calibrate the model of robot/environment interaction dynamics during evolution [6][7]. However, these approaches are limited to the tasks where interaction dynamics between robot and environment is properly modeled.

Therefore, we focus on conducting the evolutionary run entirely in a real environment where an evolved controller must work well, while we consider to shorten the total amount of experimental time to evolve all robots. Our approach is to make genetic search space as small as possible. In other words, if the length of the genotype becomes short, it would be possible to shorten the experimental time whatever evolutionary algorithms are employed. Moreover, two types of GAs are applied, which are the standard GA and an extended GA, in order to achieve higher final fitnesses as well as achieve high fitnesses faster.

In this paper, we propose simply coded evolutionary artificial neural networks in order to shorten the time to evolve robots and investigates the performance of them for the GAs on evolutionary robotics tasks using simulated robots. The reasons why our work conducts computer simulations in advance although we argue against simulation-based approaches in evolutionary robotics are as follows; (1) Good performances of the proposed approach in computer simulations can be considered as the precondition for good performances in real environments that the one is expected to be able to show. (2) It takes extraordinary time to investigate the ability of the proposed approach in real environments for all experimental conditions, which would be tough works. The paper is organized as follows. Section II describes how to simply code artificial neural networks using a binary string for GAs. Section III and IV define the robot control problems where the evolved neural networks are evaluated, and give the results of our experiments. Conclusions are given in the last section.

II. Simply Coded Evolutionary Artificial Neural Networks (SCEANN)

This section describes how to simply code artificial neural networks for GAs.

A. Artificial Neural Networks

Artificial neural networks (ANN) is used with N_s sensory neurons, N_o fully interconnected motor neurons and N_h fully interconnected hidden neurons for a robot's controller.

The output of the i-th neuron at time t is given by:

$$x_i(t) = f(\sum_j \omega_{ij} x_j(t-1)) \tag{1}$$

where ω_{ij} is the connection weight from the neuron j to the neuron i, and $f(x)$ is the output function of neurons, given by the sigmoid function shown in Figure 1. Namely, their outputs are given by:

$$f(x) = \frac{1}{1 + exp(-x/T)} \tag{2}$$

where T is a positive parameter to control the slope of the sigmoid function. The output range is $[0, 1]$. The output

Yoshiaki Katada and Jun Nakazawa are with the Department of Electrical and Electronic Engineering, Setsunan University, 17-8 Ikeda-nakamachi, Neyagawa, Osaka 572-8508, JAPAN (phone/fax: +81 728 39 9148; email: katada@ele.setsunan.ac.jp).

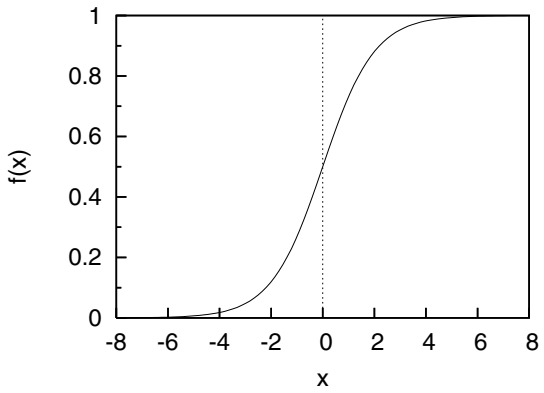


Fig. 1. Sigmoid function

of neurons given by the sigmoid function becomes easily saturated by getting successive inputs whether excitatory or inhibitory. Therefore, refractoriness[8] is introduced to reset the state of a neuron to 0 when the total amount of inputs exceeds its threshold, $\pm x_c$. In this work, $|x_c|$ is set at 8. In a behavioral level, these resets prevent a robot from being stagnated in its environment.

In general EANN, network's connection weights, firing thresholds for each neuron (the slope when the sigmoid function employed), the architecture of networks and learning rules are evolved [9]. In our approach, only connection weights are evolved as variables of GAs in order to shorten the length of the genotype. Therefore, the slope of the function, T, is set at 1, which is usually employed in pattern classification.

We proposed to use three kinds of architecture. Each architecture is genetically represented by a binary string. The details are described as follows.

B. SCEANN 1

The first one is originally proposed by Floreano [10][11], which was used for evolving real robots. A string is composed of a series of blocks, each block defined for a neuron in hidden and motor neurons (Figure 2). The first bit of a block encodes the sign $\{+, -\}$ of all the connection weights from sensory, hidden and motor neurons to a corresponding neuron and the remaining bits encode the presence/absence

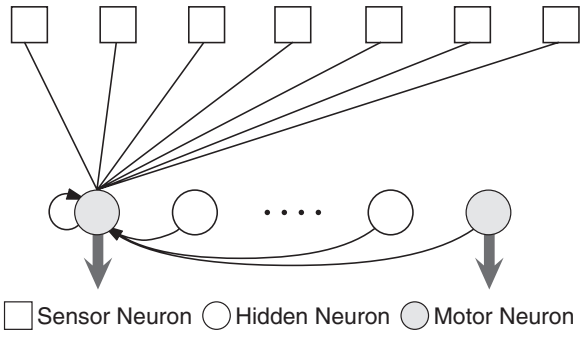
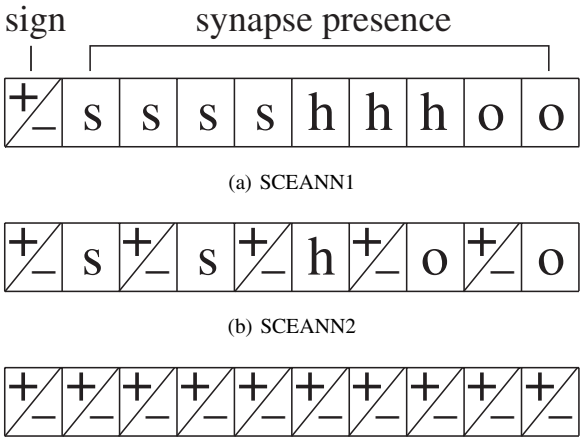


Fig. 2. Architecture of ANNs for 1 block of a string



(a) SCEANN1

(b) SCEANN2

(c) SCEANN3

Fig. 3. Genetic representation of one block

$\{1, 0\}$ of a connection from each neuron (Figure 3(a)). The synaptic strengths of all existing connections are set at 1. Therefore, the total length of the string is $L = (N_h+N_o)(1+N_s+N_h+N_o)$ bits, where N_s, N_h and N_o are the number of sensory, hidden and motor neurons, respectively.

C. SCEANN 2

The second one is an extended form of SCEANN1, which we propose. The signs of all the connection weights from sensory, hidden and motor neurons to a corresponding neuron are encoded in order to increase variety in the architecture. Thus, each block is composed of the signs $\{+, -\}$ and the presence/absence $\{1, 0\}$ of all the connection weights (Figure 3(b)). Therefore, the total length of the string is $L = 2(N_h + N_o)(N_s + N_h + N_o)$ bits.

D. SCEANN 3

Sensory, hidden and motor neurons are fully connected without coding the presence/absence of a connection from each neuron like SCEANN1 and SCEANN2. Thus, each block is composed of only signs of all the connection weights from sensory, hidden and motor neurons (Figure 3(c)). The synaptic strengths of all connections are set at 1. The total length of the string is $L = (N_h + N_o)(N_s + N_h + N_o)$ bits.

In the following sections, we apply these SCEANNs to evolutionary robotics tasks using simulated robots in order to investigate performances of SCEANNs.

III. Goal Reach Problem for a Mobile Robot

A. The Task and the Fitness Function

A two-wheeled robot was used in this experiment (Figure 4). The environment of the robot was a rectangular arena surrounded by walls with a target placed at the upper right corner. The control task used in this section was a goal reach problem where a robot approaches to a target (goal). At the beginning of each trial, a robot was always placed at the same initial position, the bottom left corner, at random orientations

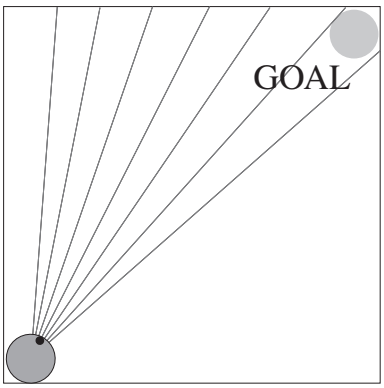


Fig. 4. Experimental setup for a goal reach problem

(Figure 4). One trial ends either when the robot reaches the goal or when 200 steps are performed without the goal. The performance measure to be maximized was as follows:

$$Fitness = \frac{1}{NumTrials} \sum_{i=1}^{NumTrials} (1 - \frac{Step_i}{MaxStep}) \quad (3)$$

where $NumTrials$ is the number of trials for a robot (8 trials for each initial orientation of a robot) and $MaxStep$ is set at 200. The fitness function increases as the robot reaches the goal more quickly.

The robot was provided with N_s infrared proximity sensors which have a maximum detection range in the environment. If a target intersects a proximity sensor, the sensor outputs a value inversely proportional to the distance between the target and the sensor. Employing a mathematical model of a mobile robot (Figure 5), the displacement of the robot was computed as follows:

$$\begin{aligned} x_{t+1} &= x_t + \frac{V_R + V_L}{2} \cos\theta_t + \omega_{xt+1} \\ y_{t+1} &= y_t + \frac{V_R + V_L}{2} \sin\theta_t + \omega_{yt+1} \\ \theta_{t+1} &= \theta_t + \frac{V_R - V_L}{2R} + \omega_{\theta t+1}, \end{aligned} \quad (4)$$

where V_R and V_L are the velocities applied to the right and left wheel respectively, R is the radius of a robot, $2R$ is the interval between the wheels and $\omega_{[\cdot]t}$ is the system error. It has been known that the system error in Equation (4) is modeled as the normal distribution $N(0, \sigma)$ with mean zero and standard deviation, σ [12]. For this experiment, σ was set at the 10 percent of the displacement at the current step.

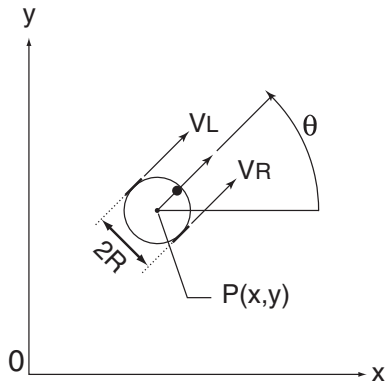


Fig. 5. Simulated model for a mobile robot

B. Simulation Conditions

For this experiment, a robot's controllers were SCEANNs with N_s sensory neurons, 2 fully interconnected motor neurons and N_h fully interconnected hidden neurons. The performances of the controllers were investigated over the relatively large N_s range, where $Ns \in \{1, 2, 3, 5, 7\}$, because the number of sensors generally depends on the setting of such an experiment that EANNs are applied to. The performances over the relatively small N_h range were also investigated, where $N_h \in \{1, 2, \cdots, 5\}$, because SCEANNs aim to shorten the length of the genotype. Such results as to vary N_s and N_h respectively would be useful for EC practitioners.

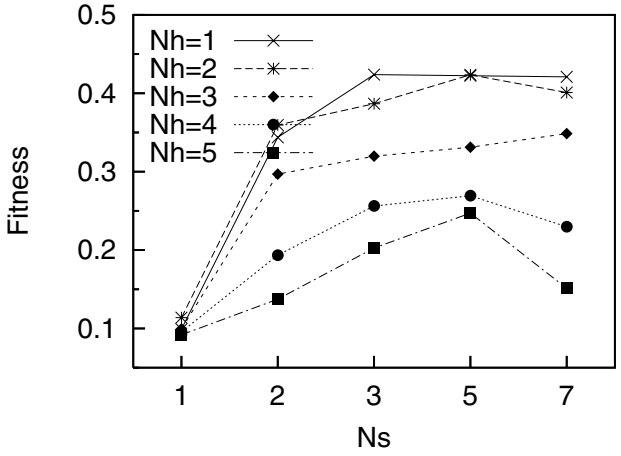
Computer simulations were conducted using populations of size 25. The standard GA (SGA) and the operon-GA (OGA)[13] were employed to evolve the string of SCEANNs. The OGA uses standard bit mutation and five additional genetic operators: *connection*, *division*, *duplication*, *deletion* and *inversion*. The probabilities for genetic operations were set at 0.3 for *connection* and *division*, 0.6 for *duplication* and 0.3 for *deletion* and *inversion*, based on our previous results in [14][15]. The genetic operation for the SGA was standard bit mutation. For both GAs, the per-bit mutation rate, q, was set at $1/L$. Crossover was not used for either GA [14]. Tournament selection was adopted. *Elitism*[1] was applied. The tournament size was set at 2 for the SGA and 6 for the OGA because the SGA prefers low selection pressure while the OGA prefers high selection pressure [14]. A generational model was used. Each run lasted 50 generations. We conducted 50 independent runs. All results were averaged over 50 runs. In this work, the performances in 50 generations are compared for both GAs to examine whether the evolving time of robots can be reduced. Thus, acquisitions of a feasible solution in that generations are considered to be the validity of the proposed approach.

C. Simulation Results

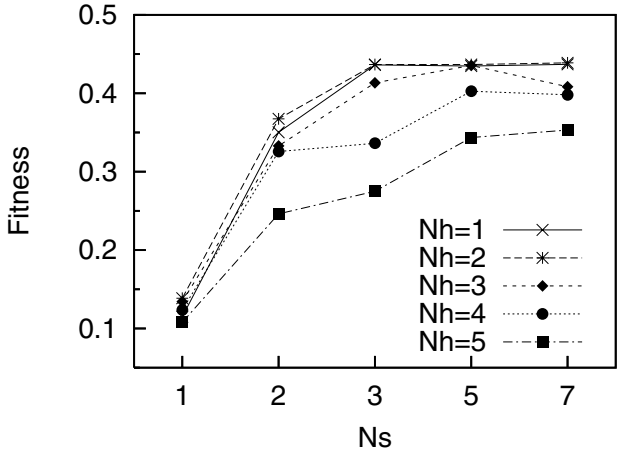
Figure 6 shows the maximum fitness at the final generation for the SGA and the OGA for controllers with $N_s \in \{1, \cdots, 7\}$ and $N_h \in \{1, \cdots, 5\}$. Figure 6(a), 6(b), 6(c) and 6(d) show the results for SCEANN1 and SCEANN2, respectively. The higher final fitnesses were achieved with the increase of N_s and with the decrease of N_h for both GAs. The OGAs performed better than the SGAs for each N_s and N_h.

Figure 6(e) and 6(f) show the results for SCEANN3. Also, the higher fitnesses were achieved with the increase of N_s. The final fitnesses became higher in order of $N_h = 1 \rightarrow$

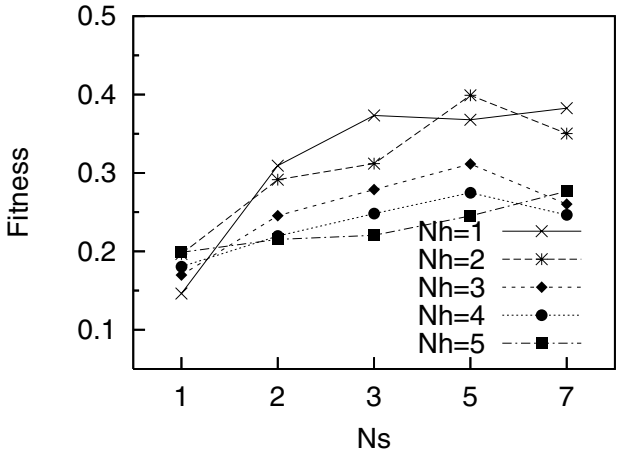
[1]The fittest individual of each generation was passed un-mutated to the next generation (if several individuals had the highest fitness, one was randomly chosen.)



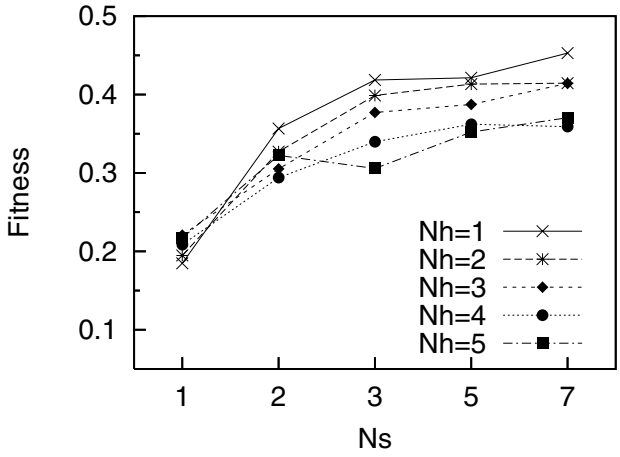
(a) SCEANN1 by the SGA



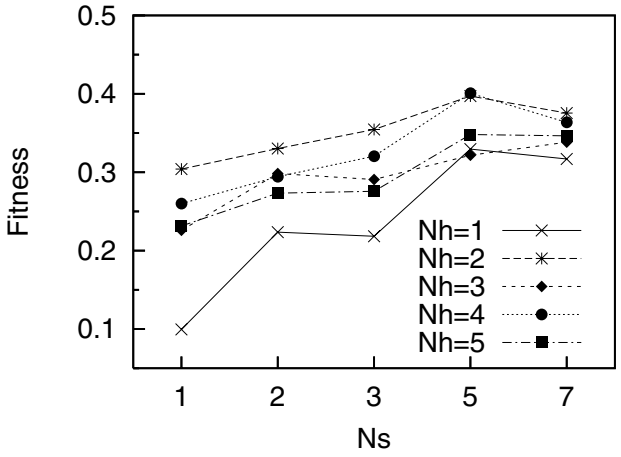
(b) SCEANN1 by the OGA



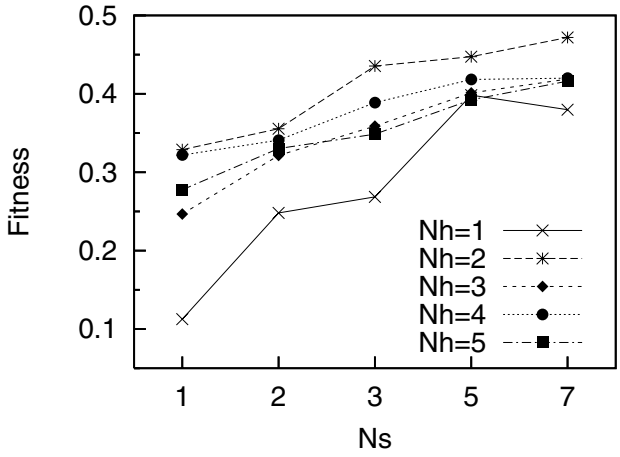
(c) SCEANN2 by the SGA



(d) SCEANN2 by the OGA



(e) SCEANN3 by the SGA



(f) SCEANN3 by the OGA

Fig. 6. Maximum fitness at the final generation for each coding by the SGA and the OGA on a goal reach problem

$5 \rightarrow 3 \rightarrow 4 \rightarrow 2$ for both GAs. Unlike SCEANN1 and SCEANN2, both GAs performed worst with $N_h = 1$ in all N_h. The OGAs performed better than the SGAs for each N_s and N_h.

For the SGA with $N_h = 1, 2, 3$, SCEANN1 performed better than SCEANN2 and SCEANN3. With $N_h = 4, 5$, SCEANN3 performed better than SCEANN1 and SCEANN2. Whereas, for the OGA with $N_h = 1, 3$, SCEANN1 performed better than SCEANN2 and SCEANN3. With $N_h = 2, 4, 5$, SCEANN3 performed better than SCEANN1 and SCEANN2.

Figure 7 shows the typical behavior of the best evolved controller (Figure 8) for SCEANN3 with $(N_s, N_h) = (7, 2)$ by the OGA. The robot turned right at the initial position to direct toward the goal, then approached to it. The trajectory of the robot was almost like a straight line.

On these results, it seems likely that the control task used in this section was easy because the task was achieved in almost all runs for each condition. In the next section, another control task was used, where performances of the codings would be more distinguishable than the task used in this section.

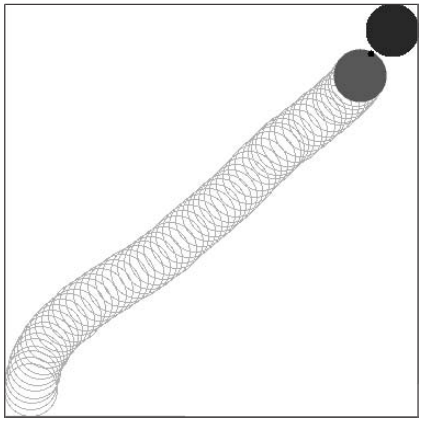


Fig. 7. Behavior of the genetically determined controller for SCEANN3 with $(N_s, N_h) = (7, 2)$ by the OGA

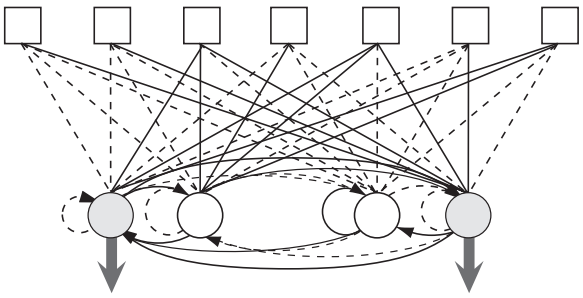


Fig. 8. A genetically determined controller for SCEANN3 with $(N_s, N_h) = (7, 2)$ by the OGA on a goal reach problem: the solid lines indicate excitatory connection weights and the dashed lines inhibitory.

IV. Pole-balancing Problem

A. The Task and the Fitness Function

The control task used in this section was a pole-balancing problem[16][17], which is a classical test problem not only in evolutionary robotics but also in control theory. A single pole is centered on a cart, which may move left or right horizontally along the track (Figure 9). The cart must move left or right in order to keep the pole balanced and avoid the track boundaries. In this experiment, the length of the pole is 1.0 m. The mass of the pole is 0.1 kg and the one of the cart is 1.0 kg. The maximum force to push the cart horizontally is 10 N. The cart and pole system were simulated for $3,000$ time steps using Euler's method with the time step, 0.02 seconds. The performance measure to be maximized was the number of time steps in which the system keeps balanced. One trial ends either when the pole falls past 36 degrees or when the cart reaches the boundary of the 4.8 meter track.

The state of this system is defined by the following state variables: the position of the cart on the track, x, the angle of the pole from vertical, θ, the velocity of the cart, $\dot{x}$, and the angular velocity of the pole, $\dot{\theta}$.

B. Simulation Conditions

For this experiment, controllers were SCEANNs with Ns sensory neurons, where $Ns \in \{2, 4\}$, 1 output neuron and N_h fully interconnected hidden neurons, where $N_h \in \{1, 2, \cdots, 5\}$. For $Ns = 4$, the state information, x, θ, $\dot{x}$ and $\dot{\theta}$, is given to the controller, and for $Ns = 2$, the amount of state information is restricted, that is, only providing x and θ. The state variables were scaled to $[0.0, 1.0]$ before being input to the controller. Every time step, the controller outputs a force value in the range $[-10.0, 10.0]$ N. The initial angle for the pole was set at 1° so that the pole naturally falls without any control.

Computer simulations were conducted using populations of size 25. The SGA and the OGA were employed to evolve the string of SCEANNs with the same settings as those in the previous section. Each run lasted 50 generations. We conducted 50 independent runs. All results were averaged over 50 runs.

C. Simulation Results

Figure 10 shows the maximum fitness at the final generation for the SGA and the OGA for controllers with

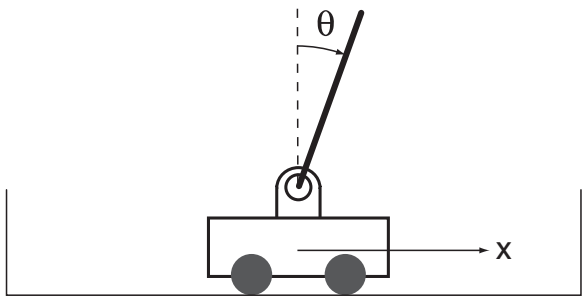
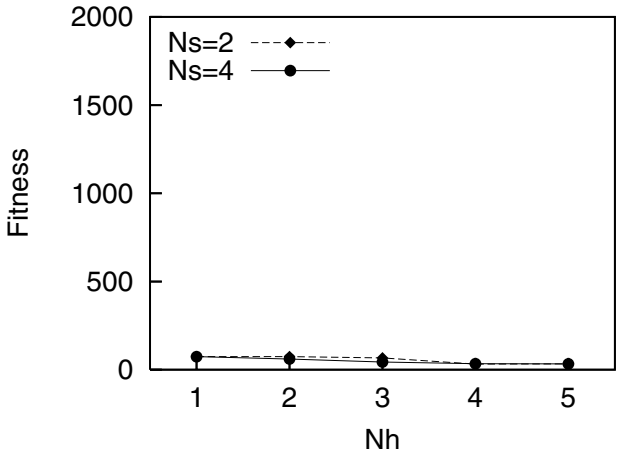
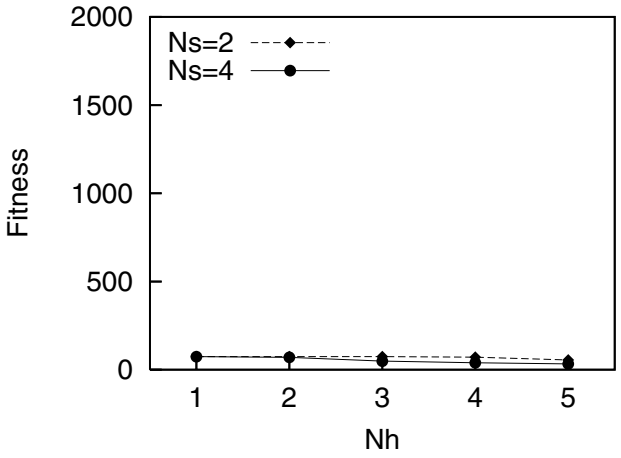


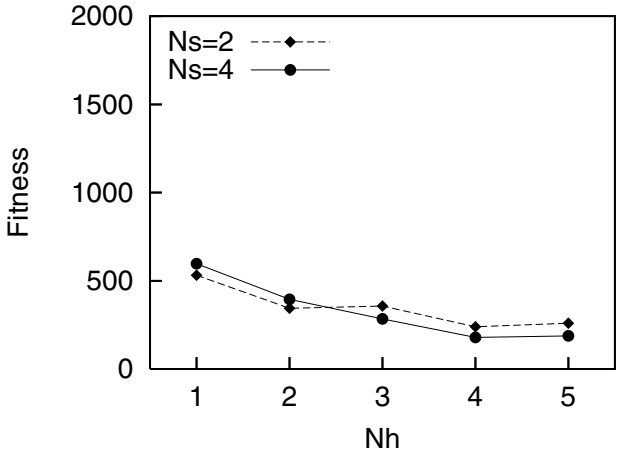
Fig. 9. Experimental setup for a pole-balancing problem



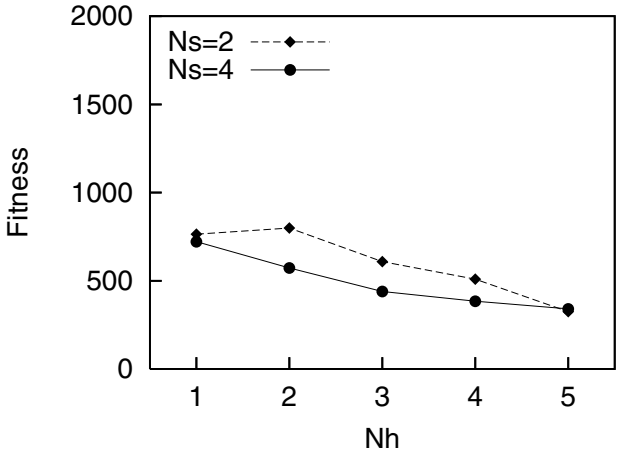
(a) SCEANN1 by the SGA



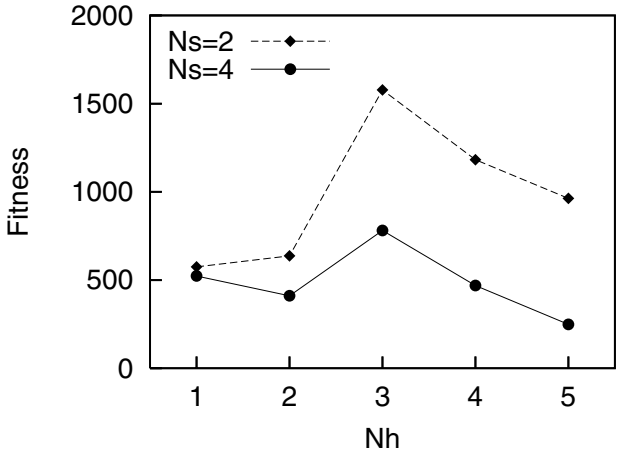
(b) SCEANN1 by the OGA



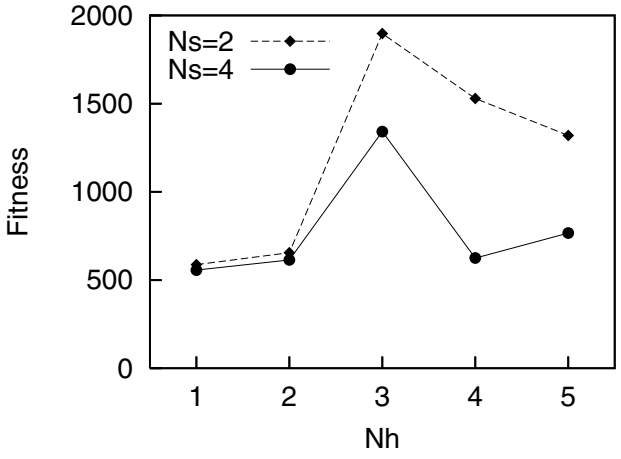
(c) SCEANN2 by the SGA



(d) SCEANN2 by the OGA



(e) SCEANN3 by the SGA



(f) SCEANN3 by the OGA

Fig. 10. Maximum fitness at the final generation for each coding by the SGA and the OGA on pole-balancing problems

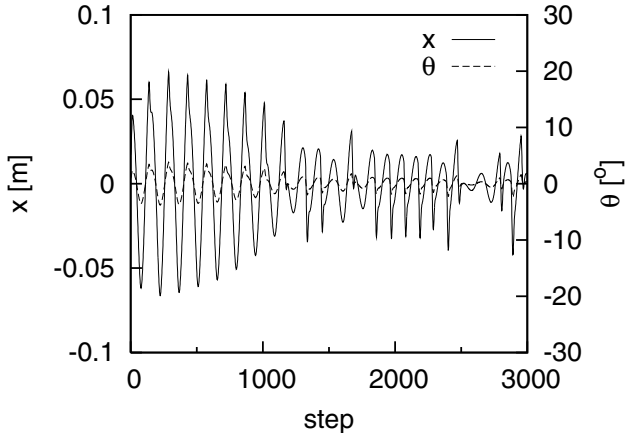


Fig. 11. Position of the cart and the angle of the pole at each time step

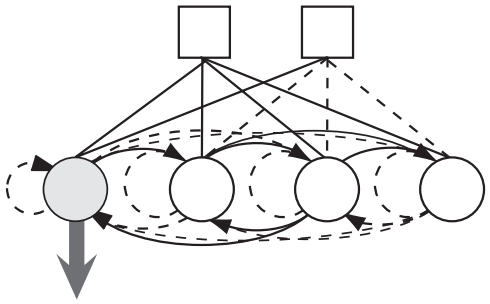


Fig. 12. A genetically determined controller for SCEANN3 with $(N_s, N_h) = (2, 3)$ by the OGA on a pole-balancing problem: the solid lines indicate excitatory connection weights and the dashed lines inhibitory.

$N_s \in \{2, 4\}$ and $N_h \in \{1, \cdots, 5\}$. Figure 10(a) and 10(b) show the results for SCEANN1. For this coding, the pole lost its balance in the early time steps under all conditions.

Figure 10(c) and 10(d) show the results for SCEANN2. The lower fitnesses were achieved with the increase of N_h for both GAs. The OGAs performed better than the SGAs for each N_s and N_h.

Figure 10(e) and 10(f) show the results for SCEANN3. The higher fitnesses were achieved with the increase of N_h for both GAs but they fall sharply when N_h exceeds 4. The differences between for $Ns = 2$ and for $Ns = 4$ were much more pronounced than for SCEANN2, that is, the fitnesses were higher for $Ns = 2$ than those for $Ns = 4$ for both GAs. This result is different from those obtained in general cases because it is known [17] that pole-balancing problems without velocity information are more difficult than those with it. As with SCEANN2, the OGAs performed better than the SGAs for each N_s and N_h.

Figure 11 shows the behavior of the best evolved controller (Figure 12) for SCEANN3 with $(N_s, N_h) = (2, 3)$ by the OGA. The controller was able to balance the pole and the cart by keeping them swinging left and right. The amplitudes of the oscillation were so small as not to reach the boundaries.

In this experiment, SCEANN3 with $(N_s, N_h) = (2, 3)$ for the OGA shows best performance in all conditions. However, those results cannot be considered fully satisfactory because the success rates of SCEANN3 achieving the task (balancing the pole and the cart for the maximum time steps) were not so high (Table I and II).

TABLE I

SUCCESS RATE (%) OF SCEANN3 FOR THE SGA IN 50 RUNS

N_s \ N_h	1	2	3	4	5
2	0	0	34	26	20
4	0	0	12	4	0

TABLE II

SUCCESS RATE (%) OF SCEANN3 FOR THE OGA IN 50 RUNS

N_s \ N_h	1	2	3	4	5
2	0	0	44	36	26
4	0	0	26	8	6

V. CONCLUSIONS

In this work, we proposed simply coded evolutionary artificial neural networks in order to shorten the time to evolve robots and investigated the performance of them on evolutionary robotics tasks using simulated robots. Two types of GA were employed to achieve higher final fitnesses as well as achieve high fitnesses faster. Our results can be summarized as follows:

- In a goal reach problem, increasing N_s improved the performance of SCEANNs for both GAs. SCEANN1 shows good performances when N_h is small, and so does SCEANN3 when N_h is relatively large. The controllers reaching the goal were obtained in almost all runs.
- In a pole-balancing problem, SCEANN3 for $N_h = 3$ shows the best performances for both GAs. Also, SCEANN3 achieved higher final fitnesses without velocity information than those with velocity information. The controllers keeping the pole balanced for the maximum time steps were obtained in many runs.
- The OGAs performed better than the SGAs for each N_s and N_h on both tasks.

These results suggest that SCEANNs are applicable to relatively easy control tasks, such as the navigation of mobile robots. From the analytical point of view, the simplicity of the obtained architectures would make behavioral analyses in a neuron level easier.

Establishing the generality of these results will require investigating the performance of SCEANNs on real robot problems.

REFERENCES

[1] R. Pfeifer. and C. Scheier, Understanding Intelligence, MIT Press, Cambridge, 1999.

[2] S. Nolfi and D. Floreano, Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines, MIT Press, 2000.

[3] R. A. Brooks, Artificial Life and Real Robots, In Proceedings of the First European Conference on Artificial Life, pp.3-10, 1992.
[4] N. Jakobi, Half-baked Ad-hoc and Noisy: Minimal Simulation for Evolutionary Robotics, In Proceedings of the Fourth European Conference on Artificial Life, pp.348-357, 1997.
[5] O. Miglino, H. H. Lund and D. Nolfi, Evolving Mobile Robots in Simulated and Real Environments, Artificial Life 2, pp.417-434, 1995.
[6] D. Keymeulen, M. Iwata, K. Konaka, R. Suzuki, Y. Kuniyoshi and T. Higuchi, Off-line Mode-free and On-line Model-based Evolution for Tracking Navigation Using Evolvable Hardware, In Proceedings of the First European Workshop on Evolutionary Robotics, Springer-Verlag, Paris, 1998.
[7] Y. Katada and K. Ohkura, An Update Method of Computer Simulation for Evolutionary Robotics, Intelligent Autonomous Systems 9 (IAS-9), pp.357-364, 2006.
[8] W. Maass and C.M. Bishop, Pulsed Neural Networks, MIT press (1998)
[9] X. Yao, Evolving Artificial Neural Networks, In Proceedings of the IEEE, 87(9):1423-1447, 1999.
[10] D. Floreano, C. Mattiussi, Evolution of Spiking Neural Controllers, In Gomi, T. (ed.): Evolutionary Robotics: From Intelligent Robots to Artificial Life (ER'01), AAI Books, Springer-Verlag, pp. 38-61, 2001.
[11] J.C. Zufferey, D. Floreano, M. van Leeuwen and T. Merenda, Evolving Vision-based Flying Robots, 2nd International Workshop on Biologically Motivated Computer Vision (BMCV'2002), Lecture Notes in Computer Science, pp. 592–600, 2002.
[12] K. Komoriya, E. Oyama and K. Tani, Planning of Landmark Measurement for the Navigation of a Mobile Robot, Journal of the Robotics Society of Japan, Vol. 11, No. 4, pp.533-540, 1993 (in Japanese).
[13] K. Ohkura, K, K. Ueda, Adaptation in Dynamic Environment by Using GA with Neutral Mutations, International Journal of Smart Engineering System Design, 2, pp.17-31, 1999.
[14] Y. Katada, K. Ohkura, K. Ueda, Tuning Genetic Algorithms for Problems Including Neutral Networks -A More Complex Case: The Terraced NK Problem-, In Proceedings of the 7th Joint Conference on Information Sciences, pp.1661-1664, 2003.
[15] Y. Katada, K. Ohkura, K. Ueda, An Approach to Evolutionary Robotics Using the Genetic Algorithm with Variable Mutation Rate Strategy, In Proceedings of the 8th Parallel Problem Solving from Nature (PPSN VIII), pp.952-961, 2004.
[16] D. E. Moriarty and R. Miikkulainen, Efficient Reinforcement Learning through Symbiotic Evolution, Machine Learning, Kluwer Academic Publishers, Vol. 22, pp. 11–33, 1996
[17] F. J. Gomez and R. Miikkulainen, Solving Non-Markovian Control Tasks with Neuro-Evolution, In Proceedings of The International Joint Conference on Artificial Intelligence (IJCAI99), pp. 1356-1361, 1999