Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Evolving neural behaviour control for autonomous robots

by

Martin Hülse, Bruno Lara, Frank Pasemann

and Ulrich Steinmetz

Preprint no.: 19 2001

Evolving Neural Behaviour Control for Autonomous Robots

Martin Hülse[1], Bruno Lara[1], Frank Pasemann[1,2], and Ulrich Steinmetz[2]

[1] *TheorieLabor, Friedrich Schiller University, D-07740 Jena*

[2] *Max Planck Institute for Mathematics in the Sciences, D-04103 Leipzig*

Abstract

An evolutionary algorithm for the creation of recurrent network structures is presented. The aim is to develop neural networks controlling the behaviour of miniature robots. Two different tasks are solved with this approach. For the first, the agents are required to move within an environment without colliding with obstacles. In the second task, the agents are required to move towards a light source. The evolution process is carried out in a simulated environment and individuals with high performance are also tested on a physical environment with the use of Khepera robots.

1 Introduction

Within the frame of Synthetic Modeling and Embodied Cognitive Science [4] an evolutionary algorithm is presented. This algorithm, called ENS^3 for Evolution of Neural Systems by Stochastic Synthesis [7]), evolves the structure and size of artificial neural networks and optimizes the corresponding parameters at the same time. It is designed especially to generate networks with recurrent connectivity.

The starting hypothesis for the presented experiments is that internal dynamical properties of recurrent networks are the basis for cognitive capabilities and therefore should be also used for the control of robot behavior. It is well known that even small recurrent networks present a variety of dynamical characteristics such as bifurcating attractors, hysteresis effects, and chaotic attractors co-existing with periodic or quasi-periodic attractors [6]. So the idea is to use the ENS^3 algorithm to generate recurrent network structures for agents having to exist and survive in a complex environment.

For the experiments described in this paper the agents are Khepera robots [2]. Evolution is carried out using a Khepera simulator [1]. Finally, the recurrent networks with highest performance are tested on the real robot. Section 2 of this paper introduces the ENS^3 evolutionary algorithm. Section 3 describes the experimental set-up and the results are outlined in Section 4 which presents also a discussion of these results. Finally Section 5 concludes the paper.

2 ENS^3 Description

The algorithm is inspired by a biological theory of coevolution. Originally it was designed to study the appearance of complex dynamics and the corresponding structure-function relationship in artificial sensorimotor systems. The basic assumption is that in "intelligent" systems the behavior relevant features of neural subsystems, called neuromodules, are due to their internal dynamical properties which are provided by their recurrent connectivity structure. But the structure and interaction of such nonlinear subsystems with recurrences in general can not be designed to produce a reasonably complex dynamical behavior. Thus, the main focus of the ENS^3-algorithm is to evolve an appropriate structure of artifical neural networks.

To start the algorithm one first has to decide which type of neurons to use for the network. We prefer to have standard additive neurons with sigmoidal transfer functions for output and internal units, and use input units as buffers. The number of input and output units is chosen according to the

definition of the problem; that is, it depends on the pre-processing of input and post-processing of output signals. Nothing else is determined, neither the number of internal units nor their connectivity, i.e. self-connections and every kind of recurrences are allowed, as well as excitatory and inhibitory connections. Because input units are only buffering data, no backward connections to these are allowed.

To evolve the desired neuromodule we consider a population $p(t)$ of $n(t)$ neuromodules undergoing a variation-evaluation-selection loop, i.e. $p(t+1) = S \cdot E \cdot V \cdot p(t)$. The *variation operator* V is defined as a stochastic operator, and allows for the insertion and deletion of neurons and connections as well as for alterations of bias and weight terms. Its action is determined by fixed per-neuron and per-connection probabilities. The *evaluation operator* E is defined problem-specific, and it is usually given in terms of a fitness function. After evaluating the performance of each individual network in the population the number of network copies passed from the old to the new population depends on the *selection operator* S. This operator performs the differential survival of the varied members of the population according to evaluation results. In consequence of this selection process the average performance of the population will tend to increase. Thus, after repeated passes through the variation-evaluation-selection loop populations with networks solving the problem can be expected to emerge.

3 Experimental Set-Up

The aim of the experiments is to find recurrent neural networks which are able to control specific behavior of the Khepera robots. To achieve this the networks are evolved using the ENS^3 algorithm described in Section 2. As aforementioned the only necessary initial conditions, on terms of structure and size, are the input and output neurons. These constrains are defined by the structure of the Khepera robot.

The Khepera robots are miniature cylindrical robots (diameter of 55 mm), with the basic configuration the robot has 8 Infra Red (IR) sensors (6 at the front of the robot and 2 at the rear), and two wheels, each controlled by a DC motor with an incremental encoder (10 pulses per mm of advancement of the robot). Each sensor can be used in two modalities: as a proximity sensor (by emitting and measuring the reflected Infra Red light), and as a light sensitive sensor (by measuring the Infra Red component of the environment light).

Two basic behaviors were studied using an average population size of 50 individuals, obstacle avoidance and light seeking. The time interval for evaluating these individuals was set to 5000 simulator time steps. Furthermore,

we can influence the size of resulting networks by changing the probabilities of increase and/or decrease of neurons and/or synapsis and adding cost terms for neurons and connections to the given fitness functions, as described in Section 2.

For evaluating the performance of the robots the two dimensional Khepera simulator [1] was used. A suitable interface for the communication between the Kephera simulator and the evolutionary program was implented. In this way every neuromodule of a generation is sent to the simulator and will be directly used for controlling the simulated Kephera. The start position of the robots is randomly set for every generation but fixed for the evaluation of all neuromodules in one generation. The fitness function for the calculation of the performance is implemented in the simulator. Performance of every neuromodule (controlling the simulated Khepera) is calculated for the life span of each individual in each generation and then sent back to the ENS^3.

3.1 Creating Obstacle Avoidance Behavior

We want to evolve neuromodules which allow the Khepera robot to move in a given environment as long as possible without colliding with any obstacle. A typical environment used for this evolution task is shown in the Fig. **??**. Environments are build on the Khepera simulator. It is worth noting that the evolved network structures are expected to behave at least with the same performance in the physical robot as they do in the simulated environments.

For evolving neuromodules which have an obstacle avoidance behavior the 8 infra-red proximity sensors of the Khepera are used. The number of sensors determines the number of input neurons of the neurocontrollers. To control the motors governing the robot wheels positive and negative signals are required. A *tanh* transfer function satisfies this requirement and is therefore used for the output neurons as well as for hidden neurons. The bias term of the *tanh* function is set constant to zero for all of the neurons throughout the whole evolution process. The initial structure of the individual neuromodules have only 8 linear input neurons, as buffers and two nonlinear output neurons.

To generate an obstacle avoidance behavior we introduce a fitness function F_1 which states: For a given time t go straight ahead as long and as fast as possible. This is coded in terms of the two network output signals M_0 and M_1 as follows:

$$F_1 := \sum_{t=1}^{5000} M_0(t) + M_1(t), \tag{1}$$

where $-1 \leq M_0, M_1 \leq 1$. A second condition for calculation of the perfor-

mance is used.

If the robot has a collision before 5000 time steps the evaluation of the neuromodule stops and the value of the current performance is taken as the maximum performance of the individual. The maximum performance for the neuromodules in these runs is 10000.

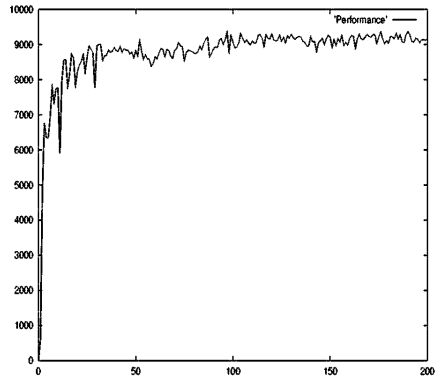

Figure 1: Performance of the best neuromodule of each generation.

The evolutionary process can be followed in Fig. ?? where the performance of the best individualsof the first 200 generations is shown. It shows the performance of the best neuromodule in every generation. At the beginning robots do not move or are spinning around having only one wheel active or both with opposite signals; as can be seen the performance is low. After a few generations displacement of the robots in straight trajectories can be observed. After 35 generations the robots move fast, turning away from the walls and exploring nearly all of the accessible space at the simulated environment. Fig. ?? shows a typical path of such simulated robot controlled by an evolved neuromodule.

During this phase the networks generally grow in size. To make or keep the network size small we increase the probabilities of deleting neurons and synapses within the evolutionary algorithm. This is done around the 50th generation, where it can be seen that the performance remains constant. After the probability change, the network size (number of synapsis and hidden neurons) decreases, but the performance and the observable behavior remains constant.

3.2 Light Seeking

For the second experiment in addition to the 8 proximity sensors we use the 8 light sensors of the Khepera; i.e., we now have 16 sensor inputs. The goal in

this experiment is to find a light source as fast as possible and move towards it. In the case of a moving light source the robot should follow it. Aiming to find a network structure only for this type of behavior the individuals should not be concerned with obstacles. The fitness function is coded as:

$$F := \sum_{t=1}^{5000} \alpha \cdot M(t) + \beta \cdot (1 - \Delta_m(t)) + \gamma \cdot (1 - P_l s(t)) \;, \tag{2}$$

where $0 \leq M \leq 2$, $0 \leq \Delta_m \leq 1$, $0 \leq P_l s \leq 1$ and α, β and γ are appropriate constant parameters.

The fitness function states: move forward, as M is the sum of the unsigned output to the motors, minimize turning, as Δ_m is the difference between the unsigned motors output, and move towards light, as $P_l s$ is a value proportional to the activity of the light sensitive sensor with the highest activity.

4 Results and Discussion

4.1 Avoiding Obstacles

A network, which generates good behavior in the simulated environment as well as on the physical robot is shown in figure 2. Input neurons are represented by black circles with the position they have in the robot. This network was randomly selected and represents the best individual of the 150th generation. It uses two hidden neurons, which are self-excitatory, one of this is larger than 1. It is known that for a single neuron with positive self-connection larger than 1 there is an input interval over which hysteresis phenomena (flipping between two stable states) can be observed [5]. It seems that this is used for instance to get out of situations which usually generate deadlocks.

The output neuron driving the left motor is self-inhibitory, and has also an inhibitory connection from the other output neuron, which drives the right motor.

The behavior in the real robot generated by this network is similar to the plotted paths in Fig. **??**. The robot presents obstacle avoidance as well as exploration abilities. Letting the robot move in its physical environment after a while it will have visited almost all areas within reach. This can of course not be achieved by pure wall following behavior, as it is usually learned by robots.

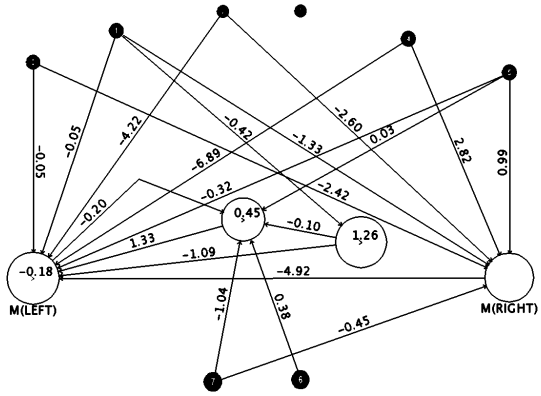

Figure 2: Network with highest performance of the 150th generation.

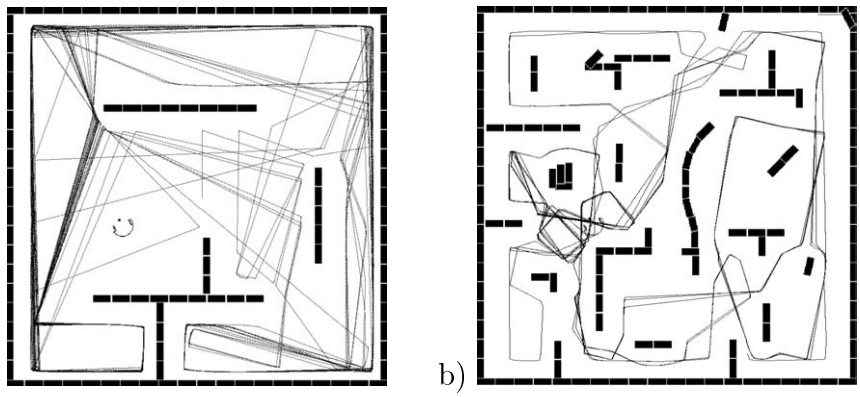

Figure 3: Typical paths followed by the simulated robots with evolved neurocontrollers for obstacle avoidance.

4.2 Light Seeking

After only a few generations the evolved structures presented light seeking behaviour. The individuals advance towards the light when this is within the reach of the light sensors. The individual with the highest performance of the 50th generation is shown in Fig. 4. Again, black circles show input neurons, the first group of 8 neurons represents the proximity sensors, the second group represents the light sensors.

It can be observed that there exist connections from the proximity sensors, even though there was no specific data requirement from their input in the definition of the fitness function. These connections are related to the straight trajectory of the robot in the absence of light.

When the physical robot was used, the individuals go in straight trajectories towards the light. If the light is moved they follow it. When there

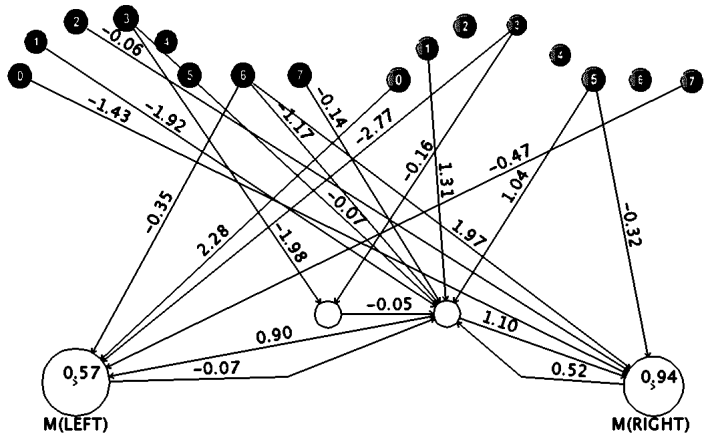

Figure 4: The network with the best performance of the 50th generation.

is no light present in the environment the individuals move in a wide semi-circular path. If the connections coming from the first 8 inputs neurons were manually deleted the robot presented a different behaviour. In the absence of light, it will turn in its own central axis, but when presented with the light source it would advance towards it and keep following it if the light source was in movement. Trajectories, with different starting positions, followed by the best individual of generation 50 are shown in Fig.??.

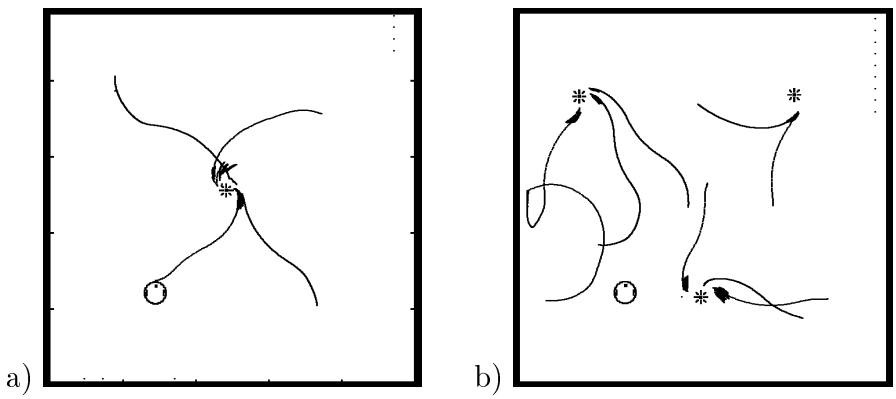

Figure 5: Typical paths followed by the simulated robots with evolved neuro-controllers for light seeking.

For the first simulated environment there is only one light source in the center. It is seen that disregarding the starting position the robot will always move towards the light and stay around it. On the second environment the robot moves towards the light source which finds closest at the front.

5 Conclusions

We have presented an evolutionary algorithm that optimizes the structure and characteristics of recurrent neural networks for control of miniature robots. Two different *tasks* were investigated. For both tasks, the results of evolution present small structures that exhibit complex behaviour after relatively small number of generations.

Future research will be done on the different possible approaches to the interface modules performing different tasks.

References

[1] Michel, O., *Khepera Simulator* Package version 2.0: Freeware mobile robot simulator written at the University of Nice Sophia-Antipolis by Oliver Michel. Downloadable from the World Wide Web at `http://wwwi3s.unice.fr/~om/khep-sim.html`

[2] F. Mondala, E. Franzi, and P. Ienne. Mobile robots miniturization: a tool for investigation in Control Algorithms. In *Proceedings of ISER' 93*, Kyoto, October 1993.

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

[4] Pfeifer, R., and Scheier, C. (2000), *Understanding Intelligence*, MIT Press, Cambridge.

[5] Pasemann, F. (1993), Dynamics of a single model neuron, *International Journal of Bifurcation and Chaos*, **2**, 271-278.

[6] Pasemann, F., Structure and Dynamics of Recurrent Neuromodules, *Theory in Biosciences,* **117***, 1-17*

[7] Pasemann, F. (1998), Evolving neurocontrollers for balancing an inverted pendulum, *Network: Computation in Neural Systems*, **9**, 495–511. –